PsPM: Psychophysiological Modelling

November 7, 2023

Version 6.1.0

by the PsPM team[1]:

Dominik R Bach, Giuseppe Castegnetti, Laure Ciernik, Samuel Gerster, Saurabh Khemka, Christoph Korn, Samuel Maxwell, Tobias Moser, Philipp C Paulus, Ivan Rojkov, Matthias Staib, Yanfang Xia, Eshref Yozdemir, Teddy Zhao and collaborators

---

[1]If you have comments on or error corrections to this documentation, please send them to the PsPM team or post them on: bachlab.org/pspm

Contents

Part I

Background

1 What is psychophysiological modelling?

1.1 The psychophysiological inverse problem

Psychophysiology is concerned with the relationship between the mind (i.e. psychological processes) and the body (including the nervous). This relationship is bidirectional: the mind influences the body, but processes in the body are monitored by the mind via proprioception and interoception. Knowledge on this bidirectional relationship affords for instance examining more closely how psychological factors influence health, or how the interceptive abilities of an individual affect his psychological well-being.

However, a very common application of psychophysiological knowledge is to measure processes in the body (sweating, heart activity, respiration, etc.) to assess processes in the mind. This is an inverse problem. The researcher is not actually interested whether skin conductance is different between two conditions (the forward relation). They are interested, for example, whether a subject successfully learned an association between a conditioned stimulus (CS) and an electric shock, and use skin conductance to measure this learning process. To make this inverse inference, the "turn the forward model around" - they ask, what has happened in the mind, given what is measured in the body. There are several way of doing this, and all require good knowledge of the forward relationship.

1.2 Operational analysis

Operational approaches seek to identify data features ("indices") that closely follow a psychological state of interest. Operational data analysis algorithms extract these data features, to "index" central states. For example, to infer stimulus-evoked sympathetic arousal (SA) from skin conductance responses (SCR), one may filter the SCR data to reduce observation noise, define a response window after the stimulus, and define some criteria do detect peaks within this window. The amplitude of such peaks may then be taken as "index" of SA. The development of such indices is an important application of psychophysiology. Such indices are usually based on qualitative or semi-quantitative models of how the index relates to the psychological state of interest. Model-based analysis has precisely the same goal as operational analysis - but seeks to make these implict models explicit (i.e. testable) in mathematical form - see [1] and [2] for a review.

1.3 Model-based analysis

Model-based approaches start with explicit, mathematical models that formulate how observed data are generated by psychological processes. For example, a model may formulate the relation between sympathetic arousal (SA) and skin conductance responses (SCR), $SA \mapsto SCR$, in mathematical form. This kind of model is often termed a "forward model": it predicts a data time series (SCR) from a known psychological process (SA). Another term for this kind of model is "generative", because it describes how a psychological process generates the data. When we analyse experimental data, we are faced with the opposite situation: we know the observed data data but not the central process, and seek to infer the central process from the data. In order to do so, one has to turn the forward model backwards, to arrive at the relation $SA \leftarrow SCR$. In statistics, this process is often termed "model inversion". It provides the most likely estimates of the central process, given the observed data and the model. Both conventional and model-based analysis aim to infer central process from observed data. The difference is that model-based methods use a stringent mathematical language and probabilistic model inversion to do so. Note that probabilistic models acknowledge the existence of noise in the data, and they do not seek to explain this noise with variation in psychological states. Hence, the criterion for the quality of a model-based method is not how well the model "fits the data", but how well it can recover the psychological variables of interest. In other words, the model should be able to fit the data but at the same time to ignore noise in the data.

2 Method comparison

How can we infer that one method of inferring a central state is better than another? Each method returns indices or estimates of the psychological state, but which ones are more precise? Psychological variables cannot be measured directly, so how can we compare our estimates to ground truth? The solution we propose is to experimentally create psychological states that are known. We can then evaluate how well a method recovers these known states. In the simple example of fear conditioning, we could compute a paired t-test on the CS+/CS- difference in the estimates and look at the t-value. the method that yields the highest t-value is most precise. But is it also significantly more precise than another method? This can be assessed with a formal model comparison.

In our example, we can try to predict CS identity (CS+/CS-) from the model estimates. This means we quantify evidence for a model in which individual participants' CS+ and CS- estimates are drawn from two distributions with different means, rather than the same mean. In other words,

we are using a regression model that seeks to predict the known psychological state (dependent variable) from the estimated psychological state (independent variable). Note that this is different from a more convential regression or ANOVA model in which the data are the dependent variable and experimental conditions are the independent variable. This difference is important. In our approach, when comparing different methods, the dependent variable (the true state) will be the same, and the independent variable (the state estimates) will differ. This means that the total sum of squares (TSS) in the model will be constant. Thus, we can use the residual sum of squares (RSS) to compute model evidence and compare models - i.e. compare evidence for the statement that the method's estimates of the psychological state predict the true psychological state. We have termed this "predictive validity". Note that the model is not predictive in time - we are not trying to predict unknown data values. We are trying to predict true psychological state from the estimated state.

To quantify model evidence, PsPM uses the following approximation to Akaike Information Criterion (AIC) [3]:

$$AIC = n \log \left(RSS/n \right) + 2k = -2 \log \left(L \right) + 2k,$$

where n is the number of data points in the predictive model, L is the maximum of the likelihood function, and k the number of parameters in the predictive model. k is constant for all methods that are evaluated in a methods comparison. We only interpret AIC differences, so this complexity term disappears from methods comparison. AIC differences divided by 2 can be interpreted as log Bayes Factors (LBF). In our case this quantifies the relative evidence that one method's estimates predict the true psychological state, relative to the other method. An absolute LBF of greater than 3 is usually regarded as decisive. This is because for a classical significance threshold of $\alpha = .05$, the probability of the data given the null hypothesis is $p < .05$. Similarly, for an LBF difference larger than 3, the relative probability that the inferior method predicts the true psychological state given the data is $p < \exp \left(-3 \right) \approx 0.05$ [4].

All methods included in PsPM have empirically been shown to provide better or equal predictive validity than the best available corresponding conventional index. Although in most validation experiments only two conditions need to be distinguished (a discrimination problem), the goal of PsPM is to provide minimum-variance estimates of cognitive processes also in situations in which the cognitive process varies across more than just two levels. This is why we do not use discrimination methods to evaluate the models. Also, since most psychophysiological measures are influenced by a variety of cognitive processes, inference on a specific process unavoidably rests on a suitable experimental design in which different conditions only differ on this dimension, and not on other dimensions.

Of course, this approach to method comparison can also be used to improve operational indices, or event to create new indices in a blind classification approach based on large data sets.

> *Historical remark:* Some papers from the PsPM team have used the formula: $NLL = n \log (RSS/n)$. If NLL is read as an arbitrary variable name, this formula is consistent with the previous paragaphs. However, note that NLL in this formula is not the negative log likelihood (as confusingly implied in the papers), but really the AIC without complexity correction. The numerical value of the negative log likelihood would in this terminology be $NLL/2$.

3   General model structure

3.1   General

All model in PsPM split the relation between mind and body into two models, a peripheral and a neural model. The peripheral model describes how a neural impulse is converted to a measurable signal. The neural model describes the relation between the psychological process and the neural input into the peripheral system. This distinction is historically based on SCR models. The neural signal that elicits SCR can be measured by intraneural recordings, and its properties are in principle known. Hence it appeared plausible for various research groups to create a peripheral model [5, 6, 7] that takes an arbitrary neural input, and several neural models were then combined with this peripheral model. This distinction has later been used for other modalities, too: first because it appeared simple from a practical standpoint, secondly because it is often plausible to assume that the peripheral process is not directly influenced by psychological states, but only via a neural input. Note, however, that the peripheral/neural distinction in the model implementation is not always consistent with biophysical reality and tends to follow a need for manageable inversion algorithms. Indeed, some models collapse peripheral and stereotypical neural processes into the peripheral model. Examples are the GLMs for fear-conditioned HPR, PSR, and RAR - this is unlike the non-linear SCR model in which parameters of the neural model are estimated explicitly.

3.2   Peripheral models

The peripheral models in PsPM collapse all physiological and biophysical processes involved in the generation of the measured signal. For most psychophysiological measures, these processes are not known at the level of detail required to build true biophysical models from theory. In fact, biophysical models for the best-investigated modality, SCR, tend to not fit the

observed data very well [8, 5]. This is why PsPM uses purely phenomenolog-
ical models which regard the peripheral system as a black box. By applying
controlled inputs into the system, and measuring the output, one can then
approximate the workings of this black box in mathematical form, without
knowing its content.

LTI systems

Crucially, all models in PsPM assume that the peripheral system can be
described by one (or several) linear time invariant (LTI) systems with the
defining properties linearity and time invariance. A LTI system is unam-
biguously described by its response function (RF). By linearity, input and
output are linearly mapped so the responses to several inputs can be simply
obtained by summing the responses to individual inputs. Time invariance
means that the output does not explicitly depend on the time, i.e. that the
shape of the RF is constant over time while the amplitude can vary. It is
usually also assumed that the RF is constant over individuals, although this
assumption is not necessary in the PsPM framework.

In principle, linearity ensures pure summation of two overlapping inputs.
This is not realistic if the system has limited dynamic range and saturates
due to a quick succession of inputs (e. g. the heart rate cannot increase
indefinitely). One can formulate limits, for each data modality, within which
the model is a useful approximation to reality.

For some modalities, the two properties linearity and time invariance can
be explicitly tested by neural recordings (e.g. for SCR). For other modali-
ties, this is not possible.

Mathematically, the output $y(t)$ of a LTI system can be fully described by
convolving input $u(t)$ with the system's response function $h(t)$ and can be
written as

$$y(t) = u(t) * h(t) = \int_0^\infty u(t - \tau)h(\tau)d\tau.$$

This is the equation for a linear filter.

> *Note:* Convolution is more generally defined by integration over $\mathbb{R}$. In-
> tegrating over $\mathbb{R}^+$ (or alternatively, setting $h(t) = 0,\ t < 0$) ensures the
> causal structure of the model - only past inputs can have an effect only
> into the future, not future inputs into the past. It also reflects the imple-
> mentation via Matlab's vector convolution function, `conv.m`

In many cases, the peripheral system is not exactly LTI but can be ap-
proximated by the combination of several LTI systems. The common case is
the inclusion of RF derivatives to account for between-subject variability in
RF shape and latency, and a special case the combination of a dilation and a
constriction RF in pupil size modelling. In our terminology, the RF then has
k "components": $h(t) = h_{1..k}(t)$.

In such cases, the amplitude estimates for several response functions need to be combined. In PsPM, this is usually (unless indicated otherwise) done by multiplying each component of the RF with its amplitude estimate, and adding them. The peak of highest absolute amplitude is then identified, and its signed amplitude extracted as estimate of neural input amplitude.

Signal frequencies in the data that are not present in the RF do not contribute to the model inversion and can be filtered out. If the true RF were known, the matched filter theorem provides a way of theoretically deriving a filter that maximises the signal-to-noise ratio for the model inversion. In most cases, the true RF is not precisely known, and also varies between individuals. This is why PsPM seeks to empirically determine the filter characteristics that maximise predictive validity of psychological state estimates.

3.3 Neural models

The neural model in PsPM describes the form of the neural input that the peripheral model can take, and thus, the relation between mind and neural system. In general, model inversion benefits from constraining the neural model, i.e. from reducing the number of parameters that have to be estimated. We have empirically shown for SCR models that in many cases, a strongly constrained neural model will result in higher predictive validity [9, 10, 11].

Most PsPM models assume an instantaneous (delta) neural input into the peripheral system at a known time point t_i:

$$u\left(t\right) = \delta\left(t_i\right).$$

Only the amplitude of the neural input is then estimated, and is taken as estimate of the psychological state. This renders the model suitable for GLM inversion (see below). Less constrained models require non-linear inversion methods and can also deal with variable onset, dispersion, or shape, of the neural input.

3.4 Model inversion

3.4.1 Hierarchical summary statistic approach

It is common in psychophysiology (and psychology in general) to compute summary statistics for each cell of the design, and for each subject. These estimates of are then entered into a group-level model. To implement this approach, PsPM estimates parameters for each participant separately, either per experimental condition, or per trial with subsequent averaging within the experimental condition. In line with fMRI terminology (and different from many branches of psychology), we use the terms "first-level" and "second-level" for the individual and group level.

3.4.2 Multi-level modelling

Multi-level models, also termed linear mixed effects model (LME) or random-effect models, are becoming increasingly popular in psychology [12]. The idea is to take variance between trials into account: a summary statistic that is computed from individual data points with high variance is less precise that a statistic from low-variance data, and such information is lost when summarising within each condition. PsPM offers the possibility to compute trial-by-trial estimates (by modelling each trial as a separate condition in GLM, or by default in DCM for SCR). This approach has been used for SCR, PSR and SEBR [13], see also [14] and [15]. For GLM, modelling trial-by-trial amplitudes and averaging within conditions yields the same result as modelling condition-by-condition amplitudes, if the design matrix is of full rank and there is only one basis function. With more than one basis function, the results will deviate for the second and further basis functions in the basis set, because the orthogonalisation with the basis set is not a linear operation.

In theory it would also be possible to account for variability within the time series, i.e. between data points, not only between trials. Such approaches are implemented for fMRI, but they are not trivial. First, model inversion is much slower by the inclusion of many subjects. Secondly, in the time-series data analysed in PsPM, errors are usually not independent. This is usually not a problem for parameter estimation but impacts on the effective degrees of freedom and thus renders statistical tests in these models invalid, and this would apply to multi-level models, too. fMRI softwares solve this problem by estimating an auto-regression model on the first level but this is usually done on data from many data channels/voxels, and this is not possible for single-channel data in psychophysiology. This is why PsPM does not offer statistical tests on the time-series level. For the trial-wise or condition-wise summary statistic approach in PsPM, the dependencies between data points on the first level are unproblematic.

3.4.3 Probabilistic model inversion

All peripheral and neural models in PsPM are approximations to reality. It is hence assumed that there is measurement error (imprecision of the peripheral model) ε in the data, and that the neural input is not truthfully described by the neural model such that it contains latent error, ω. While this distinction between ε and ω is theoretically interesting and often physiologically plausible, the model inversion methods collapse these two sources of error:

$$y\left(t\right) = \varepsilon\left(t\right) + \left[u\left(t\right) + \omega\left(t\right)\right] * h\left(t\right)$$

$$= \varepsilon\left(t\right) + \omega\left(t\right) * h\left(t\right) + u\left(t\right) * h\left(t\right)$$

$$= \epsilon\left(t\right) + u\left(t\right) * h\left(t\right).$$

3.4.4 General linear convolution model (GLM)

Under the assumption of delta neural model, we can harness GLM to estimate the neural input amplitude. A GLM can be written as

$$Y = X\beta + \epsilon.$$

Here, Y is the vector of observations (time series data), β is a vector of input amplitude parameters and ϵ is the error. X is design matrix in which each column is obtained by convolving impulse functions at known time points (e.g. event onsets for one condition) with each component of the RF. Each column takes the form:

$$X_{ij}\left(t\right) = u_i\left(t\right) * h_j\left(t\right),$$

where $u_i\left(t\right)$ is the neural input with unit amplitude for condition i, and j is the index of the RF component. Finally, X also contains a column for the intercept. It is possible to concatenate data from different experimental sessions, and an intercept is modelled for each session. The maximum-likelihood amplitude estimates are then computed using the Moore-Penrose pseudoinverse X^+, implemented in the Matlab function `pinv.m`:

$$\hat{\beta} = X^+Y.$$

This allows to deal with situations in which X is not of full rank.

3.4.5 Non-linear model inversion

Models in which timing and/or dispersion of the neural input need to be estimated require non-linear model inversion. PsPM contains several efficient non-linear model inversion methods:

- a Variational Bayes (VB) algorithm [16] for dynamic systems. This can be applied either to an entire data set, or on a trial-by-trial basis, and requires the peripheral system to be written in the form of an ordinary differential equation (ODE).

- a Matching Pursuit algorithm that provides a fast approximation to the VB inversion for some models

4 Skin conductance (SCR) models

4.1 General

SCR arise from opening of sweat glands and are elicited via the sympathetic nervous system with negligible parasympathetic transmission (see [17] for the physiology of SCR). Nerve fibres carrying impulses to the sweat glands are termed "sudomotor nerve" (SN) fibres and can be measured by intraneural recordings. SN fibres are slow C-fibres. From their end terminal, the neurotransmitter Acetylcholine diffuses through the skin to reach sweat glands, a process on the time scale of up to a second.

The frequency and amplitude of SCR is mainly influenced by psychological arousal. Because not all forms of psychological arousal may influence SCR in the same way, we term the relevant psychological state "sympathetic arousal" (SA). Hence, the PsPM model is $SA \mapsto SN \mapsto SCR$.

4.2 Peripheral LTI model

4.2.1 Model evaluation

The peripheral LTI model can be evaluated explicitly (by intraneural recordings or stimulation) and implicitly (by assessing properties of the model $SA \mapsto SN \mapsto SCR$, an approach that collapses LTI violations with imprecision in the neural model).

Time invariance: the system's response does not explicitly depend on time, or in other words, a given input always produces the same output.

- Direct evidence: The amplitude of individual SN bursts is linearly related to the amplitude of ensuing SCR, with a considerable scatter [18]; to the maximal rate of sweat expulsion; and, somewhat more weakly, to the integrated sweat production during the skin response [19]. After initial dishabituation, constant SN stimulation leads to SCR, with constant amplitude and latency [20]. Repeated SN stimulation yields slightly different SCR shapes [21], although this could be due to variation in elicited neural responses that were not measured downstream. In summary, these findings are consistent with (but do not prove or disprove) the time invariance principle. [22] re-analysed data from an experiment in which brachial sudomotor neves were regularly stimulated under thoracic block and SCR recorded [20, 23]. They found that for stimulation rates below 0.6 s (1 stimulus every 1.7 s) around 95% of variance could be explained by an LTI model.

- Indirect evidence: We have shown that for short events (< 1 s duration) that are separated by at least 30 s, more than 60% of the variance in (high pass filtered) SCR can be explained under an LTI model, for aversive white noise bursts, aversive electric stimulation, aversive pictures, auditory oddballs, and a visual detection task [24]. Is this residual variance (40%) due to violations of the LTI assumptions? We have shown that in the absence of any event for 60 s, signal variance amounts to more than 60% of total variance during stimulus presentation. That is to say, by introducing events, residual baseline variance is reduced by 20%. This suggests that residual variance is not caused by violations of the time invariance assumption but variation of the neural input (for example, spontaneous fluctuations), and observation error.

Linearity: the system's response does not depend on previous inputs.

- Direct evidence: Latency of sweat expulsion is slightly shortened when sweat expulsion rate is very high (i. e. for very strong SN bursts) but not when SN firing is frequent [19]. SCR to individual SN stimulations depend linearly on skin conductance level, and are slightly suppressed upon repetition of the stimulation after 1 s [25]. This suggests that the linearity principle does not hold under all conditions. [22] re-analysed data from an experiment in which brachial sudomotor neves were regularly stimulated under thoracic block and SCR recorded [20, 23]. They found that for stimulation rates above 0.6 s (1 stimulus every 1.7 s), the response function looked very different from lower frequencies, suggesting non-linearities at that stimulation frequency. They did not quantitatively characterise these non-linearities.

- Indirect evidence: The linearity principle amounts to saying that SCR are not influenced by the (preceding) baseline signal. We tested this assumption by presenting two subsequent aversive white noise bursts with an inter stimulus interval (ISI) of either 2 s, 5.5 s, or 9 s, such that the baseline signal at these three time points differed markedly. Violations of the linearity principle in the peripheral system would imply that the amplitude of the subsequent response is dependent on the baseline signal, and thereby, upon the ISI. The second response was always smaller than the first. However, this was not dependent on the ISI, and hence not on the baseline signal. We interpret this effect as central repetition suppression (of the neural inputs into the peripheral system) and conclude that linearity is appropriate for the peripheral system. In other words, the peripheral response to one input is not modulated or affected by the response to another, even when they

overlap in time [24].

Model limits Both properties appear good approximations to reality as long as the time interval between two SN firing bursts is not very short (< 2 s) or when the SN firing burst is not extremely strong. It would be possible to model non-linearities in a linear model, using Volterra kernels, as in the analysis of functional magnetic resonance images in the software SPM [26, 27]. Such possibility is however not implemented in PsPM.

4.2.2 Skin Conductance Response Function (SCRF)

The peripheral model in its general form does not specify a particular form of the SCRF. There are three principled ways of defining a SCRF in PsPM:

1. Individual response function. The most precise option is to measure the SCRF for each individual by providing brief inputs into the peripheral system and measuring the response. PsPM allows specifying individual response functions. The benefit of this approach has not been empirically tested.

2. Canonical response function. The most parsimonious method is to use a so-called canonical SCRF which is assumed to be constant across individuals. We have previously shown that a canonical SCRF can explain on average 50% of the variance in individual SCR elicited by various stimuli, while individual SCRFs explain about on average 75% of the variance [24]. Using a canonical SCRF, linear and non-linear models show a good predictive validity, superior to peak-scoring methods [9, 11].

3. Constrained response function. One can estimate an SCRF from the data of an actual experiment and use this for analysis. It is usually necessary to constrain the shape of the SCRF to make the model estimable. PsPM provides two options for this:

  • In GLM, one can estimate parameters of the (orthogonalised) time derivative of the SCRF. In order to combine this with the canonical SCRF, the response can afterwards be reconstructed for each experimental condition, and the peak amplitude taken as estimate of SA. This approach has higher predictive validity than using the canonical SCRF alone, or using a less constrained SCRF (canonical with time and dispersion derivative) [9].

  • In the non-linear model for event-related SCR, one can use the combined data from all evoked responses from one individual to estimate an SCRF. This approach only considers data within the inter-trial interval. In a validation experiments study with ITI of 7-11 s, this had

lower predictive validity than using the canonical SCRF, and we would
not recommend it unless the ITI is longer than 20 s, so that the SCRF
tail can be unambigouosly estimated [11].

Canonical Skin Conductance Response Function   The canonical SCRF
that is currently implemented was derived from a large dataset of 1278
SCRs from 64 individuals subjected to different experimental manipulations
(white noise 85 dB and 95 dB, painful electric stimulation, aversive IAPS
pictures, detection of auditory oddballs, detection of a visual target in a
stimulus stream). We extracted and mean-centred the 30 s following each
stimulus onset and performed a principal component analysis. The first
principal component served as empirical SCRF [24].

Formulation for linear models   The first PCA component of the empirical
SCRF was approximated was fitted by a Gaussian smoothed bi-exponential
function (also named bi-exponentially modified Gaussian function), and this
provided better fit than other function categories, in particular Gaussian
smoothed monoexponential function, exponential-Gaussian hybrid function,
or smoothed sigmoid-exponential function. The function is thus defined as

$$h\left(t\right) \propto \int_0^t N\left(t - \tau\right)\left(E_1\left(\tau\right) + E_2\left(\tau\right)\right) d\tau \, , \; t \geqslant 0, \tag{4.1}$$

$$N\left(t\right) = \frac{1}{\sqrt{2\pi}\sigma}e^{-(t-t_0)^2/2\sigma^2},$$

$$E_i\left(t\right) = e^{-\lambda_i t},$$

with estimated parameters: $\hat{t}_0 = 3.0745$ seconds for peak latency; $\hat{\sigma} = 0.7013$ for definition of the rise time; $\hat{\lambda}_1 = 0.3176$ and $\hat{\lambda}_2 = 0.0708$ to define
the two decay components. This function is normalised to its maximum such
that an infinitely short SN impulse with unit amplitude elicits an SCR with
unit amplitude. This renders parameter estimates from linear models easily
interpretable in relation to conventional (peak-scoring) analysis.

> *Historical remarks:*
> (a) The appendix of [9] contains a typographic error in the definition of the canonical response function, in particular in the integration limits of eq. (4.1). However, in all versions of the software (PsPM 1.0-now) the canonical SCRF was constructed using the Matlab function `conv()` which defines vector convolution analogous to the integral defined above. This implementation has not changed since the initial formulation of the model.
> (b) The initial formulation of the SCRF [6] used a Gaussian-smoothed Gamma distribution, based on a smaller dataset. In the extended data set published in [9], the polynomial part of the Gamma distribution was estimated to be one, resulting in a Gaussian smoothed exponential function.

Formulation for non-linear models An alternative formulation is provided for non-linear models which require a definition of the SCRF in the form of an inhomogeneous ordinary differential equation (ODE). We use an third-order linear ODE to describe the data $y\,(t)$:

$$\dddot{y} + \vartheta_1 \ddot{y} + \vartheta_2 \dot{y} + \vartheta_3 y - u\,(t - \vartheta_4) = 0; \qquad (4.2)$$

where dot notation is used for time derivatives, and this formulation embeds convolution of the SCRF with the SN time series. Parameters were estimated from the empirical SCRF by using a Gaussian bump as canonical SN input (see section on SN below) as $\hat{\vartheta}_1 = 1.342052$, $\hat{\vartheta}_2 = 1.411425$, $\hat{\vartheta}_3 = 0.122505$, $\hat{\vartheta}_4 = 1.533879$. Amplitude of the response function is rescaled such that a canonical Gaussian bump SN impulse with unit amplitude elicits an SCR with unit amplitude.

Formulation for spontaneous fluctuations For modelling spontaneous fluctuations (SF), a different database was used that contained 1153 semi-automatically detected SF from 40 male participants [28] The first PCA component was taken as empirical SCRF which has slightly different shape than the empirical SCRF for evoked responses. A number of reasons may account for this difference, from participant selection over data pre-conditioning to differences in the canonical SN input. In the absence of further knowledge about the reason for this difference, we propose to use this empirical response function for SF. We estimated the ODE parameters (eq. 4.2): $\hat{\vartheta}_1 = 2.1594$, $\hat{\vartheta}_2 = 3.9210$, $\hat{\vartheta}_3 = 0.9235$, $\hat{\vartheta}_4 = 1.533879$ seconds. Amplitude of the response function is rescaled such that a canonical Gaussian bump SN impulse with unit amplitude elicits an SF with unit amplitude.

> *Note:* In the Matlab implementation, the third order ODE is converted to a 3-dimensional system of first-order ODEs. In that process, the order of the parameters $\vartheta_{1..3}$ is reversed, and they are renamed. With substitutions $y_1 = y$, $y_2 = \dot{y}$, $y_3 = \ddot{y}$, the ODE becomes:
>
> $$\dot{y}_1 = y_2$$
> $$\dot{y}_2 = y_3$$
> $$\dot{y}_3 = -\vartheta_1 y_3 - \vartheta_2 y_2 - \vartheta_3 y_1 + u\left(t - \vartheta_4\right)$$

4.3 Neural model and model inversion

4.3.1 Physiological evidence

[22] recorded from C-fibres in the superficial branch of the common peroneal nerve proximal to the ankle, while participants heard loud sounds, or responsed to oddball sounds. They found that the average SN response was well described by a Gaussian bump with 0.25 - 0.30 s standard deviation. PsPM models SN bursts as a Gaussian bump with 0.30 s standard deviation.

4.3.2 General Linear Model (GLM) for evoked SA

This model was developed for experiments in which short experimental events with evoke SA with amplitude a. For each experimental condition i with onset vector t_{o_i}, we assume $u_i\left(t\right) = a_i \delta\left(t_{o_i}\right)$. See 3.4.4 for details on the model inversion.

4.3.3 Non-linear model (DCM) for event-related SA

This model [29] assumes that event-related SA with amplitude a causes a Gaussian bump-like SN firing burst after some delay and with a specified dispersion. For each experimental event:

$$u_{exp}\left(t\right) = a \exp \frac{-\left(t - \mu\right)^2}{2\sigma^2} + \omega,\ 0 < \mu < \mu_{max},\ \sigma_{min} < \sigma < \sigma_{max}.$$

For situations in which short external stimuli elicit SA, this model uses a canonical central burst latency of zero and dispersion of $\sigma_e = 0.3$ seconds ("fixed response"). For situations in which central processes translate SA into SN with unknown parameters, latency and dispersion are estimated from the data, alongside the SA amplitude, using the following constraints: $\sigma_{min} = 0.1$, $\sigma_{max} = 0.5\mu_{max}$ where μ_{max} is the maximally allowed response latency ("flexible response").

The full model also accounts for spontaneous fluctuations (next subsection) and baseline changes between trials and can be written as:

$$y\left(t\right) = y_{exp}\left(t\right) + y_{SF}\left(t\right) + y_{SCL}\left(t\right) + \epsilon$$

The SCR components are defined by eq. (4.2) with respective parameters and a respective input function $u\left(t\right)$. Specifically, the experimental SCR component models evoked and event-related responses, and the SF component models spontaneous fluctuations between trials (part of the latent error). The SCL term is an explicitly modelled error term that absorbs baseline fluctuations and is defined as

$$u_{SCL}\left(t\right) = a\exp\frac{-\left(t-\mu\right)^2}{2\sigma^2},\ \sigma = \sigma_0$$

where dispersion is set to $\sigma_0 = 1.0$ seconds, and μ is estimated from the data. By default, PsPM automatically mdoels SN bursts that occur more than 5 s after the previous trial, and more than 2 s before the next trial. These limits ensure that the modelled SCL and SF do not absorb variance caused by the experiment. The use of different limits has not empirically been tested.

In this model, the parameter space is too large to invert it at once. Inversion is done in a trial-by-trial approach: a number of trials, $T_{1..n}$ is inverted. The ensuing parameter estimates for T_1 are extracted, and the parameter estimates for $T_{2..n}$ are used as prior values for the next inversion for trials $T_{2..n+1}$ where the estimated hidden state values at the start of trial 2 are taken as starting values of the hidden states x. This is continued until all trials have been estimated. This scheme ensures that inversion is kept tractable, and at the same time overlapping responses are taken into account. It also means that estimation errors accumulate over trials which is why part of the latent error is explicitly modelled as SF and SCL. At an inter trial interval of around 10 seconds, inverting 2 or 3 trials at the same time yields comparable predictive validity [11].

4.3.4 Non-linear model for tonic SA (spontaneous fluctuations, SF)

Spontaneous SN bursts cause spontaneous fluctuations (SF) in skin conductance. The number of spontaneous SN bursts is thought to depend on SA. Each spontaneous SN burst is modelled as a Gaussian bump with unknown timing and amplitude and fixed dispersion. For each spontaneous SF burst:

$$u_{SF} = a\exp\frac{-\left(t-\mu\right)^2}{2\sigma^2} + \omega,\ \sigma = \sigma_0,\ a > 0$$

The dispersion is set to $\sigma_0 = 0.3$ seconds. This model is inverted using a Variational Bayes algorithm [30, 16]. Because the number of SN bursts

is difficult to estimate directly, the SF model inversion uses a fixed rate of SN bursts (set to $f = 0.5$ Hertz by default) the amplitude of which is estimated from the data. An amplitude threshold is defined to separate "true" SN bursts from noise. An amplitude threshold of 0.1 µS rendered the predictive highest validity in a validation paper [31]. The results from this paper have been replicated in an unpublished re-analysis using AIC as objective function in a manner similar to [9]. Re-analysing the second data set described in [31], the optimal threshold was between 0.15 - 0.20 µS. Further systematic research is needed to fine tune the optimal threshold.

Furthermore, PsPM uses a Matching Pursuit (MP) algorithm as fast (< 1 s per minute of data) approximation to the DCM inversion (ca. 1 min. per minute of data). Using simulated data, this method was less precise, but empirically it was comparable to DCM results in three different data sets [32].

4.4 Data conditioning

4.4.1 Filtering

Skin conductance signals return to a baseline after an SCR, but this baseline can change over time within a large dynamic range. Such fluctuations of the skin conductance level need to be filtered out to make the data accessible to an LTI model. A unidirectional 1st order Butterworth high pass filter with cut off frequency 0.05 Hz was optimal for GLM [9]. The design matrix of the GLM is filtered with the same frequency to account for distortions of response shape. For the non-linear model (DCM) for event-related responses, an optimal filter frequency was found at 0.0159 Hz when using a canonical SCRF and inversion of 2 trials at the same time [11]. Filter characteristics of SF models have not been investigated. In order to reduce computation load, data are resampled to 10 Hz sampling rate for all data analyses which requires a low-pass filter cut off frequency of 5 Hz.

Historical remark: The SCRF was developed using a high pass filter with cut off frequency of 0.0159 Hz as commonly employed in conventional analysis (bidirectional for phasic responses, unidirectional for spontaneous fluctuations). However, this turned out to not be the optimal filter for GLM in follow-up analyses of optimal data conditioning. Note that the canonical SCRF was developed from data filtered with the original settings, and this has not changed since.

4.4.2 Normalisation

For within-subjects analysis, we propose data normalisation after filtering to remove SCR scaling differences due to peripheral factors such as skin property. This increases predictive validity (unpublished analyses). For methods

that yield trial-by-trial estimates, post-hoc normalisation of amplitude estimates across trials, before averaging within conditions, yields even better predictive validity [11]. For between-subjects analysis, this is not an option as it removes between-subjects variance.

4.5 Implicit estimates of tonic sympathetic arousal

Our model for SF implies that

$$\int_I [y(t) - y_0] \, dt \propto \sum_{i=1}^{n} a_i$$

where I is a time interval, and $a_{1..n}$ are the amplitudes of the n sudomotor bursts in this interval and y_0 is the skin conductance level at rest. That is, by simply taking the time integral, or area under the curve (AUC), of the baseline-corrected data, we get a measure of the number x amplitude of spontaneous SN bursts [28]. This measure is implemented in the software as well. However, we have subsequently shown that the number of spontaneous SN bursts is a better predictor of tonic SA than AUC, and hence we recommend to use the non-linear model for tonic SA.

4.6 Comparison with other model-based methods for SCR

Several further approaches to model-based analysis of SCR have been introduced in the literature [33, 5, 7, 8, 34] - for one of them, the implementation is publicly available in the software package Ledalab. In this method, SN is calculated from SCR using a deterministic inverse filter. Peaks in the SN time series are then identified and used as operational index of SA. In a head-to-head comparison, predictive validity to identify states with known SA was compared for experiments involving short stimuli and evoked SA. PsPM had better predictive validity than Ledalab in 4 out of 5 tested contrasts, and equal validity in the fifth. Ledalab did not consistently perform superior to classical peak scoring analysis [10].

4.7 Recommendations

- Experimental design:

    - SOA > 2 s between events eliciting SCR. SCR amplitudes appear to be linear with an SOA of 3.5 s between subsequent stimuli [24] and non-linear with SOA < 1 s [25].

- Model set up:

    - Use GLM wherever possible. GLM is more sensitive than non-linear models in evoked SCR paradigms [9].

- **–** Use default filter frequencies (0.05 - 5 Hz unidirectional filter for GLM [9]; 0.0159 - 5 Hz bidirectional filter for non-linear model [11]).

- **–** GLM: include all experimental events that evoke SCR (including block starts, break messages, etc.).

- **–** GLM: canonical SCRF with time derivative and reconstruction of amplitudes is more sensitive than just canonical SCRF, or SCRF plus time and dispersion derivative [9]. The benefit of subject-specific SCRFs has not been tested.

- **–** non-linear model: canonical SCRF is more sensitive than using subject-specific SCRFs based on short (~ 10 s) ITIs [11]. The use of individual SCRFs based on long ITI paradigms has not been tested.

- **–** non-linear model: for fear conditioning paradigms, the best way of modelling anticipatory SCR is currently under investigation. It is possibly suboptimal to model one anticipatory "flexible" response, in particular at longer CS/US SOAs when this flexible response may absorb SCR elicited by US or US omission.

- **–** GLM and non-linear model: z-normalisation of data is recommended for within-subjects designs, and should not be used for between-subjects designs.

- Model inversion:

  - **–** SF model: the VB inversion is theoretically better suited than the MP inversion when the expected number of SF is > 10/minute.

- Post-processing

  - **–** non-linear model: trial-by-trial normalisation of amplitude estimates before averaging within conditions can increase sensitivity in within-subjects designs [11] but should not be used in between-subjects designs.

5   Heart Period (HP) Models

contributed by Philipp C. Paulus and Giuseppe Castegnetti, and edited by Dominik R. Bach

5.1   General

Cardiac rhythm is generated locally in the sinoatrial node, but is modulated by central neural input. Therefore, heart rhythm can be used to infer psychological states. To quantify phasic changes in cardiac chronotropy, two

measures are common: heart rate (beats per minute) or heart period (milliseconds). Heart period appears to relate to autonomic input linearly, as revealed in experiments in which autonomic nerves in rodents were electrically stimulated with varying frequencies to elicit heart period changes [35]. This is why PsPM models heart period. Heart period information is only available at discrete time points. To facilitate analysis within the existing algorithms, PsPM assigns each heart period to its following heartbeat and linearly interpolates this time series at 10 Hz resolution.

For the analysis of event-related, i.e. phasic cardiac responses, PsPM includes two models: A model for event-related HPR [36] and a model for fear-conditioned bradicardia [37]. As in previous models for Skin Conductance Responses (SCR), both models are optimised with regard to *predictive validity*, i.e. the ability to separate experimental conditions (e.g., CS+ vs. CS-).

Operational analysis has shown that event-related HPR pattern strongly depends upon properties of the input - i.e. modality, intensity and presentation duration of the stimulus material. It follows that both models can only be applied in experiments with similar conditions, including stimulus presentation times (i.e. ≤ 1 s for event-related HPR). Pharmacological studies have suggested a differential time course of sympathetic and parasympathetic nervous input, but it is not yet known how this maps onto the response components modelled here.

5.2   Peripheral and neural model

In the absence of information on the timecourse of cardiac information, it is difficult to separate a neural and a peripheral model. Hence, these two are collapsed for the purpose of quick and simple inversion with GLM (see 3.4.4). This means that the two implemented models - for evoked HRP and fear-conditioned HRP - use a different RF, although of course the peripheral system is the same in both cases. Castegnetti et al. [37] relate the two models to each other, and derived the neural input producing fear-conditioned bradycardia (see figure 1 of [37]).

5.3   Model for event-related HPR

The model for event-related HPR [36] was developed on data from 84 participants that underwent three different experiments: An experiment using loud white noise sounds ($\sim 85dB$), an experiment using an auditory oddball task, and an experiment using emotional pictures from the International Affective Picture System (IAPS; [38]). All stimuli were presented with long ITIs (> 30 s) to allow the cardiac system to return to baseline after stimulus presentation and avoid overlapping responses. The first three principal

response components were extracted, and individual peaks fitted with Gaussian Functions as RF:

$$h\left(y\right) = \frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(t-\mu)^2}{2\sigma^2}} \tag{5.1}$$

To test whether a modeled peak qualified as a RF, we tested whether it explained interesting variance in the data. We started with a single RF and included further RFs. Specifically, we created a first-level GLM for each participant by convolving the event onsets with the specific subset of RFs, and extracted estimates of the response amplitudes for each RF and condition from the continuous heart period data. In order to qualify as RF, parameter estimates for this RF were either required to depict a stable response (i.e., acceleration or deceleration) across all experiments, tested by a one-sample t-test, or to add to the *predictive validity* of the model. This second criterion was evaluated by subjecting the parameter estimates to a between-subject analysis of variance (ANOVA) testing for a main effect of experimental condition.

Hence, the following steps were taken:

Step 1. Test all potential RFs modeled from PCs individually and retain each single RF if it depicts a stable acceleration or deceleration across all experiments or if it allows separation of at least two of the three experiments. All potential RFs fulfilled one of these criteria.

Step 2. Take the chronologically first RF from Step 1 and combine it with the chronologically second RF from Step 1, or with any other RF that overlaps with this second RF. Orthogonalize each set in temporal order, using a Gram-Schmidt algorithm. Retain the best set if additional RFs allow the separation of the three experiments.

Step 3-5. Take the current set, and add the chronologically next untested RF from Step 1, or any other RF that overlaps in time with this second RF. Repeat the strategy of Step 2, and retain the best set if additional RFs allow the separation of the three experiments.

This testing procedure yielded a basis set of six RFs. Model constants are depicted in Table 1.

In an independent model-validation experiment with short ISI (10 s) that used white-noise sounds of varying intensities (∼85 vs. ∼65 dB) and both negative and positive pictures from the International Affective Picture System (IAPS; [38]), the following RFs allowed separation of the different experimental conditions:

RF1 allowed the discrimination of 85 dB white-noise sounds and IAPS pictures as well as Positive and Negative IAPS pictures. RF2 allowed the discrimination of 85 dB and 65 dB white-noise sounds, as well as Positive and Negative IAPS pictures. RF3 allowed the discrimination of Negative and Positive IAPS pictures. RF4 did not allow discrimination of any experimental conditions in the validation experiment, but showed a trend to significance for the discrimination of 85 dB white-noise sounds and IAPS pictures

Table 1: Model constants for evoked HPR

Response Function (RF)	Parameters of Gaussian Function	
	μ	σ
1	1.0	1.9
2	5.2	1.9
3	7.2	1.5
4	7.2	4.0
5	12.6	2.0
6	18.85	1.8

Note. μ =mean; σ = standard deviation

($p = .051$) and a trend for discrimination of Positive and Negative IAPS pictures ($p = .068$).

Because the data are interpolated, apparent responses to stimuli can start before stimulus presentation. This is evident in these RFs some of which overlap time point 0 and therefore start before the actual stimulus. In PsPM this phenomenon is automatically accounted for.

5.4   Model for fear-conditioned HPR

For the analysis of fear conditioned HPR, we sought to build a data-driven RF for discriminating between HPR to CS+ and CS- [37]. This model-based analysis was developed on the basis of four independent data sets, acquired during diverse implementations of a fear conditioning protocol. The shape of the response was determined by visually identifying a gamma distribution as the function that qualitatively best resembled the difference between grand means. This function

$$y = \frac{A}{\Theta^k \Gamma(k)}(x - x_0)^{k-1} e^{-\frac{x - x_0}{\Theta}}$$

was fitted it by finding the values of the shape parameter k, the scale parameter Θ, the time onset x_0 and the amplitude A that minimised the residual sum of squares (RSS). The resulting parameters were $k = 1373$, $\Theta = 0.0311$, $x_0 = -38$, with A left free to vary during the model inversion. Shifts in the response are modelled by its first time derivative. The ensuing model-based analysis reliably distinguishes HPR to CS+ and CS-, and does so significantly better than model-free analogues (e.g., peak scoring, area under the curve).

5.4.1 Adaptation to different SOAs

By considering fear conditioning sessions with different SOAs between CS and US, we investigated how this changes the anticipatory response. Our analysis suggested that for different SOAs the anticipatory HPR is time-locked to the US, with no changes in shape. This can be modelled by time-locking the RF (developed for 3.5 s) to the US. In PsPM, the default SOA is 3.5 s, and it can be modified to assume any positive value. However, we recommend not using SOAs shorter than 2 s. This procedure has been successfully tested with SOAs of 4 [37] and 6 s [39].

5.5 Data Conditioning

5.5.1 QRS detection

PsPM uses a modified offline version of the Pan & Tompkins [40] QRS detection algorithm. The algorithm analyses slope, amplitude, and width of QRS complexes. It uses digital band-pass filters to reduce noise and its thresholds are periodically adjusted to low values. The algorithm works best on data with a time-resolution of 200 Hz. Data with higher sampling rate will automatically be downsampled to the optimal time-resolution. The QRS complex is marked at the rising edge of the integrated waveform signal that usually corresponds well with the position of the R-spike of the ECG.
For the offline version of the Pan & Tompkins QRS detection algorithm some adjustments were made:

- The originally used cascaded integer filters were replaced by a second order Butterworth filter that approximates the desired passband from 5 to 15 Hz better than the original filters.

- To increase detection accuracy heart rate values were restricted to the interval of 20 bpm (IBI of 3000 ms) to 300 bpm (IBI of 300 ms).

- To allow for a manual inspection and possible correction of detection failures QRS detection procedure also containes a tool to visually inspect the results of QRS detection (see 13.2). In order to manually check only those QRS complexes that are likely to be wrong, only those IBIs are checked that fall outside a user-defined range. To avoid the introduction of bias, the rater should blind to the experimental conditions during manual correction of the QRS detection.

Depending on the quality of the recordings QRS detection can achieve a very high accuracy in detecting QRS complexes of up to 99.66% [36].

PsPM also offers a QRS detection algorithm for ECG data acquired in MRI environments. The algorithm is described in [41]. The implementation is obtained from the software link given in the paper.

5.5.2 Interpolation and Filtering

To transform the discontinuous heart-beat information into a continous signal of heart-period, the heart-beat events are interpolated at a fixed sampling rate of 10 Hz using the built-in function `pspm_hb2hp`. The interpolation shifts some early evoked responses such that they appear to start before an event - this is automatically being taken care of by a response function that starts slightly before an event definition.
Before obtaining parameter estimates for the individual response functions the interpolated heart beat time-series is filtered with a second-order Butterworth band-pass filter. For evoked HPR, PsPM uses a pass band of 0.01-2 Hz [36]. For fear-conditioned HPR, PsPM uses a 0.015-0.5 Hz bidirectional 4th order Butterworth filter (cf. [37]).

5.6 Recommendations

- Experimental design:

    - evoked HPR model: use only those RFs that peak within the SOA between subsequent events that elicit an HPR. A formal test of the linearity assumption has not been done yet. Therefore, the number of RFs that can be applied depends upon the SOA.

    - fear-conditioned HPR: use default model for CS/US-SOAs between approximately 2-8 s. The model is tested for 3.5 - 6 s.

- Model set up:

    - Use default filter frequencies (0.01 - 2 Hz unidirectional 2nd order Butterworth filter for evoked HPR [36]; 0.015 - 0.5 Hz bidirectional 6th order Butterworth filter for fear-conditioned HPR [37], although there is apparently no strong impact of filter frequency.

6 Pupil size response (PSR) models

contributed by Christoph W. Korn

6.1 General

Pupil size is under control of two antagonistic muscles [42]: Dilation (mydriasis) results from sympathetic inputs on the M. dilatator pupillae. Constriction (miosis) results from parasympathetic inputs on the M. sphincter pupillae. Pupil size reacts to changes in illuminance (light falling onto the eye, directly related to the luminance of screen presentations). It also responds to many cognitive processes including attention, perception, memory, and emotion (see [43] for references). Such cognitive processes are

supposed to be relayed via the Edinger-Westphal nucleus and noradrenergic inputs into the locus coeruleus, among others [42]. On the basis of two experiments that elicited (il)luminance changes, we have developed a model for steady state pupil size and, more importantly, an model for changes of pupil size that is based on two LTI systems. These LTI systems can be harnessed to infer the neural input into the pupillary system that is elicited by cognitive processes (under the assumption of a common final pathway of (il)luminance-mediated inputs and cognitive inputs). We have also developed and validated a specific LTI system that allows distinguishing responses to CS+US- and CS- in fear-conditioning setups employing different sensory modalities and timings [44]. In this latter model, neural and peripheral system are collapsed for the purpose of quick and simple inversion with GLM (see 3.4.4).

6.2   Model for pupil size changes elicited by (il)luminance changes

The system controlling pupil size is non-linear. To be able and use linear methods for analysis, PsPM splits the model into several parts.

6.2.1   Model for steady-state pupil size

An exponential function is used to relate illuminance (in$[lx]$) or luminance (in$\frac{cd}{m^2}$) to steady-state pupil size (in arbitrary units as determined by the Eyelink system or in mm). These functions were specified on the basis of the first experiment described in [43] (see 13.4). The functional relationship should cover the (il)luminance range employed in most cognitive experiments.

$$d(E_v) = C + A \exp(B E_v).$$

Here, d is the z-scored steady-state pupil diameter and E_v is the respective illuminance level in (in $[lx]$). We obtained the following parameter values: $A = 49.79$; $B = -0.50[\frac{1}{lx}]$; $C = -1.05$. Whether the non-linearity occurs in the peripheral or neural system is currently not known.An extension that is similar to our model within the (il)luminance range specified in our experiments, but extends across a wider range, can be found in [45].

6.2.2   Model for pupil dilation/constriction

Dilations and constrictions differ in shape, and can thus not be modeled by a single LTI system (which would assume that positive and negative changes do not differ in shape but just in sign). Hence, we identified a combination of two LTI systems that provide a good approximation of pupil size changes elicited by (il)luminance changes (see first two experiments in [43]). The

first LTI-system describes changes to continuous (il)luminace inputs and the second LTI-system models the difference between constriction and dilation. The second system thus receives a brief input for any increase in (il)luminance. Both systems are specified by canonical response functions that take the form of gamma probability density functions.

$$d(t) = c\frac{(t - t0)^{k-1}\exp(-(t + t0)/\theta)}{\theta^k\Gamma(k)}$$

where d is the z-scored steady-state pupil diameter, t is time, Γ is the gamma function, and k, θ, c and $t0$ are free parameters. The fitted parameter values were: System 1: $k = 2.40$, $\theta = 0.29s^{-1}$, $c = 0.77$; System 2: $k = 3.24$, $\theta = 0.18s^{-1}$, $c = 0.43$. Since the latency of the empirical responses was around 200 ms, $t0$ was set to 200 ms prior to fitting. (We also modelled the first system with a Gaussian smoothed biexponential function; $d(t) = aN(t)*(E_1(t)+E_2(t))$; where $*$ is the convolution operator; $N(t)$ is a centered Gaussian function $N(t) = (2\pi\sigma)^{-0.5}\exp(-t^2/2\sigma^2)$ and $E_1(t)$ and $E_2(t)$ are exponential functions of the form $E(t) = \exp(-\lambda t)$; parameter values: $\sigma = 0.27$, $\lambda_1 = 2.04s^{-1}$, $\lambda_2 = 1.48s^{-1}$, $a = 0.004$).

In GLM-based analyses, these systems explain considerable variance in pupil size time series in two experiments with (il)luminance inputs of long and short duration (with the second LTI system being particularly relevant for brief, i.e., sub-second, inputs). The modeled pupil size time series can be easily specified given a time series of (il)luminance values (see 13.4). These time series can be used as nuisance regressors in GLMs. Because the end stage of the pupil system does not depend on where the neural input comes from, the specified canonical response functions can also be used to estimate the shape (i.e., the temporal characteristics) of likely inputs into the pupillary systems for cognitive tasks. We provide examples for this for auditory oddball tasks with three different timings, for an emotional words task, and for a visual detection task (see [43]).

6.3 Model for pupil size changes elicited by fear conditioning

This model collapses peripheral and neural processes into one RF, in order to use GLM for the inversion. On the basis of four experiments employing simple and complex auditory, visual, and somatosensory CS, we have determined a canonical response function that reliably distinguishes pupil responses to CS+US- and CS- [44]. Predictive validity exceeds (or is at least equal to) peak scoring and calculating the area-under-the-curve. The response function was initially based on a CS to US SOA of 3.5 s but the same function can be used for an SOA of 6 s. The overall best performing model did not include the temporal derivative but this is still accessible

in PsPM. See: 14.5. The gamma probability density function followed the same form specified above with the parameter values $k = 6.03$, $\theta = 0.73s^{-1}$, $c = 1.6$, $t0 = 0.029$ ($t0$ was fitted for this model).

6.4   Data preconditioning

We provide procedures to import pupil data from Eyelink, SMI and View-Point files, but they can also be imported from other sources via matlab or text files. All PsPM models specify pupil diameter. Eyelink import automatically converts input into diameter in absolute units (mm). SMI diameter values are returned in pixel values. ViewPoint diameter values are returned in ratio units as they are given in the data file. For other sources it may be necessary to convert the data using the respective conversion function. Missing data due to blinks, saccades, or slight head movements can be linearly interpolated before z-scoring. These missing data can then be retained, or not, for the GLM.

Pupil area appears smaller to the eyetracking camera when the participant does not maintain fixation. Whether this "Pupil foreshortening error" (PFE) influences recordings depends on the eyetracker. Geometrically, it would be possible to fit the visible pupil shape with an ellipse and retain the long axis as pupil diameter, which is invariant to eye movements. However, apparently many eyetrackers simply report the number of pupil pixels, ie. an area measure. For example, although the Eyelink 1000 eyetracker has the possibility to choose an ellipsoid model, this is according to the manual (User Manual. v.1.4.0, p. 23) only used to determine pupil position, but not pupil area or diameter ("Measure of Area or Diameter are always based on the Centroid Model").

To account for loss of fixation, PsPM includes two options: exclude these time intervals, or correct pupil size with a geometric model. (1) To detect loss of fixation there is a specific function (see 13.5). The user has to specify the distance between screen and eye, the screen size, the position of the fixation point (by default the middle of the screen is assumed but additionally a time series of fixation points can be specified), and the threshold for exclusion set in degrees of visual angle. (2) PsPM also includes a method to perform pupil foreshortening error (PFE) correction as described in [46]. The implementation itself is eyetracker or measurement method independent; however, we use Eyelink in its name because the method was proposed specifically for Eyelink 1000 eyetracker. To perform PFE correction, PsPM requires extra information about the recording setup. Specifically, screen resolution, screen size and the screen-camera-eye geometry setup must be known.

Pupil size obtained using eyetrackers may be contaminated with high noise. Before using this data in various statistical models, it might be beneficial to preprocess it in order to increase signal-to-noise ratio. PsPM of-

fers a pupil size preprocessing utility that is independent of the eyetracker used. The method is described in [47]. We use a modified version of the software published in this paper. All the modifications are documented and the changelog is part of PsPM. The preprocessing method can be used without any additional information about the recording setup. PsPM exposes all the preprocessing parameters used by the underlying software to the user for finetuning.

Finally, we found that filtering provided equal or worse proportions of explained variance and therefore there is no default filtering although this may be beneficial for other experimental designs.

7 Respiratory response models

contributed by Giuseppe Castegnetti and Dominik R Bach

7.1 General

Brain stem centres regulate respiration via autonomic nervous efferents, and these centres are influenced by higher cognitive processes [48, 49]. This may allow inferring psychological states from measured respiration, although it is not very well know which psychological variables or events elicit phasic respiratory responses. PsPM implements four models for respiratory responses, derived from a single chest belt system. In principle, this system only allows assessing respiration period, while for precise quantification of respiratory amplitude, a double-belt system would be required to measure both thoracic and abdominal compartments [50]. However, if the ratio between thoracic and abdominal contribution is relatively constant within any individual, it may still be possible to approximate respiratory amplitude up to a linear constant from the single-belt system. We confirmed in [51] that respiration amplitude from a single chest belt system can be usefully analysed. PsPM considers the following measures:

- Respiration period (RP) is the duration of a breathing cycle. PsPM models RP rather than the more common respiration rate, its inverse, in analogy to analysis of event-related cardiac responses where heart period has been shown to linearly relate to autonomic nervous system input [35]

- Respiration amplitude (RA) is the amplitude in rib cage excursion on that cycle, which linearly relates to current tidal volume (VT) [50].

- Respiration flowrate (RFR), RA/RP, which is linearly related to tidal volumetric flow rate, i.e. tidal volume per time unit. RFR is computed per cycle with variable duration. Otherwise it is similar to minute

volume, or RLL (respiration line length), which are computed over fixed intervals rather than respiratory cycles.

In these measures, we identified the following pattern of phasic responses [51]:

- Respiratory period responses (RPR) are elicited by various external events (aversive sounds, electric shocks, picture viewing, visual detection) and reliably distinguish events from non-events

- Respiratory amplitude responses (RAR) are elicited by aversive sounds and shocks, and by picture viewing. Response amplitudes are smaller for picture viewing (including aversive pictures) than aversive sounds. RAR are also elicited by fear-conditioned stimuli.

- Respiratory flow-rate responses (RFRR) are elicited by aversive sounds and shocks, and by picture viewing. Response amplitudes are smaller for picture viewing (including aversive pictures).

7.2 Peripheral and neural model

In the absence of information on the timecourse of input into breathing control, it is difficult to separate a neural and a peripheral model. Hence, these two are collapsed for the purpose of quick and simple inversion with GLM (see 3.4.4). This means that the two implemented models for RAR - evoked RAR and fear-conditioned RAR - use a different RF, although of course the peripheral system is the same in both cases.

7.3 Models for evoked respiratory responses

These models are based on responses from 46 participants and on the following task conditions: visual detection, aversive electric shocks, IAPS picture viewing. The model was validated on an independent sample of 20 participants and the following task conditions: aversive and less aversive white noise sounds, positive and negative IAPS picture viewing.

We modelled responses with Gaussian functions:

$$h\left(y\right) = \frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(t-\mu)^2}{2\sigma^2}}.$$

Model constants after filter optimisation are summarised in table 2. Later response components that were visually identified in the development data set did not contribute to detecting overall responses or separating experimental conditions. Modelling the time derivative of the RF improved the RA and RFR models, but not the RP model [51].

Table 2: Model constants for evoked respiratory responses

Response Function (RF)	Parameters of Gaussian Function	
	μ	σ
RPR	4.20	1.65
RAR	8.07	3.74
RFRR	6.00	3.23

Note. μ = mean; σ = standard deviation

7.4   Model for fear-conditioned RAR

After showing that RAR may be better suited to distinguish cognitive processes than respiration period [51], we investigated whether RAR allow assessment of fear memory in cued fear conditioning. Following the same procedure as in [37], we built the RF from data collected during six experiments, involving diverse stimuli and two types of breathing belts (bellows and cushion systems). The ensuing winning model was the combination of an early and a late response components, modelled by gamma distributions

$$y = \frac{A}{\Theta^k \Gamma(k)} (x - x_0)^{k-1} e^{-\frac{x-x_0}{\Theta}}$$

with parameters $k_e = 2.570 * 10^7$, $\Theta_e = 3.124 * 10^{-4}$, $x_{o,e} = -8.024 * 10^3$ and $k_l = 3.413$, $\Theta_l = 1.107$, $x_{0,l} = 7.583$, respectively. This method distinguishes RAR to CS+ and CS- better than model-free scoring of RAR (e.g., peak scoring). Compared to other measures of fear learning, RAR turn out to perform similar to SCR, but are less sensitive than HPR. By comparing CS/US SOAs of 3.5, 4, and 6 s, it is evident that the RAR are most likely time locked to the US. In the model, both components of the RF are shifted in time as a function of the SOA. To avoid deformation of the early RF due to violations of the linearity assumptions, however, we recommend to use this method only with experiments involving SOAs longer than 2.5 s.

7.5   Data Conditioning

7.5.1   Breathing cycle detection

Bellows system   Transducer output is filtered offline with an anti-aliasing bidirectional first-order Butterworth low-pass filter and a cut-offfrequency of 5 Hz. Data are then downsampled to 10 Hz resolution. Respiration traces are mean-centred, filtered with a bidirectional Butterworth band pass filter and cut-off frequencies of 0.01 Hz and 0.6 Hz, and median filtered over 1 s. A negative zero-crossing is taken as start of inspiration. After each detected cycle, PsPM imposes a 1 s refractory period, to account for residual signal

noise with might cause several zero-crossings on the same cycle. In the validation data set, we compared this method to visual detection as gold standard and found, for the automated method, a sensitivity of 99.3%, and a positive predictive value of 99.5%. For the time difference between visual and automated detection, we found a median delay of 0.1 s and a standard deviation of 0.1 s [51].

Cushion system For data obtained from the cushion/belt system, the above algorithm must be adapted to account for the different response of cushion/-belt system to ventilator activity. Specifically, we defined the onsets of the inspirations as the minima of the respiratory trace. Hence, the algorithm was adapted to extract zero crossings of the derivative of the time series (i.e., the extrema), from which the positive ones (i.e., the minima) were set as the start of the inspiration.

7.5.2 Interpolation and Filtering

To transform the discontinuous RP, RA and RFR information into a continous signal, they are assigned to the start of the following inspiration cycle and linearly interpolated with 10 Hz sampling frequency. PsPM also allows writing out the respiration time stamps for quality control. The interpolation shifts some early evoked responses such that they appear to start before an event - this is automatically being taken care of by a response function that starts slightly before an event definition. The best filter frequencies in [51] were: 0.01-1 Hz for RP, 0.001-1 Hz for evoked RA and RFR. For fear-conditioned RA a 0.01-2 Hz bidirectional 6th order Butterworth filter provided the best results (cf. [39]).

7.6 Recommendations

- Experimental design:

    - evoked respiratory response models: make sure the SOA between subsequent events that elicit an respiratory response is longer than the peak of the respective RF. A formal test of the linearity assumption has not been done yet.

    - fear-conditioned RAR: To avoid deformation of the early RF due to violations of the linearity assumptions, we recommend to use this method only with experiments involving SOAs longer than 2.5 s.

- Model set up:

    - Use default filter frequencies (0.01-1 Hz for RP, 0.001-1 Hz for evoked RA and RFR. 0.001-2 Hz for fear-conditioned RA.)

8   Startle eye blink response models

8.1   General

The startle response is a fast defensive reflex to an unexpected intense auditory, visual, or haptic stimulus. Functionally, it appears to protect an organism from an imminent blow to the head [52]. The human startle response encompasses a postural change and an eyeblink response which is easy to measure [53]. While the startle response itself is very stereotypical, its amplitude is susceptible to various physiological and psychological manipulations, many of which can be summarised in a decision-theoretic model in terms of optimising the cost of the startle response, and of a putative predator attack [54]. In this context, the most important phenomenon is fear-potentiated startle, a stronger startle response during a CS+ than CS- in fear conditioning [55].

8.2   Peripheral and neural model

In the absence of information on the millesecond-time course of neural response in the centres controlling startle, it is difficult to separate a neural and a peripheral model. Hence, these two are collapsed for the purpose of quick and simple inversion with GLM (see 3.4.4). Because the startle response is so stereotypical, the same RF is used for all applications, and the amplitude estimated. There appears to be some variability in the latency of the startle response between trials and participants, such that the neural model contains a (constrained) latency parameter that is estimated from the data. The startle eye blink response function (SEBRF) is based on data from 19 participants that heard startle sounds with no other manipulation. Pre-processing parameters were optimised to distinguish CS+/CS- in a fear retention task with 20 participants in which startle onsets were recorded for enhanced temporal precision. They were then validated on two independent samples of 15 and 14 participants in a fear retention, and fear acquisition task, respectively. The SEBRF is described as a gamma distribution

$$y = \frac{A}{\Theta^k \Gamma(k)} (x - x_0)^{k-1} e^{-\frac{x-x_0}{\Theta}}$$

with parameters $k = 3.5114$, $\Theta = 0.0108$, $x_o = 0.0345$ [56].

8.3   Data preconditioning

The algorithm expects raw data from orbicularis oculi EMG. Data filtering and preprocessing is taken care of by a separate PsPM function (see 13.10) and includes a 4th order Butterworth band-pass filter with cutoff frequency

of 50 Hz and 470 Hz, removal of mains noise with a 50 Hz notch filter. Filtered continuous EMG signals are rectified and smoothed using a 4th order Butterworth low-pass filter with a time constant of 3 ms corresponding to a cutoff frequency of 53.05 Hz. No more data pre-conditioning is applied during GLM inversion, which allows to visually inspect the filtered EMG data in advance to the inversion. To specify startle sound onsets with high precision, you can use the function 13.10 to detect these onsets from recorded audio signals.

8.4   Recommendations

- Experimental design:

    - record audio output from your startle equipment and use PsPM to detect sound onsets

- Preprocessing:

    - use default pre-processing

- Model setup:

    - model each trial as separate event type such that the response latency can be estimated per trial, rather than per condition

9   Scanpath speed (SPS) model

9.1   General

Various salient stimuli, e.g. threat-conditioned cues, capture overt attention in a bottom-up process [57], as shown in multiple stimulus competition paradigms using aversive cues. This yields a possibility to assess threat memory by overt attention. In cue competition paradigms, overt attention is usually measured as gaze patterns; for example, fixation duration, saccade initiation latency, and number of erroneous saccades. Here, we implement a model that captures gaze patterns during exclusive presentation of a single cue, by assessing scanpath speed, the length of scanpath per time unit, quantified as degree visual angle per second [58]. In our publication, we refer to scanpath length, which is the integral of scanpath speed over time. Since we used constant cue presentation times in our publication, average scanpath speed and scanpath length were identical up to a multiplicative constant.

9.2   Model for fear-conditioned SPS

Scanpath speed/length allow differentiating CS+ and CS- in fear-conditioning paradigms with visual cues, and without fixation instructions or fixation cross. PsPM implements two simple models that build on the GLM inversion algorithm (see 3.4.4). The first model is to compute average scanpath speed during a 2-s time window before US (shock) delivery. This was validated on three independent samples in [58]. The model is implemented with a simple boxcar response function without mean-centering and is the default option. Parameter estimates can be interpreted as average scan path speed per second, and if they are multiplied with 2 then the resulting values can be interpreted as scan path length during the 2 s preceding the US. We also implement a gamma RF that describes the shape of the scanpath speed during the CS-US interval better than the boxcar function, but did not yield higher retrodictive validity in inferring the CS-/+ difference. This RF was fitted to the data of 21 participants and validated in an independent sample; it is described by a gamma probability density function:

$$y = \frac{A}{\mu^k \Gamma(k)} (t - t_0)^{k-1} e^{-\frac{t-t_0}{\mu}}$$

with parameters $k = 10.0913, \mu = 0.4213, t0 = -1.9020 + (SOA\text{--}3)$ [58], where SOA is the duration of the interval between CS onset and US onset in seconds.

9.3   Data preconditioning

PsPM provides procedures to import scan path speed directly. More commonly, it is convenient to import gaze coordinates in pixels, distance units, or visual angle, and convert them to scanpath speed. This requires interpolation of missing values which is done automatically. Conversion from pixels to mm or visual angle in degree is also possible. No filtering or smoothing is required for both RF during model inversion with GLM.

9.4   Recommendations

- Experimental design:

  – use visual cues during fear conditioning and avoid fixation guides.

  – use a SOA longer than 2 s. Tested SOAs are 3.0 s and 3.5 s.

- Model setup:

  – use the default boxcar function and specify custom SOA.

Part II

User Guide

10 Installation

You can find and download the PsPM package at the following URL:

`http://bachlab.org/pspm`

Decompress and copy the files to any directory of your choice. Then add the PsPM files to the Matlab search path by entering the following command in the Matlab command line:

`addpath([your Matlab path])` where [your Matlab path] is a string of the file location on your hard drive, e.g.

`addpath('C:\Program Files\MATLAB\R2013a\toolbox\PsPM\')`.

Do not add subfolders of the PsPM directory to the matlab path.

Type `pspm` in the command line to start the package. Alternatively, each function can be called individually from the command line. In that case, you can use `pspm_init` to load the settings, and `pspm_quit` to remove PsPM subfunctions from the path and delete the settings.

PsPM requires Matlab 2009a or higher.

11 General points

11.1 File types

PsPM knows three different file types:

- Data files - usually the original file name is prepended with 'pspm_' during import. Various pre-processing steps prepend additonal letters to the name.

- First-level model files - all models use a similar data structure. The name can be freely specified by the user. These files also contain reconstructed responses in GLM, and contrast values.

- Second-level model files. The name can be freely specified by the user.

11.2 Functions that create new data files

- Import - prepended with 'pspm_'

- Interpolation - prepended with 'i'

- Trim - prepended with 't'

- Artefact removal - prepended with 'm'

- Split sessions - appended with '_sn<nr>'

- Downsample data - prepended with 'd'

These functions ask whether an existing file with the same name should be overwritten - unless you specify 'overwrite' in the options.

11.3   Functions that create or modify data channels in an existing file

- Preprocess heart data

- Preprocess respiration data

- Preprocess startle eyeblink EMG

- Find valid fixations

- Pupil foreshortening error correction

- Pupil preprocessing

- Convert data

- Find startle sound onsets

- Interpolate missing data

These functions allow you to select where to store a modified data channel. The following options exist:

- add: A new channel is created with the new data

- replace: If one or several channels of the output type exists, the last of these will be overwritten. For some functions, this can overwrite the original data, i. e. if a conversion function produces an output of the same type as the input (e.g. when units are converted, or artefacts removed). If no channel of the output channel type is found, a channel will be added.

12   Data preparation

12.1   Import

Related function: `pspm_import`
    Import external data files for use by PsPM. First, specify the data type. Then, other fields will come up as required for this data type. The imported data will be written to a new .mat file, prepended with 'pspm_'.

Data Type

CED Spike

Text   Text files can only contain numbers (i.e. no header lines with channel names) and one data column per channel. Make sure you use the decimal point (i.e. not decimal comma). At the moment, no import of event time stamps is possible, but continuous event marker channels are supported.

Matlab   Each input file must contain a variable called data that is either a cell array of column vectors, or a data points × channels matrix. The import of event markers is supported. Marker channels are assumed to be continuous if the input data is a matrix or if the input data is a cell and the given samplerate is larger than 1 Hz. A sample rate has to be specified for any type of data.

Biopac AcqKnowledge (up to v. 3.9)   Imports original files.

exported Biopac AcqKnowledge   Converted with manufacturer's export function.

bioread-converted Biopac AcqKnowledge (any version)   Loads mat files which have been converted using the bioread tool acq2mat. Bioread can be installed using pip (installed by python) or can be downloaded and installed manually from here `https://github.com/njvack/bioread`. It requires python and the python libraries numpy and scipy.

Labchart (any Version, Windows only)   Supports the import of any original Labchart (.adicht) file. Since it uses an external library, this import is restricted to Windows systems only and does not work on any other operating system.

Labchart exported (up to v. 7.1)   Export data to matlab format (plugin for the LabChart software, available from www.adinstruments.com)

Labchart exported (v. 7.2 and higher)

VarioPort

Biograph Infiniti exported   Export data to text format, both "Export Channel Data" and "Export Interval Data" are supported; a header is required

Mindmedia Biotrace exported

BrainVision

Windaq (Manufacturer's version)   Requires an ActiveX Plugin provided by the manufacturer and contained in the subfolder Import/wdq for your convenience. This plugin only runs under 32 bit Matlab on Windows.

Windaq (PsPM Version)   Windaq import written by the PsPM team. Is platform independent, thus has no requirements for ActiveX Plugins, Windows or 32bit Matlab. Imports the *.wdq files. Up to now the import has been tested with files of the following type: Unpacked, no Hi-Res data, no Multiplexer files. A warning will be produced if the imported data-type fits one of the yet untested cases. If this is the case try to use the import provided by the manufacturer (see above).

Observer XT compatible

NeuroScan

BioSemi

Eyelink   Eyelink output files (with extension *.edf) must first be converted to ASCII format (extension *.asc). This is done with the utility edf2asc.exe (normally included in the Eyelink software in <Path to Program Files>\SR Research\EyeLink\EDF_Access_API\). Otherwise there is a Data viewer, available at `http://www.sr-research.com/dv.html` (registration needed), which installs a utility called 'Visual EDF2ASC'. This also allows the conversion and does not require a license. The composition of channels depends on the acquisition settings. Available channels are Pupil L, Pupil R, x L, y L, x R, y R, Blink L, Blink R, Saccade L, Saccade R. The channels will be imported according to a known data structure, therefore channel ids passed to the import function or set in the Batch will be ignored. If the import function defines PsPM file channels that are not available in the data file, these channels will be filled with NaN values. Periods of blinks and saccades will be set to NaN in the gaze and pupil channels. To reduce the impact of loss of fixation on recorded pupil size, we recommend setting ELCL_PROC to ELLIPSE during acquisition.

Table 3: Coefficients Used for Eyelink Data Conversion

Coefficient	Variable in `pspm_get_eyelink.m`	Value	Unit
$dist_{ref}$	`reference_distance`	700	mm
$m_{diameter}$	`diameter_multiplicator`	0.00087743	mm
m_{area}	`area_multiplicator`	0.119	mm

tab:eyelink_coefficients

Pupil Channels and Eyetracker distance   Distance between eyetracker camera and recorded eyes in length units. If the given eyetracker distance is given as a positive numeric value, it enables the pupil data to be converted from arbitrary area or diameter units to diameter (mm) and then to the length unit of the eyetracker distance. If eyetracker distance is not given, or if it is not a positive numeric value, then the conversion to mm is disabled. In this case, pupil data will be imported in arbitrary (Eyelink) units. (Use only if you are interested in relative values)

The relation between arbitrary units (a.u.) and diameter is assumed to be linear such that for diameter data

$$d = m_{diameter} \cdot \frac{dist}{dist_{ref}} \cdot d_{a.u.}$$

and for area data [46]

$$d = m_{area} \cdot \frac{dist}{dist_{ref}} \cdot \sqrt{d_{a.u.}}$$

for data d in mm, data $d_{a.u.}$ in arbitrary units, reference camera-eye distance $dist_{ref}$ in mm, used camera-eye distance $dist$ in mm and a proportionality factor m in mm. In our equations, $m_{area}/dist_{ref}$ ratio corresponds to the scaling factor in [46]. For area-based conversion, we use the scaling factor $\alpha = 1.70 \times 10^{-4}$ as given in [46]. For a reference distance of 700 mm this translates to $m_{area} = 0.119$ mm. For diameter-based conversion, we determined reference distance and the proportionality factor in a simpler calibration with three different artificial pupil sizes setting using linear regression. We note that for a different Eyelink eye-tracker setup, conversion with these values yielded results with slight deviance from the expected value, such that they are either not entirely precise, or there may be small differences between different eyetrackers. The table below summarises all coefficients:

For custom setups reference distance (field `reference_distance`) and proportionality factor (`diameter_multiplicator` or `area_multiplicator`) can be set at the top of `pspm_get_eyelink.m` file.

Gaze Channels Gaze channels are reported exactly as given in the datafile without any conversion. Gaze channel units are pixels.

Distance unit The length unit in which the eyetracker distance is given. It can be mm, cm, m or inches.

Blink/Saccade Edge Discard Samples surrounding a blink or saccade period may be noisy. For this reason, Eyelink import provides a parameter that allows users to specify the amount of samples they want to discard on each side of every blink and saccade period. This value is multiplied by the sampling rate of the recording to determine the number of samples to discard from one end. Therefore, for each blink/saccade period, $2 \cdot factor \cdot F_{sampling}$ many samples are discarded in total, and effectively blink/saccade period is extended.

This value also corresponds to the duration of samples to discard on one end in seconds. For example, when it is 0.01, we discard 10 ms worth of data on each end of every blink/saccade period.

The default value has been changed to 0 in PsPM revision r803 to reduce the amount of discarded data. Note that this might result in noisy samples around blink/saccade points. Therefore, it is highly recommended to perform pupil size data preprocessing 13.7 and gaze data filtering 13.5.

SMI (SensoMotoric Instruments) SMI output files (with extension *.idf) must first be converted to ASCII format (extension *.txt) using IDF converter.

SMI iView X EyeTracker sample and optionally also event files can be imported. The SMI sample file contains pupil, gaze and position related information. The SMI event file contains information about blink/saccade events. Available channels to import are Pupil L, Pupil R, Gaze x L, Gaze x R, Gaze y L, Gaze y R, Blink L, Blink R, Saccade L, Saccade R, Marker, Custom. All channels except custom are imported according to a known data structure. Therefore, there is no need to specify the column indices of channels. If specified channels are not available, they will be filled with NaN values. If an event file is given so that blinks/saccades are available, blink and saccade periods will be set to NaN.

There are multiple ways to specify pupil information in SMI sample files. Instead of importing all the channels for a given eye, PsPM selects only one of the available ones according to the following precedence order:

1. Mapped Diameter (mm)

2. Dia X (mm)

3. Dia (squared mm)

4. Dia X (px)

5. Dia (squared px)

If the chosen pupil channel is in squared px or squared mm, then it is converted to diameter by assuming the shape is a circle. However, no conversion from pixels to milimeters is performed for pupil channels. Therefore, if a pixel or squared pixel channel is chosen, it is returned in pixel units.

Returned gaze coordinates assume that (0, 0) point of the calibration area is the top left corner. x coordinates increase towards right and y coordinates increase downwards. If the resolution of the whole stimulus window is given, the gaze values are returned in the desired target unit. Otherwise, they are returned in pixels. It is important to note that the values can be negative or larger than the sides of the screen, if gaze is outside the calibration area.

Gaze values are reported in pixels by SMI. The conversion from pixels to milimeters is performed by using the size and the resolution of the stimulus window to calculate px/mm ratio, and then using this ratio for scaling.

Viewpoint (Arrington Research)   Arrington Research ViewPoint EyeTracker files can be imported into PsPM. Available channels to import are Pupil L, Pupil R, Gaze x L, Gaze x R, Gaze y L, Gaze y R, Blink L, Blink R, Saccade L, Saccade R, Marker, Custom. Currently, blink and saccade events can only be imported if they are stored as asynchronous messages in the datafile. This is enabled by "Include Events in Datafile" option in ViewPoint EyeTracker software. All channels except custom are imported according to a known data structure; therefore, there is no need to specify the column indices of channels. If specified channels are not available, they will be filled with NaN values.

ViewPoint reports data for eye A in monocular mode and for eyes A and B in binocular mode. PsPM performs the following mapping from A/B to L/R:

1. If only one eye is specified by user (only L or only R) and there is only eye A in data, then data for eye A is returned as the eye specified by user

2. In all other cases, eye A is mapped to eye R and eye B is mapped to eye L.

All gaze and diameter values are reported as ratios by ViewPoint:

1. Gaze value for a given axis is the ratio of the gaze coordinate to the length of that axis.

2. Pupil diameter is the ratio of pupil width to the EyeCamera window width.

Since ViewPoint data files contain the size of the stimulus window in milimeters, PsPM converts normalized gaze values to values in milimeter. This is performed by multiplying the ratio with the length of the corresponding stimulus window axis. Returned gaze coordinates assume that (0, 0) point of the stimulus window is the top left corner. x coordinates increase towards right and y coordinates increase downwards. The gaze values are returned in the desired target unit. It is important to note that the values can be negative or larger than side of the screen which correspond to looking outside the stimulus window.

Pupil diameter is returned as ratio without any conversion.

European Data Format (EDF)

Philips Scanphyslog   This functions imports psychophysiology files from Philips MRI scanners. The physlog ascii file contains 6 channels with physiological measurements:

Channel id	Event type
1	ECG1
2	ECG2
3	ECG3
4	ECG4
5	Pulse oxymeter
6	Respiration

Depending on your scanner settings, there are 10 trigger channels (see complete list below) of which channel 6 marks time t of the last slice recording. After importing, a time window from t minus $(\#volumes)*(repetitiontime)$ seconds until t should be used for trimming or splitting of sessions to constrain data in the imported file to the EPI recording window and easier matching with experimental events from a separate source.

Available trigger channels are:

Channel id	Event type
1	Trigger ECG
2	Trigger PPG
3	Trigger Respiration
4	Measurement ('slice onset')
5	Start of scan sequence
6	End of scan sequence
7	Trigger external
8	Calibration
9	Manual start
10	Reference ECG Trigger

Data File(s)

Specify which data file(s) to import. You can import multiple files that all have the same structure.

New Channel

Define each channel that you want to import.

Channel Type   Specify the type of data in this channel. Currently supported data types: SCR, ECG, heart beat markers, interpolated heart rate, interpolated heart period, respiration traces, interpolated respiration period, pupil size, EMG, marker channels.

Channel /Column Number   Specify where in the original file to find the data - depending on your data type this will be a channel or column number. You can either specify a number (i. e. the n-th channel in the file), or (for many data types) search for this channel by its name. Note: the channel number refers to the n-th recorded channel, not to its number during acquisition: if you did not save all recorded channels, these might be different for some data types.

- Search [if available]: Search for channel by its name – this only works if the channel names are unambiguous, and not for all data types.

- Specify Channel Number [if required according to data type, otherwise a default channel number will be assigned]: Specify the n-th channel. This counts the number of channels actually recorded.

Sample Rate [if required]   Sample rate in Hz (i. e. samples per second). For most data types, this is read from the original data file.

SCR Transfer Function [for SCR data only] Enter the conversion from recorded data to Microsiemens.

The transfer function supports two recording systems, which are either a conductance based system (default) or a resistance based system. Depending on the recording system the relation between measured conductance G in μS to recorded voltage U in V is either

$$G = \frac{1}{\frac{c}{U - offset} - R_s * 10^{-6}},$$
(12.1)

or

$$G = \frac{1}{\frac{U - offset}{c} - R_s * 10^{-6}}.$$
(12.2)

For a transfer constant c in $V/\mu S$ (for 12.1) or in $V/M\Omega$ (for 12.2), series resistor R_s in Ohm, and a fixed offset in V which was added to the initially measured voltage. If there is no offset and series resistor, this is equivalent to

$$G = \frac{U}{c},\ U = cG$$

or

$$G = \frac{c}{U},\ U = cR = \frac{c}{G},$$

There are three options how to enter this information:

- File: Enter the name of the .mat file that contains the variables: 'c', 'Rs' (in Ohm) and 'offset', 'recsys' (possible values: 'conductance' or 'resistance').

- Input: Enter the transfer constants manually.

- None: No transfer function. Use this only if you are not interested in absolute values, and if the recording settings were the same for all subjects.

12.2 Trim

Related function: `pspm_trim`

Trim away unnecessary data, for example before an experiment started, or after it ended. Trim points can be defined in seconds with respect to start of the data file, in seconds with respect to first and last marker (if markers exist), or in seconds with respect to a user-defined marker. The resulting data will be written to a new file, prepended with 't'.

Data file

Choose the data file you want to trim.

Reference

Choose your reference for trimming: file start, first/last marker, or a user-defined marker. All trimming is defined in seconds after this reference – choose negative values if you want to trim before the reference.

File   Trim from xx seconds after file start to xx seconds after file start.

- From (seconds after file start): Choose a positive value.

- Trim To (seconds after file start): Choose a positive value larger than the 'from' value.

First/Last Marker   Trim from xx seconds after first marker to xx seconds after last marker.

- From: (seconds after first marker): Choose a positive (after first marker) or negative (before first marker) value.

- To: (seconds after last marker): Choose a positive (after last marker) or negative (before last marker) value.

- Marker Channel: If you have more than one marker channel, choose the reference marker channel (default: use the first marker channel in the file)
  - Default (first marker channel=0)
  - Number

Any Marker   Trim from xx seconds after any marker of your choice to xx seconds after any marker of your choice.

- From: Choose marker number and trimming point in seconds after this marker.
  - Marker Number x: Choose an integer value.
  - Seconds after Marker x: Choose a positive (after this marker) or negative (before this marker) value.

- To: Choose marker number and trimming point in seconds after this marker.
  - Marker Number y: Choose an integer value.
  - Seconds after Marker y: Choose a positive (after this marker) or negative (before this marker) value.

- Marker Channel: If you have more than one marker channel, choose the reference marker channel (default: use the fist marker channel in the file)

  - Default (first marker channel=0)

  - Number

Marker according to values or names   Trim from nn seconds after first marker with value or name x, to mm seconds after first marker with value or name y.

- From: Choose first marker with value or name x, and trimming point in seconds after this marker.

  - First marker with value or name x: Either choose a numeric marker value or a marker name.

  - Seconds after marker with value/name x: Choose a positive (after this marker) or negative (before this marker) value.

- To: Choose first marker with value or name y, and trimming point in seconds after this marker.

  - First Marker with value or name y: Either choose a numeric marker value or a marker name.

  - Seconds after marker with value/name y: Choose a positive (after this marker) or negative (before this marker) value.

- Marker Channel: If you have more than one marker channel, choose the reference marker channel (default: use the fist marker channel in the file)

  - Default (first marker channel=0)

  - Number

Overwrite Existing File

Choose "yes" if you want to overwrite existing file(s) with the same name. Default: no.

13   Data preprocessing

13.1   Preprocess heart data

Related functions: `pspm_ecg2hb, pspm_ecg2hb_amri, pspm_hb2hp, pspm_ppg2hb`
    Convert ECG to heart beat (Pan & Tompkins) detects QRS complexes in ECG data and write timestamps of detected R spikes into a new heart beat

channel. This function uses an algorithm adapted from [40] (see 5.5.1). Hidden options for minimum and maximum heart rate become visible when the job is saved as a script and should only be used if the algorithm fails.

Convert ECG to heart beat (AMRI) similary detects QRS complexes in ECG data nad write timestamps of detected R peaks into a new heart beat channel. The algorithm used is described in [41]. The algorithm is obtained using the software link described in this paper.

Convert Heart Beat to heart period interpolates heart beat time stamps into continuous heart period data and writes to a new channel. This function uses heart period rather than heart rate because heart period varies linearly with ANS input into the heart.

Convert Peripheral pulse oximetry to heart beat first creates a template from non-ambiguous heart beats. The signal is then cross correlated with the template and maxima are identified as heart beats.

Convert ECG to heart period allows to directly convert continuous ECG data into continuous heart period data. This function is a combination of the two functions "Convert ECG to heart beat" and "Convert Heart Beat to heart period".

If, for ECG conversions, the semi automatic mode is enabled, the ECG editor will be shown. This allows manual inspection and correction of potentially wrong HB events. If the semi automatic mode is disabled, the conversion is done automatically, without showing the ECG editor. See ECG editor (13.2) for further information.

Data File

Specify data file.

Preprocessing

Add different preprocessing steps here. The converted data will be written into a new channel in the same file.

Convert ECG to Heart Beat (Pan & Tompkins)

Convert ECG data into Heart beat time stamps using modified Pan & Tompkins algorithm.

Channel   Number of ECG channel (default: first ECG channel).

Options

- Semi automatic mode: Allows manual correction of all potential beat intervals (default: off).

Convert ECG to Heart Beat (AMRI)

Convert ECG data into heart beat timestamps using AMRI algorithm [41]. The algorithm performs template matching to classify candidate R-peaks after filtering the data and applying Teager Energy Operator (TEO).

Channel   Number of the channel to preprocess (default: first ECG channel found in the datafile)

Options

- Signal to use: Which signal to feed to the core heartbeat detection procedure. 'ecg' corresponds to filtered ECG signal. 'teo' corresponds to the signal obtained by filtering the ECG signal even more and applying the Teager Enery Operator. 'auto' option picks the one of these two options that results in higher autocorrelation.

- Feasible heartrate range: Define the minimum and maximum possible heartrates for your data.

- ECG bandpass filter: Define the cutoff frequencies (Hz) for bandpass filter applied to raw ECG signal.

- TEO bandpass filter: Define the cutoff frequencies (Hz) for bandpass filter applied to filtered ECG signal before applying TEO.

- TEO order: Define the order k of TEO. Note that for signal $x(t)$, $TEO[x(t); k] = x(t)x(t) - x(t-k)x(t+k)$.

- Minimum cross correlation: Define the minimum cross correlation between a candidate R-peak and the found template such that the candidate is classified as an R-peak.

- Minimum relative amplitude: Define the minimum relative amplitude of a candidate R-peak such that it is classified as an R-peak.

Convert Heart Beat to Heart Period

Convert heart beat time stamps into interpolated heart period time series. You can use the output of the ECG to Heart beat conversion, or directly work on heart beat time stamps, for example obtained by a pulse oxymeter.

Channel   Number of Heart Beat channel (default: first Heart Beat channel).

- Default: First Heart Beat channel

- Number

- Processed channel: Convert a channel already preprocessed with ECG to Heart beat. Specify the preprocessed channel with a number corresponding to the position in the list of preprocessings.

Sample rate   Sample rate for the interpolated time series (default: 10 Hz).

Limit   Define unrealistic values which should be ignored and interpolated.

- Upper limit: Values bigger than this value (in seconds) will be ignored and interpolated. Default: 2

- Lower limit: Values smaller than this value (in seconds) will be ignored and interpolated. Default: 0.2

Convert ECG to Heart Period

Channel   Number of ECG channel (default: first ECG channel).

Options

- Semi automatic mode: Allows manual correction of all potential beat intervals (default: off).

Sample rate   Sample rate for the interpolated time series (default: 10 Hz).

Limit   Define unrealistic values which should be ignored and interpolated.

- Upper limit: Values bigger than this value (in seconds) will be ignored and interpolated. Default: 2

- Lower limit: Values smaller than this value (in seconds) will be ignored and interpolated. Default: 0.2

Convert Peripheral pulse oximetry to Heart Beat

Channel   Number of Peripheral pulse oximetry channel (default: first Peripheral pulse oximetry channel)

Channel action

Choose whether to add the new channels or replace a channel previously added by this method. (default: replace)

13.2   ECG editor

Related function: `pspm_ecg_editor`

The ECG editor is integrated into the QRS detection and can also be used offline, after QRS detection. It allows to graphically inspect and correct (if necessary) the detected QRS markers. To faciliate QRS inspection in data sets with known artefact epochs, the user can load epoch files which define such artefact periods. This allows the editor to either ignore or even remove QRS markers which fall into artefact periods.

Startup

The ECG editor can be started from the PsPM window, from the Matlabbatch window or from the command line.

From the command line the ECG editor can be started by typing `pspm_ecg_editor(argument)` into the Matlab command line. For this the PsPM folder has to be added to the matlab path beforehand. Arguments are either a data struct (generated usually by `pspm_ecg2hb`) or a path to a PsPM file.

Data modes

Similar to the data editor the ECG editor supports two types of operation modes, which depend on the startup arguments.

File mode   Is entered when the editor is started from the PsPM window, from the Matlabbatch or when a file path or no argument is specified at the command line startup. If the first argument is a file path, a second (ECG channel) and a third argument (Options struct) is possible. The second argument specifies the ECG channel to be plotted. The third argument is expected to be a struct with the following fields:

- hb - specifies a HB channel which contains the QRS markers to be inspected

- factor - specifies the minimum factor faulty QRS markers should deviate from the standard deviation (default is 2)

- semi - specifies whether to navigate between potentially wrong hb events only (semi mode), or between all hb events (manual mode; semi mode is enabled by default).

- limits - with fields 'lower' and 'upper' defines hard limits for the faulty detection (default is 120bpm [upper] and 40bpm [lower])

The second and third argument is optional. In file mode the 'Input / Output' panel on the bottom of the left side is visible. It allows to specifiy and change

input files and channels (ECG and HB) and allows to choose whether the edited data should be added as a new HB channel or replace the existing HB channel.

Inline mode The inline mode is usually used together with `pspm_ecg2hb`. Therefore a struct which is generated by the latter function is expected. The following fields are needed for the editor:

- settings

    - .outfact - specifies to which factor faulty QRS markers should deviate from the standard deviation

    - .sr - specifies the sample rate of the data

- data

    - .x - contains the ecg data values

    - .r - contains n x 4 vector where n corresponds to the ecg data values and columns 1-4 tell whether there is a QRS marker (column 1-4) and whether a QRS marker is faulty (column 2). Columns 3 and 4 are ignored and regenerated during data inspection

- set

    - .R - contains sample ids of all the QRS markers

Editor output

The output always contains as a first argument a status (sts). If the status equals 1 no error happend while creating the output. Otherwise the editor could not successfully create the output. The second output depends on the data mode and in file mode contains the channel id of the written channel. In inline mode the second output argument is a data vector which contains the sample ids of the resulting QRS markers.

Data inspection

Navigation The ECG editor uses QRS markers as anchor points for the navigation. One option to navigate between QRS markers is to use the listbox placed in 'Heart beat events' on the right side of the plot. Selecting a QRS marker in the listbox will cause the plot to center the selected marker with some space left before and after depending on the zoom state. If manual mode is active, the listbox contains all QRS markers available (all colors), otherwise it will only contain QRS markers marked as faulty (orange). The navigation buttons ('<<' and '>>') below the listbox allow moving back and forth in the plot. In manual mode the step size equals the size of the plot

window. If manual mode is inactive, the next/previous step equals jumping to the next/previous faulty QRS marker.

Zooming The zoom buttons are placed on the upper right. Zooming in will cause the time window to be shortened by the factor of two. Zooming out will cause the time window to be extended by the factor of two.

Add QRS marker The button for this is placed on the upper right side. Once clicked the cursor will turn into a crosshair. After that, clicking on the plot will cause an additional QRS marker to be added at that position (a green QRS marker appears).

Remove QRS marker The button is also placed on the upper right side. The cursor aswell will turn into a crosshair but different to adding QRS markers, clicking on the plot (without releasing the mouse-button) will cause a selection rectangle to be drawn. Upon release of the mouse-button all QRS markers which lie in the selection rectangle will be marked for removal (violet).

Artefact epochs

Settings for the artefact epochs can be found in the 'Artefact epochs' panel.

Load / Disable Clicking the 'Load' button causes a file dialog to appear, which allows to choose a PsPM epoch/timing file containing the definition of artefact epochs. The button 'Disable' will cause the artefact epochs to be unloaded and artefact settings to be disabled. Once an artefact epoch file is loaded, QRS marker stems and listbox font color will turn grey if they lie within an artefact epoch.

Display settings There are three different display options on how to deal with QRS markers which fall into artefact epochs.

- use all QRS markers - do nothing; all markers will be displayed and used for faulty detection

- exclude artefact periods from faulty QRS detection - still all markers will be displayed but QRS markes in artefact periods will be ignored for faulty detection

- hide QRS markers during artefact periods - QRS markers in artefact periods will be hidden and ignored for faulty detection

Output settings There are two different output options on how to deal with QRS markes which fall into artefact epochs.

- include QRS during artefact periods - all markers (faulty and valid) including those lying in artefact periods will be either returned or written to the new HB channel.

- remove QRS during artefact periods - only markes (faulty and valid) outside of artefact periods will be either returned or written to the new HB channel.

Faulty detection

Settings for faulty detection can be changed in the 'Heart beat events' panel. The following settings are possible.

- Detection factor - specifies the minimum factor faulty QRS markers should deviate from the standard deviation

- Upper limit - Heart rates higher than the upper limit will be marked as faulty

- Lower limit - Heart rates lower than the lower limit will be marked as faulty

When changed settings should be applied, clicking 'Find faulty' will cause the plot to be refreshed.

Color coding

Stem / Listbox background colors

- Valid QRS marker ⇒ blue

- Faulty QRS marker ⇒ orange

- Added QRS marker ⇒ green

- Removed QRS marker ⇒ violet

Artefact periods The color of QRS markers lying in artefact periods will be converted to greyscale. This applys for the marker stems (not the tips) and the listbox font color.

Matlabbatch

Possible options when the ECG editor is started from Matlabbatch.

Data file   Specify the PsPM datafile containing the ECG data

ECG channel   Specify the channel containing the ECG data.

- Default

- Number

HB channel   Specify the channel containing the HB data.

- None

- Number

Artefact Epochs   Specify a PsPM timing file defining artefact epochs.I

- Disabled

- File

Faulty detection settings   Settings for the faulty detection.

- Factor: The minimum factor potentially wrong QRS complexes should deviate from the standard deviation.

- Limit: Define hard limits for the faulty detection.

  – Upper limit: Values bigger than this value (in bpm) will be marked as faulty. Default: 120.

  – Lower limit: Values smaller than this value (in bpm) will be marked as faulty. Default: 40.

13.3   Preprocess respiration data

Related function: `pspm_resp_pp`

Convert continuous respiration traces into interpolated respiration period, amplitude, or RFR, or into time stamps indicating the start of inspiration. This function detects the beginning of inspiration (see 7.5.1), assigns period/amplitude/RFR of the last cycle to this data point, and interpolates data (see 7.5.2). This function outputs respiration period rather than respiration rate in analogy to heart period models - heart period linearly varies with ANS input to the heart. RFR (respiratory flow rate) is the integral of the absolute thorax excursion per respiration cycle, divided by the cycle period. Converted data are written into new channel(s).

Data File

Specify data file. The processed respiration data will be written to a new
channel in this file.

Sample Rate

Sample rate for the new channel. Default: 10 Hz. Will be ignored for
datatype "respiration time stamps".

Channel

Number of respiration channel (default: first respiration channel).

Options

System type   Type of the measuring system: bellows or cushion (default:
bellows).

Datatype   Choose for each possible process datatype either yes or no (de-
fault: yes):

- Respiration period: Create a channel with interpolated respiration pe-
  riod

- Respiration amplitude: Create a channel with interpolated respiration
  amplitude

- Respiratory flow rate: Create a channel with interpolated respiratory
  flow rate

- Respiration time: Create a channel with respiration time stamp

Diagnostic plot   Specify whether a respiratory cycle detection plot should
be created (Yes) or not (No) (default: No).

Channel action

Choose whether to add the new channels or replace a channel previously
added by this method. (default: add)

13.4 Prepare illuminance GLM

Related function: `pspm_process_illuminance`

Transform an illuminance time series into a convolved pupil response time series to be used as nuisance file in a GLM. This allows you to partial out illuminance contributions to pupil responses evoked by cognitive inputs. Alternatively you can analyse the illuminance responses themselves, by extracting parameter estimates relating to the regressors from the GLM.

The illuminance file should be a .mat file with a vector variable called Lx. In order to fulfill the requirements of a later nuisance file there must be as many values as there are data values in the data file. The data must be given in lux (lm/m^2) to account for the non-linear mapping from illuminance to steady-state pupil size. A function to transform luminace (cd/m^2) to illuminace values is provided under PsPM > Data processing > Pupil & Eyetracking. References: [43]

Illuminance file

Select a file that contains illuminance data. The file should contain a variable 'Lx' which should be an n x 1 numeric vector containing the illuminance values.

Sample rate

Specify the sample rate of the illuminance data.

Basis function options

Specify options for the basis functions.

- Duration: Specify the duration of the basis function in seconds (default: 20 s).

- Offset: Specify an offset of the basis function in seconds (default: 0.2 s).

- Dilation: Specify the basis function to model the dilation response.

  - LDRF_GM: Use gamma probability density function to model the dilation response (default).
  - LDRF_GU: Use a smoothed gaussian function to model the dilation response.

- Constriction: Specify the basis function to model the constriction response.

  - LCRF_GM: Use gamma probability density function to model the constriction response (default).

Output directory

Specify the directory where the .mat file with the resulting nuisance data will be written.

Filename for output

Specify the name for the resulting nuisance file.

Overwrite existing file

Choose "yes" if you want to overwrite an existing file with the same name. (Default: No)

13.5   Find valid fixations

Related function: `pspm_find_valid_fixations`

Pupil data time series can contain missing values due to blinks and head movements. Additionally, pupil measurements obtained from a video-based eye tracker depend on the gaze angle, therefore breaks of fixation can be excluded. Valid fixations can be determined by setting an a priori threshold with respect to the fixation point (in degree visual angle) for x- or y-gaze-positions. The input are x- and y-gaze positions converted to length units (see 13.8). The output is a time series with NaN values during invalid fixations (discriminated according to parameters passed to the function). References: [43, 46]

Data file

Specify the PsPM datafile containing the gaze recordings in length units.

Eyes

Choose eyes which should be processed. If 'All eyes' is selected, all eyes which are present in the data will be processed. Otherwise only the chosen eye will be processed. (All eyes [Default], Left eye, Right eye)

Validation method

You can either validate the data by indicating a range on the screen (in degree visual angle) and fixation point(s) or by passing a bitmap representing the screen and holding a 1 for all the fixations that are valid.

- **Validation settings**: Settings to validate fixations within a given range on the screen (in degree visual angle).

- *Visual angle:* Range of valid fixations (around a fixation point). The value has to be in degree visual angle.
- *Distance*: Distance between eyes and screen in in length units.
- *Unit*: Unit in which the distance is given.
- *Resolution*: Resolution to which the fixation point refers (maximum value of the x- and y-coordinates). This can be the resolution set in cogent / psychtoolbox (e.g. [1280 1024]) or the width and height of the screen in length units (e.g. [50 40]).
- *Fixation point:* X- and y-coordinates for the point of fixation with respect to the given resolution.
  - ∗ Default: If default is set, the middle of the screen will be defined as fixation point.
  - ∗ File: .mat file containing a variable F with an n x 2 matrix. N should have the length of the recorded data and each row should define the fixation point for the respective recorded data row.
  - ∗ Point: If the fixation point does not change during the acquisition, specify x- and y-coordinates of the constant fixation point.
- **Bitmap file**: .mat file containing a variable called 'bitmap', which is an n x m matrix. The bitmap represents the display (n=height and m = width) and holds for each position a 1, where a gaze point is valid, and a 0 otherwise. Gaze data at invalid positions (indicated by bitmap or outside the display) are set to NaN.

Channels

Enter a list of channels (numbers or names) to work on. Default is pupil channels. Channels which depend on eyes will automatically be expanded. E.g. pupil becomes pupil_l.

Missing

If enabled an additional channel containing additional information about valid data points will be written. Data points equal to 1 describe epochs which have been discriminated as invalid during validation (=missing). Data points equal to 0 describe epochs of valid data. This function may be enabled in combination with enabled interpolation.

Enabled   Missing is enabled.

Disabled   Missing is disabled.

Output settings

Define how the validated data should be stored.

File output   Write data to a new file or overwrite original data file.

- Overwrite original file: Overwrite original data file.

- Create new file: Write data into given file. The original data will be copied to the given file and the new data channels will, according to the channel output, either be added or replace the original channels.

  - File path: Path to new file.
  - File name: Name of new file.

Channel action   Choose whether to add the new channels or replace a channel previously added by this method. (default: add)

13.6   Pupil Foreshortening Error Correction

Related function: `pspm_pupil_correct`, `pspm_pupil_correct_eyelink`

Perform pupil foreshortening error correction using the equations described in [46]. To perform correction, we define the coordinate system centered on the pupil. In this system, x coordinates grow towards right for a person looking forward in an axis perpendicular to the screen. y coordinates grow upwards and z coordinates grow towards the screen.   In order to perform PFE, we need both pupil and gaze data. If the gaze data in the given file is in pixels, we need information about the screen dimensions and resolution to calculate the pixel to milimeter ratio. On the other hand, if the gaze data is in mm, cm, inches, etc., there is no need to enter any screen size related information. If the gaze data is in pixels and screen information is not given, the function emits a warning and exits early.

Data file

Specify the PsPM datafile containing the pupil and gaze recordings.

Screen resolution

Specify screen resolution ([width height]) in pixels.   Screen resolution is required only if at least one gaze channel is in pixels.

Screen size

Specify screen size ([width height]) in milimeters. Screen size is required only if at least one gaze channel is in pixels.

Correction method

Choose the correction method:

- auto: In auto mode, you need to enter C_z value. Other values will be set using the optimized parameters in reference paper. **Note that you should only use auto mode if your geometric setup matches exactly one of the layouts given in** [46]. For ease of reference, we include the page from [46] that describes the three layouts below.

- manual: In manual mode, you need to enter all values defining the geometry.

Geometry Setup

To perform correction, we define the coordinate system centered on the pupil. In this system, x coordinates grow towards right for a person staring at the screen. y coordinates grow upwards and z coordinates grow towards the screen.

- C_x :Horizontal displacement of the camera with respect to the eye. (Unit: milimeters).

- C_y: Vertical displacement of the camera with respect to the eye. (Unit: milimeters).

- C_z: Eye to camera Euclidean distance if they are on the same x and y coordinates. (Unit: milimeters).

- S_x: Horizontal displacement of the top left corner of the screen with respect to the eye. (Unit: milimeters).

- S_y: Vertical displacement of the top left corner of the screen with respect to the eye. (Unit: milimeters).

- S_z: Eye to top left screen corner distance if they are on the same x and y coordinates. (Unit: milimeters).

Refer to the geometry setup from [46]:
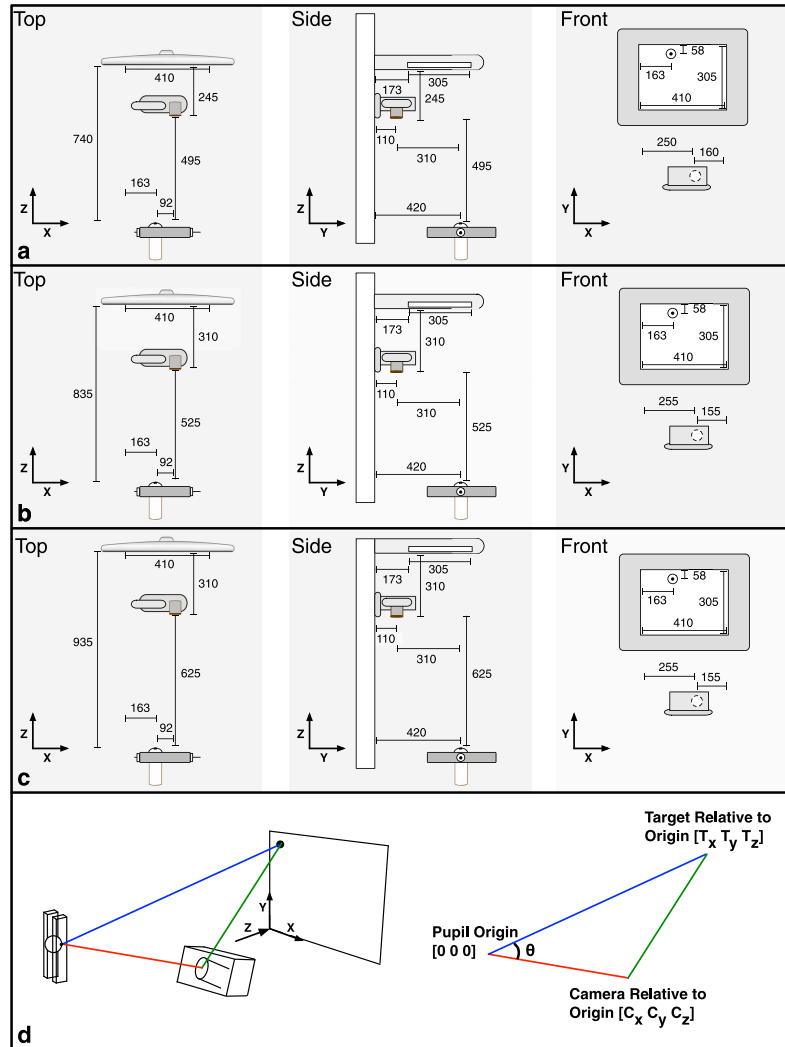
Fig. 3 *Experimental layouts and geometric model.* Three separate geometric layouts (panels **a–c**) varied the relative positions of the camera, artificial eye, and monitor. Each layout is diagrammed from three vantage points (top, side, and front). All distances are in millimeters. Pupil calibration (experiment 1) and pupil foreshortening (experiment 2) were measured for each layout. The geometric model (**d**) estimates the foreshortening of the pupil area as a function of the cosine of the angle θ between the eye-to-camera axis OC and the eye-to-target axis OT. The origin O of the coordinate system is at the center of the artificial pupil. The x-axis is horizontal and parallel to the bottom edge of the screen, growing rightward. The y-axis is vertical, growing upward. The z-axis is perpendicular to the screen, growing inward

Figure 13.1: Hayes & Petrov, 2015, page 5

Channel to correct

- Channel definition: Choose the channel definition. Only the last channel in the file corresponding to the selection will be corrected.

- Channel number: Number of the pupil channel in the PsPM file.

Channel action

- add: Add a new preprocessed channel to the PsPM file. The new channel type will contain a 'pp' suffix at the end if the input channel did not contain a 'pp' suffix.

- replace: Replace previously existing preprocessed channel. If no preprocessed channel exists, this option behaves the same way as add option. (default)

13.7   Pupil Size Preprocessing

Related function: `pspm_pupil_pp, pspm_pupil_pp_options`

Pupil size preprocessing using the steps described in [47]. This function allows users to preprocess two eyes simultaneously and average them in addition to offering single eye preprocessing. Further, users can define segments on which statistics such as min, max, mean, etc. will be computed. We shortly describe the steps performed below:

1. Pupil preprocessing is performed in two main steps. In the first step, the "valid" samples are determined. The samples that are not valid are not used in the second step. Determining valid samples is done by

   (a) Range filtering: Pupil size values outside a predefined range are considered invalid. This range is configurable.

   (b) Speed filtering: Speed is computed as the 1st difference of pupil size array normalized by the temporal separation. Samples with speed higher than a threshold are considered invalid. The threshold is configurable.

   (c) Edge filtering: Samples at both sides of temporal gaps in the data are considered invalid. Both the duration of gaps and the invalid sample duration before/after the gaps are configurable.

   (d) Trendline filtering: A data trend is generated by smoothing and interpolating the data. Then, samples that are too far away from this trend are considered invalid. These two steps are performed multiple times in an iterative fashion. Note that the generated trend is **not** the final result of this function. The smoothing, invalid threshold and the number of passes are configurable.

(e) Isolated sample filtering: Isolated and small sample islands are considered invalid. The size of the islands and the temporal separation are configurable.

2. In the second step, output smooth signal is generated using the valid samples found in the previous step. This is done by performing filtering, upsampling and interpolation. The parameters of the filtering and upsampling are configurable.

Data file

Specify the PsPM datafile containing the pupil and gaze recordings.

Channel to correct

- Channel definition: Choose the channel definition. Only the last channel in the file corresponding to the selection will be corrected.

- Channel number: Number of the pupil channel in the PsPM file.

Channel to combine

Optionally choose the additional channel to use so that a new, combined preprocessed channel can be constructed using the two input channels. The combined channel is created by computing the mean of the two preprocessed channels. Note that main and combined input channels **must** correspond to different eyes.

- Channel definition: Choose the channel definition. Only the last channel in the file corresponding to the selection will be corrected.

- Channel number: Number of the pupil channel in the PsPM file.

Channel action

- add: Add a new preprocessed channel to the PsPM file. The new channel type will contain a 'pp' suffix at the end if the input channel did not contain a 'pp' suffix.

- replace: Replace a previously existing preprocessed channel. If no preprocessed channel exists, this option behaves the same way as add option. (default)

Settings

- default: Use the default settings provided in the underlying preprocessing implementation.

- custom: In custom settings mode, you can specify any of the settings below according to your preferences.

    - Minimum allowed pupil diameter: Minimum allowed pupil diameter. (Unit: in the same unit as the pupil channel)

    - Maximum allowed pupil diameter: Maximum allowed pupil diameter. (Unit: in the same unit as the pupil channel)

    - Island separation min distance: Minimum distance used to consider samples 'separated'. Separated samples can later be considered invalid depending on other parameters. (Unit: ms)

    - Min valid island width: Minimum temporal width required to still consider a sample island valid. If the temporal width of the island is less than this value, all the samples in the island will be marked as invalid. (Unit: ms)

    - Number of medians in speed filter: Number of median to use as the cutoff threshold when applying the speed filter.

    - Max gap to compute speed: Only calculate the speed when the gap between samples is smaller than this value. (Unit: ms)

    - Min missing data width: Minimum width of a missing data section that causes it to be classified as a gap. (Unit: ms)

    - Max missing data width: Maximum width of a missing data section that causes it to be classified as a gap. (Unit: ms)

    - Reject before missing data: The section right before the start of a gap within which samples are to be rejected. (Unit: ms)

    - Reject after missing data: The section right after the end of a gap within which samples are to be rejected. (Unit: ms)

    - Deviation filter passes: Number of passes deviation filter makes.

    - Number of medians in deviation filter: The multiplier used when defining the threshold. Threshold equals this multiplier times the median. After each pass, all the input samples that are outside the threshold are removed. Note that all samples (even the ones which may have been rejected by the previous devation filter pass) are considered.

    - Butterworth sampling frequency: F_S for first order Butterworth filter. (Unit: Hz)

- **–** Butterworth cutoff frequency: Cutoff frequency for first order Butterworth filter. (Unit: Hz)

- **–** Store intermediate steps: If true, intermediate filter data will be stored for plotting. Set to false to save memory and improve plotting performance.

- **–** Interpolation upsampling frequency: The upsampling frequency used to generate the smooth signal. (Unit: Hz)

- **–** Lowpass filter cutoff frequency: Cutoff frequency of the lowpass filter used during final smoothing. (Unit: Hz)

- **–** Lowpass filter order: Filter order of the lowpass filter used during final smoothing.

- **–** Interpolation max gap: Maximum gap in the used (valid) raw samples to interpolate over. Sections that were interpolated over distances larger than this value will be set to NaN. (Unit: ms)

Segments

Optionally define any number of segments. Simple statistics for each segment will be calculated. These segments will be stored in the output channel and also will be shown if plotting is enabled.

- Segment start: Start of the segment in seconds.

- Segment stop: End of the segment in seconds.

- Segment name: Name of the segment.

Plot data

Choose whether to plot the result of preprocessing and possibly various filtering steps performed.

13.8 Convert data

Related function: `pspm_convert_area2mm, pspm_convert_pixel2unit`
Provides conversion functions for the specified data (e.g. pupil data). Currently conversions from area to diameter and from pixel to length units are available.

Data file

Specify the PsPM datafile containing the channels to be converted.

Conversions

Channel   Specify the channel which should be converted. If 0, functions are executed on all channels.

Mode   Choose conversion mode.

- *Area to diameter*

- *Pixel to unit*: Convert pupil gaze coordinates from pixel values to unit values. The unit values can be distance units or degree visual angle. In the latter case, the data are first converted into distance units and then to visual angle. Visual angle is expressed in spherical coordinates as two-element vector where the first element refers to the azimuth angle and the second to the elevation angle. The azimuth angle is the counterclockwise angle in the x-y plane measured in degree from the positive x-axis (This axis is parallel to the screen pointing toward your right). The elevation angle is the angle in degree from the x-y plane in z-axis direction. The range for the azimuth indicates the angle of the left left edge of the screen to the angle of the right edge of the screen. The range for the elevation indicates the angle of the bottom edge of the screen to the angle of the top edge of the screen. (mathworks.com/help/matlab/ref/cart2sph.html).

    - Width: Width of the display window. Unit is 'mm' if 'degree' is chosen, otherwise Unit.

    - Height: Height of the display window. Unit is 'mm' if 'degree' is chosen, otherwise Unit.

    - Unit: Unit to which the measurements should be converted. The value can contain any length unit (mm, cm, inches, etc.) or 'degree'. In this case the corresponding data is firstly convertet into 'mm ' and afterwards the visual angles are computed.

- *Visual angle to scanpath speed:* Takes pairs of channels with gaze data in spherical coordinates (i.e. visual angle) and computes scanpath speed (i.e. scalar distance per second). Saves the result into a new channel with chaneltype 'sps' (Scanpath speed).

    - Eyes: Indicates on which eyes the function should be evaluated.

Channel action

Choose whether to 'replace' the given channel or 'add' the converted data as a new channel. (default: replace)

13.9   Find startle sound onsets

Related function: `pspm_find_sounds`

Translate continuous sound data into an event marker channel. The function adds a new marker channel to the given data file containing the sound data and returns the added channel number. The option threshold, passed in percent to the maximum amplitude of the sound data, allows to specify the minimum amplitude of a sound to be accepted as an event.

Data file

Specify the PsPM datafile containing the imported startle sound data.

Channel

Number of channel containing the startle sounds (default: first sound channel).

Threshold

Percent of the maximum sound amplitude still being accepted as belonging to a sound (important for defining the sound onset).(Default: 0.1 [= 10%]).

Region of interest

Region of interest for discovering sounds. Only sounds between the 2 timestamps will be considered.

- Whole file

- Region

Output

Create channel   The new data channel contains by default all marker onsets found in the specified data file. If you want specific sounds defined by a marker, use the diagnostics option.

- Channel action: Add will append the new marker channel as additional channel to the specified PsPM file. Replace will overwrite the last marker channel of the PsPM file (be careful with reference markers). (Default: 'add')

Diagnostic   Analyze delays between existing marker channel and detected sound onsets.

- Diagnostics output

  - Text only: Output delay statistics as text (amount, mean, stddev).
  - Histogram & plot: Display a histogram of the delays found and a plot with the detected sound, the marker event and the onset of the sound events. Color codes are green (smallest delay) and red (longest delay).

- Create channel with specific sounds: Create new data channel which contains only marker onsets which could have been assigned to a marker in the specified marker channel (Default: No).

  - Channel action: Add will append the new marker channel as additional channel to the specified PsPM file. Replace will overwrite the last marker channel of the PsPM file (be careful with reference markers). (Default: 'add')

- Marker channel

- Max delay: Upper limit in seconds of the window in which sounds are accepted to belong to a marker. Default: 3 s

- Min delay: Lower limit in seconds of the window in which sounds are accepted to belong to a marker. Default: 0 s

- Expected sound count: Checks for correct number of detected sounds. If too few sounds are found, threshold is lowered until specified count is reached.

13.10   Preprocess startle eyeblink EMG

Related function: `pspm_emg_pp`

Preprocess startle eyeblink EMG data for further analysis. Noise in EMG data will be removed in three steps: Initially the data is filtered with a 4th order Butterworth filter with cutoff frequencies 50 Hz and 470 Hz. Then, Mains frequency will be removed using a notch filter at 50 Hz (can be changed). Finally, the data is smoothed and rectified using a 4th order Butterworth low-pass filter with a time constant of 3 ms (= cutoff at 53.05 Hz). The applied filter settings are according to the literature[56]. While the input data must be an 'emg' channel, the output will be an 'emg_pp' channel which is the requirement for startle eyeblink GLM.

Data file

Specify the PsPM datafile containing the EMG data channel.

Options

Channel   Channel ID of the channel containing the unprocessed EMG data.

Mains frequency   The frequency of the alternating current (AC) which will be filtered out using bandstop filter.

Channel action   Defines whether the processed data should be added as a new channel or replace the last existing channel of the same data type. After preprocessing the data will be stored as 'emg_pp' channel. (default: add)

13.11   Gaze Preprocessing

Related function: `pspm_convert_gaze_distance`

Convert gaze data from degree or distance units such as pixels or mm into degree data or scanpath speed. The gaze data must be oriented such that the point 0,0 relates the bottom-left of the screen, and the maximal x and y points relate to the top-right of the screen. These functions do not accept input channels, they will seacrch the provided data file for channels with the gaze_[x|y]_[l|r] chantype in the specified from unit.

Data file

Specify the PsPM datafile containing the gaze recordings.

Conversion

Degree To Scanpath Speed   Convert existing degree data to scanpath speed, stored as chantype: sps_[l|r]

Distance To Degree   Convert distance x/y coordinate data to degrees stored as chantype: gaze_[x|y]_[l|r], unit: degree

- From - unit to covert from, [pixel, mm, cm, inches, m]

- Width - The width of the screen in mm

- Height - The height of the screen in mm

- Distance - The distance to the screen from the subjects eyes, in mm

Distance To Scanpath Speed Convert distance x/y coordinate data to scanpath speed, stored as chantype: sps_[l|r]

- From - unit to covert from, [pixel, mm, cm, inches, m]

- Width - The width of the screen in mm

- Height - The height of the screen in mm

- Distance - The distance to the screen from the subjects eyes, in mm

Channel Action

Defines whether the processed data should be added as a new channel or replace the last existing channel of the same chantype and unit.

13.12 Preprocessing SCR

Related function: `pspm_scr_pp`

Applies skin conductance response (SCR) preprocessing for the given data. The function applies to two rules: (1) microsiemens values must be within range (0.05 to 60), (2) Absolute slope of value change must be less than 10 microsiemens per second.

Data file Specify a file name or a cell array of file names.

Parameters

- Minimum value: Minimum SCR value in μS, default as $0.05\mu S$.

- Maximum value: Maximum SCR value in μS, default as $60\mu S$.

- Maximum slope: Maximum slope in $\frac{\mu S}{s}$, default as $10\frac{\mu S}{s}$.

- Missing epochs filename: Name of the .mat file where the missing epochs will be stored.

- Deflection threshold: Amplitude threshold which excludes epochs from the filter if the overall deflection over this epoch is below it. Default: 0.1 .

- Island threshold: Duration threshold (seconds) which determines the maximum length of data between NaN epochs. Islands of data shorter than this threshold will be removed. Default: 0 s (no threshold).

- Expand epochs: Duration threshold (seconds) which determines by how much data on the flanks of artefact epochs will be removed. Default: 0.5 s.

- clipping_step_size: the step size in moving average algorithm for detecting clipping, default as 2.

- clipping_threshold: the proportion of local maximum in a step, default as 0.1.

Channel action   Defines whether the new channel should be added or the previous outputs of this function should be replaced, default as replaced.

14   First level

14.1   GLM (modality-independent options)

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`
   General linear convolution models (GLM) are powerful for analysing evoked responses in various modalities that follow an event with (approximately) fixed latency. This is similar to standard analysis of fMRI data. The user specifies events for different conditions. These are used to estimate the mean response amplitude per condition. These mean amplitudes can later be compared, using the contrast manager.

Model Filename & output directory

Specify file name and directory where the *.mat file with the resulting model will be written.

Modality specific Channel

Indicate the channel containing the modality specific data. By default the **last** modality specific channel is assumed to contain the data for this model. If the last modality specific channel does not contain the data for this model (e. g. there are two modality specific channels), indicate the channel number (within the modality specific file) that contains the data for this model.
   Note: If modality is pupil, the function loads the channels according to a precedence order and then uses the last channel among all these loaded ones. The precedence order is given below:

1. Combined pupil channels (by definition also preprocessed): These channels are constructed by averaging the data from both left and right eye after performing pupil preprocessing on both of these channels.

2. Preprocessed pupil channels corresponding to best eye: Best eye is defined as the channel having less amount of missing data. If there is only one eye in PsPM file, then that eye is defined as the best eye.

3. Preprocessed pupil channels: In case there are two eyes and the best eye channel is not preprocessed, then PsPM prefers the preprocessed channel.

4. Best eye pupil channels: When there is no preprocessed channels, PsPM prefers the best eye channel.

Time Units

Indicate the time units on which the specification of the conditions will be based. Time units can be specified in 'seconds', number of 'markers', or number of data 'samples' . Time units refer to the beginning of the data file and not to the beginning of the original recordings e. g. if data were trimmed.

Option for marker channels:

- Marker Channel: Indicate the marker channel. By default the first marker channel is assumed to contain the relevant markers. Markers are only used if you have specified the time units as 'markers'.

Data & Design

Add the required number of data files or sessions here. These will be concatenated for analysis.

Data File Add the data file containing the modality specific data (and potential marker information). If you have trimmed your data, add the file containing the trimmed data.

Missing Epochs Indicate epochs in your data in which the modality specific signal is missing or corrupted (e.g., due to artifacts). NaN values in the signal will be interpolated for filtering and downsampling and later automatically removed from data and design matrix.

- No Missing Epochs: The whole time series will be analyzed (default).

- Define Missing Epochs: Define the start and end points of the missing epochs either as epoch files or manually. Missing epochs will be excluded from the design matrix. Start and end points have to be defined in seconds starting from the beginning of the session.

  - Missing Epoch File: Indicate an epoch file specifying the start and end points of missing epochs. The mat file has to contain a variable 'epochs', which is an m x 2 array, where m is the number of missing

epochs. The first column marks the start points of the epochs that are excluded from the analysis and the second column the end points.

- Enter Missing Epochs Manually (m: nr. of epochs): Enter the start and end points of missing epochs manually. Specify an m x 2 array, where m is the number of missing epochs. The first column marks the start points of the epochs that are excluded from the analysis and the second column the end points.

Design   Specify the timing of the events within the design matrix. Timing can be specified in 'seconds', 'markers' or 'samples' with respect to the beginning of the data file. See 'Time Units' settings. Conditions can be specified manually or by using multiple condition files (i.e., an SPM-style mat file). There are several possibilities to specify the design. All of them require you to state, for each condition:

- Name of the experimental condition

- Onsets: vector of event onsets for this condition, expressed in seconds, marker numbers, or samples, as specified in time units. If seconds, do use precise values according to the data type (e. g. at least 0.1 s time resolution for SCR)

- Durations (optional, default 0): vector of durations of each event for this condition. You need to use 'seconds' or 'samples' as time units

- Parametric modulators: this is used to specify regressors that specify how responses in an experimental condition depend on a parameter to model the effect e.g. of habituation, reaction times, or stimulus ratings. For each parametric modulator a new regressor is included in the design matrix. To create this column, the normalized parameters are multiplied with the respective onset regressors. Parametric modulators require:

    – Names for each parametric modulator for this condition

    – Actual values: a vector for each parameter for this condition, containing as many numbers as there are onsets

    – Polynomial degree. A value of 1 leaves the parametric modulator unchanged and thus corresponds to a linear change over the values of the parametric modulator (first-order). Higher order modulation introduces further columns that contain the non-linear parametric modulators [e.g., second-order: (squared), third-order (cubed), etc].

- Condition File: Create a file with the following variables:

- **names:** a cell array of string for the names of all experimental conditions
- **onsets:** a cell array of number vectors for the event onsets with the same number of elements as names
- **durations** (optional, default 0): a cell array of vectors for the duration of each event
- **pmod** (optional): a struct array with the same number of elements as names, and containing the fields
  * **name:** cell array of names for each parametric modulator for this condition
  * **param:** cell array of vectors for each parameter for this condition, containing as many numbers as there are onsets
  * **poly** (optional, default 1): specifies the polynomial degree
- e.g. produce a simple multiple condition file by typing: `names = {'condition a', 'condition b'}; onsets = {[1 2 3], [4 5 6]}; save('testfile', 'names', 'onsets');`

- Enter conditions manually:

  - **Name:** Specify the name of the condition.
  - **Onsets:** Specify a vector of onsets.
  - **Durations:** Typically, a duration of 0 is used to model an event onset. If all events included in this condition have the same length specify just a single number. If events have different durations, specify a vector with the same length as the vector for onsets.
  - **Parametric Modulator(s)** [optional] (n: nr. of pmods): If you want to include a parametric modulator, specify a vector with the same length as the vector for onsets.
    1. **Name:** Specify the name of the parametric modulator.
    2. **Polynomial Degree:** Specify an exponent that is applied to the parametric modulator.
    3. **Parameter Values:** Specify a vector with the same length as the vector for onsets.

- Define conditions from distinct values or names of event markers: This option defines event onsets according to the values or names of events stored in the a marker channel. These values or names are imported for some data types.

  - **Condition-defining values/names**
  - **Name:** the names of the conditions in the same order as the conditioning-defining values/names

- No condition - If there is no condition, it is mandatory to specify a nuisance file (e. g. for illuminance GLM, see 13.4).

Nuisance File (optional)   You can include nuisance parameters such as motion parameters in the design matrix. Nuisance parameters are not convolved with the canonical response function. This is also used for the illuminance GLM (see 13.4). The file has to be either a .txt file containing the regressors in columns, or a .mat file containing the regressors in a matrix variable called R. There must be as many values for each column of R as there are data values in your data file. PsPM will call the regressors pertaining to the different columns R1, R2, ...

Normalize

Specify if you want to z-normalize the modality specific data for each subject. For within-subjects designs, this is highly recommended, but for between-subjects designs it needs to be set to false.

Centering

Specify if you want to mean centering the convoluted X data. By default this centering is applied to every model but for some specific cases such as some scanpath length analysis it has to be set to 0 (i.e. False).

Filter Settings

Specify how you want filter the modality specific data.

Default   Standard settings for the Butterworth bandpass filter. These are the optimal settings for the modaliy specific data.

Edit Settings   Create your own filter settings (not encouraged).

- Low-Pass Filter
    - Enable
        1. Cutoff Frequency: Specify the low-pass filter cutoff in Hz.
        2. Filter Order: Specify the low-pass filter order.
    - Disable
- High-Pass Filter
    - Enable
        1. Cutoff Frequency: Specify the high-pass filter cutoff in Hz.

2. Filter Order: Specify the high-pass filter order.

– Disable

- New Sampling Rate: Specify the sampling rate in Hz to down sample modality specific data. Enter NaN to leave the sampling rate unchanged.

- Filter Direction {uni, bi}, (default: uni): A unidirectional filter is applied twice in the forward direction. A 'bidirectional' filter is applied once in the forward direction and once in the backward direction to correct the temporal shift due to filtering in forward direction.

Create information on missing data values

Option to extract information over missing values in each condition of the GLM. This option extracts the ratio of NaN-values over all trials for each condition, and whether this ratio exceeds a cutoff value. The information is stored in the GLM structure and will be used in future releases for excluding values during extraction and first-level contrasts.

- *No*

- *Yes*: Need to define the segment length and a cutoff value

    – Segment length: Length of segments.
    – Cutoff: Value which represents upperbound for the ratio of NaN-values in the trials.

Overwrite Existing File

Specify whether you want to overwrite existing mat files.

14.2   GLM for SCR

Related function: `pspm_glm`, `pspm_glm_recon`, `pspm_get_timing`
   This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM [14.1]. References: [6, 24, 9, 10].

Basis Function

Basis functions for the peripheral model. Standard is to use a canonical skin conductance response function (SCRF) with time derivative for later reconstruction of the response peak.

SCRF0   SCRF without derivatives.

SCRF1   SCRF with time derivative (default). This is the recommended option based on the publication [10].

SCRF2   SCRF with time and dispersion derivative.

FIR   Uninformed finite impulse response (FIR) model: specify the number and duration of time bins to be estimated.

- Arguments

  - N: Number of Time Bins: Number of time bins.

  - D: Duration of Time Bins: Duration of time bins (in seconds).

14.3   GLM for evoked HPR

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`
   This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM[14.1]. References: [36]].

Basis Function

Basis functions for the peripheral model.

HPRF_E

- Arguments

  - **–** Number of basis functions

FIR   Uninformed finite impulse response (FIR) model: specify the number and duration of time bins to be estimated.

- Arguments

  - **–** N: Number of Time Bins: Number of time bins.

  - **–** D: Duration of Time Bins: Duration of time bins (in seconds).

14.4   GLM for fear-conditioned HPR

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`
   This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM[14.1]. References: [37].

Basis Function

Basis functions for the peripheral model.

Function

> **HPRF_FC0**   HPRF_FC without derivatives.

> **HPRF_FC1**   HPRF_FC with time derivative.

SOA   Specify custom SOA for response function. Tested values are 3.5 s, 4 s and 6 s. Default: 3.5 s

14.5   GLM for fear-conditioned PSR

Related function: `pspm_glm`, `pspm_glm_recon`, `pspm_get_timing`
    This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM[14.1]. References: [43].

Basis Function

Basis functions for the peripheral model.

PSRF_FC0   PSRF_FC with CS only and without derivatives.

PSRF_FC1   PSRF_FC with CS and derivatives for CS (default).

PSRF_FC2   PSRF_FC with CS and US. Without derivatives.

PSRF_FC3   PSRF_FC with US only and without derivatives.

Default Channel

Best eye   Use eye with the fewest NaN values.

First left eye   Look for first left eye channel.

First right eye   Look for first right eye channel.

14.6 GLM for evoked RAR

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`
This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM[14.1]. References: [51].

Basis Function

Basis functions for the peripheral model.

RARF_E0 RARF_E without time derivative.

RARF_E1 RARF_E with time derivative (default).

14.7 GLM for fear-conditioned RAR

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`
This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM[14.1]. References: .

Basis Function

Basis functions for the peripheral model.

RARF_FC0 RARF_FC early and late response.

RARF_FC1 RARF_FC early response with derivative.

14.8 GLM for evoked RPR

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`
This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM[14.1]. References: [51].

Basis Function

Basis functions for the peripheral model.

RPRF_E0 RPRF_E without time derivative.

RPRF_E1 RPRF_E with derivative.

14.9   GLM for evoked RFRR

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`

   This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM[14.1]. References: [51].

Basis Function

Basis functions for the peripheral model.

RFRRF_E0   RFRRF_E without time derivative.

RFRRF_E1   RFRRF_E with derivative.

14.10   GLM for SEBR

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`

   This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM [14.1]. References: [56].

Latency

Latency is either 'fixed' or 'free'. If latency is 'free', the model estimates the best latency within the given time window for each regressor (using a dictionary matching algorithm) and then inverts the GLM with these latencies. See [56] in the context of SEBR.

Fixed latency

Free latency

- Time window - In seconds, specifies over which time window latencies should be evaluated. Positive values mean that the response function is shifted to later time points, negative values that it is shifted to earlier time points.

Basis Function

Basis functions for the peripheral model.

SEBRF0   SEBRF without time derivative (default).

SEBRF1 SEBRF with derivative.

14.11 GLM for SPS

Related function: `pspm_glm, pspm_glm_recon, pspm_get_timing`

This section only covers modality specific topics for GLM. Settings applying to all modalities are discussed in the general section GLM [14.1].

Basis Function

Basis functions for the peripheral model.

Average scanpath speed This option implements a boxcar function over 2 s before the US time point, and yields the averaged scan path speed over that interval (i.e. the scan path length over that interval, divided by 2 s). (default).

SPSRF_FC This option implements a gamma function for fear-conditioned scan path speed responses, time-locked to the end of the CS-US interval.

SOA Specify custom SOA for response function. Tested values are 3.0 s and 3.5 s. Default: 3.5 s

14.12 Non-Linear Model for SCR

Related function: `pspm_dcm, pspm_dcm_inv`

Non-linear models for SCR are powerful if response timing is not precisely known and has to be estimated. A typical example are anticipatory SCR in fear conditioning – they must occur at some point within a time-window of several seconds duration, but that time point may vary over trials. Dynamic causal modelling (DCM) is the framework for parameter estimation. PsPM implements an iterative trial-by-trial algorithm. Different from GLM, response parameters are estimated per trial, not per condition, and the algorithm must not be informed about the condition. Trial-by-trial response parameters can later be summarized across trials, and compared between conditions, using the contrast manager. References: [29, 11]

Model Filename & Output directory

Specify file name for the resulting model. Specify directory where the mat file with the resulting model will be written.

SCR channel

Indicate the channel containing the SCR data. By default the first SCR channel is assumed to contain the data for this model. If the first SCR channel does not contain the data for this model (e. g. there are two SCR channels), indicate the channel number (within the SCR file) that contains the data for this model.

Data & design

Add the appropriate number of data files or sessions here. Results will be concatenated.

Data File   Add the data file containing the SCR data. If you have trimmed your data, add the file containing the trimmed data.

Design   Specify the timings of individual events from your design either by creating a timing file or by entering the timings manually. The DCM framework allows you to specify two types of events:

Fixed latency (evoked): An event is assumed to elicit an immediate response. The amplitude of the sympathetic arousal will be estimated, while the latency and duration of the response are fixed. This event type is meant to model evoked responses.

Flexible latency and duration (event-related): An event is assumed to elicit sympathetic arousal within a known response window, but with unknown amplitude, latency and duration. For each event of this type, specify a time window in which the response is expected. PsPM will then estimate the amplitude, duration and latency of the response. An example for this type of event is an anticipatory response which might vary in timing between trials (e. g. in fear conditioning).

- Timing File: The timing file has to be a .mat file containing a cell array called 'events'. For a design with n trials and m events per trial (n × m events in total), the cell array has to be structured in the following way: Create m cells in the cell array (one per event type that occurs in each trial). Each cell defines either a fixed or a flexible event type. A cell that defines a fixed event type has to contain a vector with n entries, i.e. one time point per trial, in seconds, samples or markers. A cell that defines a flexible type has to contain an array with n rows and two columns. The first column specifies the onsets of the time windows for each trial, while the second column specifies the offsets of the time windows. It is assumed that all trials have the same structure, i.e. the same number of fixed and flexible event types. For individual trials with a different structure you can enter negative values as time

information to omit estimation of a response. For later comparison between trials of different conditions, it is absolutely mandatory that they contain the same types of events, to avoid bias. Hence, if one condition omits an event (e. g. unreinforced trials in conditioning experiments), the omitted event needs to be modelled as well.

- Enter Timing Manually: In the DCM framework, a session is structured in n individual trials. Each trial contains m fixed and/or flexible event types. All trials need to have the same structure. Create event types to define the structure of a trial and enter the timings for all events.

  - Name (optional): Enter a name of the event. This name can later be used for display and export.

  - Onsets: For events with a fixed response, specify a vector of onsets. The length of the vector corresponds to the number of trials (n). For events with a flexible response, specify a two column array. The first column defines the onset of the time window in which the response occurs. The second column defines the offset. The number of rows of the array corresponds to the number of trials (n). All timings are specified in seconds.

Conditions (optional)   Specify the conditions that the individual trials belong to. This information is not used for the parameter estimation, but it allows you to later access the conditions in the contrast manager.

- Name: Specify the name of the condition.

- Index: Specify a vector of trial indices between 1 and n. The length of the vector corresponds to the number of events included in this condition.

Missing epochs   Indicate epochs in your data in which the signal is missing or corrupted (e.g., due to artifacts). Data around missing epochs are split into subsessions and are evaluated independently if the missing epoch is at least as long as subsession threshold. NaN periods within the evaluated subsessions will be interpolated for averages and principal response components.

- No Missing Epochs: Missing epochs are detected automatically according to the data option 'Subsession threshold'.

- Define Missing Epochs: Define the start and end points of the missing epochs either as epoch files or manually. Start and end points have to be defined in seconds starting from the beginning of the session.

    – Missing Epoch File: Indicate an epoch file specifying the start and end points of missing epochs (m). The mat file has to contain a variable 'epochs', which is an m x 2 array, where m is the number of missing epochs. The first column marks the start points of the epochs that are excluded from the analysis and the second column the end points.

    – Enter Missing Epochs Manually: Enter the start and end points of missing epochs (m) manually. Specify an m x 2 array, where m is the number of missing epochs. The first column marks the start points of the epochs that are excluded from the analysis and the second column the end points.

Note: pspm_dcm calculates the inter-trial intervals as the duration between the end of a trial and the start of the next one. ITI value for the last trial in a session is calculated as the duration between the end of the last trial and the end of the whole session. Since this value may differ significantly from the regular ITI duration values, it is not used when computing the minimum ITI duration of a session.

   Minimum of session specific min ITI values is used

1. *when computing mean SCR signal*

2. *when computing the PCA from all the trials in all the sessions.*

In case of case (2), after each trial, all the samples in the period with duration equal to the just mentioned overall min ITI value is used as a row of the input matrix. Since this minimum does not use the min ITI value of the last trial in each session, the sample period may be longer than the ITI value of the last trial. In such a case, pspm_dcm is not able to compute the PCA and emits a warning.

   The rationale behind this behaviour is that we observed that ITI value of the last trial in a session might be much smaller than the usual ITI values. For example, this can happen when a long missing data section starts very soon after the beginning of a trial. If this very small ITI value is used to define the sample periods after each trial, nearly all the trials use much less than available amount of samples in both case (1) and (2). Instead, we aim to use as much data as possible in (1), and perform (2) only if this edge case is not present.

   In PsPM 5.1.0, pspm_dcm can determine the last trials that is less than the given cut-off value (default as 7s).

Data Options

Normalization Specify if you want to normalize the SCR data for each subject. For within-subjects designs, this is highly recommended. During

analysis, data are always z-transformed, but the parameter estimates are transformed back if this is set to false.

Filter Settings   Specify if you want filter the SCR data.

- Default: Standard settings for the Butterworth bandpass filter. These are the optimal settings from [11]: bidirectional Butterworth band pass filter with cutoff frequencies of 0.0159 Hz and 5 Hz, downsampling to 10 Hz.

- Edit Settings: Create your own filter (discouraged).

    – Low-Pass Filter
        1. Enable
            - Cutoff Frequency: Specify the low-pass filter cutoff in Hz.
            - Filter Order: Specify the low-pass filter order.
        2. Disable
    – High-Pass Filter
        1. Enable
            - Cutoff Frequency: Specify the high-pass filter cutoff in Hz.
            - Filter Order: Specify the high-pass filter order.
        2. Disable
    – New Sampling Rate: Specify the sampling rate in Hz to down sample SCR data. Enter NaN to leave the sampling rate unchanged.
    – Filter Direction {uni, bi}, (default: bi): A unidirectional filter is applied twice in the forward direction. A 'bidirectional' filter is applied once in the forward direction and once in the backward direction to correct the temporal shift due to filtering in forward direction.

Subsession threshold   Specify the minimum duration (in seconds) of NaN periods to be considered as missing epochs. Data around missing epochs is then split into subsessions, which are evaluated independently. This setting is ignored for sessions having set missing epochs manually.

Last trial cutoff [s] (default: 7s)   Define a cutoff value for using the parameters estimated from the last trial. If there are fewer data after the end of the last trial in a session than the given cutoff value (in second), the estimated parameters from this trial will be assumed inestimable and set to *NaN* after the inversion. This value can be set as *inf* to always retain the parameters in the last trial. The default value for this field is 7 s, corresponding to the time at which the canonical SCRF has decayed to around 80% of its peak value.

Constrained model   This option can be set to one if the flexible responses have fixed dispersion (0.3 s SD) but flexible latency. If the option is set, the value must be 0 or 1. The default value is 0.

Response Function Options

Estimate the Response Function from The Data   A response function can be estimated from the data and used instead of the canonical response function (default 0 for fully flexible and 1 for fix and flex/fix paradigms). This is not normally recommended unless you have long inter trial intervals in the range of 20-30 s [11].

Only Estimate RF (Do Not Do Trial-Wise DCM)   This option can be used to estimate an individual response function to be used in analysis of another experiment.

Call External File to Provide Response Function   Call an external file to provide a response function, which was previously estimated using the option "only estimate RF".

Inversion Options

Number of Trials to Invert at The Same Time (default: 2 s)   The iterative DCM algorithm accounts for response overlap by considering several trials at a time. This can be set here.

SF-Free Window Before First Event [s] (default: 2 s)   The DCM algorithm automatically models spontaneous fluctuations in inter trial intervals. Here, you can define a time window before the first event in every trial in which no spontaneous fluctuation will be estimated.

SF-Free Window After Last Event [s] (default: 5 s)   The DCM algorithm automatically models spontaneous fluctuations in inter trial intervals. Here, you can define a time window after the last event in every trial in which no spontaneous fluctuation will be estimated.

Maximum Frequency of SF in ITIs [Hz] (default: 0.5 Hz)   The DCM algorithm automatically models spontaneous fluctuations in inter trial intervals. Here you can define the minimal delay between two subsequent spontaneous fluctuations.

SCL-Change-Free Window Before First Event [s] (default: 2 s) The DCM algorithm automatically models baseline drifts in inter trial intervals. Here, you can define a window before the first event in every trial in which no change of the skin conductance level will be assumed.

SCL-Change-Free Window After Last Event [s] (default: 5 s) The DCM algorithm automatically models baseline drifts in inter trial intervals. Here, you can define a window after the last event in every trial in which no change of the skin conductance level will be assumed.

Display Options

Display Progress Window Show a on-line diagnostic plot for each iteration of the estimation process.

Display Intermediate Windows Show small plots displaying the progress of each iteration in the estimation process.

14.13 SF models

Related function: `pspm_sf`

This suite of models is designed for analysing spontaneous fluctuations (SF) in skin conductance as a marker for tonic arousal. SF are analysed over time windows that typically last 60 s and should at least be 15 s long. PsPM implements 3 different models: (1) Skin conductance level (SCL): this is the mean signal over the epoch (2) Area under the curve (AUC): this is the time-integral of the above-minimum signal, divided by epoch duration. This is designed to be independent from SCL and ideally represents the number x amplitude of SF in this time window. (3) Number of SF estimated by DCM (Dynamic Causal Modelling) or MP (Matching Pursuit): this is a non-linear estimation of the number of SF, and is the most sensitive indicator of tonic arousal. It relies on absolute data values as it implements and absolute threshold for data peaks. References: [31, 32].

The parameter estimates of these different methods have interpretable physical units: (1) SCL parameters have the same units as the input data, most often μS, (2) AUC parameters have the same units as the input data, most often μS, and (3) DCM and MP parameters are the frequency of above-threshold SF, stated in $Hz = 1/s$.

Data File

Add the data file containing the SCR data (and potential marker information). If you have trimmed your data, add the file containing the trimmed data.

Model Filename & Output directory

Specify file name for the resulting model. Specify directory where the mat file with the resulting model will be written.

Method

Choose the method for estimating tonic sympathetic arousal: AUC (equivalent to number x amplitude of spontaneous fluctuations), SCL (tonic skin conductance level), DCM or MP. The latter two estimate the number of spontaneous fluctuations, requiring absolute data units as both methods implement an absolute amplitude threshold. In theory, DCM provides highest sensitivity but is slow [31]. MP is a very fast approximation to the DCM results, and comparable in sensitivity for analysis of empirical data [32]. In simulations, it is less accurate when the expected number of SF exceeds 10/min.

Time Units

Indicate the time units on which the specification of the conditions will be based. Time units can be specified in 'seconds', number of 'markers', or number of data 'samples' . Time units refer to the beginning of the data file and not to the beginning of the original recordings e. g. if data were trimmed. Choose 'seconds', 'markers', 'samples', or 'whole' to analyse the entire data file as one epoch.

Seconds/Markers/Samples

- Epochs - define data epochs to analyse

    – Epoch File
    – Enter Epochs Manually

Markers

- Marker Channel (default: 0): Indicate the marker channel. By default the first marker channel is assumed to contain the relevant markers. Markers are only used if you have specified the time units as 'markers'.

Filter Settings

Specify if you want filter the SCR data.

Default   Standard settings for the Butterworth bandpass filter: unidirectional Butterworth bandpass filter with cut off frequencies of 0.0159 Hz and 5 Hz, down sampling to 10 Hz.

Edit Settings   Create your own filter (discouraged).

- Low-Pass Filter

    - Enable

        1. Cutoff Frequency: Specify the low-pass filter cutoff in Hz.
        2. Filter Order: Specify the low-pass filter order.

    - Disable

- High-Pass Filter

    - Enable

        1. Cutoff Frequency: Specify the high-pass filter cutoff in Hz.
        2. Filter Order: Specify the high-pass filter order.

    - Disable

- New Sampling Rate: Specify the sampling rate in Hz to down sample SCR data. Enter NaN to leave the sampling rate unchanged.

- Filter Direction): A unidirectional filter is applied twice in the forward direction. A 'bidirectional' filter is applied once in the forward direction and once in the backward direction to correct the temporal shift due to filtering in forward direction.

Channel

Indicate the channel containing the SCR data. By default the first SCR channel is assumed to contain the data for this model. If the first SCR channel does not contain the data for this model (e. g. there are two SCR channels), indicate the channel number (within the SCR file) that contains the data for this model.

Overwrite Existing File

Specify whether you want to overwrite existing mat file.

Threshold (used for DCM and MP only)

Threshold for SN detection - default 0.1 µS.

Display Progress Window (used for DCM and MP only)

Show a on-line diagnostic plot for each iteration of the estimation process (DCM) or for the result of the estimation process (MP).

Display Intermediate Windows (used for DCM only)

Show small plots displaying the progress of each iteration in the estimation process.

14.14   Review First-Level Model (via Batch editor)

Related function: `pspm_review`
   This module allows you to look at the first-level (within-subject) model to investigate model fit and potential estimation problems. This is not necessary for standard analyses. Further processing can be performed directly on the second level after first-level model estimation.

Model File

Choose model file to review.

Model Type

Specify the type of model.

GLM   Specify the plot that you wish to display:

- Design matrix

- Orthogonality

- Predicted & Observed

- Regressor names

- Reconstructed

Non-linear model

- Inversion results for one trial: Non-linear SCR model (DCM) - Review individual trials or sequences of trials inverted at the same time.

  – Session Number (Default 1): Data session. Must be 1 if there is only one session in the file.

  – Trial Number: Trial to review.

- Predicted & Observed for all trials: Non-linear SCR model for event-related responses - Review summary plot of all trials.

  – Session Number

> **–** Figure Name [optional]: Adding a figure name saves the figure as .fig file.

- SCRF: Non-linear model for event-related responses - review SCRF (useful if SCRF was estimated from the data).

SF Non-linear model for spontaneous fluctuations: Show inversion results for one episode

- Epoch Number: Epoch to review.

Contrasts Display contrast names for any first level model.

14.15   First-Level Model Review Manager (via GUI)

Related function: `pspm_review`

Display information stored in a model file through an easily accessible interface. Add a model file to the menu via *Add Models*. Available content of a highlighted model is listed on the right hand side under *Figure Selection*. For non-linear models, you can select the session number by entering its index in the field *Session number*; GLMs are inverted and displayed across all sessions. Click on any *Display* or *Show* button to visualize information. Details for each model type are listed in Chapter 14.14.

14.16   First-Level Contrasts (via Batch editor)

Related function: `pspm_con1`

Define within-subject contrasts here for testing on the second level. Contrasts can be between conditions (GLM), epochs (SF) or trials (Non-linear SCR models). Contrast weights should add up to 0 (for testing differences between conditions/epochs/trials) or to 1 (for testing global/intercept effects across conditions/epochs/trials). Example: an experiment realises a 2 (Factor 1: a, A) x 2 (Factor 2: b, B) factorial design and consists of 4 conditions: aa, aB, Ab, AB. Testing the following contrasts is equivalent to a full ANOVA model: Main effect factor 1: [1 1 -1 -1], Main effect factor 2: [1 -1 1 -1], Interaction factor 1 x factor 2: [1 -1 -1 1]. To test condition effects in non-linear models, assign the same contrast weight to all trials from the same condition.

Model File(s)

Specify the model file for which to compute contrasts.

Specify Contrasts on Datatype

Contrasts are usually specified on parameter estimates. For GLM, you can also choose to specify them on conditions or reconstructed responses per condition. In this case, your contrast vector needs to take into account only the first basis function. For non-linear SCR models, you can specify contrasts based on conditions as well. This will average within conditions. Use the review manager to extract condition names and their order. This argument will be ignored for other first-level models.

   - Parameter: Use all parameter estimates.

   - Condition: Contrasts formulated in terms of conditions in a GLM, automatically detects number of basis functions and uses only the first one (i.e. without derivatives), or based on assignment of trials to conditions in DCM.

   - Reconstructed: Contrasts formulated in terms of conditions in a GLM, reconstructs estimated response from all basis functions and uses the peak of the estimated response.

Contrast(s)

Contrast Name   This name identifies the contrast in tables and displays.

Contrast Vector   This is a vector on all included conditions (GLM), trials (non-linear SCR model for event-related responses), or epochs (non-linear model for SF). Shorter vectors will be appended with zeros. To specify a condition or trial difference, the contrast weights must add up to zero (e. g. was sympathetic arousal in condition A larger than in condition B: c = [1 -1]). To specify a summary of conditions or trials, the contrast weights should add up to 1 to retain proper scaling (e. g. was non-zero sympathetic arousal elicited in combined conditions A and B: c = [0.5 0.5]) .

Delete Existing Contrasts

Deletes existing contrasts from the model file.

Z-Scoring

Use parameter estimates across all conditions for each parameter/event type, subtract the mean and divide by the standard deviation, before computing the contrast. This option is only available for models with trial-wise parameter estimates. (default: false)

14.17   First level contrast manager (via GUI)

Related function: `pspm_con1`

The Manager guides you through the setup of pre-defined within-subject contrasts applied to the parameter estimates of your model.

Load Model   Import a model file from a subject containing parameter estimates for which to compute contrasts.

Define Contrast Name   Enter a name to identify a contrast. By pressing *New Contrast*, the contrast is added to the box *Contrasts*. You can reset existing contrasts by highlighting a contrast and pressing *Clear Contrast* or you can remove contrasts from your model file by pressing *Delete Contrast.*

Define Stats Type   Depending on the model (GLM, spontaneous fluctuation or non-linear model), different data types are available (see Chapter 14.16 for details). Choose the type you want to be entered into a statistical test.

Define Test   Three pre-defined types of statistics can be computed on your estimates. Parameter estimates for individual conditions or trials are grouped by the user to compute test contrasts. "Group" refers to groups of conditions/trials, not to groups of individuals or datasets. After choosing the test, you can add or remove conditions/trials to a group by clicking on them in the *Names* window. The color will indicate to which group they belong.

- Test of intercept: Compute the average of conditions entered in *Group 1*. This option is suitable if you are interested in the main effect of one or across several conditions.

- Test of condition differences: Compute the difference between the averages of conditions/trials in *Group 1* and conditions/trials *Group 2*.

- Test of quadratic effects: Here you compute the combined effect of *Group 1* and *Group 3* against *Group 2*. This option requires at least three conditions/trials.

Run   Execute computation of statistics. The contrasts are stored in your model files and can be reviewed any time using the Review Manager (Chapter 14.15).

14.18   Export Statistics

Related function: `pspm_exp`
    Export first level statistics to a file for further analysis in statistical software, or to the screen.

Model File(s)

Specify file from which to export statistics.

Stats type

Normally, all parameter estimates are exported. For GLM, you can choose to only export the first basis function per condition, or the reconstructed response per condition. For DCM, you can specify contrasts based on conditions as well. This will average within conditions. This argument cannot be used for other models.

- Parameter: Export all parameter estimates.

- Condition: Export all conditions in a GLM, automatically detects number of basis functions and export only the first one (i.e. without derivatives), or export condition averages in DCM.

- Reconstructed: Export all conditions in a GLM, reconstructs estimated response from all basis functions and export the peak of the estimated response.

Exclude conditions with too many NaN

Exclude statistics from conditions with too many NaN values. This option can only be used for GLM file for which the corresponding option was used during model setup. Otherwise this argument is ignored.

Target

Export to screen or to file?

Delimiter for Output File

Select a delimiter for the output file. Default is 'tab', you can select from a choice of several options or specify as free text.

15   Second level

15.1   Define Second-Level Model

Related function: `pspm_con2`

Define one-sample and two-sample t-tests on the between-subject (second) level. A one-sample t-test is normally used to test within-subject contrasts and is equivalent to t-contrasts in an ANOVA model A two-sample t-test is required for between-subjects contrasts or interactions. This module sets up the model but does not report results.

Test Type

Specify the test type.

One Sample T-Test

- Model File(s)

Two Sample T-Test

- Model File(s) 1: Model files for group 1.

- Model File(s) 2: Model files for group 2.

Output directory

Specify directory where the mat file with the resulting model will be written.

Filename for Output Model

Specify name for the resulting model.

Define Contrasts

Define which contrasts to use from the model files, and the second level contrast names. Names can be read from the first model file, or just be numbered.

Read from First Model File

- All Contrasts

- Contrast Vector: Index of first-level contrasts to use.

Number Contrasts

- All Contrasts

- Contrast Vector: Index of first-level contrasts to use.

Overwrite Existing File

Specify whether you want to overwrite existing mat file.

15.2   Report Second-Level Results

Related function: `pspm_rev2`
   Result reporting for second level model.

Model File

Specify 2nd level model file.

Contrast Vector (only accessible from Batch Editor)

Index of contrasts to be reported (optional).

16   Tools

16.1   Display Data

Related function: `pspm_display`
   Display PsPM data file in a new figure.

Data File

Specify data file to display.

16.2   Data editor

Related function: `pspm_data_editor`
   The data editor allows you to visualize data time series and mark epochs, for example to mark artefacts that should be excluded, for example to be used as missing epochs in GLM (see 14.1).

Startup

The Data edtior can be started from the PsPM window, the Matlabbatch and from the Matlab command line. The command line startup requires the PsPM folder to be added to the Matlab path. Once that is done, the data editor can be started by typing 'pspm_data_editor(argument)' in the Matlab command line. The command accepts two types of arguments. (i) a path to file, which causes the editor to open the specified file and to work in the '**file mode**' or (ii) a data vector which causes the editor to plot the

passed data vector and to work in the '**inline mode**'. If no argument is specified, the editor is stared in 'file mode'.

Data modes

The data editor differs between two data modes. The data mode depends on the specified startup argument.

File mode Is entered when called from the PsPM window, the Matlab-batch or a file path or no argument is speciefied at command line startup. File input/output settings aswell as the Channel view are visible on the left. The latter can be used to select the plotted channel. Multiple selection is possible and the color of the plot corresponds to the color in the channel list. File input settings can be used to change the file to be edited, while the output settings specify where the edited data should be stored to. It is also possible to specify the output file from when calling the data editor using Matlabbatch or the argument options.output_file from the Matlab command line.

Inline mode Is entered when a data vector is specified as command line startup argument. Channel view and file input/output settings are hidden. Once the user exits, the editor returns either the interpolated data or the selected epochs in a PsPM epoch format.

Data navigation

Data navigation can be done by zooming (🔍➕ 🔍➖) and panning (✋). Zooming does only change the horizontal axis. After zooming axis limits remain fixed when changing the selected channel. This sometimes may cause data not to be plotted because the data points are out of the visible range. Zooming out will make the data appear again. Panning allows the rearrangement of the plot horizontally and vertically. After panning the axis limits remain fixed as well and data may disappear when changing the selected channel.

Epochs

Add epochs New epochs can be added by clicking the 'add epoch' button (⊢✛). Once clicked the cursor will change to a crosshair and it is possible to select data in the plot, which will upon release create a new epoch in the selected range. The new epoch will appear on the right side in the epoch list and is marked in green on the plot.

Remove epochs   Existing epochs can be removed by clicking the 'remove epoch' button ( ). Once clicked the cursor will change to a crosshair and it is possible to select the range in which the epochs should be removed. If an epoch is selected partially, the rest of the epoch will remain. The epoch-list on the right side will be updated automatically.

Epoch navigation   In order to navigate between epochs it is either possible to use the epoch list on the right side or the two epoch navigation buttons ( ). The selected epoch always corresponds to the selected epoch in the epoch-list and is marked in black in the plot.

Output format

There are two possible output formats. The output, depending on the data mode, will either be written to a specified output file (file mode) or will be returned as a data vector (first argument; inline mode).

Epochs   If epochs is selected, the output will be a n x 2 vector, where n is the number of epochs. The first column marks the onset of the epoch, while the second column marks the offset of the epoch. The time unit is seconds. In file mode the data will be written into the specified mat file in the variable 'epochs'.

Interpolated data   If interpolated data is selected, the output will be the specified data itself with the selected channels (in Channel view) interpolated during the selected epochs. In file mode the original channels will be replaced. If 'Display interpolated curve' is checked the interpolated data will be displayed in the plot as a dashed line.

Matlabbatch

Possible options when the Data editor is started from Matlabbatch.

Data file   Specify the PsPM datafile to be edited.

16.3   Rename File

Related function: `pspm_ren`

Rename PsPM data file. This renames the file and updates the file information.

File

Choose how many files to rename.

File Name   Choose name of original file.

New File Name   Choose new file name.

16.4   Split Sessions

Related function: `pspm_split_sessions`, `pspm_trim`

Split sessions, defined by trains of markers. This function is most commonly used to split fMRI sessions when a (slice or volume) pulse from the MRI scanner has been recorded. In automatic mode, the function will identify trains of markers and detect breaks in these marker sequences. In manual model, you can provide a vector of markers that are used to split the file. The individual sessions will be written to new files with a suffix "_sn", and the session number. You can choose one datafile and, optionally, one missing epochs file, which will be split at the same points. By default, the function will use the first marker channel. Alternatively, you can choose a marker channel number.

Data File

Choose the data file, in which you want to split sessions. The data file should be a single file, and a cell array is not allowed.

Marker Channel

If you have more than one marker channel, choose the marker channel used for splitting sessions (default: use first marker channel).

- Default

- Number

Split behaviour

Choose whether sessions should be detected automatically or if sessions should be split according to given marker id's.

- Automatic: Detect sessions according to longest distances between markers.

- Marker: Split sessions according to given marker id's.

Missing epoch file

Split sessions can split the corresponding missing epoch file of the data file if defined. Up to one missing epoch file can be added.

Overwrite Existing File

Choose "yes" if you want to overwrite existing file(s) with the same name.

16.5 Merge files

Related function: `pspm_merge`

Allows to merge two files by stacking channels. Multiple pairs of files are allowed and are processed in a sequential manner. Which means the first element of the second file set is merged into the first element of first file set and so on. Therefore first file se and second file set must have the same number of elements. Within each pair of files, they are aligned according to the first event of a given marker channel, or to the start of the file. The output file consists of an 'm' at the beginning and the name of the first file appended.

Datafiles

Specify the PsPM datafiles to be merged.

First file(s)   Specify the first of the two files to be merged. This can be one file or a set of files. The output file will have the name of the first file prepended with an 'm'.

Second file(s)   Specify the second of the two files to be merged. This can be one file or a set of files. This set must have the same length as the set defined as first file(s).

Reference

Specify whether to align the files with respect to the first marker or with respect to the file start.

Options

Marker channel   Specify for each file (first and second) a channel which should be used as marker reference. A 1x2 vector is expected. If equal to 0, the first marker channel is used. Default: [0 0]

Overwrite existing file(s)   Specify whether existing files should be over-written (Yes) or not (No). Default: No

16.6   Artefact Removal

Related function: `pspm_pp, pspm_prepdata`

This module offers a few basic artifact removal functions. Currently, median, and butterworth filters [59] are implemented. The median filter is useful to remove short "spikes" in the data, for example from gradient switching in MRI. The Butterworth filter can be used to get rid of high frequency noise that is not sufficiently filtered away by the filters implemented on-the-fly during first level modelling.

Data File

Choose the data file.

Channel Number

Select the channel to work on.

Filter Type

Currently, median and Butterworth filters are implemented. A median filter is recommended for short spikes, generated for example in MRI scanners by gradient switching. A Butterworth filter is applied in most models; check there to see whether an additional filtering is meaningful.

Median Filter

  • Number of Time Points: Number of time points over which the median is taken.

Butterworth Low pass Filter

  • Cutoff Frequency: Cutoff frequency has to be at least 20Hz.

Overwrite Existing File

Choose "yes" if you want to overwrite existing file(s) with the same name.

16.7   Downsample Data

Related function: `pspm_down`

    This function downsamples individual channels in a PsPM file to a required sampling rate, after applying an anti-aliasing Butterworth filter at the Nyquist frequency. The resulting data will be written to a new .mat file, prepended with 'd', and will contain all channels – also the ones that were not downsampled.

Data File

Name of the data files to be downsampled.

New Frequency

Required sampling frequency.

Channels To Downsample

Vector of channels to downsample, or all channels.

Overwrite Existing File

Choose "yes" if you want to overwrite existing file(s) with the same name.

16.8   Interpolate missing data

Related function: `pspm_interpolate`

    The function interpolates missing values, either for all continuous channels in a specified PsPM data file (file mode), or for selected channel(s) only (channel mode). In file mode, the function writes all channels to a new file with the same name prepended by an 'i'. In channel mode, the function interpolates data and either replaces a channel or writes the data to a new channel.

Data file

Select data files.

Work mode

Specify whether to work on a whole file or just on the specified channel(s). If whole file is specified, the data will be written to a new file with the same name prepended by an 'i'. Otherwise the specified channel(s) will be written to a new channel or replace an existing channel in the same file.

- File mode

  - Overwrite existing file: Choose "yes" if you want to overwrite existing file(s) with the same name (Default: No).

- Channel mode

  - Interpolate channel(s): Specify which channel(s) should be interpolated.

  - Mode: Specify how to behave with the interpolated data.

    * New channel: Always add as a new channel.
    * Replace channel: Replace specified channel(s).

16.9   Extract segments

Related function: `pspm_extract_segments`

This function extracts data segments (e.g., for visual inspection of mean responses per condition).

Mode

Either extract all information from a GLM or DCM model structure/file or define the relevant information manually.

- **Automatically read from GLM or DCM**: Extracts all relevant information from a GLM or DCM model structure/file. To distinguish between conditions in a DCM, trialnames must be specified in the DCM definition (before running it).

  - *GLM or DCM* structure/file: Give a GLM or DCM structure already loaded to MATLAB or give the filepath to a GLM or DCM model.

    * When filepath is given, GLM or DCM structure will be loaded to MATLAB automatically. (From now on called the model)
    * When the model corresponds to GLM, the data used for segment extraction is `model.input.data`. This corresponds to the unfiltered input data passed to GLM model fitting part.
    * When the model corresponds to GLM, duration of each onset must be given in `model.timing.durations` as seconds.
    * When the model corresponds to DCM, the data used for segment extraction is `model.input.scr`.
    * When the model corresponds to DCM, the trials that should belong to the same condition should have the same trial name. By default, DCM generates different trial names for each trial.

If the input trial names are all different, then all the trials are assumed to be in the same case.

- **Manual:** Specify all the settings manually.

    – *Data files*: PsPM files from which data segments should be extracted.

    – *Channel*: Channel in specified data file from which the segments should be extracted.

    – *Conditions*:

        ∗ Condition files: Should be in the format of the conditions defined in a GLM model. Required fields are names, onsets, duration.

        ∗ Enter conditions manually: Specify the conditions that you want to include in your design matrix.

            · Name: Specify the name of the condition.
            · Onsets: Specify a vector of onsets. The length of the vector corresponds to the number of events included in this condition. Onsets have to be indicated in the specified time unit ('seconds', 'samples').
            · Duration: Specify the length of the condition.

- **Raw:** Specify all the settings manually (i.e. choose manual mode) and provide raw data

    – *Data*: Numeric raw data or a cell array of numeric raw data.

    – *Sampling rate*: One sampling rate or an array of sampling rates of the corresponding *Data*.

    – *Conditions*:

        ∗ Condition files: Should be in the format of the conditions defined in a GLM model. Required fields are names, onsets, duration.

        ∗ Enter conditions manually: Specify the conditions that you want to include in your design matrix.

            · Name: Specify the name of the condition.
            · Onsets: Specify a vector of onsets. The length of the vector corresponds to the number of events included in this condition. Onsets have to be indicated in the specified time unit ('seconds', 'samples').
            · Duration: Specify the length of the condition.

Options

Change values of optional settings.

- **Timeunit**: The timeunit in which conditions should be interpreted.

  – Seconds
  – Samples
  – Markers

- **Marker channel**: Channel containing the markers referenced in the conditions. Only needed if option 'Timeunit' is set to 'markers'.

- **Segment length**: Length of segments. If set durations of trials in conditions will be ignored.

- **NaN-output**: Option to visualize the ratio of NaN values for each trial for each condition. Possibilities of visualization:

  – None
  – Screen: a table of the NaN-ratios is printed on the MATLAB command window.
  – File output : a table of the NaN-ratios is saved in a file (Attention: already existing files get overwritten)
    * File name: defines the name of the file
    * Path to File: defines the path where the file should be stored

  The table printed in the option 'Screen' or 'File output' has the conditions as columns and all the trials as rows. For each row there exists only one non-NaN value, which represents the propotion of the NaN values in the correspondig trial for the corresponding condition. The last row indicates the proportion of NaN-values over all trials for each condition.

Output

Output settings.

- **Output file**: Where to store the extracted segments.

  – File path: Path to file.
  – File name: Name of file.

- **Overwrite existing file**: Overwrite existing segment files.

- **Plot**: Plot means over conditions with standard error of the mean.

16.10   Segment mean

This function creates means over extracted data segments (as extracted in 'Extract segments').

Segment files

Specify the segment files which have been created with the 'extract segment' function.

Output file

Where to store the segment mean across the specified files.

File path   Path to file.

File name   Name of file.

Adjust method

How to deal with different sample rates across segment files. 'interpolate' data segments to highest or 'downsample' data segments to lowest sampling rate.

Overwrite existing file

Overwrite existing segment mean files.

Plot

Plot means over segment files with standard error of the mean.

16.11   Extract event marker info

Allows to extract additional marker information for further processing. The information can be used to distinguish between different types of events or for other purposes. This is usually used to extract marker information from EEG-style data files such as BrainVision or NeuroScan.

Data file

Specify the PsPM datafile containing a marker channel with a markerinfo field.

Marker channel

Specify which marker channel should be used for marker info extraction.

Output

File   Specify the output file to which the marker info should be written to.

Overwrite existing file   Overwrite existing marker info files.

17   Convenience functions

This section is about function which do not directly belong to PsPM but may have at some point. There is no support for these functions. They are kept because they might be useful for some tasks or may serve as skeleton for other functions. Convenience functions are located in '<pspm-folder>/backroom'.

- *pspm_axpos*: Creates position vectors for axes() commands, which can be passed along the 'Position' argument.

- *pspm_bf_Fourier*: Fourier response function for GLM.

- *pspm_convert_...*: Conversion functions for

  - Illuminance to luminance
  - Lux to candela
  - mm to visual degree

- *pspm_find_data_epochs*: Find epochs of non-zero values in the given data.

- *pspm_get_transfer_function*: Tries to estimate a SCR transfer function.

- *pspm_ledalab*: Wrapper for Ledalab analysis from within SCRalyze / PsPM.

- *pspm_peakscore*: Calculate event-related responses by scoring the peak response against a pre-stiumulus baseline.

- *pspm_predval*: Computes evidence for a predictive model.

- *pspm_scr2ledalab*: Export SCRalyze / PsPM files to ledalab.

- *pspm_sf_get_theta*: Create parameters for f_SF, given some SCR data.

- *pspm_transfer_fit*: Calculates the mean squared error for pulse rates.

- *SCR_f_amplitude_check*: Check the scaling of SCR/SF amplitudes in functions f_SF and f_SCR.

18 Troubleshooting and known problems

18.1 Path

If the PsPM folder is added to the path with all subfolders, the batch editor will not work. Only put the PsPM main folder (containing the function pspm.m) to the path.

18.2 Diagnostic graphics for non-linear model inversion

During non-linear model inversion, a diagnostic window is displayed for each trial. Between two trials, there is a short period of time during which several tabs appear in that window. If you click on one of the hidden tabs in that moment, model inversion will fail due to a display error.

18.3 Export statistics into text format with tab delimiter

The tab delimiter in the exported files is not correctly interpreted by text editors like MS Word. Nonetheless, the data can be imported into softwares such as Excel or SPSS.

Part III
Tutorial

Contributed by Christoph Korn & Matthias Staib.

19 GLM for SCR tutorial: Appraisal data

Here, we analyze SCR data using a general linear model (GLM). The example data set comprises SCR data from 15 participants and can be downloaded from https://bachlab.github.io/PsPM/software/. Results from this data set have been previously published [9, 10]. Each participant saw 45 neutral and 45 aversive pictures within one session that included two short breaks. SCR data were recorded using a 0.5 V coupler, optical (wave to pulse) transducer, and CED Spike, with a minimum sampling rate of 100 Hz. Be aware that this example uses functions from the MATLAB Signal Processing Toolbox, make sure to have it installed before starting the tutorial.

19.1   Import

We import the SCR data and the corresponding marker data. We first import data from one subject.

- In the batch editor go to *PsPM → Data Preparation → Import*.

- Specify the Data Type. Several data formats are available. The current data set is in the smr format.

- Select the SCR Data File of the first subject `trsp_1_25.smr`.

- Specify the *Channels*. We need one SCR channel and one marker channel.

    – The *Channel Number* of the SCR channel is 2.

    – For the *Transfer Function* choose *None*.

    – For the *Channel Number* of the marker channel put in *Search* to test the automatic search (otherwise you can specify the number of the marker channel manually. In this data set, the marker channel is 6).

See Figure 19.1 for a picture of the final batch editor. We recommend saving the batch and script so that you can easily re-run your analyses at a later time point (see Chapter 21). Run the batch. A mat file with the name `pspm_trsp_1_25.mat` will be created in the same folder as the original data file.

   You can have a look at the imported data by loading this file into MAT-LAB. The cell array data contains two structures. The first one contains the SCR data and the second one the marker data (this data set contains 96 markers).
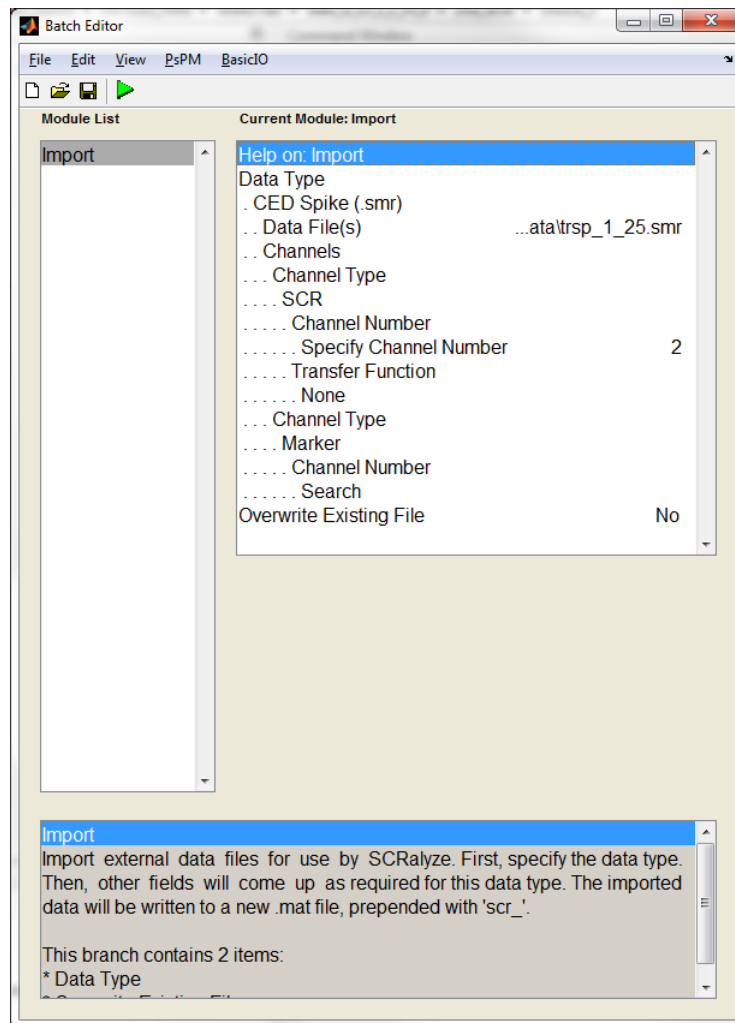
Figure 19.1: Batch editor for importing raw SCR data.

19.2   Trim

Most of the time, the SCR recording is switched on some time before the actual experiment starts and is switched off some time after it ends. These parts of the time series may contain large artifacts from subject motion, setting up the equipment, etc. Trimming excludes these parts from the analyses.
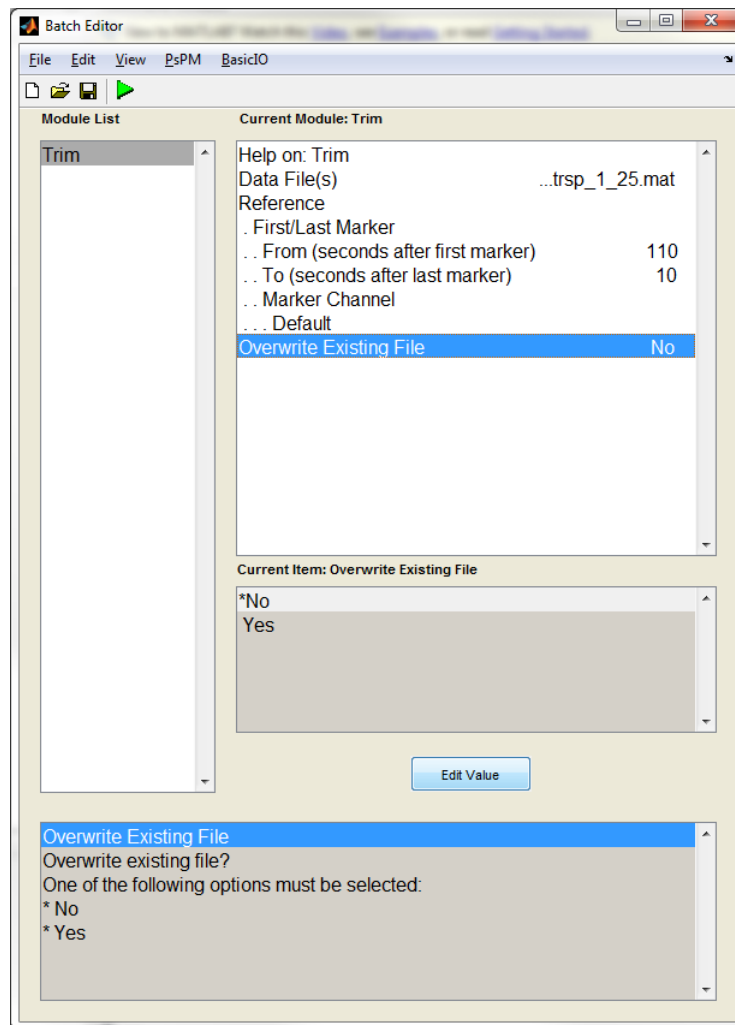
Figure 19.2: Batch editor for trimming SCR data.

- In the batch editor go to *PsPM → Data Preparation →Trim*.

    - Select the currently imported file as Data File. Alternatively, you
      can use the *Dependency* button.

- You can choose different References to specify how data should be
  trimmed. Here, select *First/Last Marker* and

    - *From (seconds after first marker)*: +110 (In the current data set
      the first 2 minutes are a task-relevant baseline period and are
      thus trimmed. Note that a positive number will delete at least the

first marker of the original data. Thus, make sure that a possible specification of the First-Level takes this into account.)

– *To (seconds after first marker)*: 10 (The specified time window should be large enough to completely include the last trial.)

You can also specify time points before the markers by using negative numbers (see above). See Figure 19.2 for a picture of the final batch editor. Again, it is recommended to save the batch and script for future use. Run the batch. A mat file with the name `tpspm_trsp_1_25.mat` will be created (i.e., a "t" will be prepended).

19.3   First-Level

Now, we specify the GLM on the first (i.e., the subject-specific) level, estimating average responses per experimental condition (this equivalent to averaging responses over trials in classical "peak scoring" analysis, or to performing a first level analysis in SPM).

- In the batch editor go to *PsPM → First Level → SCR →GLM for SCR*.

- *Model Filename*: This is the name of the resulting mat file containing the GLM. Here, we call it `glm_25`.

- *Output Directory*: The resulting mat file containing the GLM will be saved in this folder.

- *Time Units* on which the specification of the conditions will be based. Here, we want to analyze with respect to Markers.

- *Session(s)*: Similar to SPM, SCRalyze allows the specification of multiple sessions per subject. Here, we only have one session.

  – Add the trimmed data (`tpspm_trsp_1_25.mat`) as Data File. Alternatively, you can use the *Dependency* button.

  – *Missing Epochs* allows the specification of time periods which you do not want to include in the analysis for example because they contain corrupted data. In the example data set, there are no missing epochs.

  – *Data & Design*: This option allows specifying the different conditions of the experiment. You need to provide the information when the different conditions started. This time information has to be in the unit specified above (see *Time Units*). In the current data set, condition information is provided in Markers and is stored in the mat file `trSP_appraisal_01_25_onsets_1.mat`, which you should select as *Condition File*. When you open this file

in MATLAB, you will see that the file contains a 1x3 cell "names" and a 1x3 cell "onsets" specifying three conditions. The numbers in "onsets" correspond to the nth marker when the respective trial started.

– *Nuisance File* allows the inclusion of nuisance confounds such as motion or respiration parameters. Here, we leave it empty.
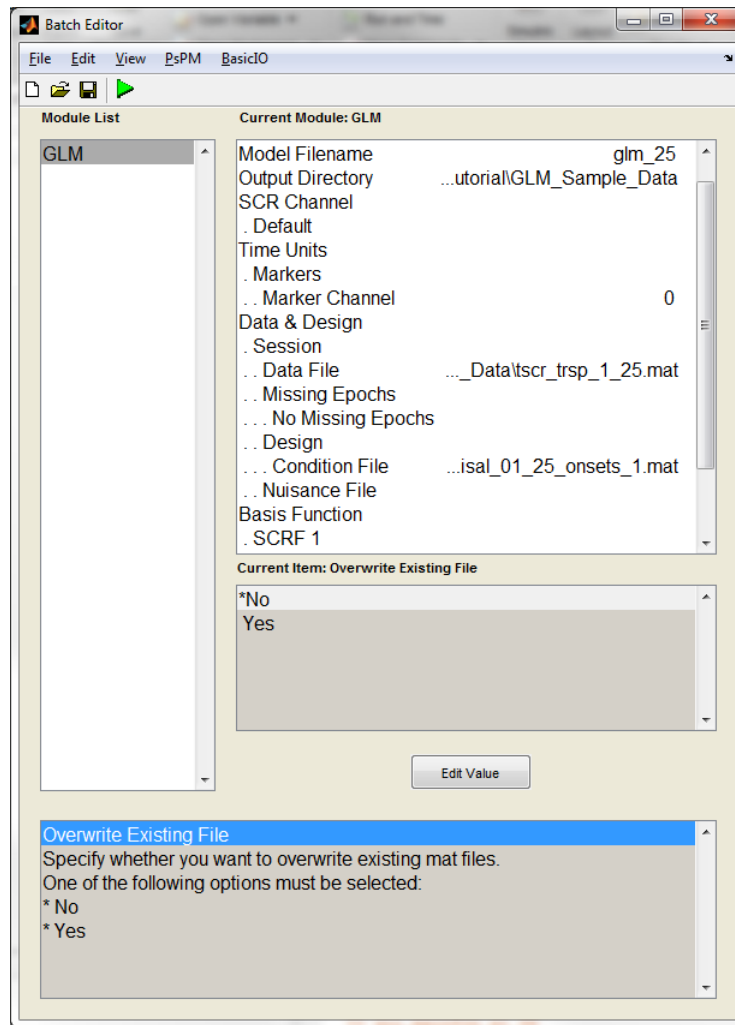

Figure 19.3: Batch editor for First-Level analysis.

• *Basis Function*: We chose the default *SCRF 1*, which is the version with time derivative. The default basis set follows the recommenda-

tions by [9].

- *Normalize*: The design of the current data set is "within-subjects." Therefore, we want to z-normalize the data.

- *Filter Settings*: We choose the default filter settings, which follow the recommendations by [9]:

    - first-order low-pass Butterworth filter with a cutoff frequency of 5 Hz

    - first-order high-pass Butterworth filter with a cutoff frequency of 0.05 Hz

    - Data are resampled to 10 Hz.

    - The filter direction is unidirectional.

See Figure 19.3 for a picture of the final batch editor. Again, we recommend saving the batch and script for future use. Run the batch. A mat file with the specified name will be created in the specified folder.

19.4   Review First-Level Model

You can have a look at plots related to the first-level model in the batch editor.

- In the batch editor go to *PsPM → First Level → Review First-level Model*.

- Specify the Model file that you want to review. Select the mat file containing the GLM (e.g., `glm_25.mat`).

- The *Model Type* is of course GLM in the current example. You can set the following options to get different types of information:

    1. *Design matrix*: On the x-axis, you see the number of regressors in the model. Here we have specified 7 regressors in total. On the y-axis, you see the time course of the experiment. Note that there are 7 regressors because the specification for the *Basis Function* was *SCRF with time derivative*. That is, 3 (conditions times) x 2 (basis functions) + 1 (constant) = 7 regressors. You can see that neutral and aversive are intermixed and fall into three blocks. See Figure 19.4.

    2. *Orthogonality*: On the x- and y-axes you see the specified regressors. Regressor names are additionally specified along the y-axes. The plot depicts the collinearity between regressors. See Figure 19.5.

3. *Predicted & Observed*: You see the model fit including the observed response over the whole time series, which you can visually compare to the predicted time series by the specified model. Note that the predicted data will never follow the observed data very closely, because we estimated the average response per condition which will deviate from the response in each individual trial. See Figure 19.6.

4. *Regressor names*: The names of the regressors as specified in the *Conditions File* under *Data & Design* are written to the MATLAB command window. The labels bf 1 and bf 2 refer to the two used basis functions (*SCRF with time derivative*).

5. *Reconstructed*: Here you see the estimated responses per condition for the first basis function. These are estimates of the true responses which contain noise. The noise term can be negative but if all estimated responses across an entire group are negative it is a good idea to look for confounds or correlations in the design. See Figure 19.7.
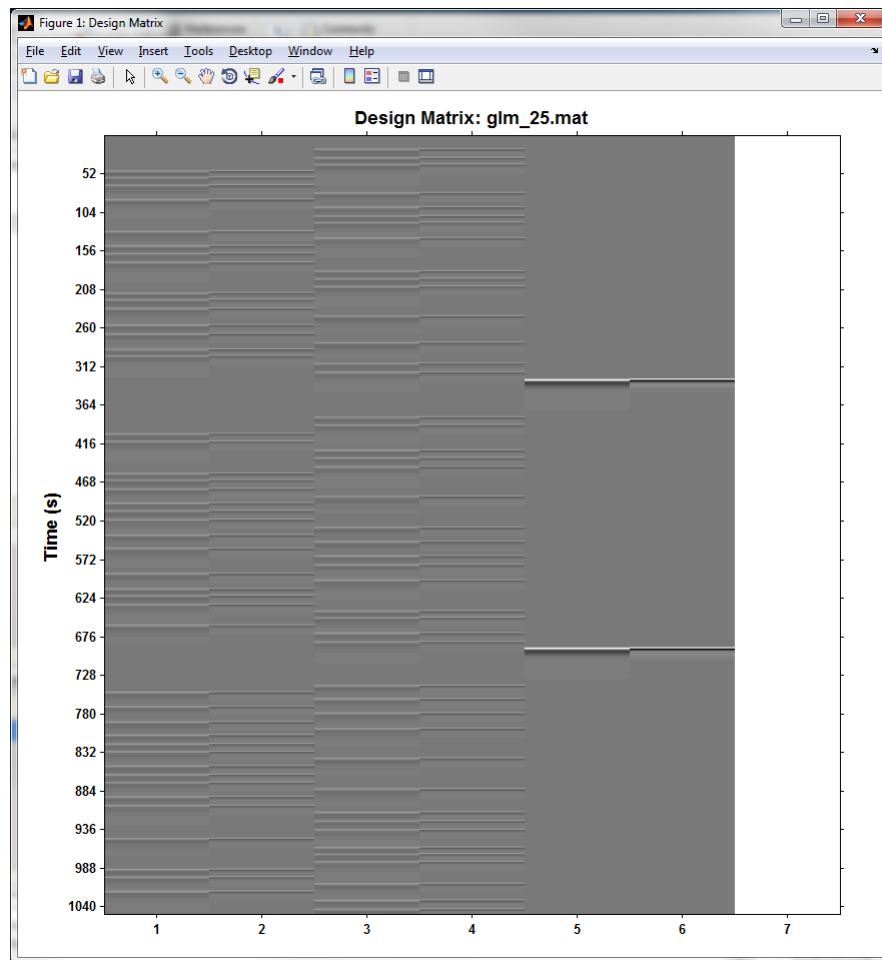
Figure 19.4: Design matrix for sample participant. On the x-axis, you see the number of regressors in the model. There are 7 regressors (i.e., 3 (conditions times) x 2 (basis functions) + 1 (constant)).
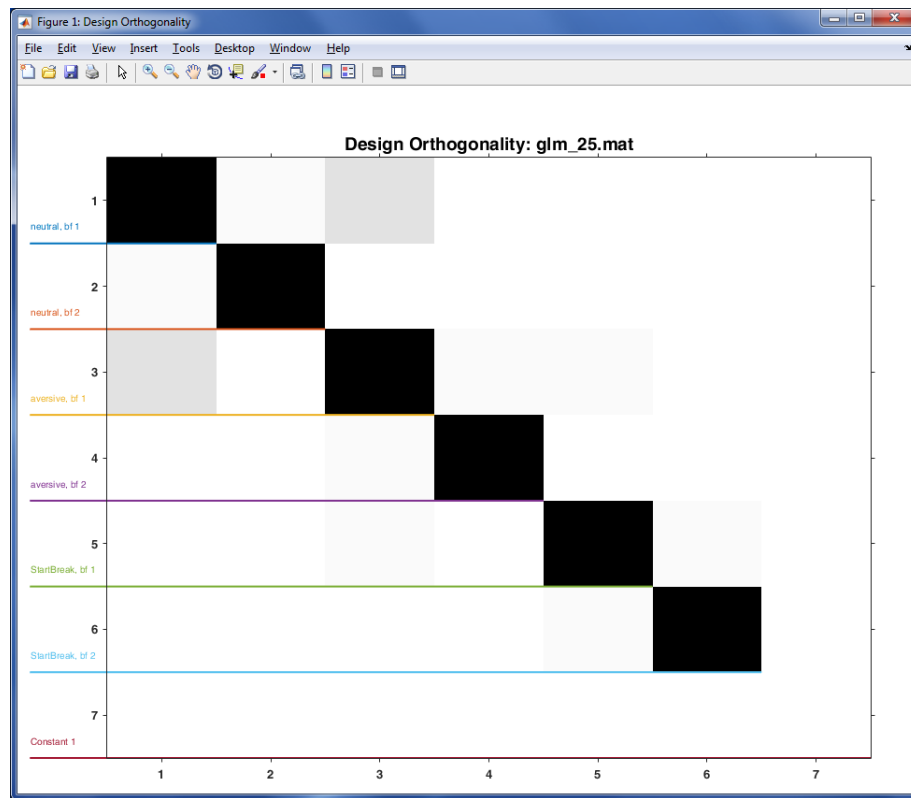
Figure 19.5: Design orthogonality plot for sample participant. On the x- and y-axes you see the 7 specified regressors.
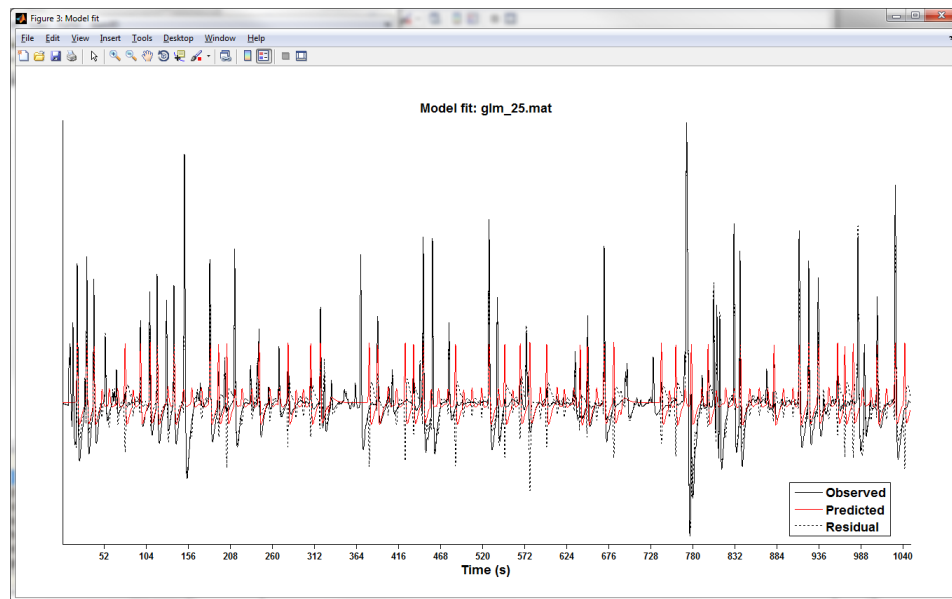
Figure 19.6: Model fit for sample participant. You see the observed response and the predicted response according to the model over the whole time series.
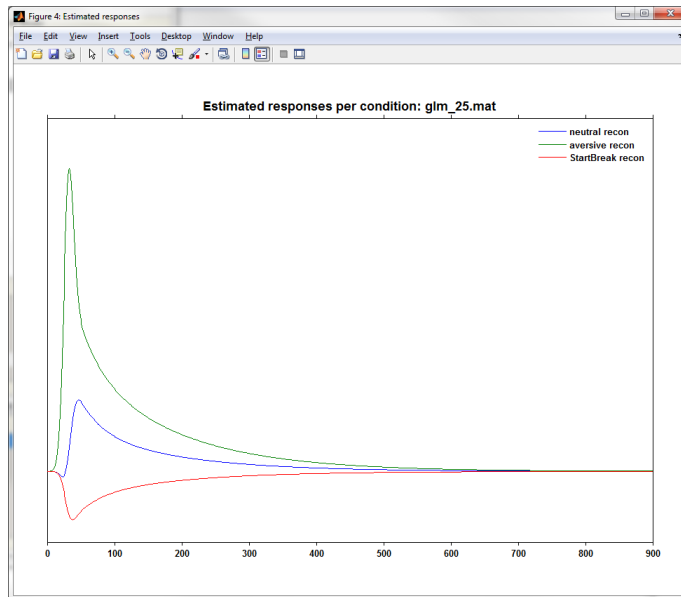
Figure 19.7: Estimated responses per condition for sample participant. You see the estimated responses per condition for the first basis function.

19.5   First-Level Contrast Manager

We want to compare the difference between aversive and neutral pictures across participants (i.e., on the second level). This can be tested in a paired t-test. A paired t-test is a one-sample t-test on difference values between two conditions. The contrast manager allows you to compute such difference values, for each participant (i.e., on the first level).

- In the batch editor go to *PsPM* → *First-Level* → *First-Level Contrasts*.

- *Model File(s)*: Select the mat file containing the GLM (here `glm_25.mat`).

- *Stats type*: You can choose between 3 different options to set up the contrasts. Here, we choose the option Condition, which specifies contrasts formulated in terms of conditions and automatically uses only the first basis function (i.e., without derivatives).

- *Contrast*

  - *Contrast Name*: Chose a name so that you can easily refer to the contrast (e.g., aversive>neutral)
  - *Contrast Vector*: Specify a vector of the contrast you want to test for. Here, we are interested in the difference between aversive and neutral pictures, i.e., the first and the second condition.

Therefore, specify the vector [-1, 1, 0]. You could also leave out the last zero because trailing zeros are automatically appended. (But you could not leave out the first zero in a contrast such as [0 -1 1].)

19.6   Export Statistics

You can use this functionality to export all necessary parameters on the screen or into an extra file, which allows you to analyze the data in the statistics program of your choice. For details see *PsPM → First Level → Export Statistics*.

19.7   Adapting scripts for multiple participants and second-level analyses

For didactic purposes, we have so far only considered one participant. But in most experiments, you want to analyze data from multiple subjects (i.e., a second-level analysis). For a description on how to do this see Chapters 21 and 22. You can find a script that runs the above described analyses for multiple participants at pspm.sourceforge.net (import_trim_glm_contr_job_loop).

20   Non-linear SCR tutorial: Delay fear conditioning data

The example data set comprises SCR data from 20 participants, with 40 trials each, which can be downloaded from https://bachlab.github.io/PsPM/software/ together with the result files of the model inversion. Participants underwent a differential delay fear conditioning paradigm where coloured circles were presented for 4 seconds each. Either the blue or orange coloured circles (balanced over participants) were probabilistically paired with an electric stimulation (unconditioned stimulus, US) that coterminated with the stimulus (conditioned stimulus, CS+) while the CS- were always presented alone.

Presenting the CS+ causes phasic sympathetic arousal in anticipation of an electric shock. This anticipatory reaction typically occurs with an unknown and variable latency after stimulus onset. For such cases, a GLM should not be used because it assumes that sympathetic arousal follows the stimulus with a fixed latency. In contrast, the non-linear model allows estimating (i) a variable onset within a user-specified time window and (ii) a variable duration of a sympathetic response. Additionally, the non-linear model includes estimation of spontaneous fluctuations that occur between trials.

In this tutorial, the non-linear model is estimated. The estimated parameters can then be used to test the within-subject hypothesis that the sympathetic arousal between CS+ and CS- are different on a group level.

20.1 Setup the Non-Linear Model

In this step you prepare the model estimation. For this, you need to provide timing information about the experimental events which can either be entered manually or as a Matlab file. For large numbers of events as in this experiment, a timing file can easily be created using a few lines of Matlab code. Here we use timing information saved in the trimmed PsPM file. In this file, a trigger was saved at every CS onset. For timing information from other sources, the code has to be adapted accordingly. Go to the Matlab command window and enter the following lines, but replace `[insert path!]` by the correct path where the filtered SCR data from pspm.sourceforge.net is saved on your hard drive, e.g. `C:\PsPM_Tutorial\nonlinear\`.

```matlab
data_path = [insert path!]; % set the correct PsPM file path here!


p = 1; % the first participant
SOA = 3.5; % delay between CS and US onset in seconds

% load events of CS onsets from trimmed scr data
pname = fullfile(data_path,sprintf('tpspm_s%i.mat',p))
[sts, infos, data, filestruct] = pspm_load_data(pname,'events');

CS_onset = data{1, 1}.data; % recorded triggers of CS onset in seconds
US_onset = data{1, 1}.data + SOA; % US starts 3.5 seconds after CS

% variable latency for CS response between CS onset and US onset/omission
events{1} = [CS_onset US_onset]; % define an interval of 3.5 seconds for each trial
% constant onset for US response
events{2} = US_onset;

save(fullfile(data_path,sprintf('event_timings_s%i.mat',p)),'events')
```

The resulting file `event_timings_s1.mat` contains the variable events with two cells.

- The first cell *events{1}* is a 40 x 2 matrix with the onset and offset of a time window for each trial in which we expect the response to the CS (both CS+ and CS- are treated the same here). We expect the response to the CS to occur with a variable onset in time. Note that the window for the CS response ends when the US starts (or is omitted), i.e. after 3.5 seconds.

- The second cell *events{2}* is a 40 x 1 vector that marks the onset of the electrical stimulation. For this event we expect an evoked response with constant latency after US occurrence. Importantly, we provide

timings *both for the occurrence and the omission of the US*. Modelling an unconditioned response in all trials is necessary to get unbiased estimates for the CS responses. In general, the trial structure should always be the same for all trials, across experimental conditions.

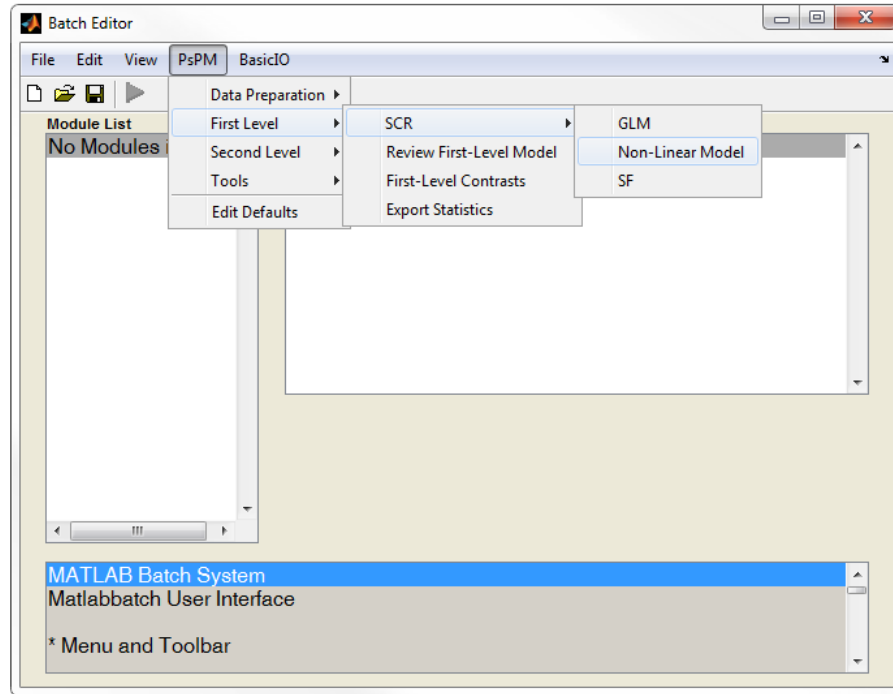Now the non-linear model estimation can be set up (Figure 20.1).

Figure 20.1: Prepare a batch to set up a non linear model for SCR analysis.

- In the batch editor go to *PsPM → First Level → SCR → Non-Linear Model*

- *Model Filename*: dcm_s1

- *Output Directory*: Specify a folder where the model will be saved

- *Chan*nel: Default

- *Data & Design:→New: Session*

  - *Data File*: choose tpspm_s1.mat from your hard drive
  - *Design→Timing File→*Select the previously created `event_timings_s1.mat` which was saved to your scr folder

– *Condition names: New Condition* here we can define conditions
that each trial belongs to. Create three new conditions. Note that
we split CS+ trials in reinforced (CS+|US+) and non-reinforced
(CS+|US-) trials because CS+|US+ trials are excluded from some
analyses.

* *Condition*
  · *Name: CS+|US-*
  · *Index: 6 7 11 14 16 18 26 29 38 40*
* *Condition*
  · *Name: CS+|US+*
  · *Index: 13 17 19 20 21 24 28 31 32 37*
* *Condition*
  · *Name: CS-*
  · *Index: 1 2 3 4 5 8 9 10 12 15 22 23 25 27 30 33 34 35 36
    39*

• *Display Options*

– *Display Progress Window*: If you plan to run the estimation in
the background, you can turn off the progress window with this
option

• *For the remaining fields you can keep the default settings*

Save the batch and script as `batch_model_s1.m` and press run. A window
will be displayed that shows the progress of model estimation for each trial.
The estimation for an entire session takes several minutes depending on
your system. The file `dcm_s1.mat` will be created. If you want to, you can re-
view the first-level model by clicking *Review model* in the PsPM user inter-
face. In the new window (Figure 20.2) add the non-linear model `dcm_s1.mat`
and choose *Display All trials for one session*. A graphics window (Fig X) will
display the original data (blue) together with the model estimation (green).
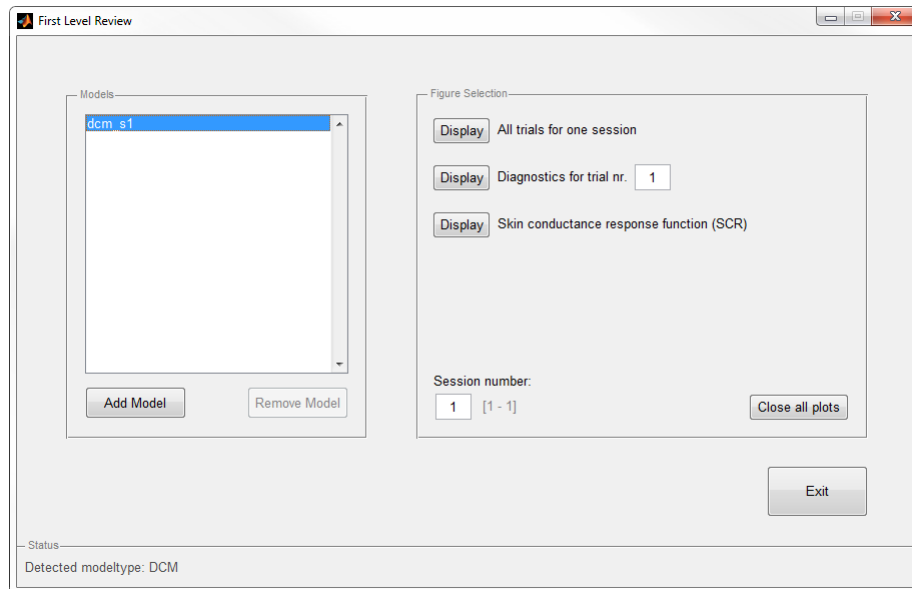
Figure 20.2: Use *Review Model* to inspect parameter estimation for one or multiple trials.

20.2   First-level Contrasts

In this step you specify the hypothesis that a CS+ elicits a stronger sympathetic arousal than a CS- for an individual participant by providing a contrast weight vector.

- In the batch editor go to *PsPM→First Level→First-level Contrasts*

- *Model File(s):* Select dcm_s1.mat

- *Specify Contrasts on Datatype: param*

- *Contrasts: New Contrast*

  – *Contrast*

    * *Name: Fear Learning*
    * *Contrast Vector: -1 -1 -1 -1 -1 2 2 -1 -1 -1 2 -1 0 2 -1 2 0 2 0 0 0 -1 -1 0 -1 2 -1 0 2 -1 0 0 -1 -1 -1 -1 0 2 -1 2* Note: A weight of +2 is assigned to the CS+|US- because we have twice as many CS- and the contrast vector has to sum up to zero.
    * *Name: Electric Stimulation* We include this contrast optionally to check if the electrical stimulation caused an increase in sympathetic arousal which is a premise for successful fear conditioning.

∗ *Contrast Vector: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 -1 -1 3 -1 3 3 3 -1 -1 3 -1 -1 -1 3 -1 -1 3 3 -1 -1 -1 -1 3 -1 -1 -1*

- *Delete: existing contrasts: No (default)*

Save the batch and script as `batch_contrast_s1.m` and press run. The model file of participant s1 now contains the contrast that can be forwarded to a statistical test on group level. Preparing analyses for more than one scr file is explained in Chapter 21.

20.3   Exporting statistics

PsPM provides basic tools for statistical analysis on the estimates of the non-linear model. Alternatively you can export the results as a text file to use them in different software applications.

- In the batch editor go to SCR→First Level→Export Data

- *Model File(s):* Select *dcm_s1.mat*

- *Specify Export on Datatype: param*

- *Target*

    – *Filename* Enter `dcm_s01`

- *Specify Delimiter for Output File*

    – *Semicolon*

The text file will be saved to Matlab's current working directory.

21   Adapting scripts for multiple participants

In most experiments, you want to analyze data from multiple subjects, which implies that you have to repeat model estimation several times. There are two simple strategies to make this easier.

- Use dependencies: You can specify all steps for one subject in one go. At some points, you need to specify a file from a previous step (e.g., when trimming you need the file containing the imported data). In these cases, you can use the button "Dependency" to specify the corresponding file, but make sure the number of output files of the preceding module corresponds with the allowed number of input files for this module.

- Loop over participants in the script: You can easily adapt the scripts for many subjects by including a loop and adapting the file names. This is independent of whether or not you use dependencies. To show you how this works, see the examples below.

21.1 Importing and trimming data

Here, we create an example script for importing and trimming data. We chose these steps since they are required in all types of analysis. The current example is based on the Appraisal dataset described in Chapter 19.

- In the batch editor specify all parameters in PsPM → Data Preparation → Import.

- You do not need to specify the SCR Data File. This will later be put in the script that loops over subjects.

- In the batch editor specify all parameters in PsPM → Data Preparation → Trim.

- Use the dependency button to select the imported Data File.

- Save batch and script.

- Open the saved job script (e.g., `batch_import_trim_job.m`) and build a loop around the lines of code that were saved. In this loop, create a subject-specific filename (and make sure to include the whole path to the files in the variable data_path).

```
data_path = [insert path!];

for p = 25:39; % participants
% load participant-specific raw data
spike.datafile = {fullfile(data_path,sprintf('trsp_1_%i.smr',p))};
spike.importtype{1}.scr.chan_nr.chan_nr_spec = 2;
spike.importtype{1}.scr.transfer.none = true;
spike.importtype{2}.marker.chan_nr.chan_nr_spec = 0;
matlabbatch{1}.pspm{1}.prep{1}.import.datatype.spike = spike;
matlabbatch{1}.pspm{1}.prep{1}.import.overwrite = false;
matlabbatch{2}.pspm{1}.prep{1}.trim.datafile(1) = ...
cfg_dep('Import: Output File', substruct('.', ...
'val', '{}',{1}, '.','val', ...
'{}',{1}, '.','val', '{}',{1}), substruct('()',{':'}));
matlabbatch{2}.pspm{1}.prep{1}.trim.ref.ref_mrk.from = 110;
matlabbatch{2}.pspm{1}.prep{1}.trim.ref.ref_mrk.to = 10;
matlabbatch{2}.pspm{1}.prep{1}.trim.ref.ref_mrk.mrk_chan.chan_def = 0;
matlabbatch{2}.pspm{1}.prep{1}.trim.overwrite = false;

% run batch
pspm_jobman('initcfg');
```

```matlab
pspm_jobman('run', matlabbatch);

end
```

Run the script. The folder data_path now contains trimmed scr files for all subjects.

21.2   Inverting non-linear models

Here we adapt the model inversion from Chapter 20 to multiple subjects. We use similar loops as above to create timing files and PsPM batches.

First, open the MATLAB script that created the timing file event_timings_s1.mat and construct a loop around it for subjects 2 to 20. Execute the script to obtain files containing timing information for subjects 2 to 20. Alternatively you can use the timing files for subjects 2 to 20 downloaded from pspm.sourceforge.net.

Next, open the saved job script (e.g., batch_model_s1_job.m) created in Chapter 20 and build a loop around the lines of code that were saved. Make sure to include the whole path to the files in the variable data_path. You can use the following code as guideline.

```matlab
% set the path where you saved the scr data files
data_path = [insert path!];

% set the path where you saved the model file of participant 1
model_path = [insert path!]; % initiate PsPM pspm_init % loop through participants 2 to 20

% initialize PsPM
pspm_init
pspm_jobman('initcfg')

for p = 2:20

clear matlabbatch dcm
dcm.modelfile = ['dcm_s' num2str(p)];
dcm.outdir = {model_path};
dcm.chan.chan_def = 0;
dcm.session.datafile = ...
{fullfile(data_path,['tpspm_s' num2str(p) '.mat'])};
dcm.session.timing.timingfile = ...
{fullfile(data_path,['event_timings_s' num2str(p) '.mat'])};
dcm.session.condition(1).name = 'CS+|US-';
dcm.session.condition(1).index = [6 7 11 14 16 18 26 29 38 40];
dcm.session.condition(2).name = 'CS+|US+';
dcm.session.condition(2).index = [13 17 19 20 21 24 28 31 32 37];
dcm.session.condition(3).name = 'CS-';
dcm.session.condition(3).index = ...
[1 2 3 4 5 8 9 10 12 15 22 23 25 27 30 33 34 35 36 39];
dcm.data_options.norm = 0;
dcm.data_options.filter.def = 0;
```

```
dcm.resp_options.crfupdate = 0;
dcm.resp_options.indrf = 0;
dcm.resp_options.getrf = 0;
dcm.resp_options.rf = 0;
dcm.inv_options.depth = 2;
dcm.inv_options.sfpre = 2;
dcm.inv_options.sfpost = 5;
dcm.inv_options.sffreq = 0.5;
dcm.inv_options.sclpre = 2;
dcm.inv_options.sclpost = 5;
dcm.inv_options.ascr_sigma_offset = 0.1;
dcm.disp_options.dispwin = 0;
dcm.disp_options.dispsmallwin = 0;

matlabbatch{1}.pspm{1}.first_level{1}.scr{1}.dcm = dcm;

% run batch
pspm_jobman('run', matlabbatch);

end
```

Similarly to the previous step, the batch of the first-level Contrast of participant 1 is adapted to participants 2 to 20. Run the following code in MATLAB, but replace [insert path!] with the path on your hard drive where you saved dcm_s1.mat:

```
% set the path where you saved the estimated non-linear model
model_path = [insert path!];

% initialize PsPM
pspm_init
pspm_jobman('initcfg')

% loop through participants 2 to 20
for p = 2:20

clear matlabbatch
matlabbatch{1}.pspm{1}.first_level{1}.contrast.modelfile = ...
{fullfile(model_path,['dcm_s' num2str(p) '.mat'])};
matlabbatch{1}.pspm{1}.first_level{1}.contrast.datatype = 'param';
matlabbatch{1}.pspm{1}.first_level{1}.contrast.con(1).conname = 'Fear Learning';
matlabbatch{1}.pspm{1}.first_level{1}.contrast.con(1).convec = ...
[-1 -1 -1 -1 -1 2 2 -1 -1 -1 2 -1 0 2 -1 2 0 2 ...
0 0 0 -1 -1 0 -1 2 -1 0 2 -1 0 0 -1 -1 -1 -1 0 2 -1 2];
matlabbatch{1}.pspm{1}.first_level{1}.contrast.con(2).conname = ...
'Electric Stimulation';
matlabbatch{1}.pspm{1}.first_level{1}.contrast.con(2).convec = ...
[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 -1 -1 ...
3 -1 3 3 3 -1 -1 3 -1 -1 -1 3 -1 -1 3 3 -1 -1 -1 -1 3 -1 -1 -1];
matlabbatch{1}.pspm{1}.first_level{1}.contrast.deletecon = true;

% run batch manually
pspm_jobman('run', matlabbatch);
```

end

Run the script. The statistics for each contrast are now added to the model file.

22 Second-level Contrast Manager

In most cases, we want to compare the difference between conditions across participants on a group level (i.e., the second level in a hierarchical model). Each participant represents one data point that goes into a one sample t-test.

Below we demonstrate how to set-up Second-level contrasts for GLM and non-linear models.

22.1 GLM

For the appraisal data of Chapter 19.

22.1.1 Define Second-Level Model

- In the batch editor go to *PsPM→Second Level→Define Second-Level Model*.

  – *Test Type: One Sample T-Test*
  – *Model File(s)*: Select the model files from all participants
  – *Output Directory:* Choose a location to save the group result
  – *Filename for Output Model*: Chose a name so that you can easily refer to the contrast
  – *Define Contrast: Read from First Model File*: Choose *All Contrasts.* This will select the contrast(s) that we created in the first level (i.e., in our case aversive>neutral)

22.1.2 Report Second-Level Results

- In the batch editor go to *PsPM →Second Level→Report Second-Level Results*.

  – *Model File*: Select the second level model file that you created in the previous step.
  – *Contrast Vector*: Leave empty because we want all contrasts to be reported.

Run the batch. A figure appears that shows you a bar graph of the mean of the parameter estimates and the statistics will be printed to the MATLAB command window.

22.2   Non-linear models

Here, we test the difference in anticipatory sympathetic arousal elicited by a CS+ vs CS- on a group level (see Chapter 20). As a supplementary hypothesis we want to check if the US+ indeed elicited a stronger response than the US-. From the 1st level contrast, eight statistics were generated for each model file. Use the *Review Model* utility from the PsPM main GUI to get a list of all contrasts. In the *First Level Review Manager*, click on *Add Model*, choose any model file created previously and press *Done*. Now press the Show button next to *Contrast names in command window* (Figure 22.1) to get a list of all contrasts:

- *Contrast 1: Fear Learning - Flexible response # 1: amplitude*

- *Contrast 2: Electric Stimulation - Flexible response # 1: amplitude*

- *Contrast 3: Fear Learning - Flexible response # 1: peak latency*

- *Contrast 4: Electric Stimulation - Flexible response # 1: peak latency*

- *Contrast 5: Fear Learning - Flexible response # 1: dispersion*

- *Contrast 6: Electric Stimulation - Flexible response # 1: dispersion*

- *Contrast 7: Fear Learning - Fixed response # 1 response amplitude*

- *Contrast 8: Electric Stimulation - Fixed response # 1 response amplitude*

Our hypothesis is concerned only with the response amplitude of the flexible event (anticipation of the CS) and the response amplitude of the fixed event (electrical stimulation), which are the first and eighth contrasts. The indices of these two contrasts are used for the setup of the Second-Level model below.
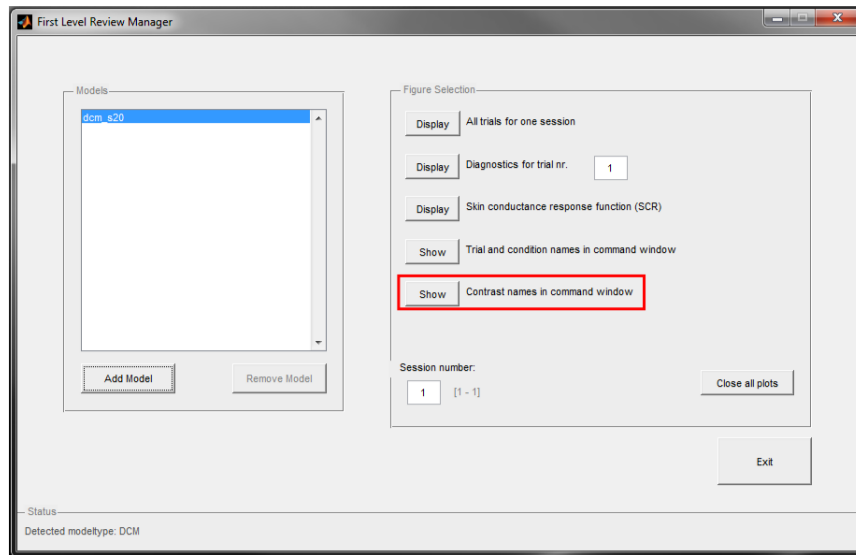
Figure 22.1: The First Level Review Manager displays all contrasts that are stored in a model file.

- In the batch editor go to PsPM→Second Level→Define Second-Level Model

- Test Type: One Sample T-Test

- Model File(s): Select the model files from all participants

- Output Directory: Choose a location to save the group result

- Filename for Output Model: Fear_Learning

- Define Contrast: Read from First Model File

    - Contrast Vector: [1, 8]

- Overwrite Existing File: No (default)

Run the batch. You can display the results of the second level contrasts by pressing *Report 2nd level result* in the PsPM user interface and choosing Fear_Learning.mat. A Figure displays the mean differences and standard error for the hypothesis that the CS+ elicits a stronger sympathetic arousal than CS- and that an electric stimulation results in an increase in sympathetic arousal (Figure 22.2). Statistical parameters are printed in the MATLAB command window (Figure 22.3).
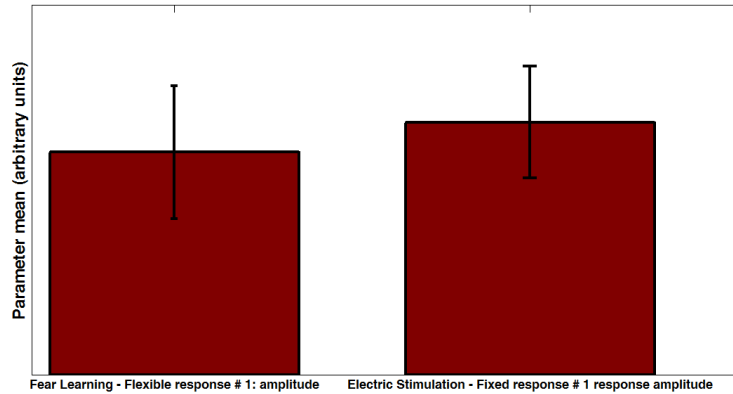
Figure 22.2: Display of results from a group analysis (2nd level). Left bar: difference in amplitude of sympathetic arousal (aSA) to CS+|US- minus CS-. Right bar: difference in aSA to US+ minus US-

```
Statistics:
----------------------------------------------------------------------------
mean    sem    t      p       df    Contrast name
----------------------------------------------------------------------------
5.16    1.54   3.34   0.0035  19    Fear Learning - Flexible response # 1: amplitude
5.85    1.30   4.52   0.0002  19    Electric Stimulation - Fixed response # 1 response amplitude
----------------------------------------------------------------------------


SCRalyze (c) Dominik R. Bach, Wellcome Trust Centre for Neuroimaging, UCL London UK
```

Figure 22.3: Statistics from one-sampled t-tests CS|US- minus CS- and US+ minus US- for 20 participants.

23   Calling PsPM functions directly from the Matlab command line

All the above mentioned steps in this tutorial can also be done in the Matlab command line without using the Matlabbatch syntax. Calling PsPM functions allows fast analysis of the data, efficient processing of multiple subject data and gives a lot more flexibility when it comes to fine tuning. However, this also requires knowledge about the Matlab command line and about the PsPM functions used. In this section we will revisit some examples from above and go through them by calling PsPM functions directly.

Basic PsPM setup

Here we setup the command line environment. This includes adding PsPM to the path as well as creating path variables for other information such as the path where the tutorial data is located.

```matlab
% The path where the 'Tutorial_data_GLM' folder is located
work_path = 'Path where the data is located';

% Where to find the tutorial data
tutorial_data = [work_path filesep 'Tutorial_data_GLM'];

% The path where PsPM is located
pspm_path = 'Path where PsPM is located';

% Add PsPM to the search path of matlab
addpath(pspm_path);

% initialise PsPM
pspm_init;

% Change path to the working directory
cd(work_path);

% Define for which subject we want to execute this code:
subject = 25;
```

Import the data

Here we will import the data from an aquisition file into a PsPM-Matlab file which we can later use to fit a GLM.

```matlab
% Compose name of datafile and combine with the path where to find it.
data_file_name = sprintf('trsp_1_%i.smr', subject);
data_file = [tutorial_data filesep data_file_name];

% With the option overwrite you can tell whether to overwrite existing
% files (=true) or not (=false).
options = struct('overwrite', true);

% Define which channels should be imported and what are the settings of
% those. This tells the function to import channel 2 as SCR channel
% with transfer function disabled and a marker channel, which should be
% found automatically (therefore no channel id is provided).
import = { ...
    struct('type', 'scr', 'channel', 2, 'transfer', 'none'), ...
    struct('type', 'marker') ...
    };

% Call PsPM function to import the file. 'spike' is the datatype of the
% file. You can check in pspm_init under defaults.import.chantypes
% which datatypes are supported. The function returns the path to the
% imported file.
imported_file = pspm_import(data_file, 'spike', import, options);
```

Trim the data

Trimming allows to cut away the data which should not be used for further

analysis.

```matlab
% Set option to overwrite existing files.
options = struct('overwrite', true);

% Remove the 110s after the first marker and 10s after the last
% marker.
trimmed_file = pspm_trim(imported_file, 110, 10, 'marker', options);
```

Fit GLM

Set the model settings and fit the GLM. The model uses a separate timing
file with timeunits 'markers'. We also specify to normalise the data and
set a custom response function. Args is passed to the response function
and the format of an argument therefore depends on the specified response
function itself. Also we tell the function to overwrite existing files by passing
options.overwrite = true;

```matlab
% set option 'overwrite' to true
options = struct('overwrite', true);

% set model settings
model = struct('modelfile', [tutorial_data filesep sprintf('glm_%i', subject)], ...
    'datafile', trimmed_file, ...
    'timing', [tutorial_data filesep sprintf('trSP_appraisal_01_%i_onsets_1.mat', subject)], ...
    'timeunits', 'markers', ...
    'norm', 1, ...
    'modelspec', 'scr', ...
    'bf', struct('fhandle', @pspm_bf_scrf, 'args', 1) ...
    );

% run GLM fitting procedure
glm = pspm_glm(model, options);
```

Create First-Level contrasts

Here we want to find out if there is a difference between neutral and aver-
sive events. Therefore we compare event type 1 with event type 2. We are
not interested event type 3. This results in the contrast [1 -1 0].

```matlab
% Set path to modelfile
modelfile = [tutorial_data filesep sprintf('glm_%i', subject)];

% Call contrasts function (function has no output)
pspm_con1(modelfile, {'neutral vs. aversive'}, {[1 -1 0]});
```

Export parameter estimates

If we are interested to do further analysis on the parameter estimates we can export them to the screen or to a file.

```
% export parameter estimates to screen
pspm_exp(modelfile, 'screen', 'param');
```

Part IV

How to reference PsPM

To justify our continued effort in the face of tight resources, we ask you acknowledge PsPM when you use it: "Data were analysed using PsPM [version number], available at `http://bachlab.org/pspm`."

We suggest you reference our papers to justify your analysis steps and models in the following way:

- General PsPM reference:

  - Bach DR, & Friston KJ (2013). Model-based analysis of skin conductance responses: Towards causal models in psychophysiology. Psychophysiology, 50, 15-22.

  - Bach DR, Castegnetti G, Korn CW, Gerster S, Melinscak F, Moser T (2018). Psychophysiological modelling - current state and future directions. Psychophysiology, 55, e13214.

- GLM for SCR

  - general reference: Bach DR, Flandin G, Friston KJ, & Dolan RJ (2009). Time-series analysis for rapid event-related skin conductance responses. Journal of Neuroscience Methods, 184, 224-234.

  - canonical SCRF: Bach DR, Flandin G, Friston KJ, & Dolan RJ (2010). Modelling event-related skin conductance responses. International Journal of Psychophysiology, 75, 349-356.

  - assumptions of the model: Gerster S, Namer B, Elam M, Bach DR (2018). Testing a linear time invariant model for skin conductance responses by intraneural recording and stimulation. Psychophysiology, 55, e12986.

  - optimised filter settings: Bach DR, Friston KJ, Dolan RJ (2013). An improved algorithm for model-based analysis of evoked skin conductance responses. Biological Psychology, 94, 490-497.

- comparison with Ledalab and peak-scoring methods: Bach DR (2014). A head-to-head comparison of SCRalyze and Ledalab, two model-based methods for skin conductance analysis. Biological Psychology, 103, 63-88.

- non-linear model for event-related SCR:

  - general reference: Bach DR, Daunizeau J, Friston KJ, & Dolan RJ (2010). Dynamic causal modelling of anticipatory skin conductance responses. Biological Psychology, 85, 163-70.

  - optimised filter settings: Staib M, Castegnetti G, Bach DR (2015). Optimising a model-based approach to inferring fear learning from skin conductance responses. Journal of Neuroscience Methods, 255, 131-138.

  - optimised forward model for short SOA paradigms (see supplementary material): Tzovara A, Korn CW & Bach DR (2018). Human Pavlovian fear conditioning conforms to probabilistic learning. PLOS Computational Biology, 14, e1006243.

- non-linear model for SF:

  - general reference: Bach DR, Daunizeau J, Kuelzow N, Friston KJ, & Dolan RJ (2011). Dynamic causal modelling of spontaneous fluctuations in skin conductance. Psychophysiology, 48, 252-57.

  - canonical SCRF: Bach DR, Friston KJ, & Dolan RJ (2010). Analytic measures for quantification of arousal from spontaneous skin conductance fluctuations. International Journal of Psychophysiology, 76, 52-55.

  - MP algorithm: Bach DR, Staib M (2015). A matching pursuit algorithm for inferring tonic sympathetic arousal from spontaneous skin conductance fluctuations. Psychophysiology, 52, 1106-12.

- GLM for evoked HPR: Paulus PC, Castegnetti G, & Bach DR (2016). Modeling event-related heart period responses. Psychophysiology, 53, 837-846.

- GLM for fear-conditioned HPR: Castegnetti G, Tzovara A, Staib M, Paulus PC, Hofer N, & Bach DR (2016). Modeling fear-conditioned bradycardia in humans. Psychophysiology, 53, 930-939.

- GLM for evoked respiratory responses: Bach DR, Gerster S, Tzovara A, Castegnetti G (2016). A linear model for event-related respiration responses. Journal of Neuroscience Methods, 270, 147-155.

- GLM for fear-conditioned RAR: Castegnetti C, Tzovara A, Staib M, Gerster S, Bach DR (2017). Assessing fear memory via conditioned respiratory amplitude responses. Psychophysiology, 54, 215-223.

- GLM for illuminance-elicited PSR: Korn CW & Bach DR (2016). A solid frame for the window on cognition: Modeling event-related pupil responses. Journal of Vision, 16, 28.

- GLM for fear-conditioned PSR: Korn CW, Staib M, Tzovara A, Castegnetti G, Bach DR (2017). A pupil size response model to quantify fear conditioning. Psychophysiology, 54, 330-343.

- GLM for SEBR: Khemka S, Tzovara A, Gerster S, Quednow BB, Bach DR (2017). Modeling startle eyeblink electromyogram to assess fear learning. Psychophysiology, 54, 204-214.

- Comparison of different measures: Xia Y, Gurkina A & Bach DR (2019). Pavlovian-to-instrumental transfer after human threat conditioning. Learning & Memory, in press.

Part V

Release notes

24   PsPM Version 3

Version 3.0

Import

New data types were implemented

- Noldus Observer compatible

- Eyelink

Untested data types   CNT data import has not been not tested – please contact the developers with sample data files.

Filtering for SCR models

Previous versions of PsPM have used a bi-directional high pass filter of 0.0159 for all SCR analyses. We have recently shown a better predictive validity for GLM with a unidirectional filter of 0.05 Hz [9]. This also implies that different filters are used for different models. These are now set as

defaults, and the way the default settings are implemented has changed. It is now possible to alter the filter settings in the model definition, although we discourage this.

New SF method

A matching pursuit algorithm is now implemented to approximate the number of spontaneous fluctuations (SF) in skin conductance [32].

General linear modelling (GLM)

Parametric modulators   Parametric modulators (pmods) are z-normalised before being entered into the design matrix. This is to account for possibly very large or very small numbers – a badly scaled design matrix can cause induced instability in the inversion. The parameter estimates of the pmods were not transformed back in previous versions, i. e. the parameter estimates of the pmods were independent of the scaling of the pmods. This is appropriate as long as they are the same for all datasets, or if analysis is done strictly on a within-subject level. Some researchers have reported designs in which inference was to be drawn on parameter estimates of pmods on a between-group level, and where the pmods systematically differed between these groups. To account for this possibility, the parameter estimates are now transformed back, such that they refer to the pmods in their original scaling/units.

Parametric confounds   Previous versions of PsPM contained an option to include a parametric modulator across all event types, to account for confounds across all conditions. For example, in an experiment with 5 conditions, one could have included 5 regressors, plus one reaction time confound across all events, without including an associated regressor that contains the event onsets for all these events. This option was removed.

Design matrix filtering   Previous versions of PsPM filtered the design matrix after orthogonalisation of basis sets. This can introduce unwanted dependencies between regressors. PsPM 3.0 filters the regressors first, then orthogonalises the basis sets.

File format

Some minor changes have been made to the data format. In particular, marker channels from previous versions can not be read with the current version - such data files have to be re-imported. Model files have changed drastically, and model files from previous versions can not be read with the current version of the software.
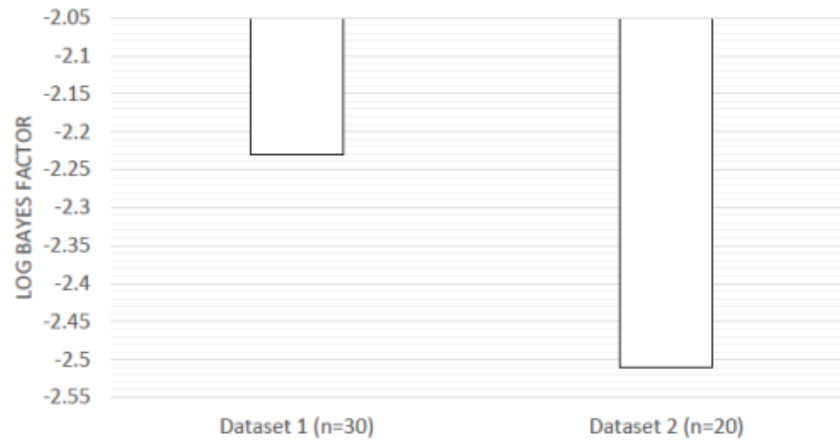
Figure 24.1: Model comparison between old and new versions of the VBA toolbox, based on two delay fear conditioning datasets. The log Bayes factor quantifies the difference between negative log likelihood (nLL) of parameter estimates obtained from model inversion using the old and new version of VBA. A difference in nLL above 3 indicates significant differences in model evidence which is not exceeded for either data set. Analysis and figure contributed by Matthias Staib.

VB inversion

The VBA toolbox (`http://mbb-team.github.io/VBA-toolbox/`) was updated in October 2014 [30]. This update incorporates bugfixes in this toolbox and slightly changed the model estimates in our test models. In terms of predictive validity, we noted that there was no consistent benefit of the old or new version of this code (Figure 24.1).

Version 3.0.1

Import

New data types were implemented

- DATAQ Windaq (no ActiveX needed)

- European Data Format (EDF)

Tools   'Preprocess respiration traces' replaces 'Convert Respiration to Respiration Period'. It supports conversions into the following datatypes:

- Respiration preriod (old funtionality)

- Respiration amplitude

- Respiration line length

- Respiration time stamp

First level contrast

Z-score parameter estimates   The function `pspm_con1` now supports z-scoring trial-by-trial parameter estimates. If selected, all parameter estimates for the same event type, across all trials, will be z-scored before computing the contrast. This option is currently only available for non-linear models.

Review first level model   Next to the regressors on the x- and y-axes, orthogonality plots newly display regressor names along the y-axes.

Version 3.0.2

GLM

Problem with multiple sessions in Design matrix fixed.

Version 3.1

General linear modelling (GLM)

New modalities   For the first time in PsPM, we introduce models for data types other than SCR:

- GLM for evoked HPR

- GLM for fear-conditioned HPR

- GLM for evoked RA

- GLM for fear-conditioned RA

- GLM for evoked RFR

- GLM for evoked RP

- GLM for fear-conditioned PS

Tools & Data preprocessing

Tools are split up into Tools & Data preprocessing. The content of each section is listed below. New functions are written in green.

Tools

- Display data

- Rename file

- Split sessions

- Artefact removal

- Downsample data

- Interpolate missing data

- Extract segments

- Segment mean

Data preprocessing

- Preprocess heart data

- Preprocess respiration data

- Prepare illuminance GLM

- Find valid fixations

- Convert pupil data

- Find startle sound onsets

Data preparation

Import Support for Philips Scanphyslog files and for bioread-converted AcqKnowledge files has been added to the import function.

25   PsPM Version 4

Version 4.0

Change from scr_ to pspm_

The prefix scr_ has been exchanged with pspm_. This applies to all functions contained in PsPM.
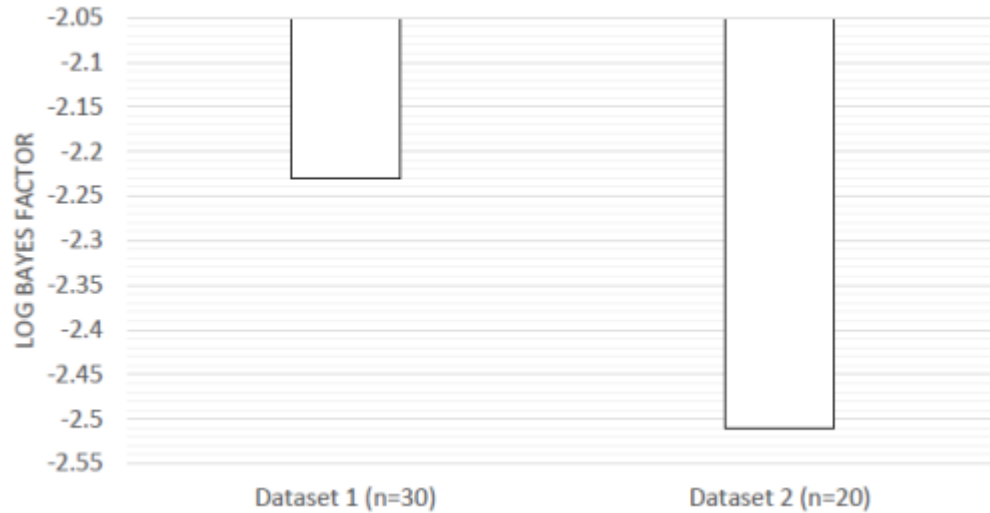
Figure 25.1: Model comparison between old and new versions of the VBA toolbox, based on one delay fear conditioning dataset (Dataset 2, see earlier comparison). The log Bayes factor quantifies the difference between negative log likelihood (nLL) of parameter estimates obtained from model inversion using the old and new version of VBA. A difference in nLL above 3 indicates significant differences in model evidence which is not exceeded for the used dataset.

Updated toolboxes

Matlabbatch (`https://git.code.sf.net/p/matlabbatch/code-git`) and the VBA toolbox (`https://github.com/MBB-team/VBA-toolbox`) were updated to the latest available versions (08.11.2016). The updated VBA version fully supports newer Matlab versions and warning messages in Matlab 2014 and newer do not appear anymore (during PsPM startup). In terms of predictive validity, we noted that there was no consistent benefit of the old or new version of this code (Figure 25.1).

Subsessions in DCM models

Until now, long periods of NaN values (which are just disregarded) could cause problems for inversion of DCMs. These periods are now split (according to threshold value) into subsessions. The subsessions will be evaluated independently, while at the end the results will be stacked together. Therefore the output-format will not change and should be compatible with earlier releases.

Import

New import function for Labchart files   The new import function allows direct import of raw labchart files without exporting them to a suitable format (as was required to date). The function is only available on Windows systems.

General linear modeling (GLM)

New modalities

- Startle eyeblink for EMG (SEB)

Pupil Models

- Blink channels were removed. Periods during blinks and saccades are set to NaN already in the import function. This applies to all channels of the respective eye. Accordingly, blink validation functionality was removed in the function pspm_find_valid_fixations. If needed, blinks and saccades can be imported as custom channels using channel ids 7-10 (two eyes) or 4-5 (one eye), while blinks and left eye come first.

- Importing pupil channels now allows direct specification of a transfer function to convert arbitrary units to mm.

- Importing of eyes which have not been recorded creates channels containing NaN values, so that the same import batches can be used for groups of subjects, even if one eye is missing in some of them.

- Sessions in eyelink files (caused by interruption of recording) will be concatenated according to the timing variable.

- In pspm_find_valid_fixations

  - the use of software aspect ratio was replaced by resolution. This allows to correctly map the gaze coordinates to the shapes mapped by the stimulus software.
  - gaze coordinates can be plotted in order to verify the validation settings.
  - interpolation option was removed and should now be used independently.

Minor changes & Bugfixes

- Version-Check bug during startup is now fixed.

- Markerinfos will now according to marker channels be converted to an event-based format.

- The data editor (pspm_data_editor) now allows to specify an output file using the command line.

- pspm_split_sessions additionally allows to specify marker ids at which the file should split.

- Artefact correction was extended with a further function (Simple SCR quality correction).

- Convert pupil data becomes convert data and currently only allows area to diameter conversion (arbitrary units to mm is integrated in the import function)

Version 4.0.1

Bugfixes

- Fix not working options in Matlabbatch module 'Preprocess heart data'

Version 4.0.2

Bugfixes

- Fix error when calling 'Convert data' from GUI

- Fix taking square root if pupil data is recorded in AREA

Version 4.1.0

New Functions

- `pspm_convert_pixel2unit`: Allows to transfer gaze data from pixel to units. This facilitates the use of pspm_find_valid_fixations which needs data in unit values.

- `pspm_compute_visual_angle`: Computes visual angle from gaze data.

- `pspm_convert_visangle2sps`: Convert visual angles from Eyelink to Scanpath speed.

- `pspm_bf_spsrf_box`: SPSRF basis function with box car.

- `pspm_bf_spsrf_gamma`: SPSRF basis function with gamma function.

New Features

- `pspm_extract_segments` supports DCM files/structures.

- GLM for SPS.

- `pspm_find_valid_fixations` can now take a bitmap as input.

- A new way to trim data: Start and end times can be chosen according to marker name or value.

- A new function to import SMI iView X EyeTracker files into PsPM.

- A new function to import ViewPoint EyeTracker files into PsPM.

Changes

- `pspm_find_valid_fixations` now computes a circle around the fixation points when run in fixations mode.

Bugfixes

- Heart period response function (bf_hprf) coefficients are fixed according to the manual

- `pspm_extract_segments` bugfixes:

    - Fix bugs related to conditions appearing in different sessions.

- `pspm_get_eyelink` now imports markers between two recording sessions (previously these markers were "lost")

- `pspm_compute_visual_angle`: Fix bug in range computation.

- `pspm_ecg2hb`: Fix out of bounds error occurring for highly noisy data with many outliers.

- `pspm_get_acq`: Fix incorrect linear transformation during ACQ import.

- `pspm_convert_unit`: Fix incorrect transformations between metric units and inches.

- `pspm_resp_pp`: Fix out of bounds error during convolution in cushion mode.

- `pspm_pp` in 'simple_qa' mode now uses the default values specified in `pspm_simple_qa`.

- Various further bugfixes.

Improvements

- Blink and saccade channels can be imported with PsPM Eyelink import.

- GLM structure holds the missing data percentage of each condition after segment extraction. Further, the decision to exclude conditions can be made depending on the percentage of missing data.

- `pspm_extract_segments` now works both with GLM files and already loaded structures

- PsPM now checks if SPM is already in MATLAB path. If so, user is asked to remove it from path before initializing PsPM.

- `pspm_load1` now returns statistics about conditions with high NaN ratios.

- `pspm_extract_segments` now is able to use data stored in GLM/DCM structures instead of relying the existence of original data files.

- Multiple new tests to validate the correctness of the functions.

Version 4.1.1

This is a hotfix release that fixes a few issues with 4.1.0 release.

Changes

- `pspm_get_eyelink` now uses the scaling factor from [46] for area based arbitrary units to milimeters conversion.

- `pspm_get_smi` does not perform pixel to milimeters conversion for pupil data anymore. Pupil values are returned as is in pixel units.

- pspm_get_smi performs pixel to milimeters conversion for gaze data only if the stimulus resolution in pixels are given as an extra option.

Improvements

- `pspm_get_viewpoint` is now able to import blink and saccade channels from sample files as well. In order to enable this feature, event files must be stored asynchronously in the datafile. See ViewPoint EyeTracker section in this manual for further information.

Version 4.2.0

New Functions

- A new pupil data preprocessing function

- A new pupil foreshortening error correction function

- A new QRS detection algorithm for fMRI environments

New Features

- Previously, Eyelink import used to discard 50 ms worth of samples from each side of every blink or saccade period. This behaviour is preserved; however, sample discard duration can now be set by the user.

- Channel processing functions now store extensive historical data in PsPM files. This data can be printed in table format using the new utility function `pspm_format_history_from_file`

- Pupil channel is now loaded according to a new precedence order. Refer to GLM documentation in this manual to learn more.

Changes

- Eyelink import now returns times and dates in a slightly different format.

- Newest version of PsPM is now obtained from the version string in pspm.sourceforge.net, not from the download link zip file name

- GLM now uses the **LAST** channel of a specified modality in a PsPM file.

- Nonlinear SCR model (DCM) now does not use the inter-trial interval duration value (ITI) of the last trial when computing session specific minimum ITI values. Previously, using these last ITI values would lead to abnormally small overall min ITI values in some input files, thereby causing the inference and PCA sections to use less data for all trials. Now, we prevent this from happening.

- Eyelink import parameter blink/saccade edge discard factor default value has been set to 0. This means no extra samples are discarded around blinks/saccades.

Bugfixes

- Fix a bug in `pspm_extract_segments` where trial onsets were not assigned to conditions correctly

- Fix a bug in Viewpoint import where files in DOS format were being parsed incorrectly

- Fix a bug in Eyelink import where blink/saccade periods where misaligned, causing the function to discard useful data and return noise instead

Improvements

- New utility functions to make PsPM more compatible across different operating systems

- An optimized Eyelink import function that is significantly faster and more memory efficient than previous versions

- An optimized SMI event import function that is significantly faster than the previous version

- Multiple new tests to validate the correctness of the functions

- API unification: Now all preprocessing functions use `channel_action` variable to choose what to do with the new channel.

Version 4.2.1

New Functions

- Three new tests (pspm_hb2hp_test.m, pspm_filtfilt_test.m, pspm_butter_test.m)

Changes

- `pspm_display` and `pspm_ecg_editor` do not perform filtering anymore.

- Treatment of missing data in DCM is now the same regardless of whether they are specified as NaN or via model.missing.

- Eyelink import parameter blink/saccade edge discard factor default value has been set to 0. This means no extra samples are discarded around blinks/saccades.

Bugfixes

- Fix a bug in `pspm_hb2hp` where the function crashed when there is no heartbeat left after lower and upper bound filtering of the heartbeat periods

- Fix a bug in `import_eyelink` which imported more markers than the number of markers in the actual .asc file

- Fix a bug in `pspm_display` which crashed when trying to plot ECG signal that contains NaN

- Fix a bug in `pspm_prepdata` which returned only NaN when input signal contained some NaN values

- Fix a bug in `pspm_version` which crashed when MATLAB was invoked with `-nojvm`

- Fix a bug in `pspm_get_viewpoint` which returned '+,=,+' lines in the marker list and which crashed when given a viewpoint data file containing multiple sessions separated with '+,=' type of markers

- Fix a bug in `import_viewpoint` which created a new session for every marker containing a '+' somewhere, e.g.: 'CS+'

- Fix a bug in `pspm_get_events` which was not able to locate a marker if it occured on the first or last sample in a given datafile

- Fix a bug in `pspm_filtfilt` which crashed when the filter parameters were of dimension one

Version 4.3.0

New Features

- In `pspm_get_events`, `import.flank` can be now set to `'all'` what would import all markers and data related to them.

- `pspm_display` allows now to display pupil size units and the gaze x & y coordinate units on the y-axis.

- `pspm_extract_segments` can be used now with raw data and thus be called easily within another function.

Changes

- `pspm_version` has a new url and thus do not send any warning about version checks anymore.

- `import_eyelink` do not import markers which are outside the session end time (END marker).

- `import_eyelink` sends a warning whenever two markers have the same timestamp.

- In `pspm_get_eyelink`, `import.flank` set to `'all'`.

- The test of `pspm_extract_segments` was adapted to the new feature.

- External functions and libraries were regrouped in one folder called `ext`.

Bugfixes

- Fix a bux in `pspm_bf_lcrf_gm` and `pspm_bf_ldrf_gm` where the offset was wrongly implemented.

- Fix a bug in `pspm_compute_visual_angle` where there was an error in the conversion factor of pixels wrt. to mm.

26   PsPM Version 5

Version 5.0.0

New Features

- Allow `pspm_data_editor` to load an epoch file.

- Allow `pspm_simple_qa` to suppress classification of discretisation oscillations as artefacts, to expand artefact windows, and to automatically remove small data islands embedded in artefacts.

- Allow `pspm_simple_qa` to store the epochs of data that are filtered out into an output `.mat` file. The accompanying GUI editor is available under 'Artefact removal' in the tools section.

- Allow `pspm_convert_gaze_distance` to convert from distance units to degrees or scanpath speed. The accompanying GUI editor is available under 'Gaze Convert' in the data preprocessing section.

- Add the possibility to select the flank in the `Import` module of the GUI of PsPM.

- Add the possibility to import DSV (delimiter separated values) as well as CSV (comma separated values) data files.

Changes

- Split the data convert functionality in tools into the 'Gaze Convert' and 'Pupil Size convert' in the data preprocessing section.

- Factor out blink/saccade edge filtering logic out of `pspm_get_eyelink` to `pspm_blink_saccade_filt`.

- Deprecate edge filtering functionality in `pspm_get_eyelink`.

- Make `pspm_pupil_correct_eyelink` use the last gaze channel when there are multiple gaze x (similarly y) channels in the file.

- Make `pspm_extract_segments` return NaN percentages and not ratios.

Bugfixes

- Scale DCM plot XTick by sample rate.

- Correct index offset when dealing with the descending flank for continuous markers.

- Allow `pspm_display` to plot any type of marker channels.

- Fix `pspm_display` behaviour when user tries to load data with less channels than the data he/she is currently displaying.

- Fix `pspm_split_sessions` behaviour when the intertrial interval in the data is random. Add an option `randomITI` (0 or 1) which in the latter case it forces the function to evaluate the mean marker distance using all the markers and not per session. Default value: 0.

- Remove `startsWith` and `endsWith` from all functions for a better backward compatibility.

- Fix bug in `pspm_trim` which was wrongly defining the starting trimming point index.

Version 5.1.0

New Features

- PsPM now has an improved GUI with a more eye-friendly colour and typeface.

- PsPM now has an alternative GUI built by using .mlapp, which shows consistent a more modern appearance across different platforms.

New Functions

- pspm_scr_pp

    - `pspm_scr_pp` replaces the classic `pspm_simple_qa`.
    - `pspm_scr_pp` now allows users to detect clipping, where the results can be exported together with the original outcome.
    - `pspm_scr_pp` now allows users to just write artefact epochs and leave data unchanged.

Changes

- General

    - Dialogs have been rewritten for avoiding endless loops.
    - *Pre-processing* menu has been reconstructed, where *pupil size convert* and *gaze convert* are now under *pupil and eye tracking*.
    - PsPM now always generates test data inside the PsPM folder.

- pspm_interpolate

    - `pspm_interpolate` now process data ending with `NaN` well without throwing warnings.

- pspm_find_sounds

    - `pspm_find_sounds` now throws warning when some data was excluded due to the strict parameter setting.

- pspm_split_sessions

    - `pspm_split_sessions` is now using `pspm_trim` for processing data.

Bugfixes

- pspm_glm

    - A bug that leads to the failure of selecting left/right eye has been fixed.

- pspm_load1 and pspm_dcm

    - A bug that leads to the failure of exporting statistics has been fixed.

- pspm_resp_pp

    – A bug that leads to the failure of opening batch for respiration data processing has been fixed.

- pspm_scr_pp

    – A bug that leads to incorrect matrix construction has been fixed.

27   PsPM Version 6

Version 6.0.0

This is a major release and may have incompatibility with previous versions.

New features

- PsPM now support a develop mode, which allows

    – Minimised terminal output.
    – Automagical parameter selection for testing.
    – Always to overwrite files.

- pspm_extract_segments

    – now allows an optional z - score for normalisation.

- pspm_find_valid_fixations

    – now allows processing preprocessed left/right/combined pupil data.

- pspm_gaze_pp

    – now allows preprocessing gaze signals.

- pspm_overwrite

    – now handles the behaviour of overwrite if not defined by the user, and is used where applicable.

- pspm_time2index

    – now processes adjustable conversion from time to index globally.

- pspm_update_chantype

    – now allows generalised behaviour of updating channel types.

Bug Fixes

- General

    - A bug which may lead to failure of pspm path searching has been fixed.

    - A bug which may display figures in the GUI has been fixed.

- pspm_bf_spsrf_box

    - now uses correct parameter settings.

- pspm_convert_ppu2hb

    - now terminates if only one pulse is detected.

- pspm_data_editor

    - now able to show figures when import a datafile correctly.

- pspm_dcm

    - now handles missing epochs that start at 0 correctly.

    - now processes variables properly to make length consistent.

- pspm_display

    - now displays plots with correct x-axis ranges.

- pspm_glm

    - now applies correct variable settings.

    - now produces results correctly when facing many short missing epochs.

- pspm_interpolate

    - now returns results correctly when applying forced extrapolation.

- pspm_resp_pp

    - now allows replace as the channel action properly.

- pspm_scr_pp

    - now uses inconsistent variables when delivering missing epochs.

    - now writes data out properly when epochs are removed.

    - now properly handles data if the first channel is not `scr`.

- pspm_sf

    – now refer to model file correctly.

- pspm_split_sessions

    – now processes missing files with an epoch starting at 0 s properly.

    – now properly catches edge cases.

- ValidSamplesModel

    – now properly throws warnings if histogram counts are zeros.

Improvements

- General

    – now supports loading `ppg` data.

    – now displays improved terminal output text.

- pspm_con1

    – now assesses the statistics in the arguments for throwing warnings when detecting invalid values.

- pspm_data_editor

    – now displays the *x* and *y* axis label and text according to the input data.

    – now sorts the epoch list according to the start data point.

- pspm_get_marker

    – now detects and update the field `flank` when applicable.

- pspm_import

    – now detects and update the field `flank` when applicable.

- pspm_load_data

    – now checks whether the input has an empty channel.

    – now autofills some variables.

- pspm_pupil_pp

    – now calls `pspm_load_data` to load single channels.

- pspm_rev_glm

    – now displays normalised data for visualisation purpose only.

- pspm_write_channel

    – now checks whether the input has an empty channel.

GUI

- "Edit defaults" in the GUI is now working properly.

- GUI typeface is now generalised across the software.

For Developers

- General

  – PsPM now uses `.editorconfig` to control the indent.

  – PsPM now uses UTF-8, LF, and MATLAB's soft tab indent as the standard code formatting style.

  – PsPM now uses test data from the repository PsPM-data.

- pspm_get_text_test

  – A bug, which leaves residual test files after code testing, has been fixed.

Version 6.1.0

General

- Minimum version of MATLAB

  – The minimum version of MATLAB required for running PsPM is MATLAB 7.1 or R14 Service Pack 3.

- Entering PsPM

  – Entering PsPM is now hybrid based on the version of MATLAB, namely through AppDesigner or Guide. PsPM will be entered through AppDesigner when supported and through Guide for older versions of MATLAB. To call PsPM, users should still use the command `pspm` in the Command Window of MATLAB, where the appropriate way of entering PsPM will be recognised automatically.

  – Users can still use their preferred way to enter PsPM. To enter PsPM through AppDesigner, please type `pspm_appdesigner`. To enter PsPM through Guide, which is the classic entrance, please type `pspm_guide`.

  – The AppDesigner version of PsPM UI is newly introduced by MATLAB and the encouraged way for launching PsPM, and it has no functionality difference than the Guide version of PsPM UI.

  - Both AppDesigner and Guide are designed and tested for Windows, macOS (either Intel or Apple Silicon) and Linux. Issues of opening PsPM through either of the two UI systems are encouraged to be reported to PsPM developing group at GitHub.

- Introducing `pspm_options`

  - A new function `pspm_options` is introduced to PsPM for controlling the default and acceptable values of the struct `options` used by most PsPM functions. The default values of the fields in the struct `options` for various functions can be directly checked by searching in `pspm_options`.
  - The default values in `pspm_options` have been checked and tested for PsPM. If preferred values are different from defaults, users can specify them when calling the corresponding PsPM functions, and the corresponding functions and `pspm_options` will always respect the users' specification with the highest priority. However, the user's specification needs to satisfy the condition set in `pspm_options`, and invalid values may be reported as errors.

- Help text

  - The help text has been updated for all PsPM functions.
  - Help text can be checked by right clicking the functions and select "help text".

UI Improvements

- The UI for the following features has been improved and tested

  - Launchpad (AppDesigner)
  - Launchpad (Guide)
  - Batch editor
  - Display data
  - Contrast manager

Bug Fixes

- UI

  - A bug that used to make PsPM crash when `pupil size convert` and `gaze convert` in `Data Preprocessing` were selected has been fixed.
  - A bug that leads to error when PsPM is started offline has been fixed.

> **–** A bug that leads to incorrect x labels in `pspm_rev_dcm` has been fixed.

- `pspm_convert_...`

  > **–** A bug that leads PsPM to crash, which is caused by outdated UI calling for `pspm_convert_...` functions, has been fixed.

- `pspm_get_eyelink`

  > **–** A bug that causes issues when importing eye link data that is scanned at both left and right sides has been fixed.

- `pspm_glm`

  > **–** A bug that incorrectly identifies the `NaNs` in the input data has been fixed.

- `pspm_sf`

  > **–** A bug that has lead to incorrect input datafile assigning has been fixed.

  > **–** A bug that leads PsPM crash when `pspm_sf` is analysing data with missing epochs has been fixed.

  > **–** A bug that leads to error when `pspm_sf` analyses data where `time` unit is defined as `marker` has been fixed.

- `pspm_split_sessions`

  > **–** `pspm_split_sessions` now considers `marker_chan_num` when calling `pspm_trim`.

Improvements

- channel

  > **–** We have generalised `channel` related variables throughout PsPM, which are given as
  >
  > * `chan/channel` to `channel`
  > * `chans/channels` to `channels`
  > * `channel_units/channels_units/chan_units/chans_units` to `channel_units`
  > * `chan_combine` to `channel_combine`
  > * `chan_action` to `channel_action`
  > * `channels_header` to `channel_header`
  > * `chantype` to `channeltype`

- import_eyelink

  * Improved import_eyelink for adding some support for importing eyelink data converted by higher version of .EDF files.

- pspm_con2

  * pspm_con2 now uses pspm_overwrite.

- pspm_dcm and pspm_dcm_inv

  * .flexevents and .fixevents are now fields of model instead of options.
  * pspm_dcm now uses pspm_get_timing to handle missing epochs.
  * Improved missing epoch support
    · The field .missing is now allocated from options to model.
    · .missing_data is used to load missing epoch data that was loaded from dcm, as an optional field.
  * The index is update to keep the first event when it is at time 0 in session.

- pspm_extract_segments

  * The first channel is always referred by marker_chan as default.

- pspm_glm

  * The fallback for options.exclude_missing has been set for pspm_glm, which is not to exclude any missing epochs.
  * A bug that leads to incorrect missing epoch checking has been fixed.
  * marker_chan_num now refers to the first marker channel as default.
  * pspm_glm now uses pspm_get_timing to handle missing epochs.

- pspm_interp1

  * pspm_interp1 now considers the data where no valid values are detected and interpolation is not possible, and warnings will be reported in this case.

- pspm_merge

  * The first two channels will be merged as the default option.

- pspm_prepdata

  * pspm_prepdata now uses interpolation to handle data with NaN.
  * Data that begins and/or ends with NaN will be filled with the first/last non-NaN values in those positions.

- pspm_sf

* Now supports missing epochs by allowing input with `NaN` that indicates missing epochs.
* Missing epochs can be specified with `options.missing` or the automatic `NaN` detection.
* Missing epochs will be verified with `options.missingthresh`.
* When the missing epochs are longer than 2s, the result out will be converted to `[]`.
* The UI of SF analysis now allows using missing epochs from files or not to define any missing epochs.
* Now uses `pspm_get_timing` to handle missing epochs.
* The field `.marker_chan_num` now refers to the first marker channel as default.

– `pspm_sf_dcm`
  * `pspm_sf_dcm` now uses `pspm_interp1` for interpolating data.
– `pspm_text`
  * The Matlab file that stores the information of help text `pspm_text.mat` will be stored inside the source folder of PsPM and will be deleted when PsPM is quit.
– `pspm_trim`
  * `marker_chan_num` now refers to the first marker channel as default.

Minor Adjustments

- `pspm_sf`, `pspm_glm` and `pspm_dcm`.

  – The option `marker_chan_num_event` is removed.
  – In default, the first marker channel / event channel is always selected and no users' customisation is allowed.
  – The last data channel / wave channel is always selected.

Reference document

- Added a new document *PsPM Reference* for checking the default values and restrictions of `options` fields.

Part VI

Acknowledgements

Various models have been developed by different researchers. These are:

- Dominik R. Bach: SCR models and GLM for evoked respiratory responses

- Matthias Staib: Optimisation of non-linear SCR model

- Philipp Paulus: GLM for evoked HPR

- Giuseppe Castegnetti: GLM for fear-conditioned HPR and RAR

- Christoph Korn: PSM models

- Saurabh Khemka: GLM for SEBR

- Yanfang Xia: GLM for ScanPath Speed

Some PsPM functions depend on external scripts/libraries implemented by other groups:

- For import of spike data, PsPM makes use of the SON library written by Malcolm Lidierth at King's College London, and of functions provided by the SPM physiology toolbox, written by Chloe Hutton and Eric Featherstone at the FIL.

- Biopac AcqKnowledge import uses a routine created by Sebastien Authier and Vincent Finnerty at the University of Montreal.

- The DCM inversion uses a variational Bayesian inversion scheme written by Jean Daunizeau.

- Some SCR quality control functions use the method described in [59].

- Pupil preprocessing uses external scripts adapted from the implementations by Mariska Kret and Elio Sjak-Shie at https://github.com/ElioS-S/pupil-size. The original reference is [47].

- Pupil foreshortening error correction implements the method proposed in [46] by Taylor Hayes and Alexander Petrov.

- QRS detection for fMRI environment uses external scripts adapted from the code by Zhongming Liu, Jacco A de Zwart, Peter van Gelderen, Li-Wei Kuo, and Jeff H Duyn at https://www.amri.ninds.nih.gov/software.html. The original reference is [41].

Part VII
References

References

[1] Bach, D. R & Friston, K. J. (2013) Model-based analysis of skin conductance responses: Towards causal models in psychophysiology. *Psychophysiology* **50**, 15–22.

[2] Bach, D. R, Castegnetti, G, Korn, C. W, Gerster, S, Melinscak, F, & Moser, T. (2018) Psychophysiological modeling: Current state and future directions. *Psychophysiology* **55**, e13214.

[3] Burnham, K. P & Anderson, D. R. (2004) Multimodel inference understanding AIC and BIC in model selection. *Sociol. Methods Res.* **33**, 261–304.

[4] Penny, W. D, Stephan, K. E, Mechelli, A, & Friston, K. J. (2004) Comparing dynamic causal models. *Neuroimage* **22**, 1157–1172.

[5] Alexander, D. M, Trengove, C, Johnston, P, Cooper, T, August, J. P, & Gordon, E. (2005) Separating individual skin conductance responses in a short interstimulus-interval paradigm. *J. Neurosci. Methods* **146**, 116–123.

[6] Bach, D. R, Flandin, G, Friston, K. J, & Dolan, R. J. (2009) Time-series analysis for rapid event-related skin conductance responses. *J. Neurosci. Methods* **184**, 224–234.

[7] Benedek, M & Kaernbach, C. (2010) Decomposition of skin conductance data by means of nonnegative deconvolution. *Psychophysiology* **47**, 647–658.

[8] Benedek, M & Kaernbach, C. (2010) A continuous measure of phasic electrodermal activity. *J. Neurosci. Methods* **190**, 80–91.

[9] Bach, D. R, Friston, K. J, & Dolan, R. J. (2013) An improved algorithm for model-based analysis of evoked skin conductance responses. *Biol. Psychol.* **94**, 490–497.

[10] Bach, D. R. (2014) A head-to-head comparison of SCRalyze and Ledalab, two model-based methods for skin conductance analysis. *Biol. Psychol.* **103**, 63–68.

[11] Staib, M, Castegnetti, G, & Bach, D. R. (2015) Optimising a model-based approach to inferring fear learning from skin conductance responses. *J. Neurosci. Methods* **255**, 131–138.

[12] Pinheiro, J. C & Bates, D. M. (2000) *Mixed-Effects Models in S and S-PLUS*, Statistics and Computing. (Springer-Verlag, New York). DOI: 10.1007/b98882.

[13] Bach, D. R, Tzovara, A, & Vunder, J. (2017) Blocking human fear memory with the matrix metalloproteinase inhibitor doxycycline. *Mol. Psychiatry* **23**, 1584–1589.

[14] Homan, P, Lin, Q, Murrough, J. W, Soleimani, L, Bach, D. R, Clem, R. L, & Schiller, D. (2017) Prazosin during threat discrimination boosts memory of the safe stimulus. *Learn. Mem.* **24**, 597–601.

[15] Tzovara, A, Korn, C. W, & Bach, D. R. (2018) Human Pavlovian fear conditioning conforms to probabilistic learning. *PLoS Comput. Biol.* **14**, e1006243.

[16] Daunizeau, J, Friston, K. J, & Kiebel, S. J. (2009) Variational Bayesian identification and prediction of stochastic nonlinear dynamic causal models. *Physica D* **238**, 2089–2118.

[17] Boucsein, W. (2012) *Electrodermal activity*. (Springer Science & Business Media).

[18] Bini, G, Hagbarth, K. E, Hynninen, P, & Wallin, B. G. (1980) Thermoregulatory and rhythm-generating mechanisms governing the sudomotor and vasoconstrictor outflow in human cutaneous nerves. *J. Physiol.* **306**, 537–552.

[19] Sugenoya, J, Iwase, S, Mano, T, & Ogawa, T. (1990) Identification of sudomotor activity in cutaneous sympathetic nerves using sweat expulsion as the effector response. *Eur. J. Appl. Physiol. Occup. Physiol.* **61**, 302–308.

[20] Kunimoto, M, Kirnö, K, Elam, M, & Wallin, B. G. (1991) Neuroeffector characteristics of sweat glands in the human hand activated by regular neural stimuli. *J. Physiol.* **442**, 391–411.

[21] Kunimoto, M, Kirnö, K, Elam, M, Karlsson, T, & Wallin, B. G. (1992) Neuro-effector characteristics of sweat glands in the human hand activated by irregular stimuli. *Acta Physiol. Scand.* **146**, 261–269.

[22] Gerster, S, Namer, B, Elam, M, & Bach, D. R. (2017) Testing a linear time invariant model for skin conductance responses by intraneural recording and stimulation. *Psychophysiology* **55**, e12986.

[23] Kirnö, K, Kunimoto, M, Lundin, S, Elam, M, & Wallin, B. G. (1991) Can galvanic skin response be used as a quantitative estimate of sympathetic nerve activity in regional anesthesia? *Anesth. Analg.* **73**, 138–142.

[24] Bach, D. R, Flandin, G, Friston, K. J, & Dolan, R. J. (2010) Modelling event-related skin conductance responses. *Int. J. Psychophysiol.* **75**, 349–356.

[25] Kunimoto, M, Kirnö, K, Elam, M, Karlsson, T, & Wallin, B. G. (1992) Non-linearity of skin resistance response to intraneural electrical stimulation of sudomotor nerves. *Acta Physiol. Scand.* **146**, 385–392.

[26] Friston, K. J, Josephs, O, Rees, G, & Turner, R. (1998) Nonlinear event-related responses in fMRI. *Magn. Reson. Med.* **39**, 41–52.

[27] Friston, K. J, Mechelli, A, Turner, R, & Price, C. J. (2000) Nonlinear responses in fMRI: the Balloon model, Volterra kernels, and other hemodynamics. *Neuroimage* **12**, 466–477.

[28] Bach, D. R, Friston, K. J, & Dolan, R. J. (2010) Analytic measures for quantification of arousal from spontaneous skin conductance fluctuations. *Int. J. Psychophysiol.* **76**, 52–55.

[29] Bach, D. R, Daunizeau, J, Friston, K. J, & Dolan, R. J. (2010) Dynamic causal modelling of anticipatory skin conductance responses. *Biol. Psychol.* **85**, 163–170.

[30] Daunizeau, J, Adam, V, & Rigoux, L. (2014) VBA: a probabilistic treatment of nonlinear models for neurobiological and behavioural data. *PLoS Comput. Biol.* **10**, e1003441.

[31] Bach, D. R, Daunizeau, J, Kuelzow, N, Friston, K. J, & Dolan, R. J. (2011) Dynamic causal modeling of spontaneous fluctuations in skin conductance. *Psychophysiology* **48**, 252–257.

[32] Bach, D. R & Staib, M. (2015) A matching pursuit algorithm for inferring tonic sympathetic arousal from spontaneous skin conductance fluctuations. *Psychophysiology* **52**, 1106–1112.

[33] Lim, C. L, Rennie, C, Barry, R. J, Bahramali, H, Lazzaro, I, Manor, B, & Gordon, E. (1997) Decomposing skin conductance into tonic and phasic components. *Int. J. Psychophysiol.* **25**, 97–109.

[34] Greco, A, Lanata, A, Valenza, G, Scilingo, E. P, & Citi, L. (2014) *Electrodermal activity processing: A convex optimization approach.* (IEEE), pp. 2290–2293.

[35] Berntson, G. G, Cacioppo, J. T, & Quigley, K. S. (1995) The metrics of cardiac chronotropism: Biometric perspectives. *Psychophysiology* **32**, 162–171.

[36] Paulus, P. C, Castegnetti, G, & Bach, D. R. (2016) Modeling event-related heart period responses. *Psychophysiology* **53**, 837–846.

[37] Castegnetti, G, Tzovara, A, Staib, M, Paulus, P. C, Hofer, N, & Bach, D. R. (2016) Modeling fear-conditioned bradycardia in humans. *Psychophysiology* **53**, 930–939.

[38] Bradley, M. M, Codispoti, M, Cuthbert, B. N, & Lang, P. J. (2001) Emotion and motivation I: Defensive and appetitive reactions in picture processing. *Emotion* **1**, 276–298.

[39] Castegnetti, G, Tzovara, A, Staib, M, Gerster, S, & Bach, D. (2017) Assessing fear learning via conditioned respiratory amplitude responses. *Psychophysiology* **54**, 215–223.

[40] Pan, J & Tompkins, W. J. (1985) A real-time QRS detection algorithm. *IEEE. Trans. Biomed. Eng.* **32**, 230–236.

[41] Liu, Z, de Zwart, J. A, van Gelderen, P, Kuo, L.-W, & Duyn, J. H. (2012) Statistical feature extraction for artifact removal from concurrent fMRI-EEG recordings. *Neuroimage* **59**, 2073–2087.

[42] McDougal, D & Gamlin, P. (2008) Pupillary control pathways. *The Senses: A Comprehensive Reference* **1**, 521–536.

[43] Korn, C. W & Bach, D. R. (2016) A solid frame for the window on cognition: Modeling event-related pupil responses. *J. Vis.* **16**, 28.

[44] Korn, C. W, Staib, M, Tzovara, A, Castegnetti, G, & Bach, D. R. (2017) A pupil size response model to assess fear learning. *Psychophysiology* **54**, 330–343.

[45] Watson, A. B & Yellott, J. I. (2012) A unified formula for light-adapted pupil size. *J. Vis.* **12**, 12–12.

[46] Hayes, T. R & Petrov, A. A. (2015) Mapping and correcting the influence of gaze position on pupil size measurements. *Behav. Res. Methods* **48**, 510–527.

[47] Kret, M. E & Sjak-Shie, E. E. (2018) Preprocessing pupil size data: Guidelines and code. *Behav. Res. Methods* **51**, 1336–1342.

[48] Cacioppo, J. T, Tassinary, L. G, & Berntson, G. (2007) *Handbook of psychophysiology*. (Cambridge University Press).

[49] Wientjes, C. J & Grossman, P. (1998) Respiratory psychophysiology as a discipline: introduction to the special issue. *Biol. Psychol.* **49**, 1–8.

[50] Binks, A. P, Banzett, R. B, & Duvivier, C. (2006) An inexpensive, MRI compatible device to measure tidal volume from chest-wall circumference. *Physiol. Meas.* **28**, 149.

[51] Bach, D. R, Gerster, S, Tzovara, A, & Castegnetti, G. (2016) A linear model for event-related respiration responses. *J. Neurosci. Methods* **270**, 147–155.

[52] Yeomans, J. S, Li, L, Scott, B. W, & Frankland, P. W. (2002) Tactile, acoustic and vestibular systems sum to elicit the startle reflex. *Neurosci. Biobehav. Rev.* **26**, 1–11.

[53] Lang, P. J, Bradley, M. M, & Cuthbert, B. N. (1997) *Attention and orienting: Sensory and motivational processes*, eds. Lang, P. J, Simons, R. F, & Balaban, M. T. (Lawrence Erlbaum, Mahwah, NJ), pp. 97–135.

[54] Bach, D. R. (2015) A cost minimisation and Bayesian inference model predicts startle reflex modulation across species. *J. Theor. Biol.* **370**, 53–60.

[55] Brown, J. S, Kalish, H. I, & Farber, I. E. (1951) Conditioned fear as revealed by magnitude of startle response to an auditory stimulus. *J. Exp. Psychol.* **41**, 317–328.

[56] Khemka, S, Tzovara, A, Gerster, S, Quednow, B. B, & Bach, D. R. (2017) Modeling startle eyeblink electromyogram to assess fear learning. *Psychophysiology* **54**, 204–214.

[57] Schutz, A. C, Braun, D. I, & Gegenfurtner, K. R. (2011) Eye movements and perception: A selective review. *J. Vis.* **11**, 1–30.

[58] Xia, Y, Melinscak, F, & Bach, D. (2020) Saccadic scanpath length: an index for human threat conditioning. *Behav. Res. Methods* **53**, 1426–1439.

[59] Kleckner, I. R, Jones, R. M, Wilder-Smith, O, Wormwood, J. B, Akcakaya, M, Quigley, K. S, Lord, C, & Goodwin, M. S. (2018) Simple, transparent, and flexible automated quality assessment procedures for ambulatory electrodermal activity data. *IEEE. Trans. Biomed. Eng.* **65**, 1460–1467.