Journal of Statistical Software

January 2007, Volume 18, Issue 3. *http://www.jstatsoft.org/*

TIMP: An R Package for Modeling Multi-way Spectroscopic Measurements

Katharine M. Mullen
Vrije Universiteit Amsterdam

Ivo H. M. van Stokkum
Vrije Universiteit Amsterdam

Abstract

TIMP is an R package for modeling multiway spectroscopic measurements. The package allows for the simultaneous analysis of datasets collected under different experimental conditions in terms of a wide variety of parametric models. Models arising in spectroscopy data analysis often have some parameters that are intrinstically nonlinear, and some parameters that are conditionally linear on estimates of the nonlinear parameters. **TIMP** fits such separable nonlinear models using *partitioned variable projection*, a variant of the variable projection algorithm that is described here for the first time. The of the partitioned variable projection algorithm allows fitting many models for spectroscopy datasets using much less memory as compared to under the standard variable projection algorithm that is implemented in nonlinear optimization routines (e.g., the `plinear` option of the R function `nls`), as is shown here. An overview of modeling with **TIMP** is also given that includes several case studies in the application of the package.

Keywords: spectroscopy, separable nonlinear least squares, superposition model, compartmental model, global analysis, target analysis.

1. Introduction

TIMP[1] is an R package for modeling multiway spectroscopic measurements. It is a fully cross-platform problem-solving environment for fitting a wide range of models to spectroscopy data, and is freely available under the GNU General Public License (GPL).

This paper outlines the capabilities, structure, and application of the package. The introduction gives an overview of the problems in scientific model discovery that the package has been designed to address. Section 2 describes some aspects of the implementation, and in-

[1]The name of the package references its origins in the **tim** collection of FORTRAN routines developed over the past fifteen years by the latter author to model *tim*e-resolved spectroscopy data; **TIMP** stands for *tim package*.

cludes a description of the partitioned variable projection algorithm that allows application of the variable projection functional to parameter estimation problems in the absence of large memory resources. Section 3 describes in brief user-accessible functions. Section 4 describes general model options. Section 5 contains descriptions of kinetic models and their specification, fitting and validation with the package, and includes a case study in the application of a kinetic model to the simultaneous analysis of two datasets. Section 6 describes spectral models and their specification, fitting and validation in **TIMP**, and includes a case study in the application of a spectral model to the description of time-dependent change in spectral bandshapes. Section 7 discusses in brief the extension of **TIMP** to new model types. Conclusions are contained in Section 8.

1.1. Interactive scientific model-discovery

We term *scientific model discovery* the identification of a statistical model able to reproduce experimentally collected measurements to a satisfactory degree of accuracy, with the additional constraint that the model be well-interpretable according to physico-chemical theory. Scientific model discovery very often requires iterating the steps of formulation of a candidate model, model fitting, and model validation. Postulation of a candidate model is guided by *a priori* knowledge of the system underlying the data as well as by exploratory analysis of the dataset (e.g., with decomposition techniques like the singular value decomposition). Fitting provides estimates for free parameters that are more statistically likely than the starting estimates provided during postulation of the candidate model. Validation considers whether the fitted parameter values are precise and likely to be correct according to physico-chemical theory, whether the residuals are sufficiently small and unstructured, and whether adjustment of the candidate model is desirable. The cycle of model formulation, fitting and validation is often iterative because validation often results in the identification of a new candidate model.

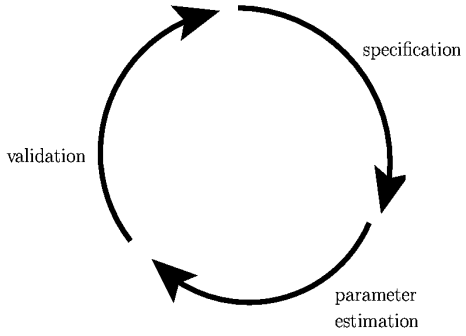


Figure 1: Scientific model discovery is often an iterative process of model specification, parameter estimation and validation.

Scientific model discovery is *interactive* in the case that the time to complete the model formulation, fitting, and validation cycle is determined primarily by the ability of the researcher to postulate and validate candidate models. To allow for interactive scientific model discovery the applied computer hardware and software must enable the researcher to quickly specify and fit a model, and must provide information for model validation that allows for efficient evaluation of model fit and physical feasibility. The primary goal of the **TIMP** package for the R system for statistical computing (R Development Core Team 2006) is to provide software for interactive scientific model discovery in the multiway spectroscopy data modeling problem domain.

1.2. Multiway spectroscopy data and models

Let Ψ_q denote a spectroscopic dataset arising under experimental conditions q. Ψ_q may be represented as the matrix

$$\Psi_q = \left[\begin{array}{c|cccc} & \lambda_1 & \lambda_2 & \dots & \lambda_n \\ \hline t_1 & \psi(t_1,\lambda_1) & \psi(t_1,\lambda_2) & \dots & \psi(t_1,\lambda_n) \\ t_2 & \psi(t_2,\lambda_1) & \psi(t_2,\lambda_2) & \dots & \psi(t_2,\lambda_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_m & \psi(t_m,\lambda_1) & \psi(t_m,\lambda_2) & \dots & \psi(t_m,\lambda_n) \end{array}\right] \tag{1}$$

Each row of Ψ_q is a spectrum in the spectral variable λ (which is often wavelength, but may be wavenumber or magnetic field strength; Laptenok, Mullen, Borst, van Stokkum, Apanasovich, and Visser (2007) describe the case that the variable is location). Spectra are represented at m instances of independent experimental variables t such as time, pH, pD, temperature, excitation wavelength or quencher concentration. The independent experimental variables are chosen so as to monitor spectral change in a manner that provides information on the dynamics of the underlying system.

Ψ_q represents a contribution from n_{ncomp} spectrally distinct components. The concentration and spectral property of each component may be represented as column l of matrices C and E, respectively, in the superposition model

$$\Psi_q = \left[\begin{array}{c|ccc} & 1 & \dots & n_{\text{ncomp}} \\ \hline t_1 & c(t_1,1) & \dots & c(t_1,n_{\text{ncomp}}) \\ t_2 & c(t_2,1) & \dots & c(t_2,n_{\text{ncomp}}) \\ \vdots & \vdots & \ddots & \vdots \\ t_m & c(t_m,1) & \dots & c(t_m,n_{\text{ncomp}}) \end{array}\right] \left[\begin{array}{c|ccc} & 1 & \dots & n_{\text{ncomp}} \\ \hline \lambda_1 & \epsilon(\lambda_1,1) & \dots & \epsilon(\lambda_1,n_{\text{ncomp}}) \\ \lambda_2 & \epsilon(\lambda_2,1) & \dots & \epsilon(\lambda_2,n_{\text{ncomp}}) \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_n & \epsilon(\lambda_n,1) & \dots & \epsilon(\lambda_n,n_{\text{ncomp}}) \end{array}\right]^\top \tag{2}$$

$$= CE^\top \tag{3}$$

Each column of C represents a concentration profile of a component in the independent variable t in which spectra are resolved. Likewise, each column of the matrix E represents a spectrum of a component.

In modeling multiway spectroscopy data Ψ_q, the inverse problem of recovery of the entries of C or E in terms of physically significant parameters (descriptive of, e.g., the decay rate of a component, or the location of the maximum of a spectrum) using Equation 3 is often of interest. Adequate parameterizations of either C or E are nonlinear, and are usually comprised of many submodels to represent various model aspects. Such parameterizations have been reviewed for the case of time-resolved spectroscopy data by van Stokkum, Larsen, and van Grondelle (2004). Very often, a model-based description is possible for either C or E, but not both matrices. Then the variable projection algorithm (Golub and LeVeque 1979; Golub and Pereyra 2003) allows the estimation of the entries of the matrix for which a model-based description is unavailable as conditionally linear parameters (clp). The resulting reduction in the size of the nonlinear parameter search space is very significant for problems arising in typical spectroscopy data modeling applications, as is discussed further in Section 2.3. **TIMP** includes an implementation of a refinement of the standard variable projection implementation, which we call *partitioned variable projection* that allows the application of the variable

projection functional to large estimation problems in the absence of large memory resources. Partitioned variable projection is discussed further in Section 2.3. Is is compared to the standard variable projection implementation (implemented in R in the `plinear` option of the `nls` function) in detail in Appendix A.

Multiway spectroscopy experiments often result in measured datasets $\Psi_1, \ldots, \Psi_x$ collected under similar but not identical experimental conditions $1, \ldots, x$. $\Psi_1, \ldots, \Psi_x$ may vary with one experimental condition, in which case the data is three-way, or with multiple experimental conditions, in which case the data is four-or-higher-way. In the latter case, distinct groups of datasets in $\Psi_1, \ldots, \Psi_x$ represent variance with respect to each experimental condition representing a dimension of the data Ψ as a whole, (for an example of five-way data Ψ resolved with respect to time, wavelength, temperature, pH, and polarization see e.g., van Stokkum and Lozier (2002)). It may be desirable to parameterize a model for all datasets $\Psi = \{\Psi_1, \ldots, \Psi_x\}$ so as to extract information regarding some model parameters from all available data, while fitting other model parameters on a per-dataset basis. Parameters fit per-dataset may then be used to describe the effects of variances in experimental conditions on the underlying system.

TIMP is designed to efficiently specify, fit and validate models for multiway spectroscopy data Ψ from possibly many experiments simultaneously. The package currently implements a wide variety of model options for the parameterization of kinetic and spectral models. Extension to other model types requires a minimum of additional code. An overview of the goals, structure and capabilities of the package is contained in this paper, along with two case studies its application.

1.3. An introduction to modeling with TIMP

The application of **TIMP** to a very simple parameter estimation task on a single simulated spectroscopy dataset will introduce the use of the package. As discussed in the previous section, a spectroscopy dataset Ψ_q can be considered to represent a superposition of the concentration profiles C and spectral properties E of components, so that $\Psi_q = CE^\top$ (Equation 3).

We will simulate a dataset in which two spectrally distinct components contribute to Ψ_q, where C represents concentration in time and E represents spectra resolved with respect to wavenumber. The simplest realistic model for C lets the time-profile of each component c_l be described by an exponential decay with decay rate parameter k_l, so that

$$c_l(t) = \exp(-k_l t), \tag{4}$$

where t is time. (for elaboration on the use of exponential models for kinetic processes, see e.g., the review by Istratov and Vyvenko (1999)). We let the two components contributing to Ψ_q have associated decay rate parameters .5 and 1, and let the concentrations be measured at 51 timepoints equidistant in the interval 0-2 ns. Then the following R commands calculate C.

```
R> C <- matrix(nrow = 51, ncol = 2)
R> k <- c(.5, 1)
R> t <- seq(0, 2, by = 2/50)
R> C[, 1] <- exp( - k[1] * t)
R> C[, 2] <- exp( - k[2] * t)
```

The most basic model for the spectrum e_l associated with a single component in wavenumber $\bar{\nu}$ is a Gaussian with parameters $\mu_{\bar{\nu}}$, $\Delta_{\bar{\nu}}$, and a_l, for the location, full width at half maximum (FWHM), and amplitude, respectively, so that

$$e_l(\bar{\nu}) = a_l \exp\left(- \ln(2) \left(2 \frac{(\bar{\nu} - \mu_{\bar{\nu}})}{\Delta_{\bar{\nu}}} \right)^2 \right), \tag{5}$$

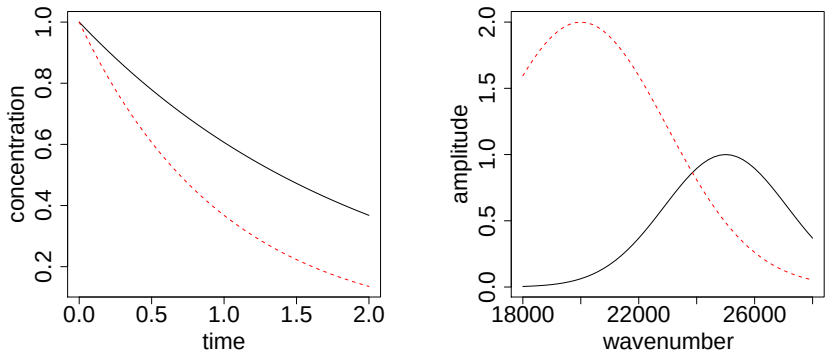
(see e.g., van Stokkum (1997) and references therein regarding the ubiquity of Gaussian models for spectra). Let us consider spectra represented by 51 wavenumbers equidistant in the interval 18000 - 28000 cm^{-1}, with locations 25000 and 20000, FWHMs 5000 and 7000, and amplitudes 1 and 2, respectively. In R we can then calculate E as

```
R> E <- matrix(nrow = 51, ncol = 2)
R> wavenum <- seq(18000, 28000, by=200)
R> location <- c(25000, 20000)
R> delta <- c(5000, 7000)
R> amp <- c(1, 2)
R> E[, 1] <- amp[1] * exp( - log(2) * (2 * (wavenum - location[1])/delta[1])^2)
R> E[, 2] <- amp[2] * exp( - log(2) * (2 * (wavenum - location[2])/delta[2])^2)
```

Given these R expressions for C and E, a dataset Ψ_q with Gaussian noise with zero mean and width $\sigma = .001$ may be simulated as

```
R> sigma <- .001
R> Psi_q  <- C %*% t(E) + sigma * rnorm(dim(C)[1] * dim(E)[1])
```

Plots of C and E are shown in Figure 2. The simulated dataset Ψ_q is shown in Figure 3



(a) (b)

Figure 2: (a) Simulated concentrations (b) simulated spectra. Component 1 is in black; component 2 is in red.

Given a dataset such as the simulated `Psi_q` shown in Figure 3 it is often the case that either a kinetic model for C or a spectral model for E is known to apply whose parameters are nonlinear. Given estimates for the nonlinear model parameters that determine one of the two matrices, the entries of the other matrix may be solved for as clp. Let us assume that the

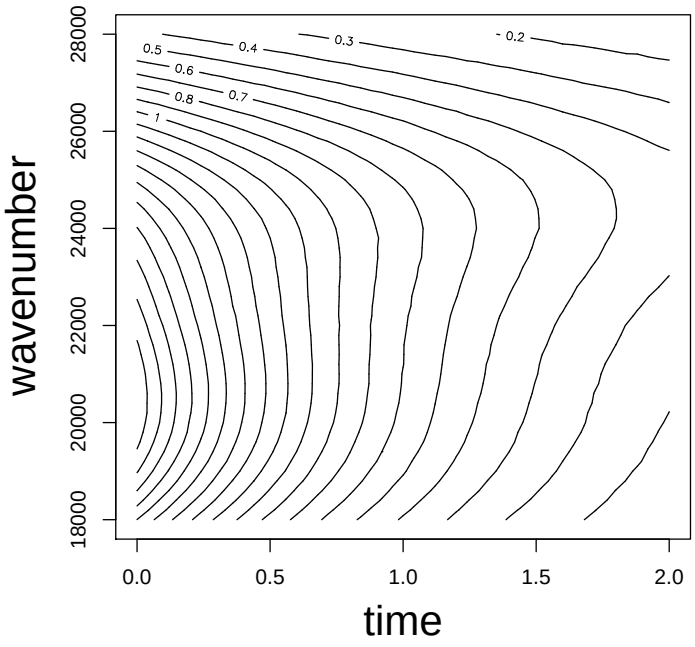


Figure 3: Simulated data; model fitting will resolve the two contributing components.

kinetic model used in simulating the data is known to describe `Psi_q`, and that approximate starting values for the two rate constants are known. Then the following R commands can be used to estimate the rate constants k and clp E.

TIMP is loaded with

```
R> library("TIMP")
```

The simulated dataset is placed into an instance of the class "dat" which is used to store data and model objects in **TIMP**. The "dat" object contains not only the data but also some information like its dimensions.

```
R> Psi_q_data <- dat(psi.df = Psi_q, x = t, nt = length(t),
+  x2 = wavenum, nl = length(wavenum))
```

A model to be applied to the data is initialized with the `initModel` function. The `seqmod=FALSE` option indicates that the components decay in parallel; the starting values for the decay rates are given in the vector `kinpar`.

```
R> kinetic_model <- initModel(mod_type = "kin", seqmod = FALSE,
+  kinpar = c(.1, 2))
```

With the next command model parameters are optimized over the course of four iterations using the `fitModel` function of **TIMP**.

```
R> kinetic_fit <- fitModel(data = list(Psi_q_data), model = kinetic_model,
+  opt = list(iter=4, paropt=list(mar=c(2,2,2,2),mgp=c(1,.2,0))))
```

The call to the `fitModel` function results in a composite plot displaying the fit of the model to the data at each wavenumber, a window showing parameter estimates for the two rate constants and information regarding the residuals, and a summary figure showing the estimated spectra and rates constants. For the case considered here the estimated spectra and concentration profiles well-approximate the entries of E and C used in simulation. **TIMP** employed the `nls` function in estimating the nonlinear parameters $k = \{k_1, k_2\}$; examination of the summary object returned by `nls`, as below, verifies that the standard errors of estimated parameters are low.

```
R> kinetic_fit$sumonls

Formula: 0 ~ rescomp(t, d)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
t1 0.4997107  0.0007933   629.9   <2e-16 ***
t2 1.0002352  0.0005581  1792.1   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.000999 on 2497 degrees of freedom

Number of iterations till stop: 4
Achieved convergence tolerance: 0.1151
Reason stopped: number of iterations exceeded maximum of 4
```

This introduction to the application of **TIMP** shows how, given a single dataset Ψ_q, a parametric description of the concentration profiles of contributing components can be fit, while the spectra E are solved for as clp. Subsequent sections will describe the application of **TIMP** to fitting more complex models to possibly many datasets.

1.4. Hierarchical models for possibly many datasets

Let $\Psi = \{\Psi_1, \ldots, \Psi_x\}$ denote multiway spectroscopy data collected over the course of possibly many experimental conditions indexed $1, \ldots, x$. Let Θ denote the nonlinear parameters of the multidataset model applied to Ψ. The goal of model fitting is to solve $\min \|\hat{\Psi}(\Theta) - \Psi\|_{F^2}$, e.g., to minimize the residuals associated with the fit of the model to the data. The model specifies a prescription to determine residual matrices $Z_1(\Theta), \ldots, Z_x(\Theta)$, where $Z_q(\Theta)$ are those residuals associated with dataset Ψ_q. $Z_1(\Theta), \ldots, Z_x(\Theta)$ are concatenated together in vectorized form to form the residual vector to be minimized with respect to Θ by a nonlinear optimization routine, (such as the `nls` function from R, which is employed by **TIMP**). The multidataset model may be such that residual matrix Z_q depends on only a subset of Θ. Parameters in Θ that determine the residuals associated with multiple datasets are said to be *linked*. Parameters that determine the residuals of only one dataset are said to be *unlinked*.

Models applied to data Ψ may be comprised of many submodels, each of which describe a distinct aspect of the underlying system giving rise to the measurements. Examples of submodels include a parametric description of an instrument response function (IRF), a coherent artifact, or the shape of a spectrum. Submodels may also include a prescription for the transformation of parameters in order to enforce constraints or apply relations between parameters, as illustrated in the case study contained in Section 5.6.

Submodels may themselves be comprised of submodels. For instance, a multidataset model is comprised of a model for each dataset, which is in turn comprised of submodels, for, e.g., an IRF, a kinetic model, etc. Description in terms of a tree is often well-representative of such parameterizations. As an example, a tree representation for a model describing two datasets $\Psi = \{\Psi_1, \Psi_2\}$ is given in Figure 4. The parameters associated with models for the individual datasets Ψ_1 and Ψ_2 are themselves comprised of submodels for various aspects of the underlying system. Note that distinct submodels may depend on the same parameter $\theta_i \in \Theta$, so that parameters are linked between datasets, as Figure 5 illustrates diagrammatically.

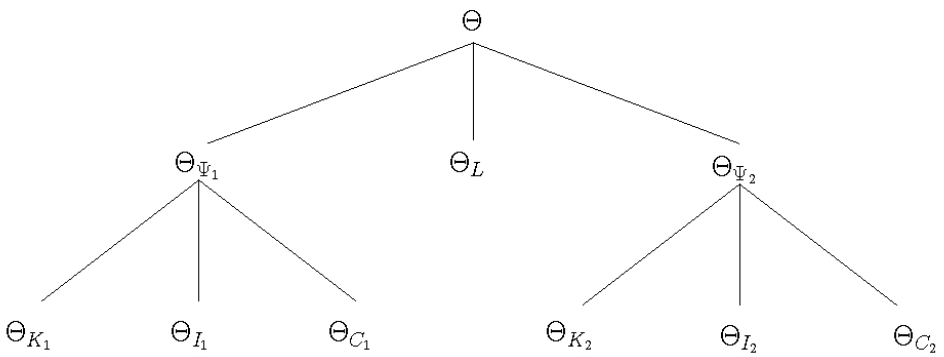


Figure 4: A hierarchical model Θ for datasets $\Psi = \{\Psi_1, \Psi_2\}$. The model is comprised of a submodel for each dataset with associated nonlinear parameters $\Theta_{\Psi 1}$ and $\Theta_{\Psi 2}$. These submodels are each comprised of a submodel for the kinetic decay rates parameterized with Θ_{K1} and Θ_{K2}, a submodel for the IRF, parameterized with Θ_{I1} and Θ_{I2}, and a submodel for a coherent artifact parameterized with Θ_{C1} and Θ_{C2}.

2. The implementation of TIMP

TIMP has been designed to facilitate interactive scientific model discovery of the multidataset hierarchical models introduced in Section 1.4 and reviewed in van Stokkum *et al.* (2004). In this section we attempt to give a brief overview of the design of the package.

2.1. The role of S4 classes and methods in TIMP

S4 classes and methods are the preferred means by which object-oriented programming is implemented in R. See e.g., Chambers (1998) for a review of the S4 classes and methods

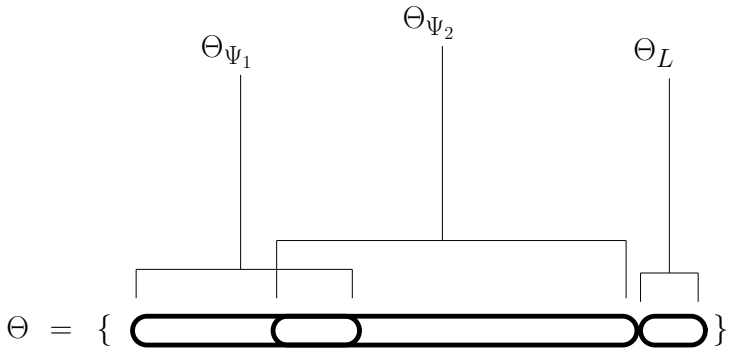


Figure 5: The vector of nonlinear model parameters Θ may parameterize a model for multiple datasets. Some parameters $\theta \in \Theta$ may be used to model more than one dataset, so that θ is *linked* between datasets, while other parameters may be used to model only one dataset. Still other parameters may be used to model a relationship *between* datasets, such as a linear scaling. In the above figure, the vectors of parameters Θ_{Ψ_1} and Θ_{Ψ_2} parameterizing the models for datasets Ψ_1 and Ψ_2, respectively, include some of the same elements from the full vector of nonlinear model parameters Θ. Parameters to determine a relationship between datasets are written as Θ_L, and apply to how the model functions associated with datasets are scaled.

system, and e.g., Bates and DebRoy (2003) for a description of their utility within a large R package.

S4 classes and methods are central to the implementation of **TIMP**. Slots of S4 classes are used to store each component of a hierarchical model. In the course of optimization (initiated with the `fitModel` function), an object is associated with the parameterization of the multidataset model and the current values of the parameters associated with this model, respectively. The object containing a prescription for the multidataset model is initialized to the (hidden) global variable `.currModel`, and is of class `multimodel`. The object used to store the current parameter estimates associated with this model is initialized to the (hidden) global variable `.currTheta`, and is of class `multitheta`.

During each iteration of parameter estimation, **TIMP** updates the object representing the current nonlinear parameter estimates `.currTheta` using the updated vector of non-linear parameter estimates and the model prescription object `.currModel`. The update is performed by the function `getThetaCl`. The object `.currTheta` is subsequently used to form the residual vector to be minimized with respect to the nonlinear parameter vector. By determining the residuals as a function of `.currTheta` as opposed to as a direct function of the "raw" vector of nonlinear parameter estimates Θ (which is typically of length $10^1 - 10^2$), the residual function implementation is relatively concise and readable. All bookkeeping necessary to account for

S4 classes defined by **TIMP**

class	represents
`dat`	single dataset Ψ_q and an associated model
`kin`	kinetic model specification, inherits from `dat`
`spec`	spectral model specification, inherits from `dat`
`theta`	parameter estimates associated with model for single dataset Ψ_q
`multitheta`	parameter estimates associated with (possibly) multiple dataset model
`multimodel`	all information associated with multiple dataset model and fit
`res`	results of fitting (possibly) many datasets

Table 1: Classes currently used in **TIMP**. These classes are initialized in the package by functions prefaced by `init.`, followed by the name of the class in the table above.

fixed parameters, parameter relations, and parameter constraints is performed in `getThetaCl`, not the residual function, further facilitating its implementation and readability. This also allows for the rapid implementation of support for model types associated with new residual functions, as elaborated in Section 7.

As elaborated in Section 4, aspects of model parameterization not specific to model type (e.g., a weighting specification, constraints, or a specification of fixed parameter values) are specified in the slots of the class `dat`. Aspects of model parameterization specific to a given sort of model (e.g., a model in which the nonlinear parameters apply to the kinetics) are specified in the slots of classes inheriting from *dat*. S4 methods switch the definition of the function that determines the residuals and the output/plotting based on the class of the model type.

2.2. Model specification

Model specification in **TIMP** has possibly two steps. First, a model applicable to one dataset in $\Psi = \{\Psi_1, \ldots, \Psi_x\}$ is specified via the function `initModel`. If no per-dataset model differences are required, the model and the dataset list become arguments to the function `fitModel`, the call of which fits the model to the data, and outputs this fit and information for model validation. In the case that the model varies per-dataset, a second specification step describing per-dataset model differences from the model defined in `initModel` is required. The description of per-dataset model differences takes the form of a list input to `fitModel`. An example of this two-step process is found in Section 5.6.

Note that when the `fitModel` function is called, its arguments are used to initialize a list of objects of class `dat`, stored in the slot `modellist` of the (hidden) global variable `.currModel`. Element i of `modellist` stores dataset i and the model associated with dataset i. Many **TIMP** functions index into `modellist`.

Breaking the model specification into two steps allows the datasets associated with a given model to be specified just prior to fitting, as an argument to the `fitModel` function. This is convenient for applying the same model to many different combinations of datasets, but is not convenient when there are many differences between datasets, as the differences must be respecified with each call to `fitModel`. In the next version of **TIMP** a function to allow multidataset model specification to be performed in a single step will be included to avoid this inconvenience. This function will also facilitate the specification of model differences between *groups* of datasets.

2.3. Parameter estimation

The variable projection algorithm that allows for the solution of separable nonlinear parameter estimation problems is central to parameter estimation with **TIMP**. Separable nonlinear parameter estimation problems often arise in spectroscopy data modeling due to the bilinear form of the superposition model $\Psi = CE^\top$, and the fact that it is often possible to well-describe either the concentration profiles C or the spectra E, but not both matrices, with a parameterized model. By separating the parameters into intrinsically nonlinear parameters and clp descriptive of the entries of the matrix that is not described in terms of a parametric model, the nonlinear search space is often very significantly reduced, since datasets Ψ are often large (of dimension on the order of $10^3 \times 10^3$) with on the order of 10^1 contributing components. Furthermore, a parametric description of C or E is also often of interest in its own right, insofar as the model and parameter estimates may be assigned physical significance, and thereby serve as a simplified system description.

Let X denote the matrix (either C or E) directly determined by nonlinear parameters Θ, and let β denote the matrix determined as conditionally linear on X. The estimation problem associated with fitting Θ to $\Psi = X(\Theta)\beta$ such that the residuals $\| \Psi - X(\Theta)\beta \|_2$ are minimal may be formulated as

$$\text{Minimize} \| (I - X(\Theta)X^+(\Theta))\Psi \|_2 . \tag{6}$$

This is the variable projection functional for which Golub and Pereyra (1972, 1973) determined an analytical gradient by deriving the derivative of the pseudoinverse X^+.

Given starting estimates for Θ, a solution to Equation 6 may be approached iteratively using gradient-based techniques. **TIMP** uses a finite difference approximation for the gradient.

TIMP's parameter estimation technique to solve Problem 6 can be summarized as follows, where "convergence" is some appropriate stopping criterion.

Algorithm Finite Difference VarPro:

1. choose starting Θ approximately
2. *for* $s := 1, 2 \ldots$ *until* convergence *do*
   determine gradient in Θ-space with finite diff. approx. of $\frac{d(I-XX^+)}{d\Theta}\Psi$
   $\Theta_{s+1} :=$ step$(\Theta_s, \text{gradient}, \ldots)$

Implementation in **TIMP** of the finite difference variable projection algorithm described above is via the `nls` function of R. `nls` is given as input starting values for Θ and a residual function that returns the vectorized version of $(I - X(\Theta)X^+(\Theta))\Psi$. Then the `numericDeriv` function is used by `nls` to determine the finite difference approximation of the gradient of this residual vector in Θ-space. The determination of the step-size to move Θ in the gradient direction each iteration (the algorithm step) is also performed by `nls`. Section 2.4 describes the extension of this methodology to the case in which X is dependent on conditions such as the wavelength or timepoint of the data, and to multiple datasets.

2.4. Partitioned variable projection

For application to multiway spectroscopy modeling problems domain we seek to address, the algorithm for finite difference variable projection given in the previous section must be

generalized to the case of simultaneously modeling multiple datasets, and to the case where the model for the matrix X is dependent on condition, where a condition is an instance of a variable with respect to which the data is resolved, such as wavelength, time, temperature, etc. The desired extension must operate on (i.e., perform QR-decomposition on, for instance) small matrices, so as to allow parameter estimation to be performed on computer systems lacking large memory resources.

Let X_q be the matrix determined by nonlinear parameters Θ that is associated with dataset Ψ_q. The model for X_q may be a function of Θ that is different for distinct datasets in $\Psi_1, \ldots, \Psi_x$. The clp may be equated between datasets, so that $\beta_1 = \beta_2 = \ldots = \beta_x$ (the "link clp" case in the following description of PartitionedVarPro) or the clp may be estimated per-dataset, or per-group of datasets, so that $\beta_p \neq \beta_q$ for datasets Ψ_q and Ψ_p in Ψ. These two possibilities for the clp correspond to the equations

$$\begin{pmatrix} \Psi_1 \\ \vdots \\ \Psi_x \end{pmatrix} = \begin{matrix} X_1 \\ \vdots \\ X_x \end{matrix} \beta = X_{\text{super}} \ \beta \tag{7}$$

and

$$\begin{pmatrix} \Psi_1 \\ \vdots \\ \Psi_x \end{pmatrix} = \begin{matrix} X_1\beta_1 \\ \vdots \\ X_x\beta_x \end{matrix} \tag{8}$$

In the case that the matrix X_q is dependent on a data condition (as in wavelength-dependent kinetic models, for instance, or in time-dependent spectral models), X_q must be re-calculated for every condition with which it varies, so that Models 7 and 8 are

$$\psi_{\text{super } p} = \begin{pmatrix} \psi_{1p} \\ \vdots \\ \psi_{xp} \end{pmatrix} = \begin{matrix} X_{1p} \\ \vdots \\ X_{1p} \end{matrix} \beta_p = X_{\text{super}} \ \beta_p \tag{9}$$

and

$$\begin{pmatrix} \psi_{1p} \\ \vdots \\ \psi_{xp} \end{pmatrix} = \begin{matrix} X_{1p}\beta_{1p} \\ \vdots \\ X_{xp}\beta_{xp} \end{matrix} \tag{10}$$

where p is the condition index, ψ_{qp} is a vector comprising the pth row or column of dataset Ψ_q, β_{qp} is the pth row of β_q, and X_{qp} is associated with dataset q at condition p.

In general p need not represent a single condition, but may represent a collection of conditions (e.g., wavelengths or times) for which the matrix X is constant.

The residuals associated with Equations 9 and 10 may be calculated using the variable projection function under the partitioned variable projection algorithm. This algorithm is presented for the general case in which p may represent a single condition or a group of conditions associated with the same prescription for X. A group of conditions associated with the same

prescription for X is termed in the algorithm description a *part*. If Equation 7 or 9 are desired, then (link clp) is true. If Equation 8 or 10 are desired, then (link clp) is false.

Note that the variable projection functional is $(I - CC^{+})\Psi = Q_2 Q_2^{\top}\Psi$ using the QR-decomposition, and that these residuals are formulated in Q-space as $Q_2^{\top}\Psi$.

```
PartitionedVarPro(model, Θ)
  for parts p in 1:n:
      if(link clp)
          get(X_super p( model, Θ))
          QR(X_super p)
          append Q_2^T ψ_super p to Z
      else
          for q in 1:x
              get(X_pq( model, Θ))
              QR(X_pq)
              append Q_2^T ψ_pq to Z
  return: Z
```

After formation of the residual vector Z, a finite difference method may be applied to determine an update of Θ as described in Section 2.3 for Finite Difference VarPro.

The ability to apply the variable projection functional without operating on large matrices is the main motivation for introduction of PartitionedVarPro. PartitionedVarPro results in the same residuals as under the standard variable projection algorithm as implemented in, e.g., the `plinear` function of `nls`. The advantage of PartitionedVarPro is that significantly less memory resources are required, allowing application of the algorithm on large datasets on a personal computer. The memory resources required by PartitionedVarPro and the standard variable projection implementation are considered in detail in Appendix A.

To examine the implementation of partitioned variable projection in **TIMP**, see the `rescomp` function which collects the residual vector Z and one of the `S4` methods for `residPart`, which returns the residuals (with a contribution from possibly many datasets) associated with a single part p.

2.5. Validation

Model validation in **TIMP** may be performed via a variety of means. Nonlinear parameter estimates as returned by `nls` may be validated via the linear approximation standard error estimates returned. Relatively large standard errors indicate an over-parameterization of the model.

For each model type, an `S4` method `plotter` is implemented to output model type-specific results, both in the form of plots and in the form of ASCII files representing the model fit and other estimates.

Analysis of the residuals is an important aspect of model validation. This analysis is often facilitated by taking a singular value decomposition (SVD) of the residual matrix, which allows structure in the misfit of the model to the data to be readily observed. Also important to an evaluation of the model fit are plots of the fit of the model to the data for each row or

column of the data (possibly for each of multiple datasets).

It is often desirable in scientific modeling applications to perform model validation after the application of the fitting algorithm (in the case of **TIMP**, *nls*) results in the satisfaction of *stopping* criteria, as opposed to convergence criteria. For example, it is often desirable to validate the model fit after a set number of iterations, or at the starting values of parameters to be optimized. In order to allow validation to be performed after a set number of iterations, the *nls* function of R was extended with new options, which are described in Appendix B. These options allow the `fitModel` of **TIMP** to return information validating model fit after any desired number of iterations.

3. User-accessible functions

TIMP is currently structured around five core user-accessible functions, described in this section in turn. More complete information regarding function arguments and output is found in the `help` functions of the package; here a higher-level description of the purpose and structure of the arguments and output is given.

3.1. `readData`

`readData` takes as an argument a string containing the path to an ASCII-file containing data and reads the data into R. There are currently three supported data formats, which are described in Appendix C. The data (and inferred attributes, such as the number of wavelengths by which spectra are represented, etc.) are returned as an object of class `dat`.

3.2. `preProcess`

`preProcess` takes an argument of class `dat` and a specification of the desired data sampling, selection, baseline correction, or axis scaling, and the dimension in which to perform the preprocessing (which may be the spectral dimension or the dimension in which spectra are resolved, e.g., time). The `preProcess` function returns an object of class `dat`.

3.3. `initModel`

The `initModel` function is used to specify a model. A string `mod_type` giving the class of the model being specified, (`"kin"` for kinetic models, `"spec"` for spectral models, and so forth) is a mandatory argument. Additional arguments may be any model options described in the `help` page for the `dat` class, plus options described on the `help` page of the desired model class given as `mod_type`. Output is an object of the desired model class, which inherits from `dat`.

3.4. `fitModel`

`fitModel` performs optimization of a model to an arbitrary number of datasets. Arguments to `fitModel` include a list of datasets to be fit, a model that is applied to all datasets, a list specifying any model differences to apply per-datasets, and a list of control and printing options. A call to this function results in optimization of free parameters. By default results are then printed to the screen, and optionally to text files and/or postscript. Returned is a list

whose elements include the output of the call to the function `nls` that is used to iteratively improve the starting estimates for nonlinear parameter values.

3.5. `examineFit`

`examineFit` takes as input the list returned by `fitModel` and re-calls the plotting and functions to write output. This function is useful for the comparison of fit of several models. An output object is not returned.

4. General model options

This section seeks to outline at a higher level than that found in the package's `help` pages model parameterization options that may be applied to all model types. The specification of each such option becomes a slot in the class `dat`, possibly after processing (within the `initModel` or `getModel` functions). Options that are specific to a given model type that inherits from `dat` (e.g., the class `kin` for kinetic model options or the class `spec` for spectral model options) are described in later sections.

The function `initModel` takes as arguments a specification of model options. Note that for multidataset models, any aspect of a parameterization of a model possible to specify in `initModel` may be modified, removed or added to the prescription of per-dataset model differences given as the `modeldiffs` argument to the fitting function `fitModel`.

4.1. Data weighting

A weighting scheme W has the form of an m by n matrix (where m by n is the dimension of the data matrix Ψ). W may lessen the weight of portions of the data known to contain less information regarding the model parameters. Such data may result from noisy experimental conditions, for instance. The application of a weighting scheme may also be desirable from first principles, e.g., in the case that the data are known to have a variance related to their magnitude as in single photon counting (SPC) experiments, in which the data are Poisson distributed, with

$$\hat{\sigma}_{\Psi(t_i,\lambda_j)} = \sqrt{\Psi(t_i,\lambda_j)}. \tag{11}$$

For the case of Poisson distributed count data, $W(i,j)$ is

$$W(i,j) = \frac{1}{\hat{\sigma}_{\Psi(t_i,\lambda_j)}}. \tag{12}$$

Once W has been determined, the Hadamard product (element-by-element product) of W and Ψ is taken, so that

$$\Psi^W(t_i,\lambda_j) = \Psi(t_i,\lambda_j) * W(t_i,\lambda_j) \tag{13}$$

which we write as

$$\Psi^W = \Psi \circ W \tag{14}$$

In the case that Ψ is transformed by a weight matrix W into Ψ^W, the associated concentration matrix becomes C^W where

$$C^W_{\lambda_j}(t_i,l) = C_{\lambda_j}(t_i,l)W(t_i,\lambda_j) = C \circ W \tag{15}$$

Ψ^W and C^W may then be used in place of Ψ and C in parameter estimation.

Specification in **TIMP***: Weighting*

The list argument `weightpar` specifies a prescription for the matrix of weights to be applied to the dataset. `weightpar` is a list of vectors. The vectors have form `c(first_x, last_x, first_x2, last_x2, weight)`. `first_x` and `last_x` are the least and greatest timepoints (or other variable with which spectra are resolved) having weight `weight`; `first_x2` and `last_x2` are the least and greatest values of the spectral variable having weight `weight`.

Note that if vector elements 1-4 are `NA`, the first-most point of the data is taken for elements 1 and 3, and the last-most points are taken for 2 and 4. For example, for a dataset in which spectra are measured in wavelength at many different times in picoseconds, the specification `weightpar = list(c(40, 1500, 400, 600, .9), c(NA, NA, 700, 800, .1))` will weight data between 40-1500 ps and 400 and 600 nm by .9, and will weight data at all times between 700 and 800 nm by .1.

For single photon counting data or other types of count datasets, `weightpar = list(poisson = TRUE)` will apply Poisson weighting to all non-zero elements of the data.

4.2. Fixed parameters

It is often of interest to set nonlinear model parameters to fixed values. Fixing model parameters may make use of *a priori* knowledge of true parameter values, or may be performed to decrease the number of free parameters of the model.

Specification in **TIMP***: Fixed parameters*

Every model parameterization option with an associated list or vector of nonlinear parameter starting values is named. For instance, the name of the starting values for kinetic decay rates is `"kinpar"`. In order to fix nonlinear parameters, the name of their list or vector of starting values is given, along with the indices into the list or vector at which parameter values should be fixed. This specification is contained in a list `fixed`.

For instance `fixed = list(kinpar = c(1,3,5), parmu = list(c(1,1), c(1,2), c(1,3)))` will fix the 1st, 3rd, and 5th elements of the `kinpar` vector of starting values for kinetic decay rates, and the 1st, 2nd, and 3rd elements of the 1st list of parameters in the `parmu` list of starting values for parameters describing wavelength-dependence of the IRF.

4.3. Constraint of clp

The basic superposition model $\Psi = CE^\top$ (Equation 3) is of bilinear form, where the matrix C describes concentrations and the matrix E describes spectra. A nonlinear model may be used to describe either C or E, and the entries of the remaining matrix estimated as clp, as described in Section 2.3.

It is often desirable to constrain clp to account for *a priori* knowledge or to reduce the number of free parameters in the model. **TIMP** currently allows clp to be constrained to a linear relationship with a scaling parameter (that may be fixed at 1 to equate clp), or to be constrained to zero.

It is often useful to name the clp to be constrained in terms of the component they represent, (i.e., the column of C or E they are contained in).

Specification in **TIMP***: Constraint of clp to zero*

The list `clp0` contains lists that specify clp to constraint to zero. The elements of these lists are named `low`, `high`, `comp`, specifying the least and greatest absolute values of the clp dimension to constrain to zero, and the component to which to apply the zero constraint, respectively. For example, where clp represent spectra in wavelength, `clp0 = list(list(low=400, high = 600, comp=2), list(low = 600, high = 650, comp=4))` applies zero constraints to the spectra associated with component 2 between 400 and 600 nm, and to the spectra associated with component 4 between 600 and 650 nm.

Specification in **TIMP***: Constraint of clp to a linear relationship*

The list `clpequspec` contains lists that specify collections of clp to relate. The elements of these lists are named `to`, `from`, `low`, `high`. An optional element named `dataset` specifies the dataset from which to get the reference clp determining the relationship. `to` is the component from which clp are to be fixed in relation to clp from some other component; `from` is the reference component. `low` and `high` are the least and greatest absolute values of the clp dimension to constrain. For example, where clp represent spectra in wavelength, `clpequspec = list(list(low = 400, high = 600, to = 1, from = 2))` will constrain the spectra associated with the first and the second components to equality between 400 and 600 nm according to $\epsilon_1 \leftarrow \epsilon_2\theta_\epsilon$ where ϵ_1 and ϵ_2 are the spectra associated with components 1 and 2, $\leftarrow$ indicates that ϵ_1 is dependent on ϵ_2, and θ_ϵ parameterizes the linear relation.

The vector `clpequ` contains `length(clpequspec)` numerics, where the ith numeric is a starting value θ_{ϵ_i} parameterizing the linear relation specified between clp by the ith list in `clpequspec`. Fixing the starting value of an element of `clpequ` at 1 constrains the associated clp to equality.

4.4. Relations between nonlinear parameters

It may be desirable to enforce a relationship between two nonlinear parameters θ_1 and θ_2 so that $\theta_2 = f(\theta_1)$. The relationship of nonlinear parameters is usually performed in order to take into account *a priori* knowledge of the system being modeled.

A linear relationship between parameters is currently implemented, (and other functional relationships will be added).

Specification in **TIMP***: Nonlinear parameter relations*

As in the case of fixing parameters, in order to relate nonlinear parameters, the name of the associated list or vector of starting values is given, along with the indices into the list or vector at which parameter values are related. This specification is contained in a list `prelspec`.

Each element of `prelspec` is a list having elements named `what1` (a character string describing the parameter type to relate, e.g., `"kinpar"`), `what2` the parameter type on which the relation is based; usually the same as `what1`), `ind1` (an index into `what1`) and `ind2` (an index into `what2`), and the optional argument `rel` (a character string to specify the functional relation type, by default `"linear"`). For examples, `prelspec = list(list(what1 =`

`"kinpar", what2 = "kinpar", ind1 = 1, ind2 = 5))` relates the 1st element of `kinpar` to the 5th element of `kinpar` according to `kinpar[2]` ← `kinpar[1]` where ← indicates that `kinpar[2]` is dependent on `kinpar[1]`.

The vector `prel` is of length `length(prelspec)` and contains numeric starting values parameterizing the linear relationships described in `prelspec`.

4.5. Constraint of nonlinear parameters to positivity

It may be known *a priori* that the nonlinear parameters associated with a submodel should be positive. Then optimizing on the log of these parameters and transforming the results by exponentiation of the estimated parameters is a means of enforcing the desired constraint.

Specification in **TIMP**: *Constraint of parameters to positivity*

The vector `positivepar` includes a character string containing the name of the vector or list of starting parameters that should be constrained to positivity, e.g., `positivepar = c("kinpar")`.

5. Kinetic models

Kinetic models describe the concentrations of components in time. For multiway spectroscopy data modeling applications the basic model for the kinetics of each component (i.e., each column of the matrix C) is an exponential decay in time t, so that

$$\Psi = CE^\top = \sum_{l=1}^{n_{\text{ncomp}}} c_l \epsilon_l^\top = \sum_{l=1}^{n_{\text{ncomp}}} (\exp(-\phi_l t) \star i(t)) \epsilon_l^\top \tag{16}$$

where i is the instrument response function (IRF) and $\star$ is convolution. The parameter estimation problem of optimal values for the amplitudes ϵ_l of the exponential decays (i.e., the spectra E) along with the decay rates ϕ_l under least squares criteria is called the multiexponential analysis problem, and is ubiquitous in physics applications in which data is modeled by the solution of first-order differential equations, as Istratov and Vyvenko (1999) review.

Kinetic models are represented in **TIMP** with the class `kin`. Options for the parameterization of kinetic models in **TIMP** are here outlined.

5.1. Model for the decay of components

The basic model for the concentration matrix C is a sum of n_{ncomp} exponential decays parameterized as $\Theta_K = (k_1, \ldots, k_{n_{\text{ncomp}}})$, so that the entries of the concentration matrix $C(i,l)$ are given as

$$C(i,l) = e^{-k_l t_i}. \tag{17}$$

Specification in **TIMP**: *Kinetic decay rates*

The vector `kinpar` contains numeric starting values for the kinetic decay rates, which in the absence of a compartmental scheme parameterize exponential decays. The number of values given determines n_{ncomp}, the number of components dedicated to modeling kinetic decays.

5.2. Instrument response models

Multiway spectroscopy experiments often employ a short laser pulse to excite the system under study and measure the resulting spectra in time. The convolution of the shape of this exciting pulse and the detector response is the IRF. With pump-probe spectroscopy the IRF is given by the convolution a pump and a probe pulse. With Gaussian-shaped pump and probe pulses, the convolution of the two will again be Gaussian-shaped, but with an increased width.

An IRF may either be parameterized in the model, or measured. A parametric model for the IRF i as a Gaussian in the time dimension t gives the following model for the concentration matrix C

$$C(t, k_l, \mu, \Delta) = e^{-k_l t} \star i(t) = \frac{e^{-k_l t}}{2} e^{k_l(\mu + k_l \tilde{\Delta}^2/2)} \left\{ 1 + erf \left[\frac{t - (\mu + k_l \tilde{\Delta}^2)}{\sqrt{2}\tilde{\Delta}} \right] \right\} \tag{18}$$

where t is time, k_l is the lth kinetic component, μ and Δ are the location and full width half maximum (FWHM) parameters of the Gaussian distribution, respectively, $\tilde{\Delta}$ is the Gaussian width parameter (such that $\tilde{\Delta} = \Delta/2\sqrt{2\ln 2}$), and $\star$ is convolution. Recall that the FWHM of the Gaussian distribution is related to the standard deviation σ as $FMWM = 2\sqrt{2\ln 2}\sigma$.

In the case that a measured IRF is used as opposed to a model for the IRF with parameters to be estimated, its numerical convolution with the exponential decay function yielding the kinetic decays is required. **TIMP** allows for either methods of including the effects of the IRF to be applied.

Specification in **TIMP***: Gaussian IRF*

The vector `irfpar` contains starting values for the parametric description of the IRF in terms of a Gaussian. The vector is ordered `irfpar = c(`μ, Δ`)`. For example, `irfpar = c(-2, .05)` specifies a starting location parameter μ as -2 and a width Δ as .05, where the starting values are in the unit of time of the timepoints in the data, e.g., picoseconds.

Specification in **TIMP***: Measured IRF*

The vector `measured_irf` is of length equal to the number of timepoints in the data, and contains the measured IRF; if specified, it will be applied to the data.

The integer `convalg` is between 1-4 and determines the numerical convolution algorithm used. `convalg=1` is the default and is recommended.

5.3. Models for dependence of IRF parameters on spectral variable

In the case that an IRF is included in the model using a parametric description, it is often desirable that the parameters involved are dependent on spectral variable (e.g., wavelength). The dependence of IRF on the spectral variable is termed dispersion.

Time-gated spectra are measurements of an optical property (e.g., emission or absorption) as a function of a spectral variable like wavelength taken simultaneously across some range, repeated at many distinct times. In kinetic models of time-gated spectra, dispersion is often well-modeled as a smooth function of the spectral variable. Polynomial functions of degree

nparmu and *npartau* are often used to describe the variance in the spectral variable of the IRF location parameter μ and FWHM parameter Δ, the coefficients of which become additional parameters of Θ_I. Then the IRF μ and Δ are calculated per wavelength as

$$\mu(\lambda) = \mu_0 + \sum_{i=1}^{nparmu} \mu_i \left(\frac{\lambda - \lambda_c}{100} \right)^i, \tag{19}$$

$$\Delta(\lambda) = \Delta_0 + \sum_{i=1}^{npartau} \Delta_i \left(\frac{\lambda - \lambda_c}{100} \right)^i \tag{20}$$

where μ is a function that takes a wavelength or wavenumber and gives the location of the IRF, μ_0 is the location of the IRF at λ_c, Δ is a function that takes a wavelength or wavenumber and gives the scaled width of the IRF, and Δ_0 is the scaled width of the IRF at λ_c.

For data collected by measuring decay traces at many different wavelengths dispersion is often well-modeled by shift parameters for μ and Δ per-wavelength. Where nl is the number of points whereby spectra are represented, this results in nl shift parameters parameterizing the dispersion of μ and nl shift parameters parameterizing the dispersion of Δ. We refer to models of dispersion employing a shift parameter per-wavelength as *discrete*.

Specification in **TIMP***: Dispersion models*

The character strings `dispmufun` and `disptaufun` determine the functional form of the dispersion of the IRF location parameter. The default is a polynomial description. If `dispmufun` or `disptaufun` is equal to "discrete" a discrete model for dispersion of the corresponding variable is applied. For the discrete case `parmu` or `partau` should contain a starting value for the shift for every point by which spectra are represented (e.g., for each wavelength).

The numeric `lambdac` is supplied if a polynomial description of either dispersion of location or FWHM is applied. Then `lambdac` is the center-wavelength in this description (λ_c in Equations 19 and 20).

The list `parmu` contains starting values for the parameters of the model for dispersion of the IRF location. The vector `partau` contains starting values for the parameters of the model for dispersion of the IRF FWHM.

5.4. Coherent artifact/scatter models

Should the measurements contain an instantaneous response or coherent artifact due to Raman scatter, it may be desirable to include in the model for the concentrations C its description in time. This is done by adding components (which are columns of the C matrix) to represent its contribution.

A commonly used model for coherent artifact/scatter has the time characteristics of the IRF, in which case a single column is appended to the concentration matrix with the IRF time profile. A variation of this model maintains separate coherent artifact spectra for each of x datasets Ψ_q.

Another commonly used model type employs a sequential scheme with, e.g., femtosecond lifetimes, in which the signs of the amplitude of consecutive components alternates. This model type often well-describes an oscillatory coherent artifact. In the case that the instantaneous

response of the exciting pulse has both scatter and coherent response components, a linear superposition of the model that follows the IRF time profile and a model based on a sequential scheme may be applied.

An aside on the implementation of ultra-fast coherent artifact lifetimes

Models for the coherent artifact model type that employ a sequential kinetic scheme are often well-fit with ultra-fast lifetimes, and hence large exponential decay rates. For very large decay rates under Equation 18, it may be desirable to employ the exponentially scaled error function $\exp(x^2)erfc(x)$, as in the `decayirf` function of **TIMP**. An implementation of the exponentially scaled error function (*erfce*) is currently unavailable within R. Its use in **TIMP** is made possible by porting the implementation found in the Cephes Mathematical Library (Moshier 1992) to an R shared library in C.

Specification in **TIMP***: Coherent artifact/scatter models*

The list `cohspec` describes the model for coherent artifact/scatter component(s). If `cohspec$type` is `"irf"`, the coherent artifact/scatter has the time profile of the IRF. If `cohspec$type` is `"freeirfdisp"` the coherent artifact/scatter has a Gaussian time profile whose location and width are parameterized in the vector `coh` (independent from the IRF parameters). If `cohspec$type` is `"irfmulti"` the time profile of the IRF is used for the coherent artifact/scatter model, but the IRF parameters are taken per dataset (for the multidataset case), and `cohspec$numdatasets` must be equal to the number of datasets modeled. If `cohspec$type` is `"seq"` a sequential exponential decay model is applied, whose parameters are contained in `coh`. If `cohspec$type` is "mix" a sequential exponential decay model is applied along with a model that follows the time profile of the IRF; the coherent artifact/scatter contribution is then a linear superposition of these two models.

The vector `coh` contains starting values for the parameterization of a coherent artifact/scatter model.

5.5. Compartmental models

A linear time-invariant compartmental model may be used to describe allowed transitions between components, as Godfrey (1983) reviews. Transitions between compartments are described by microscopic rate constants which constitute the off-diagonal elements of the transfer matrix K. The diagonal elements of K contain the total decay rates of each compartment. The concentrations of the compartments in continuous time are described by a vector $c(t) = [c_1(t), \ldots, c_{n_{\text{ncomp}}}(t)]^\top$. Thus, a linear compartmental model with n_{ncomp} compartments is described by a differential equation for these concentrations

$$\frac{d}{dt}c(t) = Kc(t) + j(t) \tag{21}$$

where the input to the system is described by a vector $j = i(t)[j_1\ j_2\ \ldots\ j_{n_{\text{ncomp}}}]$ such that $\sum_{l=1}^{n_{\text{ncomp}}} j_l = 1$, and $j_{n_{\text{ncomp}}} \equiv 1 - \sum_{l=1}^{n_{\text{ncomp}}-1} j_l$. Also, generally $j_l \geq 0$. The IRF $i(t)$ describes the time-profile of the inputs. Under the assumption that the eigenvalues of K are different, and that $c(-\infty) = 0$, Equation 21 is solved as

$$c(t) = e^{Kt} \star j(t). \tag{22}$$

Equation 22 is used as the prescription for the concentrations of the components in discrete time. Note that Equation 22 requires evaluation of the exponential of the K matrix. This exponentiation is implemented in **TIMP** as described in van Stokkum (2005).

Analysis in the absence of a compartmental model is equivalent to the application of a K matrix with non-zero elements only on the diagonal (Figure 6). A commonly applied compartmental model is termed a *sequential model*, in which $n_{\text{ncomp}} - 1$ components decay to a single other component, one component decays directly to the ground state and no loops are present (Figure 7).

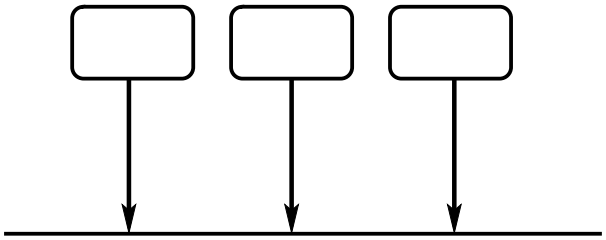


Figure 6: Diagram of a compartmental model of three components, all of which decay in parallel to the ground state.

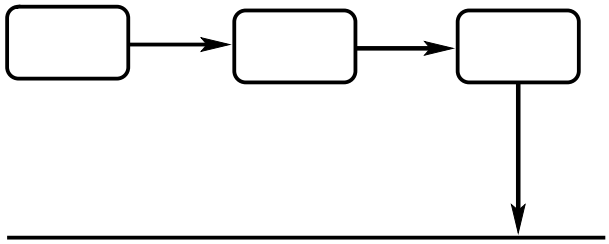


Figure 7: Diagram of a compartmental model of three components sequentially decaying to the ground state.

Specification in **TIMP**: *Parallel/sequential compartmental model*

The logical `seqmod` determines whether a sequential compartmental model is applied. If `seqmod=FALSE` then a parallel compartmental model is applied. The default is `seqmod=TRUE`.

Specification in **TIMP**: *Full compartmental model*

The array `kmat` contains an array with dimension attribute $c(2, n\_\texttt{ncomp}, n\_\texttt{ncomp})$. That is, `kmat` contains two matrices of dimension $n_{\text{ncomp}} \times n_{\text{ncomp}}$. The matrix in $\texttt{kmat}[1, 1 : n\_\texttt{ncomp}, 1 : n\_\texttt{ncomp}]$ contains i at position $\texttt{kmat}[1, j, i]$ if a transition from component i to component j is allowed, and else 0.

The matrix in $\texttt{kmat}[2, 1 : n\_\texttt{ncomp}, 1 : n\_\texttt{ncomp}]$ contains k at position $\texttt{kmat}[2, j, i]$ if the transition from component i to component j is parameterized by a branching parameter from `kinscal` with index k, and else 0.

The vector `jvec` contains the j vector descriptive of the inputs to the transfer matrix K.

The vector `kinscal` is descriptive of starting values for branching parameters of K.

5.6. Case study: Multiexperiment analysis

An example of multiexperiment kinetic modeling of data $\Psi = \{\Psi_1, \Psi_2\}$ is considered in this section. Datasets Ψ_1 and Ψ_2 were collected under the same experimental conditions except the excitation laser intensity was doubled during collection of Ψ_1 relative to the laser intensity used during collection of Ψ_2. The challenge in modeling is to obtain a parametric description of the kinetics of the underlying system as evidenced by both datasets, as well as a parametric description of how the difference in laser intensity affects the system. The system underlying the data and physical evidence informing formulation of the initial model and interpretation of model fit will be described elsewhere (*manuscript in preparation*).

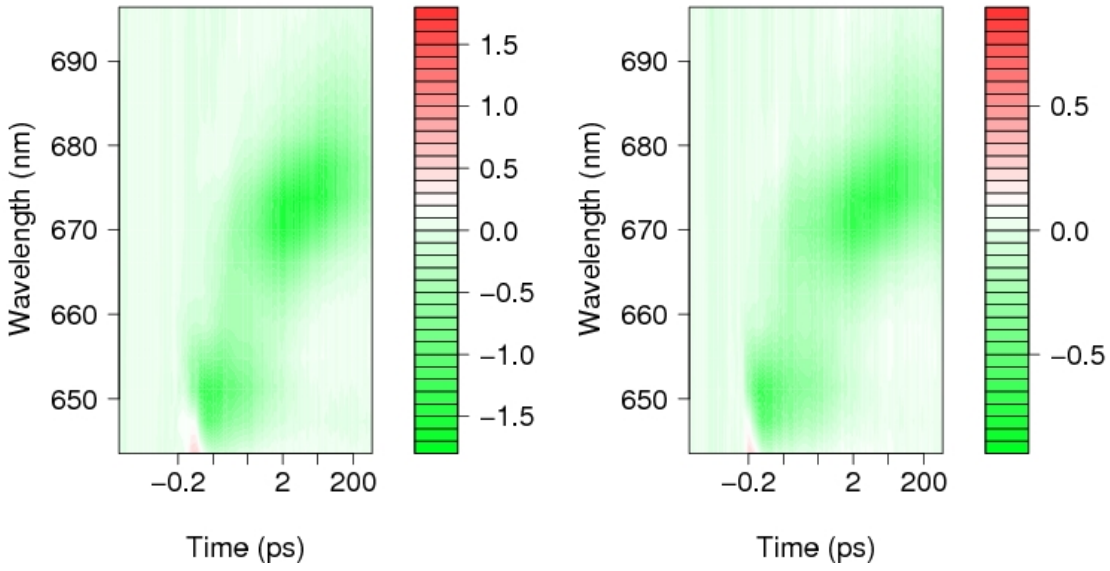


Figure 8: (Left) Dataset Ψ_1 measured using twice the laser intensity used to measure (Right) dataset Ψ_2. The color palette used to display these datasets is generated with the `diverge_hsv` function of the **vcd** package (Meyer *et al.* 2006).

Data input

The data $\Psi = \{\Psi_1, \Psi_2\}$ is read into **TIMP** in the *time explicit* format described in Appendix C via the commands

```
R> psi_1 <- readData("psi_1.txt")
Read 1 item
Read 2385 items

R> psi_2 <- readData("psi_2.txt")
Read 1 item
Read 2385 items
```

where Ψ_1 is stored in the file "`psi_1.txt`" and Ψ_2 is stored in the file "`psi_2.txt`" (distributed with this paper).

Data preprocessing

The wavelength axis of the data is scaled via the prescription $x = 3.78x + 643.5$.

```
R> psi_1 <- preProcess(data = psi_1, scalx2 = c(3.78, 643.5))
R> psi_2 <- preProcess(data = psi_2, scalx2 = c(3.78, 643.5))
```

An initial model

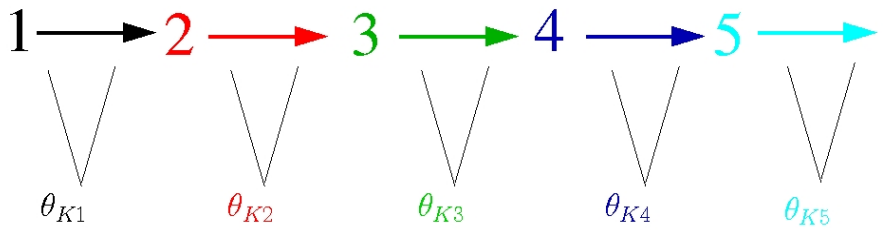


Figure 9: Kinetics are described by a five-component sequential compartmental model with identical decay rate parameters for both datasets.

It is known *a priori* that the underlying system is likely to contain five components decaying sequentially, that a Gaussian IRF model is likely to be appropriate, and that a contribution from a coherent artifact must be accounted for. Approximate starting estimates for parameter values are also known.

This leads to the following model specification in **TIMP**.

```
R> model1 <- initModel(mod_type = "kin",
+  kinpar=c(7.9, 1.08, 0.129, .0225, .00156),
+  irfpar=c( -.1018, 0.0434),
+  parmu = list(c(.230)),
+  lambdac = 650, seqmod=TRUE,
+  positivepar = c("kinpar"), title="model 1",
+  cohspec = list( type = "irf"))
```

The vector `kinpar` contains the starting values for five kinetic components. These parameters are constrained to positive values during fitting by including `"kinpar"` in the vector argument `positivepar`. The `seqmod` argument determines that the kinetics are described with a sequential compartmental model. The `irfpar` argument gives starting values for the parameters of the default Gaussian IRF model. The `parmu` argument gives starting values for the default model for dispersion in terms of a polynomial, in this case of first-order. The `lambdac` argument gives the center wavelength for the polynomial description of dispersion. The `cohspec` argument determines that a model for a coherent artifact/scatter component is to be added with the time-profile of the IRF.

Fitting and validating the initial model

For simultaneous analysis a list is initialized to group dataset Ψ_1 and dataset Ψ_2 together.

```
R> psi <- list(psi_1, psi_2)
```

In the absence of information regarding the effect of laser intensity on the underlying system, the model initialized in Section 5.6.3 may be fit to both dataset Ψ_1 and dataset Ψ_2 simultaneously, with the addition of a dataset scaling parameter Θ_L to account for the difference in amplitude between datasets due to laser intensity.

A call to the fitting function `fitModel` fits the initial model `model1` to both datasets, with the argument `modeldiffs` specifying per-dataset differences in the applied model. By visual inspection of the data it is clear that the intensity of dataset Ψ_2 is approximately half that of dataset Ψ_1. It is not obvious from visual inspection what other per-dataset differences to include in the model. Therefore `modeldiffs` only specifies that the second dataset is scaled to .5 times the first dataset (via the argument `dscal = list(list(to=2,from=1,value=.5))`). The input argument `opt` specifies plotting and output options.

```
R> res_model1 <- fitModel(psi, model1,
+  modeldiffs = list(dscal = list(list(to=2,from=1,value=.5))),
+  opt=list(iter=5, superimpose = c(1,2), divdrel = TRUE, linrange = .2,
+  makeps = "den1", selectedtraces = c(1,5,10), plotkinspec =TRUE,
+  xlabel = "time (ps)", ylabel = "wavelength",
+  paropt=list(cex.main=1.2,cex.lab=1.2,cex.axis=1.2)))
```

Figure 10 shows the fit of selected traces after fitting for five iterations. Large misfits are present. Misfits around time zero are likely to be due to an insufficient IRF model, whereas misfits at later times are likely to be due to differences in the kinetics of the two measured datasets. The root mean square (RMS) error associated with this fit is .040. Note that the datasets appear to have the same amplitudes due to the argument `divrel` to `opt` in the call to `fitModel`, which divides Ψ_2 and and the fit of the model to Ψ_2 by the estimate for the dataset scaling parameter.

Model refinement and re-validation

Fitting the IRF location parameters and the slowest two kinetic decay rate parameters per-dataset attempts to address the inadequacies of the model identified in validation.

```
R> res_model1_refined <- fitModel(psi, model1,
+  modeldiffs = list(dscal = list(list(to=2,from=1,value=.5)),
+  free = list(
+  list(what = "irfpar", ind = 1, dataset = 2, start=-.1932),
+  list(what = "kinpar", ind = 5, dataset = 2, start=.0004),
+  list(what = "kinpar", ind = 4, dataset = 2, start= .0159)
+  )),
+  opt=list(iter=5,superimpose = c(1,2), divdrel = TRUE, linrange = .2,
+  makeps = "den2", selectedtraces = c(1,5,10),
+  paropt=list(cex.main=1.2,cex.lab=1.2,cex.axis=1.2)))
```

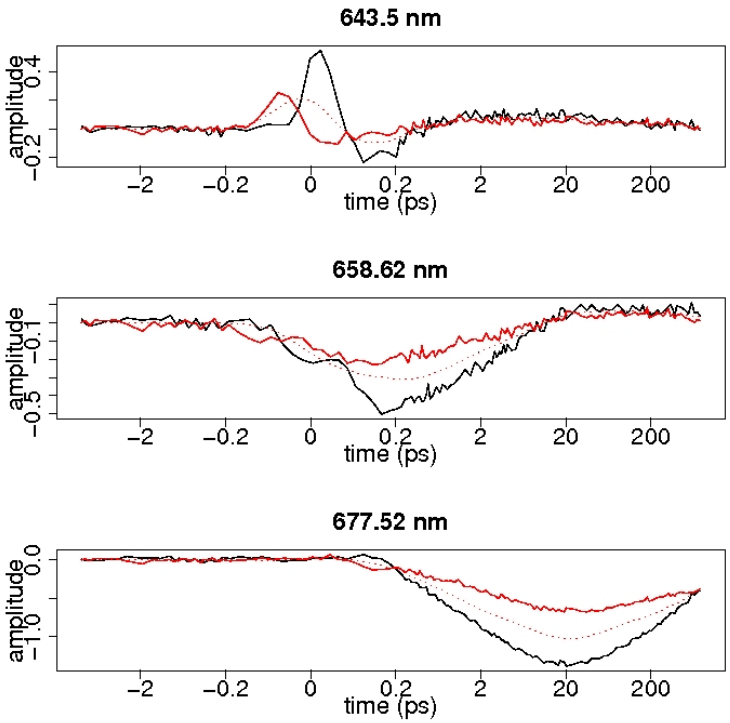


Figure 10: A plot of three selected traces resulting from the fit of the initial model to the data $\Psi = \{\Psi_1, \Psi_2\}$. Black represents data Ψ_1 (solid) and the fit of the initial model to Ψ_1 (dotted); Red represents data Ψ_2 (solid) and the fit of the initial model to Ψ_2 (dotted). The RMS error associated with this fit is .040.

The resulting fit is improved over the initial model, though a misfit remains evident at early times. The data has an oscillatory nature in some traces indicative of a contribution from a coherent artifact, as evident in Figure 11. The RMS error associated with this fit is .027.

A satisfactory model

The model for the coherent artifact has followed the time profile of the IRF, which is not sufficient to account for the oscillatory nature of its apparent contribution to the data. Therefore the coherent artifact model is replaced via a re-definition of the model (which could also be performed by specifying a different model prescription for both datasets in the model differences list):

```
R> model2 <- initModel(mod_type = "kin",
+  kinpar=c(7.9, 1.08, 0.129, .0225, .00156),
+  irfpar=c( -.1018, 0.0434),
+  parmu = list(c(.230)),
+  lambdac = 650,
+  seqmod=TRUE,
+  positivepar = c("kinpar", "coh"),
+  title="Model 2",
+  cohspec = list( type = "seq", start = c(8000, 1800)))
```

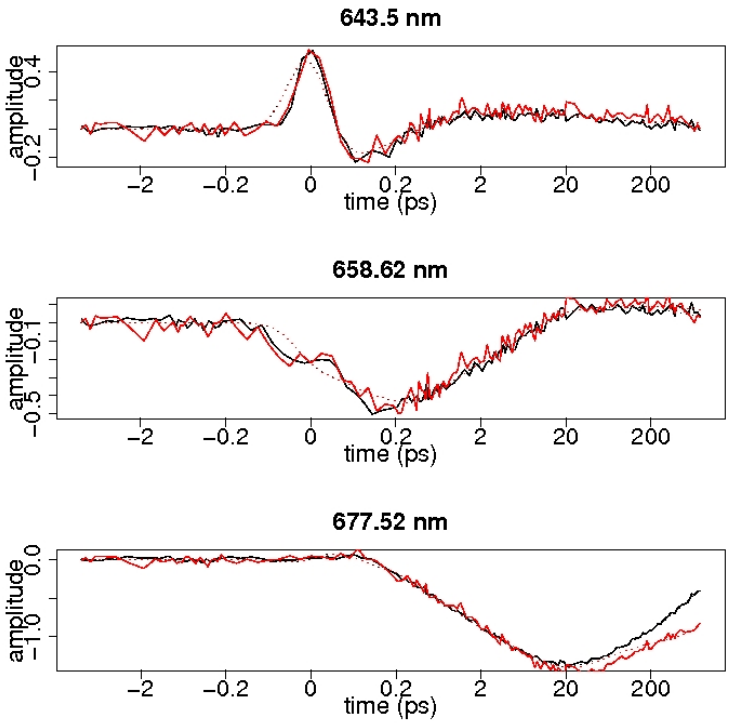


Figure 11: A plot of three selected traces resulting from the fit of the refined model to the data $\Psi = \{\Psi_1, \Psi_2\}$, with the IRF location parameters and the two slowest decay rates fit per-dataset. Black represents data Ψ_1 (solid) and the fit of the initial model to Ψ_1 (dotted); Red represents data Ψ_2 (solid) and the fit of the initial model to Ψ_2 (dotted). The RMS error associated with this fit is .027.

The new model is fit to the datasets. Again the IRF location and the two slowest data rates are fit per-dataset. The estimated value of the data scaling parameter between Ψ_1 and Ψ_2 obtained in fitting the refinement of the initial model is used as a starting value.

```
R> res_model2 <- fitModel(psi, model2,
+  modeldiffs = list(dscal = list(list(to=2,from=1,value=.457)),
+  free = list(
+  list(what = "irfpar", ind = 1, dataset = 2, start=-.1932),
+  list(what = "kinpar", ind = 5, dataset = 2, start=.0004),
+  list(what = "kinpar", ind = 4, dataset = 2,
+  start= .0159))),
+  opt=list(iter=1,superimpose = c(1,2), divdrel = TRUE, linrange = .2,
+  makeps = "den3", selectedtraces = c(1,5,10), plotkinspec =TRUE,
+  xlabel = "time (ps)", ylabel = "wavelength", breakdown = c(1,5,10),
+  paropt=list(cex.main=1.2,cex.lab=1.2,cex.axis=1.2)))
```

The resulting parameter estimates mapped to a hierarchical representation of the model are shown in Figure 15. For clarity the standard errors estimates returned by `nls` have been omitted. These standard errors are typically 1-5 in the last significant digit reported, except

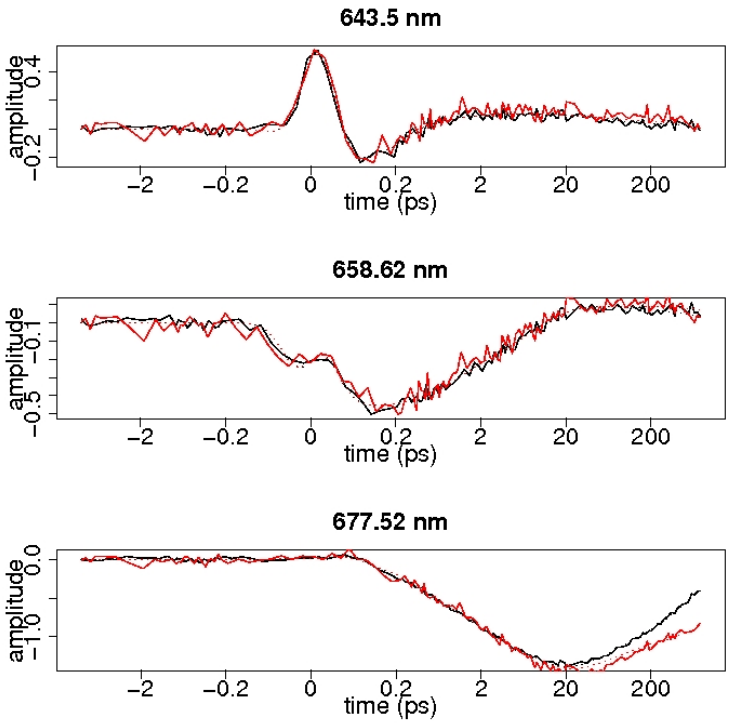


Figure 12: A plot of three selected traces resulting from the fit of a satisfactory model to the data $\Psi = \{\Psi_1, \Psi_2\}$, with the IRF location parameters and the two slowest decay rates fit per-dataset, and under application of an oscillatory coherent artifact model. Black represents data Ψ_1 (solid) and the fit of the initial model to Ψ_1 (dotted); Red represents data Ψ_2 (solid) and the fit of the initial model to Ψ_2 (dotted). The RMS error associated with this fit is .025.

for θ_C (which is not of interest) where they are huge.

Figure 13 shows just the estimates of the kinetic decay rates, with the color of each component the same as in Figure 16 and Figure 14. Figure 14 shows the contribution to the fit of the kinetic decay components and the coherent artifact (which is in pink) at three different wavelengths. The fit associated with the model after five iterations shown at three selected wavelengths in Figure 12 is deemed acceptable. The RMS error associated with this fit is .025. The spectra associated with the kinetic decay components (Figure 16) have physically plausible shapes. The discovery of an appropriate model for the data allows the differences in

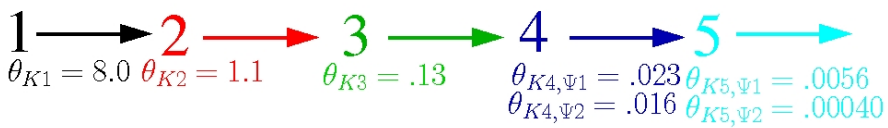


Figure 13: Transitions between the five components of the compartmental model are now fit with decay rates as labeled; the slowest two decays are fit independently for each dataset.

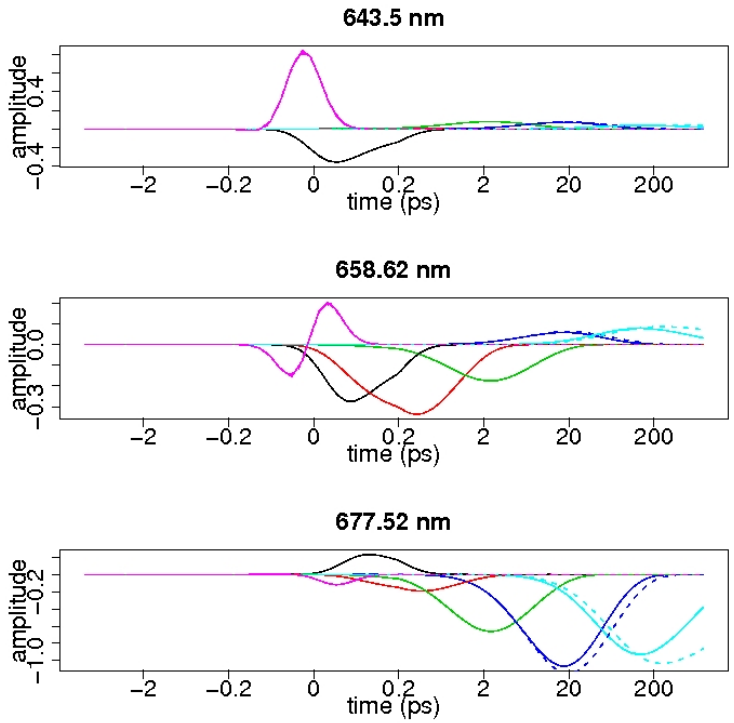


Figure 14: Contributions to fit per component show evolution. Pink represents the coherent artifact component; key to other colors is in Figure 13. Dashed lines indicate the fit of the second dataset Ψ_2, which has slower decay rate estimates.

the slow rate constant estimates between datasets to be attributed to the effect of a difference in laser power. The parameter Θ_L is also interpretable as quantifying this difference.

6. Spectral models

Spectral models are those models in which the nonlinear parameters determine the matrix of spectra E. The spectral bandshapes are typically described in terms of a linear superposition of standard band shapes (e.g., Gaussian, Lorentzian, Voigt, skewed Gaussian) or in terms of splines. In the case that a superposition of standard band shapes is used, each column of E, ϵ_l, is modeled as

$$\epsilon_l = (\text{amp}_1)(g_{1l}) + \ldots + (\text{amp}_h)(g_{hl}) \tag{23}$$

where $\text{amp}_1, \ldots, \text{amp}_h$ are amplitude parameters, and g is the band shape function. The concentration profiles are then estimated as clp.

Spectral models are represented in **TIMP** with the class `spec`. Model parameterization options for spectral models are here outlined.

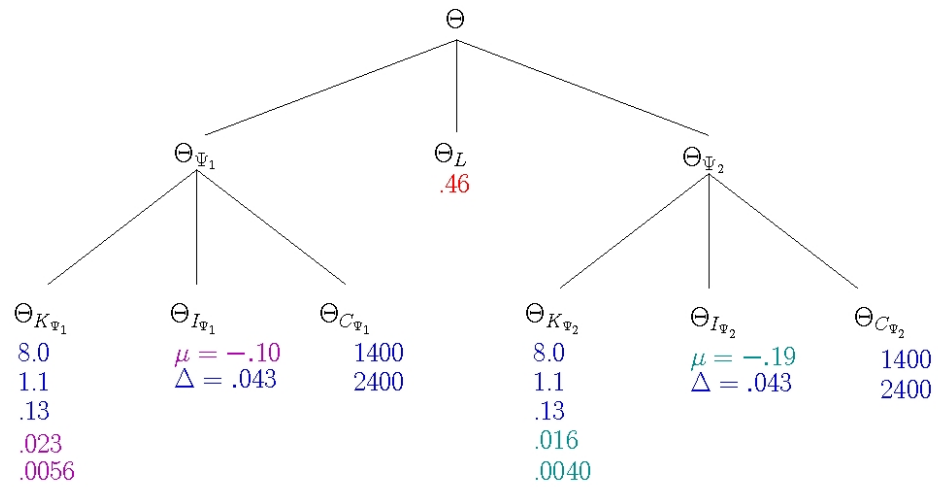


Figure 15: Parameter estimates associated with the fit of the satisfactory model mapped to a hierarchical model representation. Θ_L is a dataset scaling parameter descriptive of the difference in intensity of Ψ_2 as compared to Ψ_1. Parameter estimates linked between datasets Ψ_1 and Ψ_2 are in blue. Parameter estimates fit per-dataset are in magenta and green respectively, for Ψ_1 and Ψ_2.

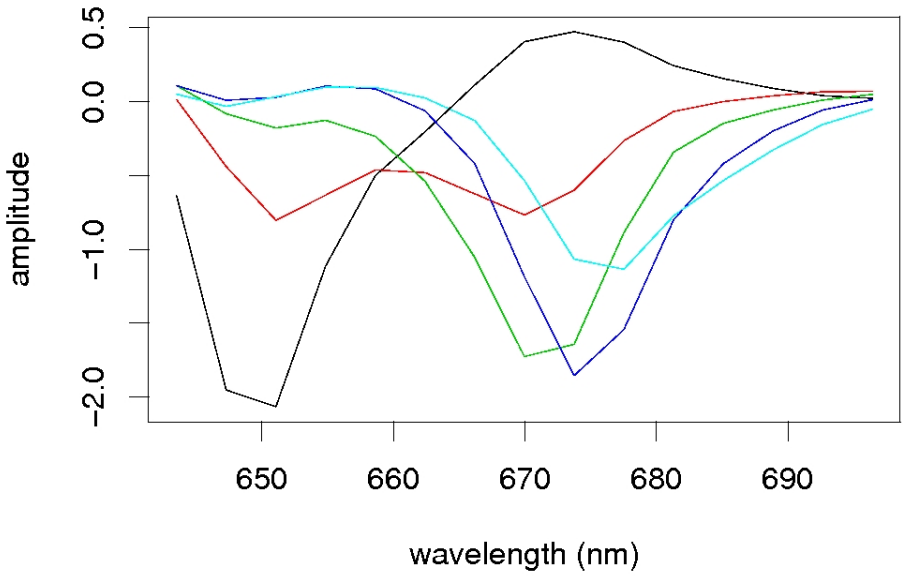


Figure 16: Estimated spectra associated with the five kinetic decays have physically plausible shapes. The mapping of spectra to kinetic decays is given in the color code of Figures 13 and 14.

6.1. Bandshape models

The most commonly applied bandshape model is a superposition of skewed Gaussians. This underlying model for E is chosen because it is a simple model capable of representing real spectra in practice and because the use of (skewed) Gaussians to represent spectral shapes is wide-spread, (see, e.g., van Stokkum *et al.* (2004) and van Stokkum (1997) and references therein).

The model for ϵ_l under a single skewed Gaussian distribution where $\bar{\nu} = 10^7/\lambda$ is

$$\epsilon_l(\bar{\nu}_{max}, \Delta\bar{\nu}, b) \equiv \bar{\nu}^{-n} \exp\left(-\ln(2)\left(\ln\left(1 + \frac{2b(\bar{\nu} - \bar{\nu}_{max})}{\Delta\bar{\nu}}\right)/b\right)^2\right), \tag{24}$$

except if $1 + (2b(\bar{\nu}_i - \bar{\nu}_{max}))/\Delta\bar{\nu} \leq 0$, in which case $\epsilon_l(\bar{\nu}_{max}, \Delta\bar{\nu}, b) \equiv 0$. When skewness $b = 0$, (and the skewed Gaussian distribution reduces to the Gaussian distribution), $\bar{\nu}_{max}$ corresponds to the maximum of the distribution, and $\Delta\bar{\nu}$ corresponds to the full width at half maximum (FWHM). When $b \neq 0$, $\bar{\nu}_{max}$ and $\Delta\bar{\nu}$ do not have this exact correspondence. The FWHM may then be determined as $\Delta\bar{\nu}\sinh(b)/b$. The average wavenumber of the skewed Gaussian is given by

$$\bar{\nu}_{avg} = \bar{\nu}_{max} + \frac{\Delta\bar{\nu}}{2b}\left(\exp\left(-\frac{3b}{4\ln(2)}\right) - 1\right) \tag{25}$$

A linear superposition of spectra as in Equation 24 is a common model for a single spectrum ϵ_l.

Specification in **TIMP***: Bandshape model*

The list `specpar` contains vectors of starting values for spectral parameters; the number of vectors gives the number of components in the resulting spectral model. Each vector contains the parameters associated with a component. e.g., `specpar = list(c(20000, 3000, .3, 21000, 2000, .4), c(18000, 1000, .2))`; the parameters in each vector are grouped `c(location_spectra, width_spectra, skew_spectra)`. The location and width parameters are given in wavenumbers. Note that this means that each component may be modeled with a superposition of many skewed Gaussians.

The character string `specfun` specifies the model used in the description of bandshapes.

Inclusion of the amplitude parameters is not yet implemented, so that all skewed bandshapes contributing to a single component spectrum ϵ_l contribute equally.

6.2. Dependence of bandshape parameters on independent variable

It is often desirable to model bandshape parameters of spectra as a function of the independent variable in which spectra are resolved, (e.g., time or temperature). Parameterization of this variation is a means of studying protein conformational stability (for temperature-resolved data), and vibrational relaxation (for time-resolved data).

Dependence on the variable in which spectra are resolved may often be well-described by a polynomial model. Where θ is some parameter contributing to the determination of a bandshape (i.e., either a location, width, skewness or amplitude parameter), t is a value of the independent variable with which spectra are resolved, v is the number of coefficients

parameterizing the polynomial (in addition to θ), t_{ref} is the center-point of the polynomial, and $\alpha_1, \ldots, \alpha_v$ parameterize the dependence, then

$$\theta(t) = \theta + \sum_{i=1}^{v} \alpha_i((t - t_{\text{ref}})/100)^i. \tag{26}$$

An exponential model of dependence on the variable in which spectra are resolved is also often of interest. Then where t_{ref} is a reference time, and other variables are as in the polynomial description,

$$\theta(t) = \theta + \alpha_1 \exp(-\alpha_2(t - t_{\text{ref}})). \tag{27}$$

A multiexponential model may also be of use in some cases. For the bi-exponential case

$$\theta(t) = \theta + ((\theta_0 - \theta)\frac{\exp(-k_1(t - t_{\text{ref}})) + \alpha_2 \exp(-k_2(t - t_{\text{ref}}))}{1 + \alpha_2}), \tag{28}$$

where $\theta(t_0) = \theta_0$ and $\theta(\infty) = \theta$.

For the general case of $i \geq 1$ multiexponential decays,

$$\theta(t) = \theta + (\theta_0 - \theta)\frac{\sum(\alpha_i \exp(-k_i(t - t_{\text{ref}})))}{\sum \alpha_i} \tag{29}$$

where $\alpha_1 \equiv 1$.

Specification in **TIMP**: *Dependence of bandshape parameters on independent variable*

The character string `parmufunc` determines the function modeling dependence of the bandshape parameters on the independent variable. Options are `"poly"` for the polynomial case (Equation 26) `"exp"` for the single exponential description (Equation 27) and `"multiexp"` for the multiexponential description (Equation 29).

The numeric `specref` gives the index of the center variable, t_{ref} in Equations 26, 27 and 29).

The list `specdispindex` defines those indices of specpar whose dependence on the variable in which spectra are resolved is to be modeled. For example, `specdispindex = list(c(1,1), c(1,2), c(1,3))` indicates that parameters 1-3 of spectra 1 are to be modeled as variable.

The list `specdisppar` contains vectors of the parameters describing time-dependence. One vector of parameters is given for each vector of indices in `specdispindex`. These parameters describe a polynomial time-dependence by default. There are three ways to interpret these vectors, depending on the value of `parmufunc`. If `parmufunc` is `"poly"` for the polynomial case, the vectors are of the length of the desired degree of the polynomial parameterization; e.g., use `specdisppar=list(c(-2000, 1, .1), c(1, .1, .01), c(.2, .1))` for a 3rd order dependence of two spectral parameters, and a 2nd order dependence on one spectral parameter. If `parmufunc` is `"exp"` the first parameter is a linear coefficient and the second is a rate; if all but the first vector have the rate omitted then rates will be linked across all parameters that are time-dependent; otherwise rates will be fit per-parameter that is dependent on the variable with which spectra are resolved. If `parmufunc` is `"multiexp"` an arbitrary number of vectors of coefficients and rates in the form $c(\alpha\_1, k\_1, \alpha\_2, k\_2, \ldots)$ may be specified as elements of the `specdisppar` list.

6.3. Case study: Time-dependence of spectral parameters

A spectral model is fit to the time-resolved dataset shown in Figure 17. A goal is the parametric description of the relaxation of the bandshape in time. As in the kinetic modeling case study, (Section 5.6) this section describes the use of **TIMP** for interactive model discovery. The system underlying the data and motivation for the initial model as well as physico-chemical interpretation of model fit will be described elsewhere (*manuscript in preparation*).

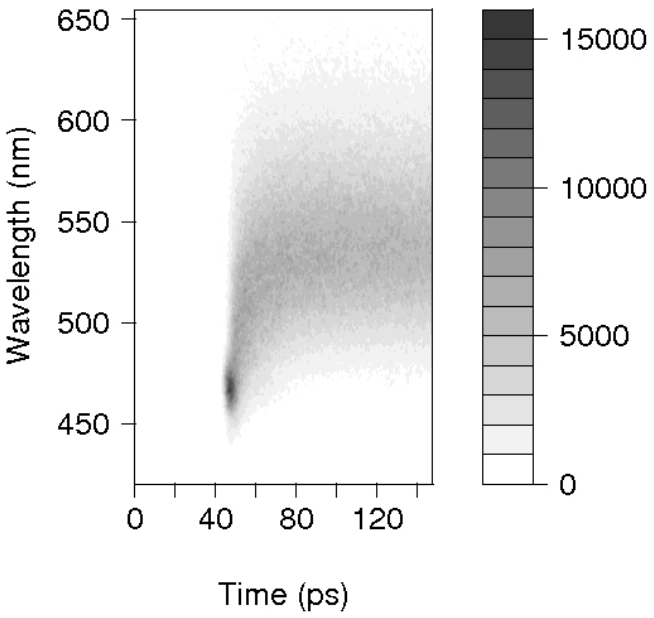


Figure 17: A dataset of time-resolved spectra. We are interested in application of a spectral model to the selection of this data shown in Figure 18.

Data input

The data $\Psi = \{\Psi_1\}$ is read into **TIMP** in the *time explicit* format described in Appendix C via the command

```
R> psi_1 <- readData("psitspec.txt")
Read 1 item
Read 181063 items
```

where Ψ_1 is stored in the file "`psitspec.txt`", (distributed with this paper).

Data preprocessing

The dataset was truncated to the wavelength range 440-640 nm, and to times after 53 ps (from time indices 178 to 478), since we are only interested in modeling processes occurring in this range.

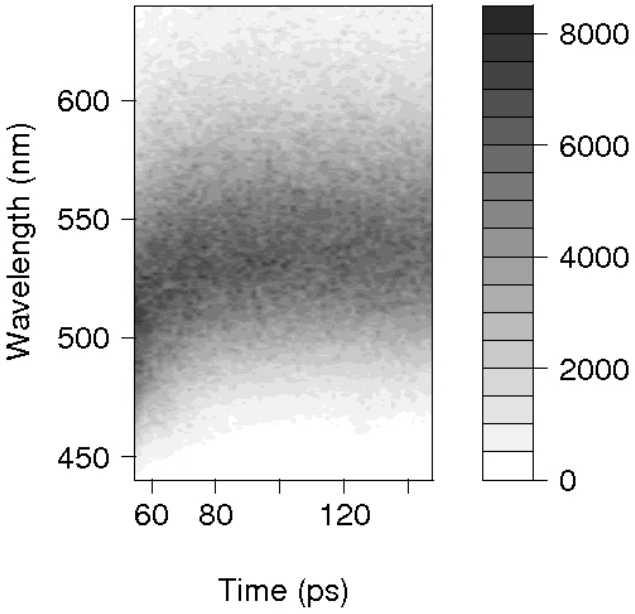


Figure 18: The dataset in Figure 17 selected to include only the range of timepoints and wavelengths of interest for modeling, and sampled to include only every fifth timepoint to expedite parameter estimation.

```
R> psi_1_full <-preProcess(psi_1, sel_time=c(178, 478),
+  sel_lambda_ab = c(440, 640))
```

To expedite parameter estimation only every fifth timepoint is sampled from the selected dataset. When a model likely to be satisfactory is identified on the sampled dataset, it may be applied to the unsampled selected dataset.

```
R> psi_1_sampled <- preProcess(psi_1, sel_time=c(178, 478),
+  sel_lambda_ab = c(440, 640), sample_time=5)
```

Figure 18 shows the selected and sampled dataset resulting from the above calls to the `preProcess` function.

Initial model: Polynomial dependence of spectral variables on time

A spectra with bandshape model comprised of a single skewed Gaussian is initialized, first with linear polynomial dependence of the spectral variables on time.

```
R> model_polylin <- initModel(mod_type = "spec",
+  specpar=list(c(20000,3100,-.3) ),
+  specdispindex = list(c(1,1), c(1,2), c(1,3)),
```

```
+  specdisppar=list(c(-2000),c(1), c(.2)),
+  specref=53, specfun="gaus",
+  iter=1, nupow=5, make_ps="badan_lin",
+  title = "Linear" )

R> res_polylin <- fitModel(data = list(psi_1_sampled),
+  model = model_polylin,
+  opt=list(iter=7, linrange = 20,
+  makeps = "polylin", nospectra = TRUE,
+  selectedspectra = seq(1, psi_1_sampled@nt, by=7),
+  residplot = TRUE,
+  xlabel = "time", ylabel = "wavelength",
+  ))
```

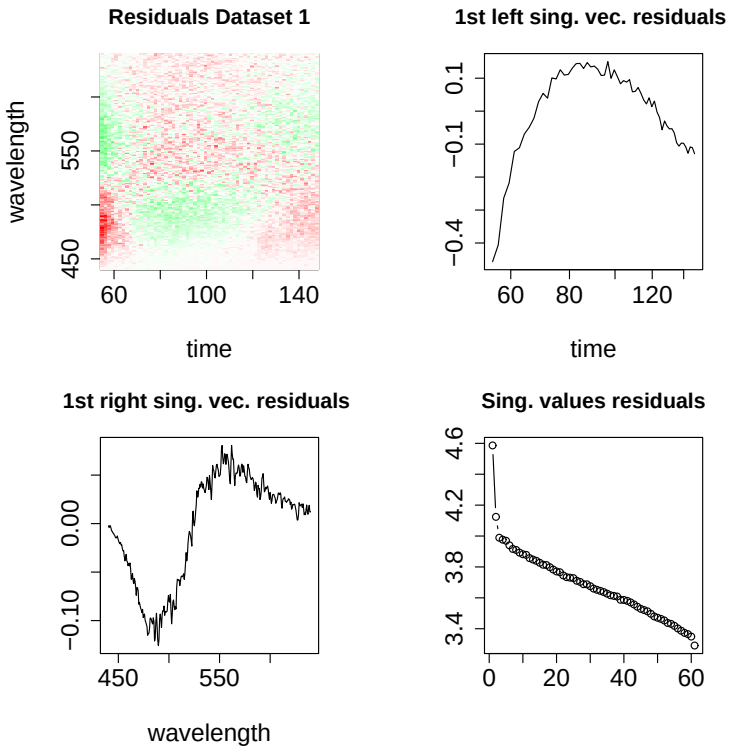


Figure 19: Residuals associated with the fit of the initial spectral model in which dependence of the spectral bandshape on parameters is described with a first-order polynomial. The first left and right singular vector and the singular values associated with their SVD are also plotted. Note the structure in the first left singular vector of the residuals, indicating inadequacy in the model fit. The RMS error associated with this fit is 414.

Structure in the residuals as evidenced by their SVD in Figure 19 indicates an inadequate model. A different parameterization of the dependence of spectral parameters on time will be applied as a model refinement.

Refined model: Exponential dependence of spectral variable on time

The initial model is refined to describe an exponential dependence of spectral variables on time, in an attempt to address the inadequacy in the fit of the initial model. The applied model is such that α_2 from Equation 27 is equated for all spectral bandshape parameters.

```
R> model_exp_linkedrates <- initModel(mod_type = "spec",
+  specpar=list(c(18000, 3200, -.1)),
+  specdispindex = list(c(1,1), c(1,2), c(1,3)),
+  specdisppar=list(c(600,1/20), c(400), c(.1)),
+  specref=53, specfun="gaus",  parmufunc = "exp",
+  nupow=5,
+  title = "Exponential parameterization of time dep., linked rates")

R> res_model_exp_linkedrates <- fitModel(data = list(psi_1_sampled),
+  model = model_exp_linkedrates,
+  opt=list(iter=5, linrange = 20, residplot=TRUE,
+  makeps = "explinked", nospectra = TRUE,
+  selectedspectra = seq(1, psi_1_sampled@nt, by=7),
+  xlabel = "time", ylabel = "wavelength"))
```

The fit of the refined spectral model that employs an exponential description of time-dependence of spectral bandshape parameters is well-fit to the data, as evidenced by plots showing the fit of the model to the data in Figure 20. Further evidence for the satisfactory model fit is contained in plots regarding the residuals in Figure 21. An SVD decomposition of the residuals shows little structure in the first left singular vector. Some structure remains evident in the first right singular vector, indicating that there remains room for improvement in the model. Despite this, we conclude that the model fit is satisfactorily descriptive of the data. The exponential rate estimate $\alpha_1 = .07$ is therefore useful as a descriptor of the relaxation of the spectral bandshape parameters in time.

7. Extension of supported model types

TIMP has been designed to allow for rapid implementation of new options and new model types. New options can be added by modification of the class definitions associated with existing model types and the class for parameter estimates `theta`, and modification of the associated residual functions to work with slots describing new options. For instance, imagine that for a model type `modelx`, a parameterization of an additional sort of submodel is desired.

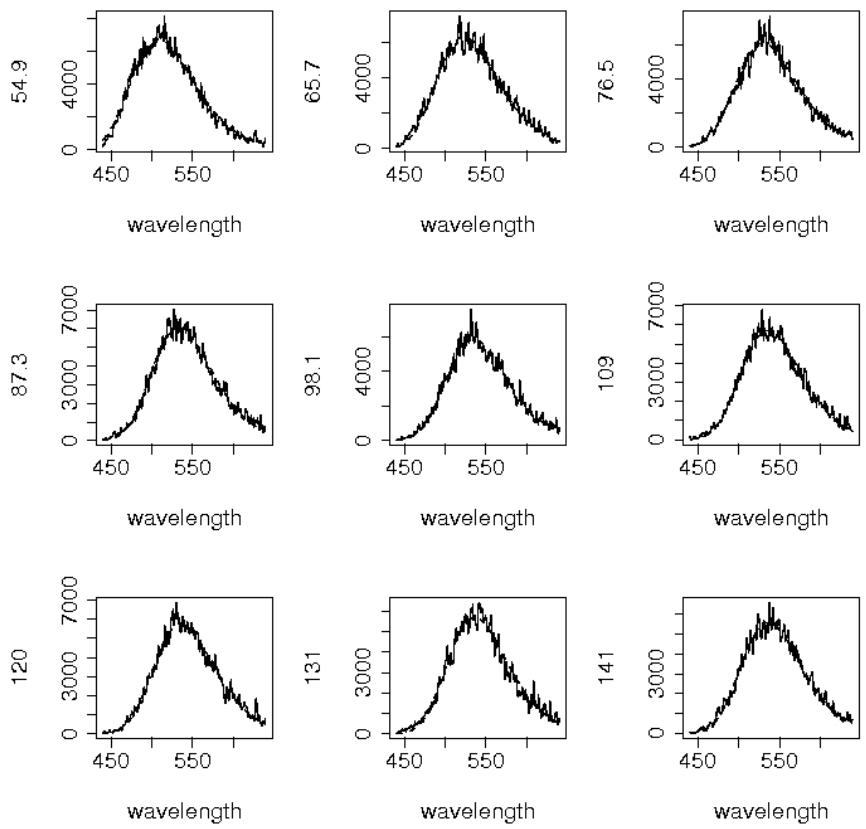


Figure 20: (Left) A plot of selected spectra resulting from the fit (dashed line) of a spectral model in which time-dependence of spectral bandshape parameters is modeled with an exponential function to data $\Psi = \{\Psi_1\}$ (solid line). Each sub-plot represents the data at the timepoint on the left axis. The RMS error associated with this fit is 312.

Then two slots are added to the class definition for model type `modelx`: a slot to represent options to parameterize the new submodel `new`, and a slot `parnew` to represent the starting values of nonlinear parameters associated with the new submodel. A slot named `parnew` is also added to the class `theta`. Then `new` and `parnew` are given as inputs to the `inputModel` function. To use the new parameterization to change how the residuals are determined, the `S4` methods associated with determining the residuals for class `modelx` are modified. The `S4` method `plotter` determining plotting/output for `modelx` is also modified to use the new options as desired. No additional modification of the code is necessary.

The process of adding an entirely new model type `newmodel` can be described in terms of four steps:

- definition of a new class for the model type `newmodel` that inherits from `dat`
- for every slot `par` representing a list/vector of nonlinear parameter value starting values in the definition of `newmodel` adding a slot for a list/vector of the same name `par` to the class `theta` so that the vector of parameters Θ can be inferred, and so that updated parameter estimates can be plugged back into the model specification each iteration

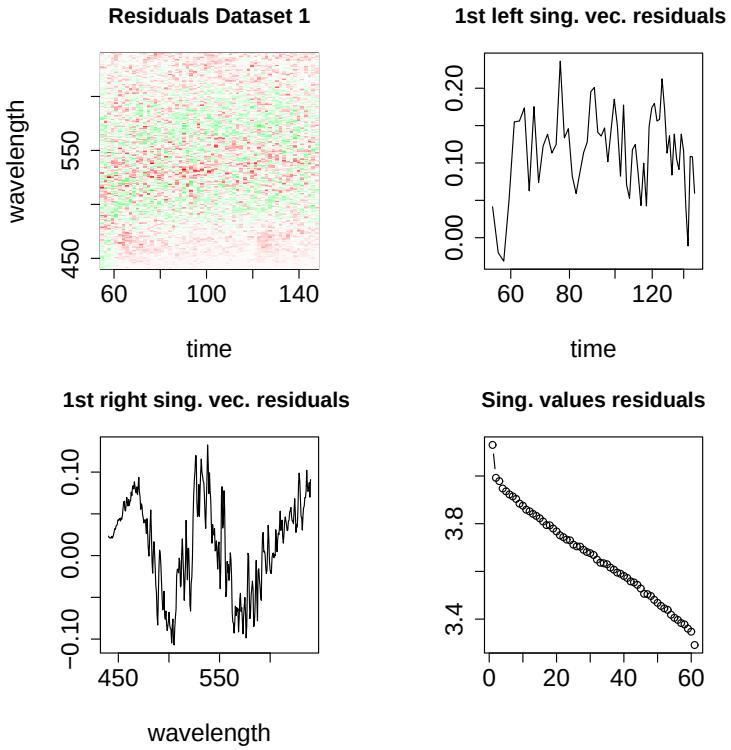


Figure 21: Residuals associated with the fit of the spectral model, and the first left and right singular vector and the singular values associated with their SVD. Note the lack of structure in the first left singular vector of the residuals, a sign that the model fit is satisfactory. There remains some structure in the first right singular vector, but we accept the model as satisfactory nonetheless.

- definition of methods for `residPart` (and/or `getClpIndepX` depending on whether `clp` are involved) that supply a prescription for the calculation of residuals for a single dataset Ψ_q given a model
- definition of desired output plots and other information via a method for `plotter`

The remaining code of **TIMP** should not require any modification. Planned extensions to additional model types will allow the application of the package to fitting models for photocycles and polarization-dependent effects.

8. Conclusions

TIMP, an R package for interactive scientific model discovery for multiway spectroscopy data has been introduced. The design of the package has been outlined. The partitioned variable

projection algorithm that is central to solving separable nonlinear least squares parameter estimation problems with **TIMP** was presented.

General options for models in **TIMP** were introduced, along with options specific to kinetic and spectral model types. A case study in application of a kinetic model to two datasets simultaneously illustrated many kinetic model options. A case study in application of a spectral model illustrated many spectral model options.

TIMP is in active development. Future work includes the development of new model types for data collected in multipulse laser experiments, anisotropy experiments, and experiments designed to extract information on photocycles, as well as the implementation of additional options to support model specification and validation. The development of a GUI to support interactivity is also planned.

Acknowledgments

This research was funded by Computational Science grant #635.000.014 from the Netherlands Organization for Scientific Research (NWO). Mikas Vengris, Denitsa Grancharova and Rienk van Grondelle provided the data modeled in Section 5.6. Rob Koehorst, Bart van Oort, Sergey Laptenok, Ton Visser and Herbert van Amerongen provided the data modeled in Section 6.3. Joris Snellenburg is thanked for constructive comments on the text. Uwe Ligges and Martin Mächler collaborated in the implementation of the `nls` options described in Section B. Achim Zeileis contributed helpful suggestions regarding the figures.

References

Bates DM, DebRoy S (2003). "Converting a Large R Package to S4 Classes and Methods." In K Hornik, F Leisch, A Zeileis (eds.), "Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria," ISSN 1609-395X, URL http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/.

Chambers JM (1998). *Programming with Data, A Guide to the S Language*. Springer-Verlag, New York.

Godfrey K (1983). *Compartmental Models and Their Application*. Academic Press, London.

Golub G, Pereyra V (2003). "Separable Nonlinear Least Squares: the Variable Projection Method and its Applications." *Inverse Problems*, **19**, R1–R26.

Golub GH, LeVeque RJ (1979). "Extensions and Uses of the Variable Projection Algorithm for Solving Nonlinear Least Squares Problems." In "Proceedings of the 1979 Army Numerical Analysis and Computers Conference," volume ARO Report 79-3, pp. 1–12.

Golub GH, Pereyra V (1972). "The Differentiation of Pseudo-inverses and Nonlinear Least Squares Problems whose Variables Separate." *Technical report*, Stanford University, Department of Computer Science.

Golub GH, Pereyra V (1973). "The Differentiation of Pseudoinverses and Nonlinear Least Squares Problems whose Variables Separate." *SIAM Journal on Numerical Analysis*, **10**, 413–432.

Istratov AA, Vyvenko OF (1999). "Exponential Analysis in Physical Phenomena." *Review of Scientific Instruments*, **70**(2), 1233–1257.

Laptenok S, Mullen KM, Borst JW, van Stokkum IHM, Apanasovich VV, Visser AJWG (2007). "Fluorescence Lifetime Imaging Microscopy (FLIM) data analysis with **TIMP**." *Journal of Statistical Software*, **18**(8). URL http://www.jstatsoft.org/v18/i08/.

Meyer D, Zeileis A, Hornik K (2006). **vcd**: *Visualizing Categorical Data*. R package version 1.0-2.

Moshier SL (1992). *Cephes Mathematical Library*. URL http://www.moshier.net/.

R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

van Stokkum IHM (1997). "Parameter Precision in Global Analysis of Time-Resolved Spectra." *IEEE Transactions on Instrumentation and Measurement*, **46**(4), 764–768.

van Stokkum IHM (2005). "Global and Target Analysis of Time-resolved Spectra, Lecture notes for the Troisième Cycle de la Physique en Suisse Romande." *Technical report*, Department of Physics and Astronomy, Faculty of Sciences, Vrije Universiteit, Amsterdam, The Netherlands. URL http://www.nat.vu.nl/~ivo/lecturenotes.pdf.

van Stokkum IHM, Larsen DS, van Grondelle R (2004). "Global and Target Analysis of Time-resolved Spectra." *Biochimica et Biophysica Acta*, **1657**, 82–104, and erratum, **1658**, 262.

van Stokkum IHM, Lozier RH (2002). "Target Analysis of the Bacteriorhodopsin Photocycle Using a Spectrotemporal Model." *Journal of Physical Chemistry B*, **106**(13), 3477–3485.

Verveer PJ, Squire A, Bastiaens PIH (2000). "Global Analysis of Fluorescence Lifetime Imaging Microscopy Data." *Biophysical Journal*, **78**(4), 2127–2137.

A. Partitioned vs. unpartitioned variable projection algorithms

This appendix considers in detail the memory requirements of the partitioned variable projection algorithm introduced in Section 2.4, and compares these requirements to those of the standard (unpartitioned) variable projection algorithm. The large memory requirements of the standard variable projection algorithm complicate its application, and sometimes prohibit its use entirely (as e.g., in the experience of Verveer, Squire, and Bastiaens (2000)). The ability to apply the variable projection functional without operating on large matrices is the main motivation for introduction of the partitioned variant of the algorithm. PartitionedVarPro and the standard implementation of variable projection, as found for instance in the `plinear` function of `nls`, return the same results. The algorithms differ only in the memory resources required.

Under PartitionedVarPro with free clp (as defined in Section 2.4) for part p, the residual vector representing dataset Ψ_q is formed by taking $QR(X_{pq})$. The matrix $Q_2^\top$ associated with this decomposition is of dimension $(m - n_{\text{ncomp}}) \times m$. Ψ_{pq} is of length m. Hence the residual for part p, dataset q is of length $(m - n_{\text{ncomp}})$. Concatenating the results for all x datasets at all n parts p yields a total residual of length $nx(m - n_{\text{ncomp}})$.

Under unpartitioned variable projection (VarPro), Ψ is not treated by part p. Hence the entire matrix Ψ is vectorized. Recall that where A, B and C are matrices of compatible dimensions, $vec(ABC) = (C^\top \otimes A)vec(B)$, where vec is the vector representation, and $\otimes$ is the Kronecker product. Let Ψ_{super} be the $mx \times n$ matrix comprised of concatenating datasets $\Psi_1, \ldots, \Psi_x$. Then

$$vec(\Psi_{\text{super}}) = vec(X_{\text{super}}\beta_{\text{super}}^\top I_n) = (I_n \otimes X_{\text{super}})vec(\beta_q^\top), \tag{30}$$

where

$$I_n \otimes X_{\text{super}} = \begin{bmatrix} X_{1\text{super}} & & & & \\ & \ddots & & & \\ & & X_{p\text{super}} & & \\ & & & \ddots & \\ & & & & X_{n\text{super}} \end{bmatrix} \tag{31}$$

$I_n \otimes X_{\text{super}}$ has zero entries except as noted, and has dimension $nmx \times nn_{\text{ncomp}}$.

PartitionedVarPro with linked clp performs n QR decompositions on matrices of dimension $mx \times n_{\text{ncomp}}$, whereas PartitionedVarPro with free clp performs nx QR decompositions on matrices of dimension $m \times n_{\text{ncomp}}$. VarPro performs one QR decomposition on a matrix of dimension $nmx \times nn_{\text{ncomp}}$. For $m = n = 1000$, $n_{\text{ncomp}} = x = 10$, the QR decomposition is performed on matrices of dimensions 10000×10 and 1000×10 in the case of linked and free clp, respectively, whereas VarPro performs the QR decomposition on a matrix of dimension $10^7 \times 10^4$. Avoiding storage and manipulation of a large matrix is a primary motivation to prefer PartitionedVarPro over VarPro. In order to make VarPro usable for realistic problems, storage of only non-zero elements of $I_{nl} \otimes QR(X_{\text{super}})$ and operation on the result with a modified QR-routine could be attempted. However, doing so would provide no advantage over PartitionedVarPro, (and furthermore, requires implementation of a QR routine to operate on the special form of the matrix).

A.1. Empirical comparison of partitioned and unpartitioned variable projection algorithms

The `nls` function of R allows application of the variable projection algorithm via the `plinear` option. We show here how the simple kinetic modeling problem considered in Section 1.3 may be fit using `nls` and the `plinear` option. We then compare the memory requirements under `plinear` to those under the partitioned variable projection algorithm implemented in **TIMP**.

We assume that a dataset `Psi_q` is simulated in R as described in Section 1.3. Commands to simulate the dataset and set up the workspace are contained in full in the file "`memory_prof_plin.R`" (distributed with this paper). In order to use `nls` under the `plinear` option, the model for concentrations is placed into a function `calcC`, which is also found in **TIMP**, as follows.

```
"calcC" <- function (k, t)
{
   conc <- matrix(nrow = length(t), ncol = length(k))
   for (i in 1:length(k)) {
       conc[, i] <- exp(-k[i] * t)
   }
   conc
}
```

The sum-of-exponentials model can then be fit to the data using `calcC`, with the `plinear` option using the standard variable projection functional to determine the residuals, and with the spectra as conditionally linear parameters.

```
R> psi_q_vector <- as.vector(Psi_q)

R> onls <- nls(psi_q_vector ~ kronecker(diag(length(wavenum)),
+  calcC(k, t)), data.frame(psi_q_vector),
+  start = list(k = c(.1,2)), alg = "plinear", trace = T)
```

To profile the memory allocated in the course of solving this problem, we apply the `gc` function, (note that more refined memory profiling is possible with the `Rprof` and `Rprofmem` functions under builds of R compiled to enable memory profiling). Before and after the call to `nls`, the results of a call `gc(verbose=TRUE)` on on our system are

```
Garbage collection 11 = 9+0+2 (level 2) ...
6.4 Mbytes of cons cells used (58%)
0.9 Mbytes of vectors used (14%)
```

and

```
Garbage collection 100 = 45+26+29 (level 2) ...
6.6 Mbytes of cons cells used (53%)
13.2 Mbytes of vectors used (32%)
```

respectively. This shows that under the `plinear` implementation of variable projection about 12 Mbytes of vector space is allocated in the course of solving this example problem.

To contrast this with the memory allocated under the partitioned variable projection implementation found in **TIMP** on the same system, we load the package and initialize a model object as described in in Section 1.3 and in the file "`memory_prof_pvarpro.R`". A call to `gc(verbose=TRUE)` before calling the `fitModel` function that applies the partitioned variable projection algorithm to fitting the sum-of-exponentials model with the spectra as clp has the following result

```
Garbage collection 35 = 31+2+2 (level 2) ...
7.5 Mbytes of cons cells used (68%)
1.0 Mbytes of vectors used (16%)
```

Then fitting the model as in Section 1.3 with

```
R> kinetic_fit <- fitModel(data = list(Psi_q_data), model = kinetic_model,
+  opt = list(iter=4, plot=FALSE))
```

and subsequently calling `gc(verbose=TRUE)` results in

```
Garbage collection 52 = 46+3+3 (level 2) ...
7.8 Mbytes of cons cells used (62%)
1.5 Mbytes of vectors used (25%)
```

This shows that in the course of applying the partitioned variable projection implementation to the problem, about .5 Mbytes of vector space is allocated, about 20 times less than under `plinear` on the same problem.

The savings in memory allocated via the use of the partitioned variable projection algorithm found in **TIMP** is very significant for problems of interest in the multiway spectroscopy modeling domain. As larger amounts of data are involved the memory requirements of the standard non-partitioned implementation found in the *plinear* option grow so large as to prohibit its use on a modern personal computer, while the memory requirements of the partitioned version of the algorithm found in **TIMP** remain modest.

B. New `nls` options

It is often desirable in scientific modeling applications to terminate the iterative optimization of free model parameters when *stopping* criteria are met, as opposed to when *convergence* criteria are met. For instance, it is often desirable to evaluate the fit of a model at a given set of starting estimates, or after fitting for a modest number of iterations. Then the stopping criteria is completion of a maximum number of iterations, after which output is desired, even though the fit may be far from satisfying convergence criteria.

It is often also desirable to examine output in the case that the fitting algorithm encountered a problem and terminated fitting with an error. For instance, if the gradient of the residual vector with respect to nonlinear parameter estimates becomes singular, examination of the current parameter estimates may shed light on how the model can be modified to be better determined with respect to the data.

The R function `nls` is widely applied in scientific model discovery and is used in **TIMP** to iteratively improve nonlinear parameter estimates. Prior to R version 2.5 `nls` did not return output in the case that any of the following conditions are met

- the maximum number of iterations x is met (as specified with `nls(..., control = list(maxiter = x, ...))`)
- the step-size is below the minimum x (as specified with `nls(..., control = list(minFac = x, ...))`)
- a singular gradient occurs

In all these cases return of the output object may be valuable for scientific model discovery, for the reasons sketched above. We have implemented the option `'warnOnly'` to determine if an output object is returned in the case that one of the above conditions is met. The implementation of this option has been incorporated into R version 2.5. A logical slot in the class `nls.control` is used to toggle the `'warnOnly'` option. To output a result object even in the case that an error is triggered, `nls` is called with `nls(..., control = list(warnOnly = TRUE, ...)`.

A better understanding of the residual surface on which optimization occurs is sometimes gained by knowledge of how many times `nls` halves the step-size in the gradient direction. We have implemented the option `'printEval'` to print the number of evaluations (of the step-size) required each iteration; this option is also included in R version 2.5. A logical slot in the class `nls.control` is used to toggle the `'printEval'` option. To print the number of evaluations required (as well as the achieved convergence tolerance), `nls` may be called with `nls(..., control = list(warnOnly = TRUE, ...)`.

C. Data formats for input into TIMP

Currently **TIMP** allows the input of data in three formats. More formats will be added in the future.

C.1. Time explicit format

The *time explicit* format for data input contains 5 lines and then a matrix of data in which each row represents a concentration profile, and each column represents spectra.

Heading line 1
Heading line 2
Time explicit
Intervalnr 5

	t_1	t_2	...	t_{m-1}	t_m
λ_1	$\Psi(t_1, \lambda_1)$	$\Psi(t_2, \lambda_1)$	...	$\Psi(t_{m-1}, \lambda_1)$	$\Psi(t_m, \lambda_1)$
λ_2	$\Psi(t_1, \lambda_2)$	$\Psi(t_2, \lambda_2)$	...	$\Psi(t_{m-1}, \lambda_2)$	$\Psi(t_m, \lambda_2)$
...	...	...	...	...	...
λ_{n-1}	$\Psi(t_1, \lambda_{n-1})$	$\Psi(t_2, \lambda_{n-1})$	...	$\Psi(t_{m-1}, \lambda_{n-1})$	$\Psi(t_m, \lambda_{n-1})$
λ_n	$\Psi(t_1, \lambda_n)$	$\Psi(t_2, \lambda_n)$	...	$\Psi(t_{m-1}, \lambda_n)$	$\Psi(t_m, \lambda_n)$

Data matrix elements are space-delimited. 'Heading line 1' and 'Heading line 2' are two lines that may be filled as desired (e.g., with a data file title). The string 'Time explicit' indicates the data format. The string 'Intervalnr' and a scalar m indicates the number of distinct points m at which spectra were measured, (note that the number of wavelengths n need not be specified). The following line contains the real-valued variable values (such as

times) $t_1, \ldots, t_m$ at which measurements were taken. The first value of each of the remaining lines represents the wavelength at which the concentration profile contained on that row was taken. The rest of each remaining row represents a (space-delimited) concentration profile $\Psi(t_1, \lambda), \Psi(t_2, \lambda), \ldots, \Psi(t_m, \lambda)$.

C.2. Wavelength explicit format

The *wavelength explicit* format for data input contains 5 lines and then a matrix of data in which each row represents spectra, and each column represents a concentration profile.

Heading line 1
Heading line 2
Wavelength explicit
Intervalnr 5

	λ_1	λ_2	$\vdots$	λ_{n-1}	λ_n
t_1	$\Psi(t_1, \lambda_1)$	$\Psi(t_1, \lambda_2)$	$\vdots$	$\Psi(t_1, \lambda_{n-1})$	$\Psi(t_1, \lambda_n)$
t_2	$\Psi(t_2, \lambda_1)$	$\Psi(t_2, \lambda_2)$	$\vdots$	$\Psi(t_2, \lambda_{n-1})$	$\Psi(t_2, \lambda_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_{m-1}	$\Psi(t_{m-1}, \lambda_1)$	$\Psi(t_{m-1}, \lambda_2)$	$\vdots$	$\Psi(t_{m-1}, \lambda_{n-1})$	$\Psi(t_{m-1}, \lambda_n)$
t_m	$\Psi(t_m, \lambda_1)$	$\Psi(t_m, \lambda_2)$	$\vdots$	$\Psi(t_m, \lambda_{n-1})$	$\Psi(t_m, \lambda_n)$

All entries above are space delimited.'Heading line 1' and 'Heading line 2' are two lines that may be filled as desired (e.g., with a data file title). The string 'Wavelength explicit' indicates the format that the input data is to take. The string 'Intervalnr' and a scalar n indicates the number of distinct wavelengths n at which measurements were taken, (note that the number of time points m need not be specified). The following line contains the real-valued wavelengths $\lambda_1, \ldots, \lambda_n$ at which measurements were taken. The first value of each of the remaining lines represent the independent variable value (such as a time) at which the spectrum contained on that row was taken. The rest of each remaining row represents a (space-delimited) spectrum$\Psi(t, \lambda_1), \Psi(t, \lambda_2), \ldots, \Psi(t, \lambda_n)$.

C.3. FLIM format

Fluorescence Lifetime Imaging Microscopy (FLIM) data is read into **TIMP** in the format described in Laptenok *et al.* (2007).

Affiliation:

Katharine M. Mullen
Department of Physics and Astronomy
Faculty of Sciences
Vrije Universiteit Amsterdam
De Boelelaan 1081

1081 HV Amsterdam, The Netherlands
E-mail: kate@nat.vu.nl
URL: http://www.nat.vu.nl/~kate/

Ivo H. M. van Stokkum
Department of Physics and Astronomy
Faculty of Sciences
Vrije Universiteit Amsterdam
De Boelelaan 1081
1081 HV Amsterdam, The Netherlands
E-mail: ivo@nat.vu.nl
URL: http://www.nat.vu.nl/~ivo/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 18, Issue 3 *Submitted:* 2006-10-01
January 2007 *Accepted:* 2007-01-10