`basicModel()`

Experimental function to fit the discrete solution of the one-component fitness-fatigue model system, with or without an estimated initial trace.

Source code	basicModel.R github.com/bsh2/fitness-fatigue-models/software/utilities/experimental/basicModel.R
File version & date	V.0.1 (22/11/2020)
Function	`basicModel()`
Importing the function using the `devtools` package	`library(devtools)` `source_url('https://raw.githubusercontent.com/bsh2/Fitness-Fatigue-Models/main/software/utilities/experimental/basicModel.R')`
Package dependencies	**Imports**: `stats, caret` **Suggests**: `ga`
Functionality	Fit the one-component IR performance model using quasi-Newton or genetic algorithms. Walk-forward cross-validation is integrated into the process to assess out-of-sample loss (but not used to further tune the model). Function includes the option to estimate an initial trace of the model component at time t=0, which decays away at the same rate as any future training.

1.1 Background

The basic model is a one-component impulse-response model describing the effect of training on performance, and arises from the following time-invariant linear system (Banister et al., 1975):

$$p(t) = p^* + Kp(t)$$

$$\frac{dP(t)}{dt} = \omega(t) - \frac{1}{\tau}p(t)$$

The first-order linear ODE above can be solved via the method of Laplace transform applied to each term, and then rearranging the substituted transforms - assumping $p(0) = 0$ - in terms of $P(s)$ to derive its transfer function (relationship between system input and output):

$$P(s) = -\frac{W(s)}{(s + \frac{1}{\tau})}$$

And then defining

$$G(s) = -\frac{1}{(s + \frac{1}{\tau})}$$

Allows $P(s)$ to be written as the product $G(s)W(s)$ which can then be solved by the convolution theorem that states the inverse laplace transform of a product is its convolution:

$$\mathcal{L}^{-1}[P(s)] = \mathcal{L}^{-1}[G(s)W(s)] = (g * w)(t)$$

Which gives

$$p(t) = (g * w)(t) = \int g(t-u)w(u)du$$

Consulting inverse laplace transform tables and substituting appropriately we get the solution

$$p(t) = \int e^{-\frac{t-u}{\tau}} w(u)du$$

Which can be discretized to give the one-component model fitted within this R function

$$\hat{p}(n) = p^* + K \sum_{i=1}^{n-1} e^{-\frac{(n-i)}{\tau}} w(i) \cdot \Delta_n$$

We have also included the option to estimate a value q of the component that represents accumulated training effects prior to the modelled period and which also decays away with future training, as follows:

$$\hat{p}(n) = p^* + q \cdot (e^{-\frac{n}{\tau}}) + K \sum_{i=1}^{n-1} e^{-\frac{(n-i)}{\tau}} w(i) \cdot \Delta_n$$

1.2 Usage

```
basicModel(inputData,
           constraints,
           method = "bfgs",
           startingValues = NULL,
           doTrace = FALSE,
           initialComponent = FALSE,
           initialWindow = NULL,
           testHorizon = NULL,
           expandRate = NULL)
```

1.3 Arguments

REQUIRED ARGUMENT | OPTIONAL ARGUMENT

Argument	Details
inputData	Data frame of time-series (i.e. sequential) data. Three columns in order of days (1:n), performances (1:n), loads (1:n). NA values used to indicate no measured performance value, 0 values used to indicate no training, on a given day. Example dataset shown here

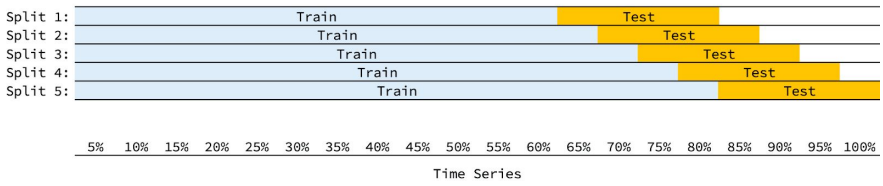
constraints	Parameter constraints (box). Supplied as a data frame with two columns (`lower, upper`) of (non-named) numeric values.
	
	`data.frame('lower' = c(),`
	`'upper' = c())`
	
	Order of each vector is `p*,K,tau` for the basic model without initial component, or `p*,K,tau,q` for the basic model with initial component. Only supply constraints for `q` if argument `initialComponent = TRUE`

`method`	Specify optimization algorithm. The quasi-Newton method L-BFGS-B is the default (`method = "bfgs"`, default). Alternatively, a genetic algorithm is available (`method = "ga"`).
startingValues	Required if using default L-BFGS-B optim method (*Not required for GA*). Supply starting point for the gradient-based optimisation method in numeric vector form in following

	order `(p*,K,tau,q)`. Only supply constraints for `q` if argument `initialComponent = TRUE`
`doTrace`	Monitor the chosen optimisation routine in the console. `TRUE`/`FALSE` argument, default is `FALSE` and no output will be provided during the fitting process.
`initialComponent`	`TRUE`/`FALSE` argument providing the option to estimate an initial decaying level for the training response component prior to the model fitting period. Note you must also supply appropriate `constraints & startingValues` values. Default is `FALSE`
`initialWindow`	Initial model training window for the out-of-sample walk forward method (expanding window). Supplied as a percentage value (without the % symbol). If `NULL`, default will be set at 60% of the data (i.e. `initialWindow = 60`)
`testHorizon`	Walk-forward testing-window size. Supplied as a percentage value (without the % symbol). If `NULL`, default will be set at 20% (`initialWindow = 20`)
`expandRate`	Walk-forward training window expansion rate. Supplied as a percentage value (without the % symbol). If `NULL`, default will be set at 4% (`expandRate = 4`)

1.4 Additional detail

Component	**Detail**
Loss function	The loss function used to assess in-sample fitness is mean-squared error (MSE), i.e. mean(RSS).
Optimisation	The default optimization algorithm is L-BFGS-B, called by the argument `method = "bfgs"`. The algorithm is a box-constrained limited memory modification of the quasi-Newton algorithm BFGS, developed by Broyden, Fletcher, Goldfarb and Shanno (1970). The algorithm is included in the `optim()` toolbox as part of the R `stats` module. The algorithm uses both the function values and gradients to iteratively search the surface of the cost function to find the global minimum. An alternative optimisation algorithm included is a genetic algorithm with stochastic local search, from the package (GA). The GA package provides a number of genetic algorithms for stochastic optimisation of real-valued problems.

<table>
<tr><td></td><td>At present, the tuning parameters selected for the problem are by default tournament selection, BLX (blend) crossover, random mutation, and a population size of minimum 10x the number of free parameters (Turner et al., 2017). These options can be changed but require adjustment of source.</td></tr>
<tr><td>Assessment of out-of-sample loss</td><td>Although the main parameter set returned is the one fitted to all the available data, the model is also fitted under a walk-forward approach (expanding window) to provide insight into out-of-sample loss, with averages provided across the splits. A diagram is provided below
 
</td></tr>
<tr><td>Bounding</td><td>Bounding is individual to each modelling problem. It might help to consider that the baseline performance value (i.e. p(0) or denoted p*) is unlikely to be higher than any maximum measured performance value in the dataset. Decay constants are unlikely to be physiologically interpretable beyond 50 days (and that's at a push), or be lower than 1 (day). The scaling values are likely to depend on the relationship between training loads and performance values.

A word of caution is not to bound too tightly, but you should provide realistic bounds for the model.</td></tr>
<tr><td>Input data</td><td>Format of input data should be consistent as described above, and as shown below:

<table>
<tr><th>days</th><th>performances</th><th>loads</th></tr>
<tr><td>1</td><td>100</td><td>0</td></tr>
<tr><td>2</td><td>NA</td><td>60</td></tr>
<tr><td>3</td><td>NA</td><td>80</td></tr>
<tr><td>4</td><td>103</td><td>0</td></tr>
<tr><td>5</td><td>102</td><td>50</td></tr>
<tr><td>6</td><td>NA</td><td>0</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td></tr>
</table>

1.5 Value (output)

For `basicModel()`, a list with components as follows:

Object	Detail

`main`	A `list` with components as follows:
	
	• `summary`
	  ○ Parameter values, in-sample loss (MSE), and basic optim info (convergence etc) (array)
	• `stats`
	  ○ In-sample fit estimates (R-squared, RMSE, MAPE) (array)
	• `predictions`
	  ○ Modelled performance values over the whole time series. (vector)
	• `optim`
	  ○ Raw output from the optimisation (list)

`cv`	A `list` with components as follows:
	
	• `summary`
	  ○ Summary statistics (min,median,mean,max,quartiles) for the training and test sets across the cross-validation splits for R-squared, RMSE and MAPE (array)
	• `metrics`
	  ○ Raw values for R-squared, RMSE, and MAPE for training and test sets of each split (array)
	• `parameters`
	  ○ Raw parameter values for each split (array)
	• `predictions`
	  ○ Raw modelled performance values over the whole time series for each split (array)
	• `raw`
	  ○ Raw optim objects for each split (list)