Matlab functions for replicate regression

Wolfram Liebermeister
Institut für Biochemie, Charité - Universitätsmedizin Berlin

1 Replicate regression package - Installation

Getting started

1. Unpack the files from the github repository
2. Include the path to the matlab directory 'replicate_regression' into your MATLAB path
3. Edit the m-file 'replicate_regression_DIR': insert the path to the matlab directory 'replicate_regression'
4. Run 'demo_replicate_regression' to see a single replicate regression
5. Run omics_data_example.m in the subdirectory replicate_regression/demo/omics_data_example to see the analysis of a small example omics data set (data and options files are provided in the same directory)

Requirements
The functions were developed and tested with matlab6.

State of the software
The functions are under development and provided 'as is'. If you would like to contribute extensions to the toolbox, please let me know.

Documentation
Documentation (in directory 'doc') has been built automatically with M2HMTL

License

Any questions?
Please send questions, comments, and bug reports to `wolfram.liebermeister@gmail.com`

2 Matlab functions

2.1 Matlab function `replicate_regression.m`

[result, options] = replicate_regression(t, y, sigma, r, flag_fix_parameters, varargin)

Bayesian replicate regression for multiple time series measured in replicate.
Data must be provided as vectors and are transformed to logarithmic scale if desired.

Function arguments

t, y, sigma, r input data (times, values, standard errors, replicate labels)
given as row vectors (see replicate_regression_core.m)

flag_fix_parameters (Boolean, optional) If set to 1, the options given in the following argument(s) will be accepted
without changes (otherwise they will be checked and updated)

varagin (optional) Either a list of property/value pairs for algorithm options (list see below).
or a structure containing the property/value pairs (this is mandatory if flag_fix_parameters is set to 1)

Function output

result matlab struct with results from replicate regression

options matlab struct with options values that were used in the calculation

The options list is supposed to be ordered by priority; earlier options override later options. The function is a wrapper for the function 'replicate_regression_core'. In converting the data to logarithms, y and sigma are either taken to be medians and geometric standard deviations, or means and standard deviations of the data values. The choice is defined by the argument 'options.transformation'

Options for matlab function `replicate_regression.m`

OPTION	IN CORE	TYPE	DEFAULT	MEANING
options.verbose		Boolean	1	Output information during regression
options.is_logarithmic		Boolean	0	Declare that data are logarithmic
options.convert_to_logarithm		Boolean	1	Convert data to logarithms for regression
options.log_transformation		string	'arithmetic'	'arithmetic', 'geometric'
options.run_crossvalidation		Boolean	0	Run crossvalidation
options.set_std		float	nan	Value to replace all data standard deviations
options.insert_std		float	1	Value to replace missing data standard deviations
options.start_at_t		float	0	Start regression curves at starting time 'start_at_t' (instead of t=0)
options.start_value		float	nan	Fixed start value for regression curves
options.shift_data		string	'mean'	Policy for shifting data before regression 'none', 'fixed_start_value', 'mean', 'initial', 'fixed_1'
options.shift_value		float	nan	Shift used when shifting the data
options.basis	X	string	'cos+sin'	Type of basis functions (see table below)
options.n_comp	X	int	nan	Fixed number of basis functions
options.n_comp_min		int	1	Minimal number of basis functions
options.n_comp_max		int	20	Maximal number of basis functions
options.use_offset	X	Boolean	1	Use constant function as one of the basis functions
options.constant_before_start	X	Boolean	0	Set all basis functions constant for t¡0
options.deviation_same_start		Boolean	0	Enforce identical start values for all replicates
options.remove_offset	X	Boolean	0	Omit offset when creating the regression curves
options.t_smooth		float	nan	Time constant for setting decreasing prior widths
options.t_jump	X	float	nan	Time constant for initial jump basis function
options.t_interp		float	t	Time points for interpolated regression curves
options.average_std	X	string	'std_dev_mean'	Type of uncertainty to be reported for average curve
options.central_offset_mean	X	float	0	Prior mean sigma_alpha_0 (for alpha_0)
options.central_offset_width	X	float	1	Prior width sigma_alpha_0 (for alpha_0)
options.central_first_mode_mean	X	float	0	Prior mean sigma_alpha_1 (for alpha_1)
options.central_first_mode_width	X	float	1	Prior width sigma_alpha_1 (for alpha_1)
options.central_mode_mean	X	vector	[]	Prior means sigma_alpha_m (for alpha_m)
options.central_mode_width	X	vector	[]	Prior widths sigma_alpha_m (for alpha_m)
options.central_jump_mean	X	float	nan	Prior means sigma_alpha_jump (for alpha_jump)
options.central_jump_width	X	float	nan	Prior widths sigma_alpha_jump (for alpha_jump)
options.deviation_offset_mean	X	float	0	Prior mean sigma_beta_0 (for beta_0)
options.deviation_offset_width	X	float	1	Prior width sigma_beta_0 (for beta_0)
options.deviation_first_mode_mean	X	float	0	Prior mean sigma_beta_1 (for beta_1)
options.deviation_first_mode_width	X	float	1	Prior width sigma_beta_1 (for beta_1)
options.deviation_mode_mean	X	float	[]	Prior means sigma_beta_m (for beta_m)
options.deviation_mode_width	X	float	[]	Prior widths sigma_beta_m (for beta_m)
options.deviation_jump_mean	X	float	0	Prior means sigma_beta_jump (for beta_jump)
options.deviation_jump_width	X	float	1	Prior widths sigma_beta_jump (for beta_jump)
options.flag_draw_sample	X	Boolean	1	Draw sample curve parameters and curve from the posterior
options.flag_time_derivative	X	Boolean	0	Compute time derivative curves

The basis functions are adjusted to the final time interval [ta,tb](from tt)

'cos'	cosine function, zero slope at t=ta and t=tb
'sin'	sine function, zero value at t=ta and t=tb
'sin_half'	sine function, zero value at t=ta
'sin_horizontal'	sine function, zero value at t=ta, zero slope at t=tb
'cos+sin'	cosine and sine functions, no restriction
'polynomial'	polynomial function, zero value at t=ta
'exp'	exponentially relaxing functions (t<0 => f=0; t>0 => f = 1-exp(t/tau)

The entire curves are shifted by a constant basis function This can be suppressed by setting options.use_offset = 0

The options marked in column "IN CORE" are also used by the underlying matlab function `replicate_regression_core.m`

2.2 Matlab function `replicate_regression_omics.m`

replicate_regression_omics(data_file, user_options_file, base_directory)

Bayesian replicate regression for omics data

Function arguments

data_file omics data file (full directory path)

user_options_file table file containing the options (full directory path)

base_directory directory name for results (full directory path)

Function output Output data and graphics are written to files

How to run a replicate regression for omics data

How to prepare data and run `replicate_regression_omics.m`

1. Create a directory for the analysis
2. Create in this directory subdirecties "data", "options", "results", and "graphics"
3. Create a data file (tab-separated text table in the format described below) and save it to the "data" subdirectory
4. Create an options file (tab-separated text table in the format described below) and save it to the "options" subdirectory
5. Start matlab and run replicate regression
   (see matlab script replicate_regression/demo/omics_data_example/omics_data_example.m)
   ```
   % Directory name for the omics set
   base_DIR = [ replicate_regression_DIR '/demo/omics_data_example/' ];
   % Name of options file
   foptions_file = [ base_DIR '/options/options_omics_data_example.csv'];
   % Run script for replicate regression of omics data
   replicate_regression_omics_analysis;
   ```

Examples of data and options files can be found in the subdirectory replicate_regression/demo/omics_data_example

Format of data file (tab-separated text file)

Line 1: Headers Headers of protein names columns (e.g., !BSUnumber, !BGnumber, !GeneName, !UniprotID), followed by sample names (as headers of data columns)

Line 2: Time points first column: !Time data columns: time points (numbers)

Line 3: Replicate numbers first column: !Replicate data columns: replicate names

Line 4 (optional): Value type !ValueType ('Value', 'Mean', or 'Std')

Further lines: numerical data

Format of options file (tab-separated text file)

Each line contains one attribute:
First column: attribute name
Second column: attribute value (string or number)

All further columns are ignored
Lines starting with the '%' character are ignored (can be used for comments)
The attribute 'options_file' allows to declare another options file containing default options
The attribute 'data_file_tsv' contains the name of the data file

Options for matlab function `replicate_regression_omics.m`

Attributes in options file (for replicate_regression_omics_analysis)

OPTION	MEANING
data_dir	directory name for data files
result_dir	directory name for result files
graphics_dir	directory name for graphics
data_file_tsv	filename for data file (tsv format, see examples)
data_file_matlab	filename for matlab data file (written during the analysis)
options_file	filename for default options file (tsv format)
options_out_tsv	filename for completed options file (tsv format, written during analysis)
translation_table_file	filename for ID mapping table (see example)
result_file_matlab	filename for
result_file_tsv	filename for hahne_salt_stress_cytosol_result.tsv
result_file_zip	filename for hahne_salt_stress_result.zip
graphics_file	file basename for graphics
data_time_unit	time unit ('min')
data_scale	'absolute' or 'log2' (also 'ln','log','log10','log2 ratio'; these are all treated like 'log2');
data_min_num_replicates	minimal number of valid replicates (genes with less valid replicates are discarded; default 1)
For "absolute" data:	
data_outliers_upper	upper threshold; points above are outliers (increase std dev by factor of 3)
data_outliers_lower	lower threshold; points below are outliers (increase std dev by factor of 3)
data_std_relative	default for relative standard deviation
data_std_minimal	minimal standard deviation
For logarithmic data	i.e., ('log2', 'ln','log','log10','log2 ratio')
data_std_log	default for standard deviation (on log scale)
data_outliers_threshold	threshold for data values (on chosen log scale) to be counted as outlier
	(check — [data value] - [median for this gene and replicate] —)
data_std_log_outlier	standard deviation (on log scale) for data counted as outliers
...	...

...	...
data_min_data_points	minimal number of data points required in the analysis (default 3)
	at least one replicate has to reach this number, points are times t¡0 do not count
	replicates with less data points are ignored
convert_to_logarithm	convert (nonlogarithmic) data to logarithms for replicate regression (Boolean)
log_transformation	type of transformation
	'arithmetic': data=mean values and plotting on absolute scale
	'geometric' : data = median values and plotting on log scale (but data on absolute scale)
ignore_std_deviations	Boolean, ignore standard deviations given in data
fixed_prior	keeping the prior fixed? (Boolean, default 0)
prior_updating	number of prior updating iterations (default 10)
updating_factor	updating factor, default 1.1
update_prior_means	change parameter means from 0 to posterior means while updating? default 0
t_smooth	time constant defining how prior widths depend on the frequency
options_start_value	fixed starting value; to be inserted into options as options.start_value
options_start_at_t	Starting time point for changes (after constant behaviour)
	to be inserted into options as options.start_at_t
options_constant_before_start	Boolean (keep curves constant before starting time)
	to be inserted into options as options.constant_before_start
regression_t_interp	time points for regression (optional)
regression_tmin	start time for regression (optional)
regression_tmax	end time for regression (optional)
crossvalidation	run crossvalidation? (Boolean, default 0)
postprocess_normalise	Boolean, default 1
graphics_individual	file basename (used in script replicate_regression_omics_selected'
graphics_scale	default 'log2', 'linear'
graphics_format	'eps', 'png' (for technical reasons, 'eps' needs to be written in single quotes)
convenience_name	type of protein names to be used in graphics (default 'SubtiWiki_20090701')
normalise_by_median	(Boolean, default TRUE)
mark_outliers_percentage	percentage of data points to be marked as outliers based on crossvalidation error

Additional attributes in options file for individual graphics (function 'replicate_regression_omics_selected')

OPTION	MEANING
graphics_scale	'log2','linear'
postprocess_normalise	1
element_id	id (or list, selected by —)
element_name	name (or list, selected by —)
delimiter_symbol	symbol for delimiting list of elements (in element_id, element_name)
title_string	title for graphics
x_label	x label for graphics
y_label	y label for graphics
plot_data	produce plot for data (single element)
plot_replicates	produce plot with replicates (single element)
plot_regression	produce plot for regression curves (single element)
plot_all	produce joint plots for all elements

3 Overview of replicate regression method

3.1 Basic idea

We developed a regression method for interpolating and combining time series data from different biological replicas of an experiment. We make the basic assumption that the replicas shows different, but similar behaviour. To put this into quantitative terms, we represent each replica curve $x_r(t)$ (with replica index r and time t) by a sum $x_r(t) = \bar{x}(t) + \Delta x_r(t)$, where $\bar{x}(t)$ represents an average behaviour shared by all replicas and $\Delta x_r(t)$ denotes the deviation of this specific replica from the average behaviour. We further assume that both $\bar{x}(t)$ and $\Delta x_r(t)$ are smooth curves and that the measured values $y_r(t)$ are noisy versions of the values $x_r(t)$. In our present implementation, which is described below, the method is well suited for relatively smooth time series (such as protein abundances), but less for time series that contain both rapid and slow behaviour (like for instance, the metabolite concentrations time series). The strength of our method is that it is relatively insensitive to systematic errors that would usually arise if biologica replicas have both systematic offsets and different sampling time points. A test example is shown in Figure 1.

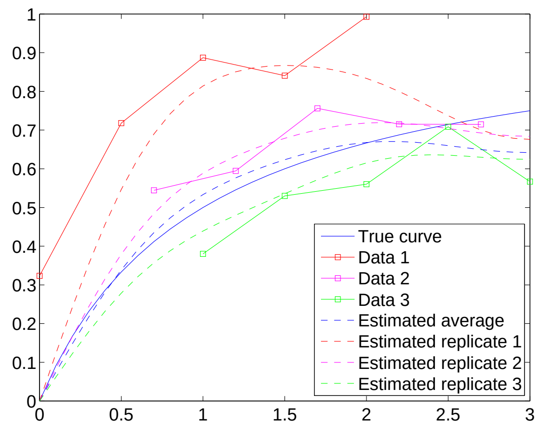

Figure 1: Example for Bayesian regression. Artificial data (squares, connected by lines to guide the eye) were created from a true curve (solid blue curve) by adding random noise (per data point) and systematic offsets (for the three replicates, shown in red, magenta, and green). Our regression for multiple time series estimates curves for the replicates (dashed curves, red magenta, green) and an average curve (dashed, blue). By the choice of the basis functions, the curves are restricted to start at the point (0,0). As a result of the systematic errors and unequal sampling, a naive regression would probably yield a regression curve that declines at the end. Our method, however, can account for such systematic errors and yields a satisfactory mean curve.

3.2 Regression model

Mathematically, we estimate the curves $x_r(t)$ for n biological replicas as well as the average curve $\bar{x}(t)$ by the following model:

$$\begin{aligned} y_r(t) &= x_r(t) + \eta_{rt} \\ x_r(t) &= \bar{x}(t) + \Delta x_r(t) = \sum_l \alpha_l\, v_l(t) + \sum_{rl} \beta_{rl}\, v_l(t) \end{aligned} \tag{1}$$

where $y_r(t)$ and $x_r(t)$ denote the measured and the true value, respectively, from replica r at time t. Random measurement errors are represented by the term η_{rt}, which is an independent standard normal random variable. In the second equation, the curves are expanded in into a sum of predefined basis functions $v_l(t)$. The first term (with coefficients α_l) describes the average curve, while the second term (with coefficients β_{rl}) describes the difference between the replicate curves and the average curve. With a predefined set of basis functions $v_l(t)$, the regression problem boils down to an estimation of the coefficients α_l and β_{rl}.

3.3 Basis functions

By an appropriate choice of the basis functions $v_l(t)$, we can ensure that the regression curves will satisfy certain constraints. For instance, with sine functions

$$v_l(t) = \sin\left(\frac{\pi\, l\, t}{T}\right) \tag{2}$$

where T is the duration of the full time series, the curves start at $x(0) = 0$ and have a zero slope at $t = T$. Other possible choices would be both sine and cosine functions (for a usual Fourier series) or powers of t (for polynomial regression). In the following, we assume that the basis functions $l = 1, 2, 3, ...$ are ordered such that higher indices l correspond to faster fluctuations. The basis function with index $l = 0$ represents a constant behaviour, $v_0 = 1$. To obtain continuous curves with a constant value for times $t < 0$ (steady state before an experimental perturbation at $t = 0$) and zero slope at $t = T$, we can use the following basis set:

$$\begin{aligned} v_0(t) &= 1 \\ v_l(t) &= \Theta(t)\, \sin\left(\frac{\pi\,(l-1/2)\, t}{T}\right) \qquad \text{for} \quad l > 0 \end{aligned} \tag{3}$$

where $\Theta(\cdot)$ denotes the step function satisfying $\Theta(x < 0) = 0$ and $\Theta(x \geq 0) = 1$.

3.4 Bayesian parameter estimation

For our regression, we require that (i) the regression curves are close to the data points; (ii) the curves are relatively smooth, so we consider only a finite number of low-frequency basis functions and try to keep the higher-order coefficients small; (iii) common variance of all replicate curves is explained by the average curve, so the coefficients β_{rl} should be kept relatively small. A compromise between these requirements can be realised by a Bayesian parameter estimation: the coefficients α_l and β_{rl} are estimated by maximising their posterior probability density,

$$\text{Prob}(\alpha, \beta | y) \sim \text{Prob}(y|\alpha, \beta)\, \text{Prob}(\alpha, \beta) \tag{4}$$

with a Gaussian priors $\alpha_l = \mathcal{N}\left(0, \sqrt{1/\lambda_l^\alpha}\right)$ and $\beta_{rl} = \mathcal{N}\left(0, \sqrt{1/\lambda_l^\beta}\right)$. By taking the logarithm of (4) and neglecting constant terms, we obtain the score function

$$F(\alpha, \beta) = \sum_{rt} \left(y_r(t) - \sum_l \alpha_l\, v_l(t) + \sum_{rl} \beta_{rl}\, v_l(t) \right)^2 + \sum_l \lambda_l^\alpha\, \alpha_l^2 + \sum_{lr} \lambda_l^\beta\, \beta_{lr}^2 \tag{5}$$

to be minimised with respect to the curve coefficients α_l and β_{lr}. As the replicas can differ in their measurement time points, the double sum runs over the replicas r and for each of them, over the respective timepoints t. The score function consists of the sum of square residuals (related to the likelihood) and a number of quadratic cost terms (related to the prior densities). The optimisation of α_l and β_{lr} is a quadratic minimisation problem that can be solved by means of linear algebra.