eleven Documentation

Release 0.1

Tim Smith

February 21, 2014

Contents

Tim D. Smith, @biotimylated.

Eleven is a Python library for performing multi-gene RT-qPCR gene expression normalization. It is a free, open-source implementation of the GeNorm algorithm described by Vandesompele et al. in 2002.

Eleven requires Python 2.7. Earlier versions will not be supported. Python 3.x support is on the roadmap. You will need a Scientific Python stack, including pandas and scipy. If you don't have these, you can install the free version of the Anaconda environment, which has everything you need.

A sample analysis session looks like this:

```
# Read PCR data into a pandas DataFrame. You want a data file where each
# row corresponds to a separate well, with columns for the sample name,
# target name, and Cq value. NTC wells should have the sample name set to
# a value like 'NTC'.
>> df = pd.read_csv('my_data.csv')

# If your Sample, Target, and Cq columns are called other things, they
# should be renamed to Sample, Target, and Cq.
>> df = df.rename(columns={'Gene': 'Target', 'Ct': 'Cq'})

# Drop the wells that are too close to the NTC for that target.
>> censored = eleven.censor_background(df)

# Rank your candidate reference genes.
>> ranked = eleven.rank_targets(censored, ['Gapdh', 'Rn18s', 'Hprt',
    'Ubc', 'Actb'], 'Control')

# Normalize your data by your most stable genes and compute normalization
# factors (NFs).
>> nf = eleven.calculate_nf(censored, ranked.ix['Target', 0:3], 'Control')

# Now, normalize all of your expression data.
>> censored['RelExp'] = eleven.expression_nf(censored, nf, 'Control')
```

Wasn't that easy? This adds the relative expression of each well as a column of the data frame. Now you can use regular pandas tools for handling the data, so

```
censored.groupby(['Sample', 'Target'])['RelExp'].aggregate(['mean', 'std'])
```

gives you a nice table of means and standard deviations for each target in each sample.

CHAPTER 1

eleven package

1.1 Module contents

eleven.**average_cq**(*seq*, *efficiency=1.0*)

Given a set of Cq values, return the Cq value that represents the average expression level of the input.

The intent is to average the expression levels of the samples, since the average of Cq values is not biologically meaningful.

Parameters

- **seq** (*iterable*) – A sequence (e.g. list, array, or Series) of Cq values.
- **efficiency** (*float*) – The fractional efficiency of the PCR reaction; i.e. 1.0 is 100% efficiency, producing 2 copies per amplicon per cycle.

Returns Cq value representing average expression level

Return type float

eleven.**calculate_all_nfs**(*sample_frame*, *ranked_targets*, *ref_sample*)

For a set of n ranked_genes, calculates normalization factors NF_1, NF_2, ..., NF_n. NF_i represents the normalization factor generated by considering the first i targets in ranked_targets.

calculate_nf (which returns only NF_n) is probably more useful for routine analysis.

Parameters

- **sample_frame** (*DataFrame*) – A sample data frame.
- **ranked_targets** (*iterable*) – A list or Series of target names, in order of descending stability (ascending M).
- **ref_sample** (*string*) – The name of the sample to normalize against.

Returns a DataFrame with columns 1, 2, ..., n containing normalization factors NF_1, ..., NF_n for each sample, indexed by sample name.

Return type DataFrame

eleven.**calculate_nf**(*sample_frame*, *ref_targets*, *ref_sample*)

Calculates a normalization factor from the geometric mean of the expression of all ref_targets, normalized to a reference sample.

Parameters

- **sample_frame** (*DataFrame*) – A sample data frame.

- **ref_targets** (*iterable*) – A list or Series of target names.
- **ref_sample** (*string*) – The name of the sample to normalize against.

Returns a Series indexed by sample name containing normalization factors for each sample.

eleven.**calculate_v**(*nfs*)

Calculates V(n+1/n) values. Useful for establishing the quality of your normalization regime. See Vandesompele 2002 for advice on interpretation.

Parameters nfs (*DataFrame*) – A matrix of all normalization factors, produced by *calculate_all_nfs*.

Returns a Series of values [V(2/1), V(3/2), V(4/3), ...].

eleven.**censor_background**(*sample_frame, ntc_samples=['NTC'], margin=None*)

Selects rows from the sample data frame that fall *margin* or greater cycles earlier than the NTC for that target. NTC wells are recognized by string matching against the Sample column.

Parameters

- **sample_frame** (*DataFrame*) – A sample data frame.
- **ntc_samples** (*iterable*) – A sequence of strings giving the sample names of your NTC wells, i.e. ['NTC']
- **margin** (*float*) – The number of cycles earlier than the NTC for a "good" sample, i.e. log2(10)

Returns a view of the sample data frame containing only non-background rows

Return type DataFrame

eleven.**collect_expression**(*sample_frame*, *ref_targets*, *ref_sample*)

Calculates the expression of all rows in the sample_frame relative to each of the ref_targets. Used in rank_targets.

Parameters

- **sample_frame** (*DataFrame*) – A sample data frame.
- **ref_targets** (*iterable*) – A sequence of targets from the Target column of the sample frame.
- **ref_sample** (*string*) – The name of the sample to which expression should be referenced.

Returns a DataFrame of relative expression; rows represent rows of the sample_frame and columns represent each of the ref_targets.

Return type DataFrame

eleven.**expression_ddcq**(*sample_frame*, *ref_target*, *ref_sample*)

Calculates expression of samples in a sample data frame relative to a single reference gene and reference sample using the Cq method.

For best results, the ref_sample should be defined for all targets and the ref_target should be defined for all samples, or else the series you get back will have lots of NaNs.

Parameters

- **sample_frame** (*DataFrame*) – A sample data frame.
- **ref_target** (*string*) – A string matching an entry of the Target column; the target to use as the reference target (e.g. 'Gapdh')
- **ref_sample** (*string*) – A string matching an entry of the Sample column.

Returns a Series of expression values for each row of the sample data frame.

Return type Series

eleven.**expression_nf**(*sample_frame*, *nf_n*, *ref_sample*)

Calculates expression of samples in a sample data frame relative to pre-computed normalization factors.

ref_sample should be defined for all targets or the result will contain many NaNs.

Parameters

- **sample_frame** (*DataFrame*) – A sample data frame.
- **nf_n** (*Series*) – A Series of normalization factors indexed by sample. You probably got this from *compute_nf*.
- **ref_sample** (*string*) – The name of the sample to normalize against, which should match a value in the sample_frame Sample column.

Returns a Series of expression values for each row in the sample data frame.

Return type Series

eleven.**log2**(*x*)

eleven.**rank_targets**(*sample_frame*, *ref_targets*, *ref_sample*)

Uses the geNorm algorithm to determine the most stably expressed genes from amongst ref_targets in your sample.

See Vandesompele et al.'s 2002 Genome Biology paper for information about the algorithm: http://dx.doi.org/10.1186/gb-2002-3-7-research0034

Parameters

- **sample_frame** (*DataFrame*) – A sample data frame.
- **ref_targets** (*iterable*) – A sequence of targets from the Target column of sample_frame to consider for ranking.
- **ref_sample** (*string*) – The name of a sample from the Sample column of sample_frame. It doesn't really matter what it is but it should exist for every target.

Returns a sorted DataFrame with two columns, 'Target' and 'M' (the relative stability; lower means more stable).

Return type DataFrame

eleven.**validate_sample_frame**(*sample_frame*)

Makes sure that *sample_frame* has the columns we expect.

Parameters **sample_frame** (*DataFrame*) – A sample data frame.

Returns True (or raises an exception)

Raises

- **TypeError** – if sample_frame is not a pandas DataFrame
- **ValueError** – if columns are missing or the wrong type

e