A statistical model summarising kinetic parameter information from the BRENDA database

Introduction

Our statistical model aims to summarise information from the BRENDA database about Michaelis constants or "Km"s. We evaluated a range of modelling approaches and chose one that balances coverage with good out-of-sample predictive performance for the biological parameters that we judged would be most interesting for systems biologists.

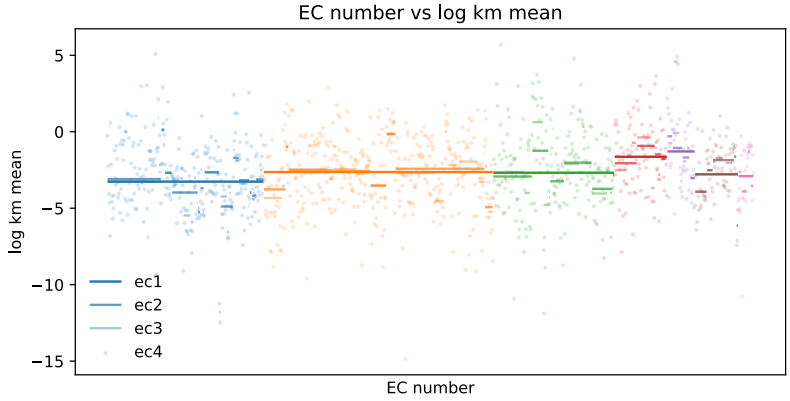
Challenges

Modelling The BRENDA dataset presents some specific challenges.

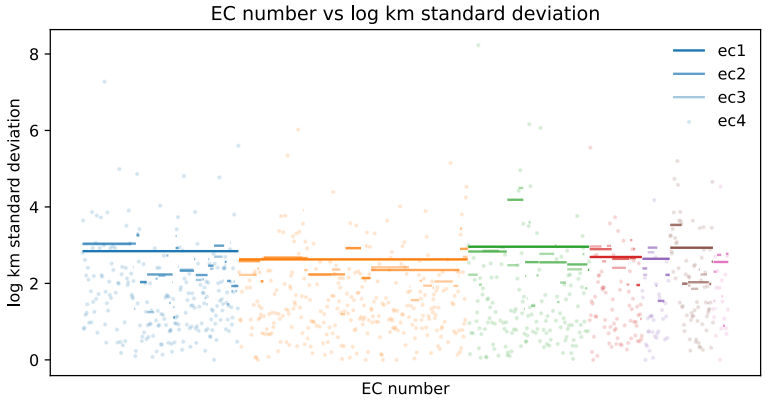
EC numbers

BRENDA classifies enzymes according to a four level tree structure represented by EC numbers. For example, the EC number 1.2.3.4 represents an oxidoreductase (EC1 group 1) that acts on the aldehyde or oxo group of donors (EC2 group 1.2) with oxygen as the acceptor (EC3 group 1.2.3) and specifically catalyses the reaction \$oxalate + O_2 + 2 H+ 2 CO_2 + H_{2O2} \$ (EC4 group 1.2.3.4).

We postulate that each component of an enzyme's EC number provide information about the values of its kinetic parameters, and would like our statistical model to be able to use this information. Unfortunately, doing so is not straightforward. As the figure below shows, there are only fairly weak systematic relationships between ec categories and average log km values.



However, the EC numbers also carry distributional information. The figure below shows the standard deviation of log km reports in each ec category. We can see that, for example, the standard deviation within EC1 category 2 tends to be somewhat lower than for EC1 category 1, and that there are also noticeable differences at the EC2 level.



From this preliminary investigation it seems like, in order to take full advantage of the information provided by the EC number hierarchy, a distributional model will be required.

Between-organism heterogeneity

It is not possible to ignore the information that BRENDA provides about organisms, as the km value for an enzyme/substrate combination is often very different for different organisms. To illustrate, the figure below shows a histogram of differences between average measured km values for enzyme/substrate combi-

nations that were available for both *Homo sapiens* and *Escherichia coli*. There are often differences of more than 3 on log scale, which is close to the average overall within-EC1 standard deviation.

Experimental conditions

Not all the measurements are at standard conditions - for example some measurements record unusual temperatures and pH values. Not all reports record these conditions, however. We therefore had to decide whether or not to attempt to model the effects of the available experimental conditions, and, if so, how to account for the cases where data about the experimental conditions is missing.

Incomplete substrate information

BRENDA's API provides two fields from which the substrates for a given km report can be inferred. One is a string called `substrate`, which is a human-readable name like 'ATP'. The other is an integer-valued id called `ligandStructureId`. Unfortunately it is not possible to link these fields with non-BRENDA identifiers, except through potentially unreliable string matching operations on the `substrate` field. As a result, we chose not to include detailed information about the substrates' chemical structures.

Non-natural substrates

Many of the measurements in the BRENDA database relate to the km values of enzyme/substrate combinations that do not occur naturally. These combinations are less interesting for general systems biology applications, and might introduce biases for the other substrates. On the other hand, it is also possible that the unnatural substrates. It was therefore not clear in advance whether or not it would be best to include the unnatural substrates in our model.

Method

Data fetching

We fetched data from BRENDA using their public SOAP API. See the script `fetch_brenda_data.py` and the directory `data/raw/` in the project github repository for details. The raw data included three tables:

- output of the SOAP method `getKmValue`, stored in `data/raw/brenda_km_measurements.csv`
- output of the SOAP method `getNaturalSubstrate`, stored in `data/raw/brenda_natural_substrates.csv`
- output of the SOAP method `getTemperatureOptimum`, stored in `data/raw/brenda_temperature_optima.csv`

Data processing

We made several significant data processing choices.

Preprocessing

The first data processing step was a non-destructive operations applied to all reports. See the function `preprocess` in the module `src/data_preparation.py` for full details, and the file `data/processed/km_preprocessed.csv` for the output. Briefly, we did the following:

- Edit column names so that they are lower case and broadly consistent with python naming conventions.
- Standardise null values such as `-999`.
- Add natural ligands information (a `frozenset` valued column of the natural ligands for each report, if available, and a boolean column indicating if the target ligand is one of the natural ligands).
- Add real-valued `temperature`, `ph` and `mols` columns by parsing the `commentary` field for each report. See the regular expressions `NUMBER_REGEX`, `TEMP_REGEX`, `PH_REGEX` and `MOL_REGEX` in the module `src/data_preparation.py` for details.
- Add substrate type column: this is either the name of the substrate if it is one of the manually specified cofactors listed in the variable `COFACTORS` in the module `src/data_preparation.py`, or else "other"
- Add a `biology` column by concatenating the columns `ec4`, `organism` and `substrate`.

Assigning biologies

An important part of the data processing step is to specify which reports our model should consider biologically identical. We call such equivalence classes 'biologies'. The natural choice is for biologies to be determined based on combinations of EC4 number, organism and substrate, as this is the finest-grained information that BRENDA provides. However, a simple assignment would leave many very sparsely populated biologies, presenting a challenge for our modelling approach.

To address this problem we used a strategy of starting with naive, maximally fine-graied biologies, and then "lumping" together sparsely populated biologies by ignoring progressively more differences. The code implementing this strategy can be found in the function `lump_biologies` in the module `src/data_preparation.py`.

Here is a table with the number of biologies and sparse biologies remaining at each stage in the procedure, with sparsity defined as having fewer than 2 unique biology/literature combinations.

Step	New definition	Biologies	Sparse biologies
1.	ec4, organism, substrate	2814	1460
2.	ec4, organism, substrate type	2438	864
3.	ec4, organism	2326	642
4.	ec4	2262	516
5.	ec3	1873	54

This procedure results in far fewer sparsely populated biologies, while allowing fine-grained differences between well-populated biologies to be taken into account.

Filtering

We performed two successive filtering steps - one at the level of reports and one at the level of biology/literature combinations. At the first step we discared reports if they matched any of the following criteria:

- null values in the columns `ec4`, `km`, `organism` or `substrate`
- negative or zero `km` value
- zero-valued `ligand_structure_id`
- `organism` value not one those specified in the variable `ORGANISMS_TO_INCLUDE` in the module `src/data_preparation.py`
- `temperature` value not between 5 and 50
- `ph` value not between 4 and 9
- `is_natural` column not `True`

At the second stage of filtering, biology/literature combinations with fewer than 2 observations were removed in order to prevent model bias due to sparsely populated groups.

Grouping

Instead of modelling reports directly, we chose to group together reports with the same biology and study, treating the mean log-scale km as a single observation. We took this approach because of the presence in the BRENDA data of different kinds of study. In some cases - presumably when the aim of a study was to discover the sensitivity of a kinetic parameter to changes in conditions - many reports with the same enzyme, organism, substrate and study are available, with a range of different Km values and different experimental conditions recorded in the `commentary` field. In other cases a study will report only a single value for one kinetic parameter.

Due to this discrepancy it seemed wrong to treat reports from better populated studies as equivalent to reports from more concise studies. While taking the mean for a given study/biology combination before modelling destroys information, we judged that it would lead to more realistic results than treating each

report as an observation, especially since we chose not to attempt to model the effects of experimental conditions.

Statistical model

All of our models were Bayesian regression models, and are determined by specifying a measurement model which allocates a probability density to any possible combination of measurements, depending on the values of some unknown parameters, and a prior model that allocates a probability density to any possible configuration of parameter values.

These models are described below using some notational conventions for conciseness:

- $N(a, b)$ represents the normal distribution with mean a and standard deviation b, i.e.

$$\frac{1}{\sqrt{2\pi}b} \exp\left(-\frac{1}{2}\left(\frac{y-a}{b}\right)^2\right)$$

- $HN(a, b)$ represents the half-normal distribution, i.e. the normal probability density function with support only for non-negative numbers.
- $ST(a, b, c)$ represents the student-T distribution with a degrees of freedom, mean b and standard deviation c, i.e.

$$\frac{\Gamma\left((a+1)/2\right)}{\Gamma(a/2)} \frac{1}{\sqrt{a\pi}\, c} \left(1 + \frac{1}{a}\left(\frac{y-b}{c}\right)^2\right)^{-(a+1)/2}$$

- $gamma(a, b)$ represents the gamma distribution with parameters a and b, i.e.

$$\frac{b^a}{\Gamma(a)} y^{a-1} \exp(-b\, y)$$

- The symbol $\sim$ represents the relation of having a probability distribution: for example $a \sim N(0, 1)$ describes a model where the variable a has a standard normal probability distribution.
- Subscripts represent indexes and superscripts represent labels. For example, the term a_c^b denotes a variable a with label b that is indexed according to c. The reason for using superscript labels is to allow symbols like μ, τ and a to be re-used when the parameters they represent perform analogous functions.

Very simple model

For comparison we tested a very simple model with just three parameters μ, ν and σ: respectively a global mean log km value and the degrees of freedom and standard deviation of the student-t measurement error distribution. The full model specification in tilde notation is as follows:

$$\begin{aligned} y &\sim ST(\nu, \mu^{\ln km}, \sigma) \\ \nu &\sim gamma(2, 0.1) \\ \mu^{\ln km} &\sim N(-1, 2) \\ \sigma &\sim HN(0, 2) \end{aligned}$$

The priors for $\mu^{\ln km}$ and σ were chosen based on their quantiles - we judged that it was very unlikely that the global mean log km would be less than -5 or greater than 3 and the $HN(0, 2)$ distribution similarly covers the range of plausible standard deviations. We chose the prior for ν following the analysis in Juárez and Steel (2010).

Simple model

We next made a more complex model, adding a parameter for each biology, as well as a partial pooling parameter $\tau^{\ln km}$.

$$\begin{aligned} y &\sim ST(\nu, \ln km_{biology}, \sigma) \\ \ln km &\sim N(\mu^{\ln km}, \tau^{\ln km}) \\ \nu &\sim gamma(2, 0.1) \\ \mu^{\ln km} &\sim N(-1, 2) \\ \tau^{\ln km} &\sim HN(0, 2)\sigma \qquad\qquad \sim HN(0, 2) \end{aligned}$$

The prior for $\tau^{\ln km}$ was chosen based on our knowledge of the range of plausible values for the variation by biology of average log-scale km values.

Final model

Our final model adds a distributional component to the simple model, according to which the standard deviation of the measurement error distribution varies according to ec3 number and substrate type. In addition, in the final model the parameter $\mu_{\ln km}$ varies by ec1 number.

$$
\begin{aligned}
y &\sim ST(\nu, \ln km_{biology}, \sigma_{biology}) \\
\sigma &= \exp(\mu^{\sigma} + a^{EC}_{EC3} + a^{sub}_{substrate\ type}) \\
a^{EC} &\sim N(0, \tau^{EC}) \\
a^{sub} &\sim N(0, 1) \\
\tau^{EC} &\sim HN(0, 2) \\
\ln km &\sim N(\mu^{\ln km}_{EC1}, \tau^{\ln km}) \\
\nu &\sim gamma(2, 0.1) \\
\mu^{\ln km} &\sim N(-1, 2) \\
\mu^{sigma} &\sim HN(0, 2) \\
\tau^{\ln km} &\sim HN(0, 2)
\end{aligned}
$$

Model evaluation procedure

We evaluated our models by fitting them to the data derived from the data fetching and processing steps described above. We then estimated the models' leave-one-out log predictive density using the Pareto-smoothed importance sampling method described in Vehtari, Gelman, and Gabry (2017) and implemented using the Python library Arviz Kumar et al. (2019).

Results

Discussion

References

Juárez, Miguel A., and Mark F. J. Steel. 2010. "Model-Based Clustering of Non-Gaussian Panel Data Based on Skew-t Distributions." *Journal of Business & Economic Statistics* 28 (1): 52–66. https://doi.org/10.1198/jbes.2009.07145.

Kumar, Ravin, Colin Carroll, Ari Hartikainen, and Osvaldo Martin. 2019. "ArviZ a Unified Library for Exploratory Analysis of Bayesian Models in Python." *Journal of Open Source Software* 4 (33): 1143. https://doi.org/10.21105/joss.01143.

Vehtari, Aki, Andrew Gelman, and Jonah Gabry. 2017. "Practical Bayesian Model Evaluation Using Leave-One-out Cross-Validation and WAIC." *Statistics and Computing* 27 (5): 1413–32. https://doi.org/%7B10.1007/s11222-016-9696-4%7D.