ISEtools: Tools for Ion Selective Electrodes

Peter W. Dillingham
University of Otago

Basim S.O. Alsaedi
University of New England

Christina M. McGraw
University of Otago

Aleksandar Radu
Keele University

Abstract

Ion-selective electrodes (ISEs) are increasingly used in demanding applications near the non-linear portion of the Nikolskii-Eisenman equation. The **ISEtools** package provides a set of tools for analysing ISE data using the Nikolskii-Eisenman equation in a Bayesian framework through R. **ISEtools** implements methods first described in Dillingham, Radu, Diamond, Radu, and McGraw (2012) and expanded on in Dillingham, Alsaedi, and McGraw (2017).

Key features of **ISEtools** include:

- Use of OpenBUGS or jags to implement the Bayesian models through R.
- Substantial automation allowing scientists with limited knowledge of Bayesian methods or R to apply these techniques.
- Characterisation of ISEs using calibration data, estimating model parameters and limit of detection (LOD).
- Analysis of experimental samples, using basic or standard addition methods.
- Compatibility with single ISEs or multiple ISEs in an array.

Two examples are used to demonstrate use of **ISEtools** and its core functions, `loadISEdata`, `describeISE`, `analyseISE`:

(1) Lead in soil: formatting requirements for external data, importing it for analysis using `loadISEdata`, and performing basic analyses using `describeISE` and `analyseISE`.

(2) Carbonate in seawater: creating a customised analysis and plot, highlighting more advanced features of the package.

Keywords: ion-selective electrodes, ISEs, calibration, electrochemistry, non-linear regression, limit of detection, LOD, loadISEdata, describeISE, analyseISE, R, OpenBUGS, jags.

---

1. Introduction

This document provides an overview of the functionality of the package **ISEtools**, and its use analysing data from ISEs. It is assumed the reader has basic familiarity with R (e.g. installing libraries, scripting), and will install **ISEtools**, the required software OpenBUGS (www.openbugs.net), and required libraries **R2WinBUGS** and **BRugs**[1] prior to running examples themselves.

Ion-selective electrodes convert analyte activity to an electrical signal through an ion-selective glass or polymer membrane, and are governed by the Nikolsii-Eisenman equation (Eisenman, Rudin, and Casby 1957; Dillingham et al. 2012), parameratised in **ISEtools** as

$$y = \texttt{a} + \texttt{b}\log_{10}(x + \texttt{c}) + \texttt{error}, \tag{1}$$

where $\texttt{error} \sim N(0, \texttt{sigma}^2)$, y is the emf response of the ISE, x is the activity of the ion of interest, `a` is a baseline emf, `b` is a slope linked to the valence of the primary ion, temperature, and natural constants, `c` is a parameter linked to the interfering ions within the chemical matrix (and dependent on materials/methods used to construct the ISE). For numerical reasons, the model also uses a parameter $\texttt{cstar} = \texttt{c}^{0.1}$.

The expected response is shown in Figure 1: the flat region occurs when the activity $x \ll \texttt{c}$ and the Nernstian portion occurs where $x \gg \texttt{c}$. **ISEtools** is designed for demanding applications where data are observed across the full response curve. For datasets entirely within the Nernstian region, standard regression theory may be used instead.

There are (typically) two independent sources of data that are collected. First, there are **calibration** data, where both x and y are observed. The calibration data are used to estimate model parameters `a, b, c, cstar, sigma`. Secondly, there may also be **experimental** data, where only y is observed and inverse methods must be used to estimate x, conditional on model parameters. Dillingham et al. (2012) describe a Bayesian approach to this problem, which allows for complex sampling distributions and non-standard data sources, while Dillingham et al. (2017) expands the methods to estimate LOD in a manner consistent with recommendations from the International Union of Pure and Applied Chemistry (IUPAC) and others (Montville and Voigtman (2003); Eksperiandova, Belikov, Khimchenko, and Blank (2010); Desimoni and Brunetti (2012)). For experimental samples, the Bayesian approach also easily accommodates standard addition data, where an experimental sample has an aliquot with known activity and volume added to the original sample and the change in emf is recorded. Standard addition techniques are useful for combating drift in `a`, but often lead to asymmetric sampling distributions.

In Section 2, data structures, key functions, and Bayesian methods are described. Section 3 provides examples demonstrating basic implementation and key features of **ISEtools**, with a brief conclusion in Section 4.

[1] Alternatively, the program jags and library **rjags** may be used. See Section 2.3 for details.

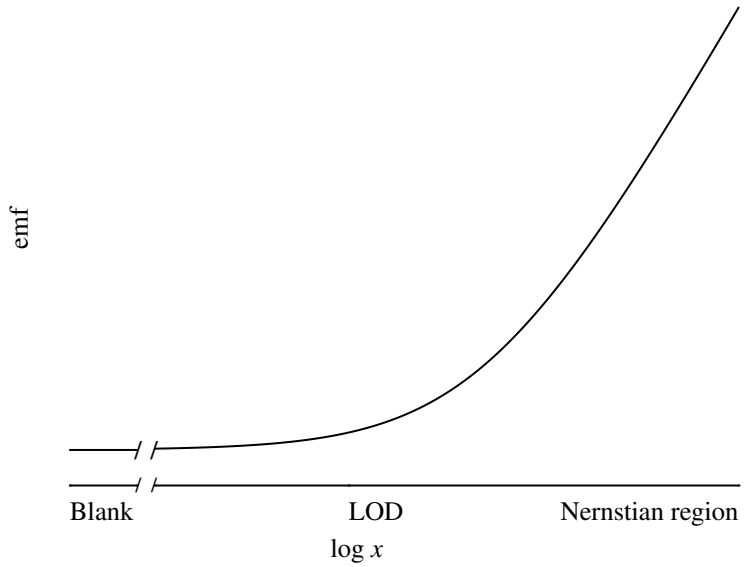

Figure 1: The response of an ISE, showing the flat region indistinguishable from a blank, the curvilinear response near the limit of detection (LOD), and the log-linear Nernstian region.

2. Methods

We describe data structures in Section 2.1, core functions in Section 2.2, and the Bayesian methods in Section 2.3. Examples showing implementation are in Section 3.

2.1. Data Structures

As described in Section 1, ISE data includes **calibration** data, where both x and y are known and used to estimate model parameters, and **experimental** data, where y is observed and inverse methods are used to estimate x. The **experimental** data may come in two formats, 'Basic' or 'Standard Addition'. Finally, data may be recorded for a single ISE, or for multiple ISEs in a sensor array. Calibration data are required, while experimental data are optional.

Data are normally stored externally, e.g. in Excel or text files, and then imported into R. The **ISEtools** library has a function `loadISEdata` described in Section 2.2 that imports tab-delimited text files and processes the data for use in other functions. Data stored in Excel (and other formats) can easily be saved in tab-delimited text format, e.g. via 'Save as'.

The text files have specific formatting requirements, demonstrated in Figure 2:

- The **calibration** data must include a header row with the variable names `ISEID`, `log10x`, and `emf`, followed by rows of data, as in Figure 2a.
  - `ISEID` indicates the ISE making the measurement. Values must be sequentially numbered, starting at 1 (even if there is only 1 ISE), e.g an array of three ISEs must

assign them the labels, 1, 2, 3.

- `log10x` is the known $\log_{10}$ activity of the calibration sample; activity is often approximated by concentration. That is, `log10x` $= \log_{10} x$.
- `emf` is the recorded emf, in mV. That is, `emf` $= y$.
- Additional variables (e.g. 'batch', as in the carbonate example in Section 3) may be included if desired: avoid spaces and special characters in variable names.
- Note that variable names in R are case-sensitive, e.g. `ISEID` is correct, `iseid` is not.

- The (optional) **experimental** data, in either 'Basic' or 'Standard Addition' formats.
  - The 'Basic' format must include a header row with the names `ISEID`, `SampleID`, and `emf`, followed by rows of data, as in Figure 2b. `ISEID` and `emf` are defined as before; `SampleID` indicates the experimental sample being measured (i.e. 1, 2, 3, ...).
  - The 'Standard Addition' format must include a header row with the names `ISEID`, `SampleID`, `emf1`, `emf2`, `V.s`, `V.add`, and `conc.add`, followed by rows of data, as in Figure 2c. `ISEID` and `SampleID` are defined as before, while:
    - `emf1` is the emf (mV) of the sample before the aliquot is added;
    - `emf2` is the emf (mV) of the sample after the aliquot is added;
    - `V.s` is the volume (ml) of the sample before the aliquot is added;
    - `V.add` is the volume (ml) of the aliquot;
    - `conc.add` is the concentration (or activity, if known) of the aliquot.

2.2. Functions

There are three core functions in **ISEtools**:

- `loadISEdata`, which imports and processes tab-delimited data files in the format described in Section 2.1 and shown in Figure 2. Inputs are a file with **calibration** data and

	A	B	C
1	ISEID	log10x	emf
2	1	-9.000116	8.558784
3	1	-6.99685	8.941667
4	1	-5.962362	13.82427
5	1	-4.970696	32.16092
6	1	-3.996123	56.68302
7	1	-3.076335	85.45835
8	2	-9.000116	-28.3812

Lead_calibration

(a) Calibration data.

	A	B	C
1	ISEID	SampleID	emf
2	1	1	25.49
3	2	1	-7.69
4	3	1	-284.32
5	1	2	29.61
6	2	2	-3.4
7	3	2	-274.83
8	1	3	32.19

Lead_experimentalBasic

(b) Experimental data, basic format.

	A	B	C	D	E	F	G
1	ISEID	SampleID	emf1	emf2	V.s	V.add	conc.add
2	1	1	25.49	51.14	25	0.02	0.1
3	2	1	-7.69	19.31	25	0.02	0.1
4	3	1	-284.32	-246.38	25	0.02	0.1
5	1	2	29.61	52.78	25	0.02	0.1
6	2	2	-3.4	21.52	25	0.02	0.1
7	3	2	-274.83	-243.84	25	0.02	0.1
8	1	3	32.19	57.68	25	0.02	0.1

Lead_experimentalSA

(c) Experimental data, standard addition format.

Figure 2: Calibration and experimental data for three ISEs measuring lead in soil, originally stored in a Microsoft Excel file and then exported to tab-delimited text files.

(optionally) a file with **experimental** data. Returns an object of class 'ISEdata'.

- `describeISE`, which implements a Bayesian model to estimate `a, b, c, cstar, sigma,` and `LOD`. Inputs an object of class 'ISEdata', as well as the valence `Z` and experimental `temperature` (in °C, defaults to 21°). Returns an object of class 'ISEdescription'.

- `analyseISE`, which implements a Bayesian model to estimate model parameters `a, b, c, cstar, sigma, LOD` *and* activities x for experimental samples. Inputs an object of class 'ISEdata' that includes **experimental** data, as well as the valence `Z` and experimental `temperature`. Returns an object of class 'analyseISE'.

- Both `describeISE` and `analyseISE` have a number of optional values, linked to numerical options (Markov chain Monte Carlo (MCMC) options, initial values, convergence diagnostics, saved output) and LOD calculation options. See `?describeISE` and `?analyseISE` for further details.

Each of these functions has generic functions `print, plot, summary` associated with them, whose functionality depends on the class of the inputted object. For example, the `summary` command applied to an object of class 'ISEdata' will actually apply the associated command, `summary.ISEdata`, while `summary` on an object of class 'ISEdescription' will actually apply `summary.ISEdescription`. The user can find more details via the affiliated help files, e.g. `?summary.ISEdata`.

2.3. Bayesian Methods

Once the dataset is loaded and the user is satisfied with data quality, Bayesian analyses may be conducted. Here, we use the functions `describeISE` or `analyseISE`, which implement the Bayesian models presented in Dillingham et al. (2012) to estimate model parameters and experimental activities (when using `analyseISE`), and the conditional-analytic method described in Dillingham et al. (2017) to estimate LOD. Dillingham et al. (2012) provides a brief introduction to Bayesian inference, e.g. how prior distributions for parameters are combined with a statistical model and data to generate posterior distributions for the parameters; the Supporting Information introduces the BUGS code that forms the basis of **ISEtools**.

In the standard implementation, analyses in **ISEtools** are conducted through the OpenBUGS variant (Thomas, O'Hara, Ligges, and Sturtz 2006; Thomas 2006) of the BUGS language (Lunn, Thomas, Best, and Spiegelhalter 2000) from within R via the **BRugs** and **R2WinBUGS** libraries (Thomas et al. 2006; Sturtz, Ligges, and Gelman 2005). The function `describeISE` calls OpenBUGS, feeding data, an (automatically chosen or user-specified) Bayesian model, and (automatically generated or user-specified) initial values into it. The user does not need to interact with OpenBUGS, but *must* have OpenBUGS installed on their computer. OpenBUGS works on

Windows and Unix/Linux,[2] or via an emulator on macOS (e.g. Wine). An alternative implementation using jags (Plummer et al. 2003) (which works across platforms) via the **rjags** library (Plummer 2013) is described on the next page.

OpenBUGS implements Markov chain Monte Carlo (MCMC) to numerically sample the joint posterior distribution of model parameters.[3] The user is given feedback during implementation that the model compiled, initialised, and ran, or lets the user know if there was an error. Where errors occur, our experience is that they are usually due to (1) incompatibility between data and the model, e.g. due to a data entry error, or (2) a problem with the automatically-generated initial values.

A key feature of **ISEtools** is that analyses are simple to implement and do *not* require knowledge of Bayesian methodology. However, those that are familiar with Bayesian methods may exercise control over technical aspects of the analysis if they so wish. The remainder of this Section describes the technical aspects of the analyses. Users may skim Section 2.3.1 or skip to the examples in Section 3 if not interested in technical details of the Bayesian models.

Technical Details

The Bayesian models are in the `/models` sub-directory of the **ISEtools** library, and are selected based on analysis and data types. There are six included files, accommodating data from single ISEs or an array of multiple ISEs, with or without experimental data, and in Basic or Standard Addition format. Key priors for these models are:

- $\texttt{a} \sim N(0, \sigma = 1,000)$, measured in mV;
- $\texttt{b} \sim N(E(\texttt{b}), \sigma = 10)$ mV/decade, where

$$E(\texttt{b}) = 8,314.33 \ln 10(\texttt{temperature} + 273.15)/(96,487\texttt{Z})$$

  is the expected Nernstian slope (Eisenman et al. 1957; Dillingham et al. 2012);
- $\texttt{cstar} \sim U(0.1, 0.5)$ and `c` calculated as $\texttt{c} = \texttt{cstar}^{10}$ (and hence a lower bound for `c` is 10^{-10} and the upper bound is approximately 10^{-3});
- $\texttt{sigma} \sim U(0, 10)$ mV;
- $\log_{10} x \sim U(-12, -2)$, which allows for samples to fall in the flat portion of Figure 1.

[2]The required library **BRugs** uses a 32-bit C compiler that may need to be installed. For example, in Fedora, `sudo dnf install @c-development glibc-devel.i686` or in Ubuntu, `sudo apt install build-essential libc6-dev-i386` may be required.

[3]For each parameter, MCMC provides a sequence of values which, if the sequence is sufficiently long, will provide a good approximation to the posterior distribution for that parameter. Across parameters, corresponding entries in the sequences represent one sample from the joint posterior distribution, allowing easy cross-comparison of two or more parameters, e.g. by plotting the sequences against each other.

In **ISEtools**, the 'cut' function is used, separating the calibration phase from the experimental phase of the study. Specifically, parameter values for `a, b, c, sigma` are estimated using calibration data only. These feed into the experimental phase, so that the priors on experimental samples do not influence the ISE parameter estimates, i.e. `describeISE` and `analyseISE` return the same parameter estimates regardless of the experimental samples. Use of the cut function is intuitively desirable in many settings (e.g. calibration data from a laboratory, experimental data from the field), but the generic implementation in OpenBUGS *may* lead to poor model behaviour due to the numerical updating procedures used (Plummer 2015). However, we have performed extensive simulation tests and determined that its use with these models does *not* lead to any substantive issues, and a model without the cut function *will* perform poorly if the the ratio of experimental to calibration samples is high.

Alternatively, we provide jags as an alternative to OpenBUGS, primarily for macOS users without access to Windows or Linux. First, install the external program jags and the R package **rjags**, and load **rjags** along with **ISEtools** (i.e. `library(ISEtools); library(rjags)`). Then, simply add the option `program="jags"` when calling `describeISE` or `analyseISE`. For `describeISE`, there is no difference between the models. However, jags does not implement the 'cut' function, so will produce different results when using `analyseISE` for both model parameters and experimental samples. With redundant ISEs, a good calibration dataset, and relatively few experimental samples, all of which lie in the Nernstian portion of the response curve, the difference is negligible. However, for a single ISE with a large number of experimental samples, including samples in the flat portion of the response curve, the difference can be substantial. A simple diagnostic is to compare parameter estimates from `describeISE` and `analyseISE`: if they are substantively different, the jags results are not trustworthy. In such cases, splitting the experimental dataset into multiple datasets, each with just a few samples, and running a series of sub-analyses may suffice. Ultimately, we recommend use of a Windows- or Linux-based machine in conjunction with OpenBUGS.

In **ISEtools**, users may also specify their own Bayesian models, so long as they have the same parameters (e.g. keeping the same structure but setting priors specific to their analysis and system). For variations beyond those which **ISEtools** can accommodate, users are encourage to explore OpenBUGS and **R2WinBUGS** (or other Bayesian programs) directly. Initial values are generated via a pre-analysis (via functions `gen.inits.single` or `gen.inits.multiple`, which users do not interact with) but can be specified by the user if required. Default MCMC options are set to ensure good numerical behaviour for most datasets (e.g. the potential scale reduction factor (Brooks and Gelman 1998), $\hat{R} \approx 1$ for most datasets using the defaults of four chains with a 25,000 iteration burn-in and an additional 25,000 samples retained per chain). See `?describeISE` and `?analyseISE` for numerical options, and the example in Section 3.2 for implementation details.

Model Output

Users may choose to only interact with model output via the associated `print, plot, summary` commands, but may also interact directly with the outputs of the functions, as follows:

- `describeISE` and `analyseISE` return estimates for model parameters `a, b, c, cstar, sigma` as `<parameter name>hat`, a vector where each entry corresponds to an ISE. E.g., the first entry of `ahat` corresponds to $\hat{a}$ for ISE #1, the second to $\hat{a}$ for ISE #2, and so on. Similarly, lower and upper limits of the 95% credible intervals are returned as `<parameter name>hat.lcl, <parameter name>hat.ucl`.
- LOD is calculated using the conditional-analytic method described in Dillingham et al. (2017) as a function of `a, b, c, sigma` for each posterior sample. The default is to base LOD on false positive and negative rates `alpha, beta`, which default to 0.05. Alternatively, LOD based on a signal-to-noise ratio is specified by providing a value for `SN`, e.g. `SN = 3`. In addition to point estimates (`LOD.hat`) and 95% credible intervals (`LOD.lcl, LOD.ucl`), the first and third quartiles for LOD are also returned as `LOD.Q1`, `LOD.Q3`. Again, these are returned as vectors where each entry corresponds to a given ISE.
- For `describeISE`, individual iterations for each parameter and LOD are returned if the option `keep.coda = TRUE` as `<parameter name>hat.coda` and `LOD.coda`. The default is to return 1000 random samples from the estimated posterior but can be modified by setting `coda.n` to the desired level. A matrix is returned for each parameter, where rows correspond to posterior samples and columns to their corresponding ISE.
- For `analyseISE`, activities for experimental samples (on the $\log_{10}$ scale) are returned as `log10x.exp`. A matrix is returned where each row corresponds to an experimental sample, the first column contains the point estimates, and the second and third columns contain the lower and upper limits of the 95% credible interval. Similarly, the two columns of `log10x.exp.IQ` contain the first and third quartiles for each experimental sample.

3. Examples

Examples for two different solid-state ISEs are provided to demonstrate use of the **ISEtools** package. In addition, each example is also chosen to highlight some of the interesting patterns that can occur with ISE data, particularly when developing new ISEs that may be noisy and/or operate close to their limits of detection.

The first example starts with data measuring lead in soil, stored in a tab-delimited `txt` file (exported from Excel). The example demonstrates how external data can be brought into R, with or without experimental samples, and with or without standard addition. It then estimates

model parameters, LODs, and experimental activities for three ISEs measuring lead in soil. The second example modifies the standard Bayesian model to analyse an array of eight carbonate ISEs to make a bespoke figure, demonstrating more advanced functionality of **ISEtools**.

3.1. Lead in soil: importing external data for analysis and plotting

McGraw, Radu, Radu, and Diamond (2008) describe the development of liquid- and solid-contact ISEs for the measurement of lead in soil. Soil samples were collected at Silvermines, County Tipperary, Ireland, where centuries of mining resulted in locally high levels of heavy metals, including lead. They analysed the samples using a range of ISEs, and compared estimated activities to atomic absorption spectroscopy (AAS) reference measurements. Here, we present their data for three solid-contact ISEs, as analysed in Dillingham et al. (2012), in order to demonstrate the three standard functions of the **ISEtools** library.

The Data

For the lead ISE data, three example files are included in the '`/extdata`' sub-folder of the **ISEtools** library (e.g. <pathname to R libraries>/ISEtools/extdata), with the files shown in Figure 2. The pathname to **ISEtools** can be found via `path.package('ISEtools')`. Data files are in tab-delimited text format[4] as described in Section 2.1.

- `Lead_calibration.txt` contains calibration data for three ISEs on six datapoints.
- `Lead_experimentalBasic.txt` contains experimental measurements on 17 samples, in the 'Basic' format.
- `Lead_experimentalSA.txt` contains experimental measurements on the same 17 samples, in the 'Standard Addition' format.

The `loadISEdata` function imports tab-delimited text files, processes them, and returns an object of class 'ISEdata'. The returned object includes the original data, as well as additional information (the number of ISEs, calibration points, experimental samples, whether the standard addition format was used) required for analysis using `describeISE` or `analyseISE`. As described in Section 2.3, the additional information is used to determine which Bayesian model is appropriate for the data, and automatically passed onto `describeISE` or `analyseISE`.

[4]User data files should be stored in a location linked to their analysis, *not* in the **ISEtools** library folder.

The `loadISEdata` function basic call is:

```
# Calibration data only
loadISEdata(filename.calibration = "<pathname>/<calibration filename>.txt",
            filename.experimental= "<pathname>/<experimental filename>.txt")
```

where '`<pathname>`' is the user-specified folder (typically) linked to the research project and '`<calibration filename>.txt`', '`<experimental filename>.txt`' are the names of the tab-delimited text files. If the dataset only has calibration data (i.e. there are no experimental samples), the call reduces to:

```
loadISEdata(filename.calibration = "<pathname>/<calibration filename>.txt")
```

- On the machine that created this vignette, the lead calibration data is loaded via:

```
lead.example1 = loadISEdata(filename.calibration =
  "C:/Program Files/R/R-3.3.2/library/ISEtools/extdata/Lead_calibration.txt")
```

Once loaded, the dataset can be printed (first 10 observations shown) and plotted (the first two ISEs are shown in Figure 3), useful for ensuring we have the data we expect before proceeding with further analysis, via:

```
print(lead.example1)
plot(lead.example1)
```

```
##    ISEID    log10x        emf
## 1      1 -9.000116   8.558784
## 2      1 -6.996850   8.941667
## 3      1 -5.962362  13.824266
## 4      1 -4.970696  32.160924
## 5      1 -3.996123  56.683022
## 6      1 -3.076335  85.458353
## 7      2 -9.000116 -28.381216
## 8      2 -6.996850 -25.458333
## 9      2 -5.962362 -20.545734
## 10     2 -4.970696  -1.579076
```

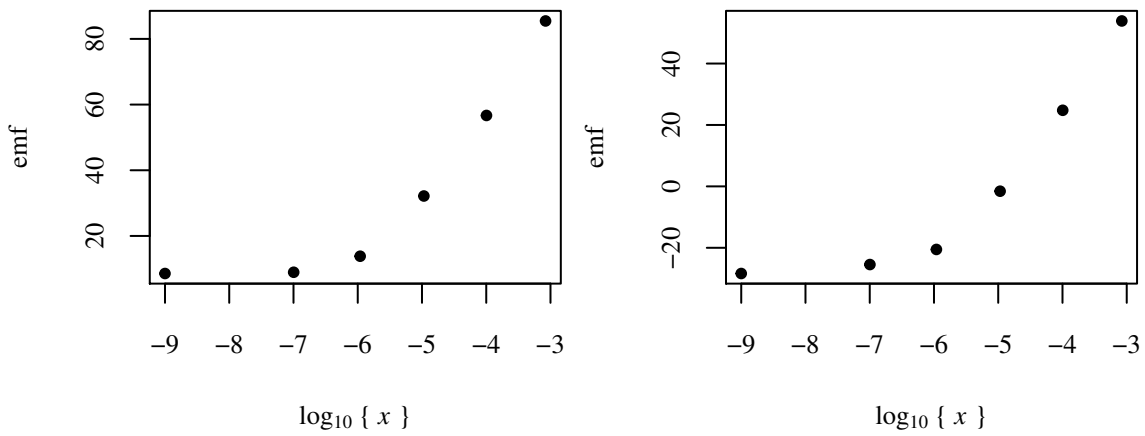

Figure 3: The response of the first two lead ISEs.

We may also see basic summary information (`N` (total number of calibration measurements), `R` (number of ISEs), a logical flag (`calibration.only`) indicating this dataset only has calibration data and no experimental samples, and missing values (`NA`) for two variables linked to experimental data, `M` (number of experimental samples) and `stdadd` (a logical T/F flag for standard addition data) using:

```
summary(lead.example1)
```

which returns

```
## $N
## [1] 18
##
## $R
## [1] 3
##
## $calibration.only
## [1] TRUE
##
## $M
## [1] NA
##
## $stdadd
## [1] NA
##
## attr(,"class")
## [1] "summary.ISEdata"
```

- To include the experimental data in Basic or Standard Addition formats (with this particular pathname), the commands are:

```
# ... and with experimental data, Basic format
lead.example2 = loadISEdata(filename.calibration =
  "C:/Program Files/R/R-3.3.2/library/ISEtools/extdata/Lead_calibration.txt",
                  filename.experimental =
  "C:/Program Files/R/R-3.3.2/library/ISEtools/extdata/Lead_experimentalBasic.txt")


# ... and with experimental data, Standard Addition format
lead.example3 = loadISEdata(filename.calibration =
  "C:/Program Files/R/R-3.3.2/library/ISEtools/extdata/Lead_calibration.txt",
                  filename.experimental =
  "C:/Program Files/R/R-3.3.2/library/ISEtools/extdata/Lead_experimentalSA.txt")
```

We note that the dataset imported and stored as `lead.example3` is also available pre-loaded in **ISEtools** as `LeadStdAdd`, accessible via

```
data(LeadStdAdd)
```

Characterising the ISEs using describeISE

The basic invocation of the `describeISE` function is straightforward, requiring:

- a dataset of class 'ISEdata', e.g. `data = lead.example1`,
- the valence `Z` of the ion, e.g. `Z = 2` for lead (Pb^{2+}), and
- the `temperature` in °C, if much different from the assumed room temperature of 21°C;
- valence and temperature form a prior distribution for `b`, per Section 2.3.1.

For the `lead.example1` data,[5] the model is run via `describeISE`[6] and saved as `example1`:

```
example1 = describeISE(lead.example1, Z=2, temperature=21)
```

[5] Or any of the other lead datasets (`lead.example2`, `lead.example3`, or `LeadStdAdd`), as they contain identical calibration data.

[6] The `loadISEdata` function recognised that the sourced data (`Lead_calibration.txt`) had calibration data for multiple ISEs, but no experimental data, and added that information to `lead.example1`. When `describeISE` was called, the appropriate Bayesian model, `Multiple_ISE_calibration_model.txt` was then automatically applied.

Once the Bayesian model has run, we can use `print, summary, plot` on the saved R object `example1`. The `print` command provides a point estimate (equal to the median of the posterior distribution) and 95% credible interval for each ISE, along with the Nikolskii-Eisenman equation (output for ISE #1 is shown below); `summary` provides similar output in abbreviated form (output not shown); `plot` displays histograms of model parameters sampled from the posterior distributions (ISE #1 shown in Figure 4).

```
print(example1)
summary(example1)
plot(example1)
```

```
##
##
## Non-linear parameter estimates and 95% CIs for
##   y = a + b log(x + c)
##
## ISE #1:
##       Parameter estimate Lower limit Upper limit
## a     1.75e+02           1.61e+02    1.91e+02
## b     2.94e+01           2.59e+01    3.36e+01
## c     2.15e-06           1.06e-06    4.30e-06
## sigma 1.38e+00           6.09e-01    5.67e+00
##
## Estimated log LOD{alpha=0.05, beta=0.05} (95% CI): -6.05 (-6.45, -5.09)
```

That is, for ISE #1, we estimate $\hat{y} = 175.3 + 29.4 \log_{10}(x + 2.15e-06)$. Figure 4 highlights the asymmetric posterior distributions, common for ISE parameters. Users may also wish to create their own plots with the automatically stored MCMC output – see Section 3.2 for an example.

The user may want to focus on a few of these parameters, e.g.

(1) Is the estimated slope $\hat{b}$ (and credible interval) consistent with the theoretical Nernstian slope,

$$1000(2.303RT)(ZF)^{-1} = 29.2\text{mV/decade},$$

(where $Z = 2$ is the valence of lead, T is temperature (°K), R is the universal gas constant, and F is Faraday's constant)?

(2) How does the limit of detection, $\log_{10} \widehat{LOD}_{\alpha=0.05,\beta=0.05} = -6.05$ (95% CI: $-6.45, -5.09$) compare to other ISEs from this or other batches?

(3) Is this ISE likely to be able to adequately measure lead activity in soil down to a specified value of interest, say $[\text{Pb}^{2+}] = 10^{-5.5}$?

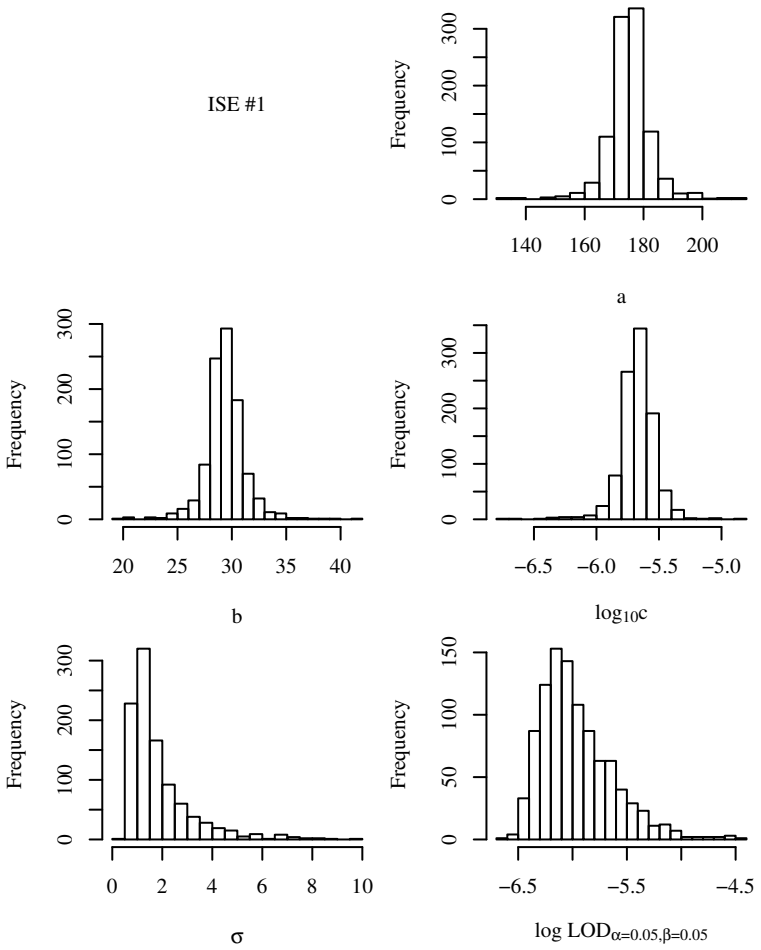

Figure 4: Posterior distribution of ISE parameter values for the first lead ISE.

In this case, the ISE appears to have (1) close to a Nernstian response; (2) have a similar LOD to ISE #2 but likely worse than ISE #3; and (3) the LOD is *at* or *near* the required level, indicating that it may perform adequately by itself but employing a sensor array of similar ISEs would be beneficial.

Analysing experimental samples using analyseISE

These ISEs were specifically developed to allow field-based measurement of lead activities in soil at Silvermines, Ireland. As part of the proof-of-concept development, ISEs were used to measure activity of lead in 17 soil samples. For reference, lead concentration was measured using AAS. Here, activity is estimated using two measurement approaches:

- Basic format: each ISE measures emf for each sample.
- Standard Addition format: two emf readings are made, comprising the original reading from the Basic format and a second reading after adding an aliquot to each sample.
- The Standard Addition format is designed to minimise any effect of drift in `a`.

The activity of lead in the 17 soil samples is estimated using the **Basic** format via:

```
example2 = analyseISE(lead.example2, Z=2, temperature=21)
plot(example2)
```

There are two aspects of this analysis that are not wholly satisfactory. First, the noted problem with drift. These particular ISEs, like many ISEs in development, exhibited temporal drift in `a`; other parameters were stable (McGraw et al. 2008). Second, the presentation in Figure 5 could be improved with a better y-axis label and narrower limits.

To combat drift, the **standard addition** method was employed: an aliquot of known volume and concentration is added to the sample, with the emf recorded before and after the addition, leading to estimators for y that do not depend on `a`. The Bayesian analysis model uses the difference in emf to estimate the unknown lead activity in each soil sample (see Equation 7 in Dillingham et al. (2012) for details).

This leads to a revised call to `analyseISE`, using `lead.example3` as the dataset and adding options to improve the plot. Non-default values were set for `ylim`, which controls the lower and upper values on the y-axis; `ylab`, which controls the label for the y-axis; and `col`, which controls the colour of the plotted symbols (see `?plot.analyseISE` for more details). Additionally, the AAS reference measurements were added onto the plot.[7]

[7] A tab-delimited text file, `aas.txt`, with these data is included in the `/extdata` sub-folder of the **ISEtools** library, and was first loaded using `aas = read.table("<ISEtools library pathname>/extdata/aas.txt", header=T)`.

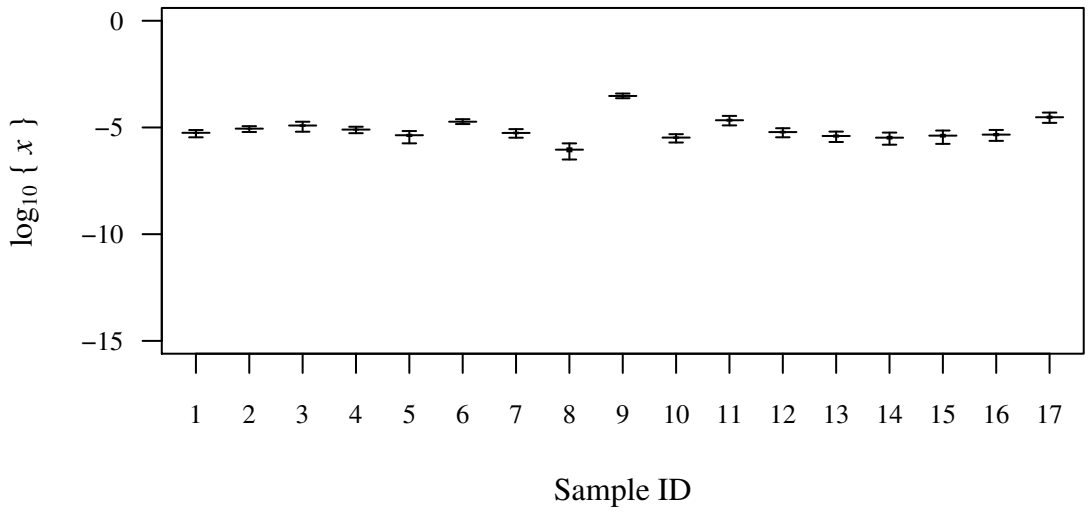

Figure 5: Posterior distribution of lead activity in 17 experimental samples, recorded in the 'Basic' format. The 95% (thin lines) and 50% (thick lines) credible intervals are shown, along with point estimates (–).

```
# Revised analysis using Standard Addition; improvements to the plot
example3 = analyseISE(lead.example3, Z=2, temperature=21)
plot(example3,
     ylab=expression(paste("log ", italic(a)[Pb^{paste("2","+")}])),
     ylim=c(-7, -3), col="steelblue")
points(aas$Sample, log10(aas$AAS), col=colours()[214], cex=0.8, pch=16)
```

Lead Analysis: Results and Discussion

There was generally good agreement between estimates from the ISE sensor array and the reference AAS measurements when using standard addition (Figure 6). AAS has high measurement precision and accuracy, with errors contained within the plotting symbol in Figure 6. ISEs, however, are field deployable and low cost, and therefore useful for a broad range of environmental analyses (Radu, Radu, McGraw, Dillingham, Anastasova-Ivanova, and Diamond 2013). Differences between ISE and AAS measurements overestimate bias,[8] but the mean difference was within an acceptable level regardless. Similarly, while estimates from the ISE array were noisy relative to AAS, the noise was also within acceptable levels. Therefore, the general agreement with AAS and the acceptable accuracy suggest that these ISEs could be reasonably employed for their intended purpose, particularly when used in a sensor array (McGraw et al. 2008; Dillingham et al. 2012).

[8] ISEs measures activity, while AAS measures concentration, which here are close but not identical.

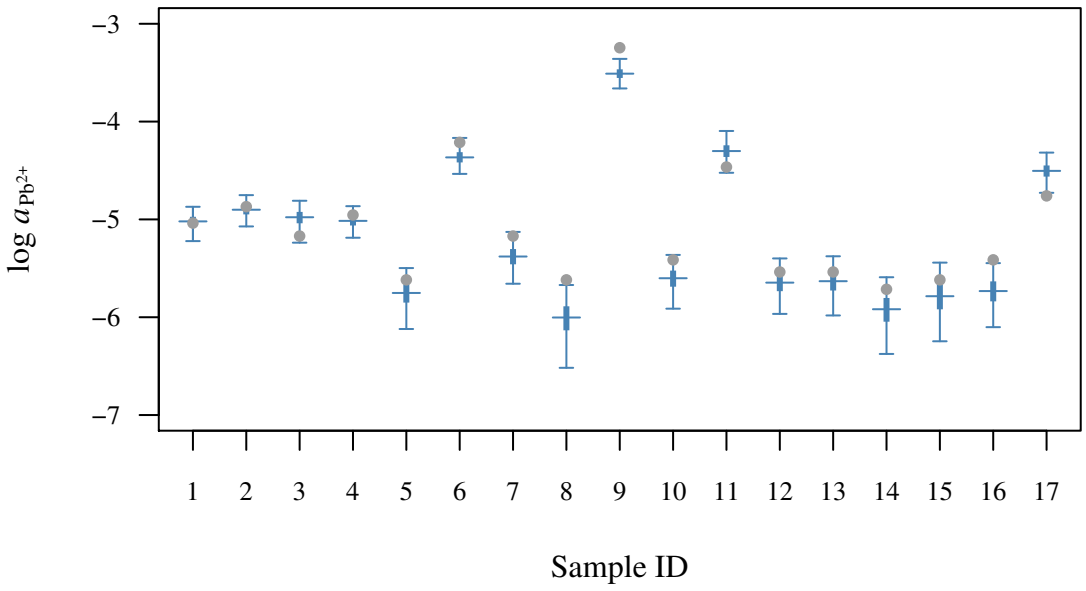

Figure 6: Posterior distribution of lead activity in 17 experimental samples, recorded in the 'Standard Addition' format. The 95% (thin lines) and 50% (thick lines) credible intervals are shown, along with point estimates (–) and AAS reference measurements (•).

3.2. Carbonate in seawater: estimating model parameters and LOD

Mendecki, Fayose, Stockmal, Wei, Granados-Focil, McGraw, and Radu (2015) described a conditioning method to develop ultrasensitive and robust polymer membrane-based ISEs, applied to carbonate in seawater. The **ISEtools** package includes carbonate data from a prototype ISE developed during that project, with data originally presented in Dillingham et al. (2017). The challenge developing these ISEs was due to the high levels of interfering ions in seawater relative to the trace levels of carbonate. However, this example is primarily about customising the analysis beyond the default values in `describeISE`.

The dataset

The `carbonate` dataset contains **calibration** data in artificial seawater, i.e. where the carbonate activity is known and the focus is on estimating ISE model parameters and limits of detection for each ISE, and is accessed via `data(carbonate)`. In this example, there is no corresponding experimental dataset, which is quite typical during the development of new ISEs. The dataset contains $N = 48$ calibration measurements, from $R = 8$ individual ISEs, measuring emf (mV) in artificial seawater at 6 carbonate levels. It includes the required variables `ISEID`, `log10x`, and `emf`, as well as an additional variable, `batch`, which indicates the manufacturing batch for these ISEs (all of the included data came from batch 2). The first 10 measurements are shown below:

```
##    ISEID log10x   emf batch
## 1      1  -8.52  35.1     2
## 2      1  -7.31  34.1     2
## 3      1  -6.16  32.0     2
## 4      1  -5.87  30.5     2
## 5      1  -4.54  21.8     2
## 6      1  -3.35   6.8     2
## 7      2  -8.52 265.2     2
## 8      2  -7.31 258.0     2
## 9      2  -6.16 249.1     2
## 10     2  -5.87 243.5     2
```

The model

In this section, we implement a customised Bayesian model for demonstration. This model has been saved in the `/models` sub-folder of **ISEtools** as `Carbonate_ISE_calibration_example.txt`. In it, we make small alterations to the default model as follows:

- These prototype ISEs were quite noisy, so the prior for `sigma` was expanded to $U(0, 15)$ mV, compared to the default $U(0, 10)$ mV.
- As prototypes, it was assumed that the ISEs might exhibit no response, sub-Nernstian response, or Nernstian response, but not greater than Nernstian response, so the prior for `b` was set to $U(E(\texttt{b}), 0)$ mV, where $E(\texttt{b}) = -29.2$ mV/decade for CO_3^{2-} at 21 °C.

A flow-on consequence is the need to specify initial values for `b`. The initial value generator for `b` is based on the default model, and allows values $< E(\texttt{b})$. However, these are incompatibile with the customised model, and values within the range $(-29.2, 0)$ mV/decade must be provided. We also specify MCMC options and the level of output to store for further analysis. This leads to the following call to `describeISE`:

```
example4 = describeISE(carbonate, Z=-2, temperature=21,
   model.path="C:/Program Files/R/R-3.3.2/library/ISEtools/models",
   model.name="Carbonate_ISE_calibration_example.txt",
   burnin=5000, iters=10000, chains=4, thin=10,
   b.init=runif(8, -25, -5),
   keep.coda=TRUE, coda.n=1000)
```

Specifically,

- The location of the customised file must be specified:
  - `model.path` contains the location of the customised model;
  - `model.name` contains the filename of the customised model.
- MCMC options are specified rather than relying on defaults:
  - `burnin` and `iters` are the number of iterations to discard and the total number of iterations per MCMC chain;
  - `chains` is the number of parallel MCMC chains to run;
  - `thin` is the thinning rate, or 1 ÷ the proportion of simulations retained (e.g. `thin = 10` retains every tenth iteration);
  - the total number of iterations calculated equals `iters` × `chains`;
  - the number of iterations stored in memory and used to estimate parameters and standard errors is (`iters - burnin`) × `chains` ÷ `thin`;
  - this control allows the user to balance robustness (via the burn-in length and diagnostics that rely on multiple MCMC chains), calculation time (via the total number of iterations), and memory usage (via thinning, as neighbouring iterations may be highly correlated with each other and add little to inference).

- Initial values for `b` are drawn from a $U(-25, -5)$ for each of the eight ISEs via `b.init = runif(8, -25, -5)`.
- `keep.coda = TRUE` specifies that the user wishes to retain MCMC output for further analysis, while `coda.n` specifies the number of samples the user wishes to retain (drawn randomly with replacement from the internally stored sample, or all of the stored samples if `coda.n` is greater than or equal to the number stored).
  - This feature is only implemented for `describeISE`.
  - The output will be used to create a plot examining relationships between parameters.

Results

Basic results can be shown using `print` and `plot` as before, with results shown for ISE #1 below and in Figure 7. In this case, the ISE appears to have (1) a sub-Nernstian response; (2) have a higher LOD than the other ISEs in the same batch; and (3) most importantly, the LOD is greater than carbonate levels expected in seawater. This leads to the conclusion that ISE #1 was not fit for the purpose of measuring carbonate in seawater. Other ISEs in the batch, especially where used in a sensor array, were able to successfully measure carbonate in seawater (Dillingham et al. 2017) but not as well as the sensors decribed in Mendecki et al. (2015).

```
##
##
## Non-linear parameter estimates and 95% CIs for
##   y = a + b log(x + c)
##
## ISE #1:
##       Parameter estimate Lower limit Upper limit
## a     -3.16e+01          -5.70e+01   -8.43e+00
## b     -1.17e+01          -1.85e+01   -6.53e+00
## c      2.36e-06           1.27e-07    1.66e-05
## sigma  1.67e+00           7.49e-01    7.23e+00
##
## Estimated log LOD{alpha=0.05, beta=0.05} (95% CI): -5.37 (-6.01, -3.24)
```

By using the stored output in `example4`, we are also able perform additional analyses. First, output is stored for each parameter and each ISE in `<parameter>hat.coda`. That is, the MCMC output for `b`, is stored in `example4$bhat.coda`, with the first nine iterations shown below (rounded to 2 decimal places). The output is a matrix with 1,000 rows (because `coda.n = 1000`) and 8 columns, where each row represents one MCMC iteration and each column represents one of the ISEs, e.g. column #1 represents ISE #1.

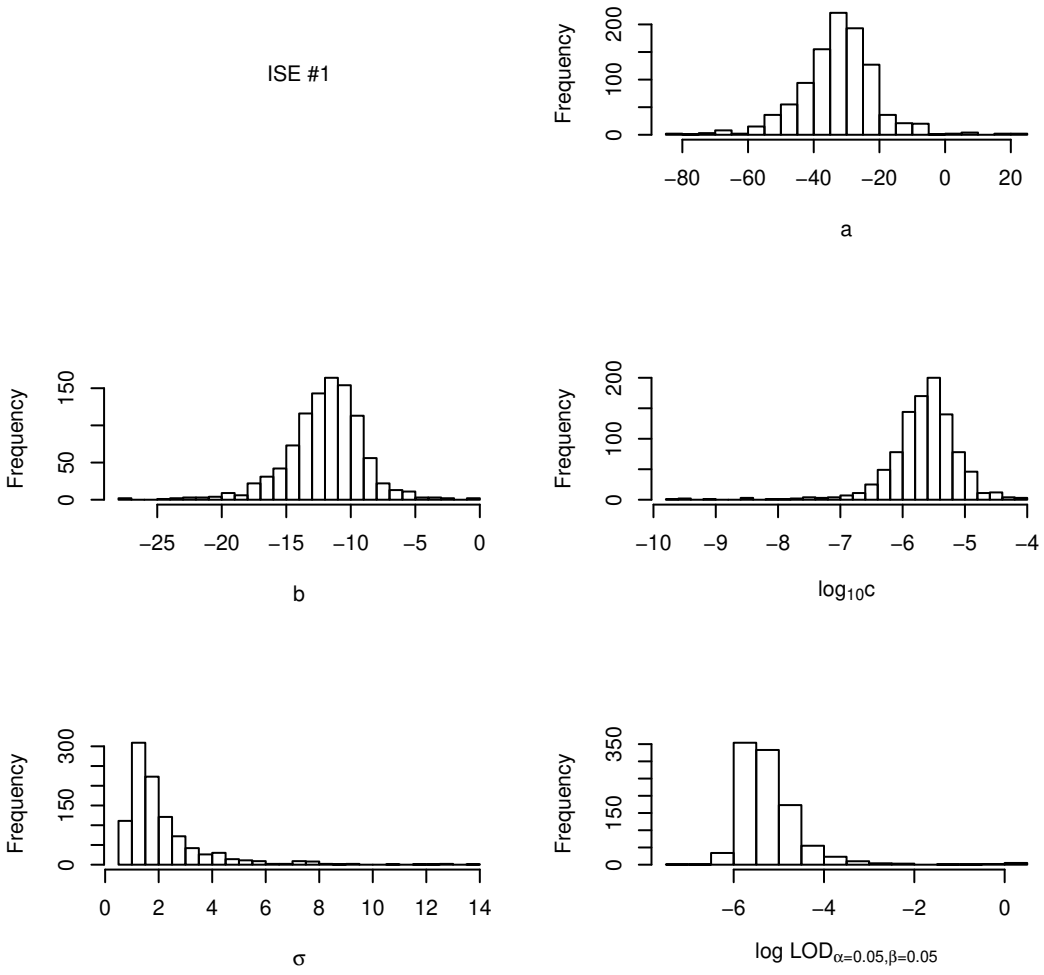

Figure 7: Sampling distribution of ISE parameters for ISE #1.

```
head(round(example4$bhat.coda, 2), n = 9)
```

```
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]
##  [1,] -11.72  -9.42 -15.84 -27.29 -17.57 -17.83 -28.38 -25.83
##  [2,] -13.97 -11.11 -15.44 -27.88 -16.86 -26.22 -28.30 -18.72
##  [3,]  -6.52  -9.64 -16.33 -28.20 -18.30 -18.73 -27.27 -16.72
##  [4,] -13.17  -8.70 -17.42 -27.04 -18.57 -20.19 -20.32 -19.14
##  [5,] -11.46 -10.76 -14.87 -29.14 -18.76 -22.44 -27.79 -14.67
##  [6,] -16.83  -9.43 -16.83 -26.27 -15.56 -20.25 -25.70 -22.15
##  [7,] -10.61 -10.92 -16.34 -23.80 -17.74 -19.95 -29.08 -20.07
##  [8,] -11.49  -9.70 -18.25 -23.52 -18.33 -17.14 -28.51 -19.65
##  [9,]  -9.57 -11.29 -17.82 -27.86 -15.44 -20.97 -28.78 -24.87
```

In Dillingham et al. (2017), the pattern of model parameters and their link to LOD was examined. Here, we create a simpler plot showing correlations between parameters for ISE #1 in Figure 8. Unsurprisingly, the intercept `a` and slope `b` are highly correlated, while both are also correlated with $\log_{10}$ `c`.

```
# Set up a plot with 9 subplots in 3 rows and columns; set margins widths
# Examine ISE #1
par(mfrow=c(3, 3), mar=c(0, 4.5, 1.5, 0))
ISE = 1

# First row of plots: a (on y-axis) vs b, log c, sigma (on x-axis)
plot(example4$bhat.coda[,ISE], example4$ahat.coda[,ISE],
     xlim=c(-30, 0), ylim=c(-100, 40), main="b",
     cex.main=1, font.main=1, axes=F, ylab="a", xlab="", pch=20, cex=0.3)
plot(log10(example4$chat.coda[,ISE]), example4$ahat.coda[,ISE],
     xlim=c(-10, -3), ylim=c(-100, 40),
     main=expression(paste(log[10],"c", sep="")), cex.main=1, font.main=1,
     axes=F, ylab="", xlab="", pch=20, cex=0.3)
plot(example4$sigmahat.coda[,ISE], example4$ahat.coda[,ISE],
     xlim=c(0, 15), ylim=c(-100, 40), main="sigma", cex.main=1, font.main=1,
         axes=F, ylab="", xlab="", pch=20, cex=0.3)

# Second row of plots: b vs blank, log c, sigma
plot(NA, xlab="", xlim=c(-30, 0), ylab="b", ylim=c(-30, 0), axes=F)
plot(log10(example4$chat.coda[,ISE]), example4$bhat.coda[,ISE],
     xlim=c(-10, -3), ylim=c(-30, 0),xlab="", ylab="",axes=F,pch=20,cex=0.3)
plot(example4$sigmahat.coda[,ISE], example4$bhat.coda[,ISE],
     xlim=c(0, 15), ylim=c(-30, 0), xlab="", ylab="", axes=F,pch=20,cex=0.3)

# Third row of plots: log c vs blank, blank, sigma
plot(NA, xlab="", xlim=c(-10, -3), ylab=expression(paste(log[10],"c", sep=" ")),
        ylim=c(-10, -3), axes=F)
plot(NA, xlab="", xlim=c(-30, 0), ylab="", ylim=c(-10, -3), axes=F)
plot(example4$sigmahat.coda[,ISE], log10(example4$chat.coda[,ISE]),
    xlim=c(0, 15), ylim=c(-10, -3), xlab="", ylab="", axes=F,pch=20,cex=0.3)
```

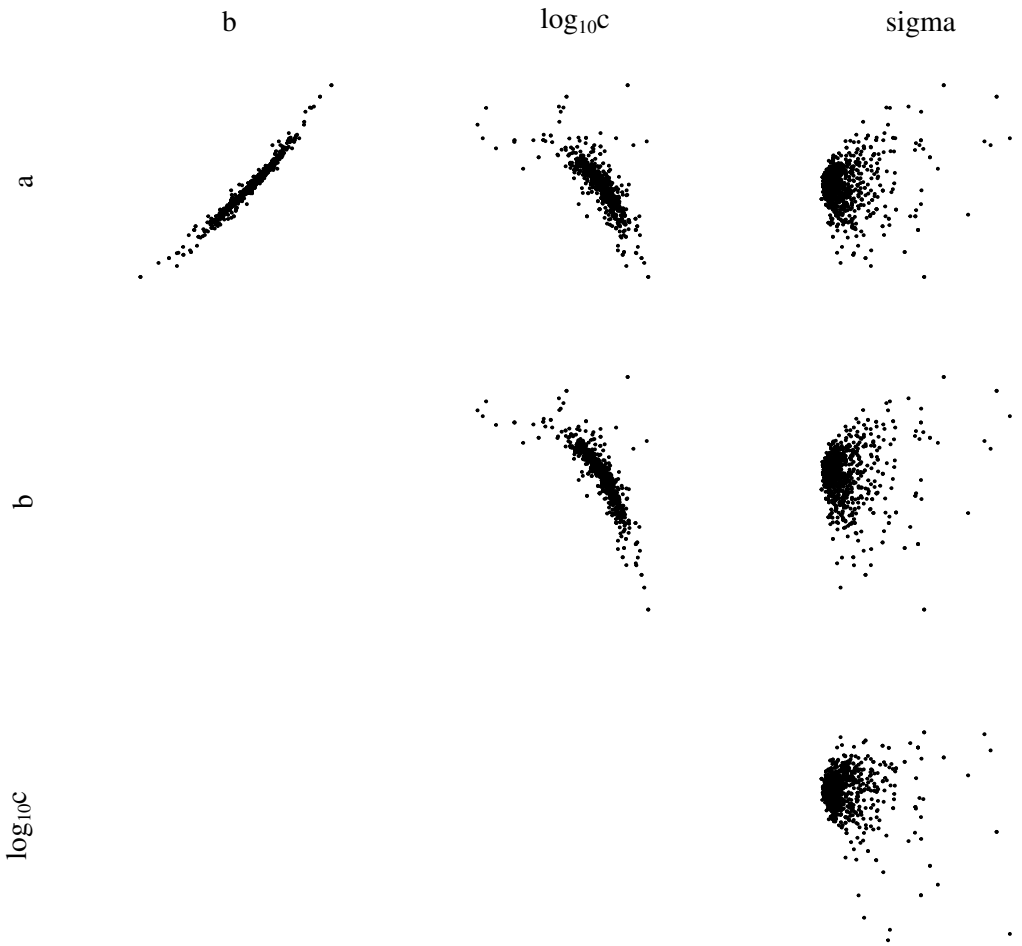

Figure 8: Correlations between ISE parameters for ISE #1.

4. Conclusion

ISEtools is designed to be as easy-to-use as possible, while also encouraging and allowing users to customise models and analyses. Three key functions, `loadISEdata`, `describeISE`, and `analyseISE` are introduced allowing researchers to analyse ISE data without requiring in-depth knowledge of Bayesian procedures or the R programming language. Summary output and standard plots are easily implemented via `print`, `summary`, `plot` commands, along with additional ability to customise the standard plots. Users may also implement their own Bayesian models and store MCMC output from their own model or the standard models. This allows customised analyses and plots, encouraging users to delve deeper according to ability and interest.

References

Brooks SP, Gelman A (1998). "General methods for monitoring convergence of iterative simulations." Journal of computational and graphical statistics, **7**(4), 434–455.

Desimoni E, Brunetti B (2012). "Glassy Carbon Electrodes Film-Modified with Acidic Functionalities. A Review." Electroanalysis, **24**(7), 1481–1500.

Dillingham PW, Alsaedi BS, McGraw CM (2017). "Characterising uncertainty in instrumental limits of detection when sensor response is non-linear." In SENSORS, 2017 IEEE, pp. 1–3. IEEE.

Dillingham PW, Radu T, Diamond D, Radu A, McGraw CM (2012). "Bayesian Methods for Ion Selective Electrodes." Electroanalysis, **24**(2), 316–324.

Eisenman G, Rudin DO, Casby JU (1957). "Glass electrode for measuring sodium ion." Science, **126**(3278), 831–834.

Eksperiandova L, Belikov K, Khimchenko S, Blank T (2010). "Once again about determination and detection limits." Journal of Analytical Chemistry, **65**(3), 223–228.

Lunn DJ, Thomas A, Best N, Spiegelhalter D (2000). "WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility." Statistics and computing, **10**(4), 325–337.

McGraw CM, Radu T, Radu A, Diamond D (2008). "Evaluation of Liquid-and Solid-Contact, Pb^{2+}-Selective Polymer-Membrane Electrodes for Soil Analysis." Electroanalysis, **20**(3), 340–346.

Mendecki L, Fayose T, Stockmal KA, Wei J, Granados-Focil S, McGraw CM, Radu A (2015). "Robust and ultrasensitive polymer membrane-based carbonate-selective electrodes." Analytical chemistry, **87**(15), 7515–7518.

Montville D, Voigtman E (2003). "Statistical properties of limit of detection test statistics." Talanta, **59**(3), 461–476.

Plummer M (2013). "rjags: Bayesian graphical models using MCMC." R package version, **3**(10).

Plummer M (2015). "Cuts in Bayesian graphical models." Statistics and Computing, **25**(1), 37–43.

Plummer M, et al. (2003). "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling." In Proceedings of the 3rd international workshop on distributed statistical computing, volume 124. Vienna, Austria.

Radu A, Radu T, McGraw CM, Dillingham PW, Anastasova-Ivanova S, Diamond D (2013). "Ion-selective electrodes in environmental analysis."

Sturtz S, Ligges U, Gelman AE (2005). "R2WinBUGS: a package for running WinBUGS from R."

Thomas A (2006). "The BUGS language." R News, **6**(1), 17–21.

Thomas A, O'Hara B, Ligges U, Sturtz S (2006). "Making BUGS Open. R News 6: 12–17."

Affiliation:

Peter W. Dillingham
Department of Mathematics and Statistics
University of Otago, New Zealand
E-mail: peter.dillingham@otago.ac.nz
URL: http://www.maths.otago.ac.nz/~dillingh/