WORKSHEET

ABC WORKSHOP, 28 JANUARY 2016, BOURNEMOUTH

This practical aims to provide hands-on experience of Approximate Bayesian Computation (ABC). The aim is to calibrate and evaluate an individual-based model (IBM); Johnston *et al.*'s [1] earthworm IBM will serve as an example. To complete Steps 1 and 2 of this practical, it's necessary to have:

- *R*, statistical software that is freely available for all operating systems; it can be downloaded from http://www.r-project.org
- the course files provided in 'ABCWorkshop.zip'; it can be downloaded from http://www.personal.reading.ac.uk/~yp905172/ABCWorkshop.zip

For Step 3, additionally required are:

- *NetLogo*, a programming platform for building IBMs, also freely available for all operating systems from http://ccl.northwestern.edu/netlogo/ (any version from 5 up to 5.2)
- the *R* package *RNetLogo* [2] which depends on *Java* – this can be a little tricky to install, so detailed instructions are provided

Step 1: Finding 95% Credible Intervals

In this step, the objective is to find the posterior parameter distributions of Johnston *et al.*'s earthworm IBM fit to two experiments [3, 4]. The simulations were generated by running the earthworm IBM 100,000 times for each experiment, with parameters drawn from uniform prior distributions ranging from half to twice the values originally chosen on the basis of the literature.

1. First, let's look at the data and simulations available. Change *R's* working directory to the ABCWorkshop folder on your machine. You may be able to do this from *R's* dropdown menu's, or you can type the `setwd` command into the *R* console, specifying the correct path; see below for examples.

```
> setwd("/Users/Elske/Desktop/ABCWorkshop") ## for a Mac
```

```
> setwd("C:/Users/Elske/Desktop/ABCWorkshop") ## for a PC
```

2. Then, to read in the empirical data, type:

```
> all.data <- read.table("Inputs/all_data.txt", header = T)
```

This stores the contents of the file `all_data.txt` in the object `all.data`. You can look at the data by typing `all.data` in the console; you'll see that it consists of 27 columns of mass data from "E1" (Experiment 1) and 19 columns of cocoon data from "E2" (Experiment 2).

3. Then, read in the files `full_priors.txt` and `full_results.txt`:

```
> full.priors <- read.table("Results/full_priors.txt",
  header = TRUE, stringsAsFactors = FALSE)
```

```
> full.results <- read.table("Results/full_results.txt",
  header = TRUE, stringsAsFactors = FALSE)
```

Each row of `full.priors` contains the parameters that were used to generate the simulation results of the corresponding row in `full.results`. To look at them, type:

```
> head(full.priors)

> head(full.results)
```

This shows the first six rows of each object; `full.priors` contains the randomly drawn values for the fourteen parameters discussed in the talk, and `full.results` has the same columns as `all.data`.

4. Now, let's look at the relevant *R* code. From *R*'s dropdown menu 'File', select 'Open Document' and find 'ParameterEstimation.R' in the folder 'Code' and then 'R'. There are four functions; all with the same structure:

```
name.of.function <- function(argument.1, argument.2, …) {
  what the function does
}
```

The first function, `create.abcEst`, actually does ABC parameter estimation, and the rest offer various ways of showing the results.

5. To load the contents of 'ParameterEstimation.R' into *R*, use `source`:

```
> source("Code/R/ParameterEstimation.R")
```

6. Then, to actually do parameter estimation using ABC, type:

```
> full.abcEst <- create.abcEst(target = all.data, priors =
  full.priors, results = full.results, rate = 0.001)
```

This calls the `create.abcEst` function and stores the result in the `full.abcEst` object.

7. To see the results of the ABC analysis, type:

```
> summary(full.abcEst)

> plot(full.abcEst)
```

These functions offer an overview of the model's approximate posterior parameter distributions given the data. In the plot, values are scaled by dividing by the mean of the prior distribution. The grey, dotted lines in back are the 95% credible intervals of the priors, while the black lines in the front are the 95% credible intervals of the posteriors.

Step 2: Model Selection

In Step 1, ABC did not result in the narrowing of all model parameters. Essentially, for some parameters, it seems not to matter what value they take – the model always fits equally well This could mean that the model mechanisms that are affected by these parameters are superfluous when it comes to fitting this empirical data. Alternatively, it could mean that the priors on these parameters are already sufficiently narrow for our purposes.

To investigate these issues, we have built a simple version of the model, where the earthworms do not move around or forage, eliminating the *s*, *h* and E_f parameters, and do not explicitly allocate energy to different processes. Instead, they reproduce and grow maximally, provided there is any food available at all.

We have run this simple model 100,000 times as well, again drawing parameter values from uniform priors ranging from half to twice the original literature values. We now use ABC model selection to compare the two models.

1. First, read in the files `simple_priors.txt` and `simple_results.txt`:

   ```
   > simple.priors  <-  read.table("Results/simple_priors.txt",
     header = TRUE, stringsAsFactors = FALSE)
   ```

   ```
   > simple.results <- read.table("Results/simple_results.txt",
     header = TRUE, stringsAsFactors = FALSE)
   ```

2. Then, open the file 'ModelSelection.R' in the folder 'Code' and then 'R' and `source` it into *R*. Again, the first function actually does ABC model selection, while the other two offer various ways of showing the results.

   ```
   > source("Code/R/ModelSelection.R")
   ```

3. The function `calc.abcSel` expects as its arguments the `target` to be fit, a list of `indexes` specifying which rows of `results` belong to which model, the `results` of all models put together, and the acceptance `rate` desired.

   As the first columns of our `priors` objects contain a model index (either `Full` or `Simple`) and the *R* command `rbind` puts objects together by row, we can give the command to do ABC model selection like this:

   ```
   > all.indexes <- c(full.priors[,1], simple.priors[,1])
   ```

   ```
   > all.results <- rbind(full.results, simple.results)
   ```

   ```
   > both.abcSel <- create.abcSel(target = all.data, indexes =
     all.indexes, results = all.results, rate = 0.001)
   ```

4. To illustrate the effects of the empirical data used, it's also worth doing ABC model selection with data from Experiment 1 or Experiment 2 only.

   First, let's define which columns relate to which experiment, like so:

   ```
   > col.e1 <- c(1:27)
   ```

   ```
   > col.e2 <- c(28:46)
   ```

   Now, let's run the model selection function for each experiment separately:

   ```
   > e1.abcSel <- create.abcSel(target = all.data[ , col.e1],
     indexes = all.indexes, results = all.results[ , col.e1],
     rate = 0.001)
   ```

   ```
   > e2.abcSel <- create.abcSel(target = all.data[ , col.e2],
     indexes = all.indexes, results = all.results[ , col.e2],
     rate = 0.001)
   ```

5. Then, let's look at the results. The easiest way is with the `summary` function:

   ```
   > summary(both.abcSel)
   ```

   ```
   > summary(e1.abcSel)
   ```

   ```
   > summary(e2.abcSel)
   ```

   This shows tables of Bayes factors, expressing how often the model in each row was accepted relative to the model in each column. The ratio of acceptances expresses how much likelier the model in each row is than the model in each column, given the data. Can you explain the outcome?

Step 3: Posterior Predictive Checking

As the last step of this practical, we will test how well our accepted models and parameter values cause the original earthworm IBM to fit the data. To do this 'posterior predictive checking' requires the *NetLogo* programming platform, as well as, ideally, the *R* package *RNetLogo* [2].

1. As a first type of 'posterior predictive checking', let's directly input the 'best values' accepted by our ABC parameter estimation into the corresponding *NetLogo* model. To open the NetLogo model for Experiment 1, open *NetLogo*, and then use the dropdown menu 'File' to 'Open' the supplied file 'Exp1_Full.nlogo' in the folder 'Code' and then 'NetLogo'.

2. To make sure the model is set to its default values, first press the buttons 'Set Basic Parameters' and 'Set Basic Interface'. After that, press the buttons 'Setup' and then 'Go' to watch the model run, using its original literature values. You can see how well the model fits by looking at the right graph, which plots the model's output against the empirical data.

3. Now, let's have the model run using ABC's values. Our previously created `full.abcEst` object also stores the parameter values of the run that minimised the distance to the empirical data. Type the following to see them:

   ```
   > full.abcEst$best.values
   ```

4. You can input these values into the *NetLogo* model by typing them into the green parameter boxes below the graphs and purple buttons; the boxes are in the same order as in `full.abcEst$best.values`. Click 'Setup' and 'Go' again to watch the model run. The fit is pretty good, right?

5. Ideally, we'd like to do this 'posterior checking' directly from *R* and plot some graphs. As a first approximation, let's re-plot the best accepted run, along with some random draws from all those accepted. Open the file 'PosteriorPlotting.R' in the folder 'Code' and then 'R' and `source` it into *R*.

   ```
   > source("Code/R/PosteriorPlotting.R")
   ```

   Then type:

   ```
   > plot.postCheck(full.abcEst, draws = 100, rerun = FALSE)
   ```

   This plots the results of the simulation runs that best fit the empirical data according to ABC. What do you think of the fit?

6. To do a true 'posterior predictive check', we need to actually re-run the earthworm IBMs. This requires the *R* package *RNetLogo* and 'RunningNetLogo.R'. Let's start with *RNetLogo*.

   Under Windows, installing *RNetLogo* should be as easy as typing:

   ```
   > install.packages("RNetLogo")
   ```

   Under MacOS X, the latest version of *RNetLogo* probably won't work – instead, manually install the previous version:

   ```
   > install.packages("http://cran.r-
     project.org/src/contrib/Archive/RNetLogo/RNetLogo_1.0-
     0.tar.gz", repos = NULL, type = "source")
   ```

   Then check your installation by typing:

   ```
   > library(RNetLogo)
   ```

   If that doesn't throw any errors, great. If it complains about missing dependencies, use the `install.packages()` command to install them. If it complains that 'JAVA_HOME' cannot be determined from the registry, you probably need to (re-)install Java – either 64 or 32 bit, whatever your *R* version is. Get it from http://www.java.com, where the standard is 32 bit. These or any other problems, let me know and I'll see what I can do.

   Then open 'RunningNetLogo.R', which is in the folder 'Code' and then 'R'. You'll see that this script starts by telling *R* how to load in NetLogo and the specific models that we'll be using. You'll need to adjust what it says to your own situation – `nl.path` should point to wherever your NetLogo application is located, and the `NLLoadModel` command requires the path to wherever you've stored 'Exp1_Full.nlogo' and its cousins.

   On a Windows machine, the correct NetLogo path will be something like:

   ```
   > nl.path <- C:\\Program Files (x86)\\NetLogo 5.2
   ```

   Once you've adjusted these, copy everything into the *R* console. To check that everything works, try to run one of the earthworm IBMs:

   ```
   > do.runExp1(make.litValues("Full"))
   ```

   This should produce the result of running the full model for Experiment 1, which is a vector of 27 masses starting with 0.011. However, you may find that the 'NLLoadModel' command throws a cryptic error – if so, the problem is probably your *NetLogo* version – *RNetLogo* currently only works with version 5.0 to 5.2.

7. We are now ready to do a true 'posterior predictive check', where the model is re-run using the accepted parameter values. This might take a while if you specify too many draws, so try:

   ```
   > plot.postCheck(full.abcEst, draws = 5, rerun = TRUE)
   ```

8. The nice thing is that we can now easily check the results of all our other ABC analyses. For instance, how well does the earthworm IBM fit the empirical data if we run it with the parameters chosen on the basis of Experiment 1 alone? Can you explain this result?

```
> e1.abcEst <- create.abcEst(target = all.data[ , col.e1],
  priors = full.priors, results = full.results[ , col.e1],
  rate = 0.001)

> plot.postCheck(e1.abcEst, draws = 5, rerun = TRUE)
```

9. How about if we try to do parameter estimation for the simple model? In Step 2, we found that the simple model is actually slightly favored over the full model when just fitting Experiment 1. So how well does it actually fit?

```
> simple.e1.abcEst <- create.abcEst(target = all.data[ ,
  col.e1], priors = simple.priors, results =
  simple.results[, col.e1], rate = 0.001)

> plot.postCheck(simple.e1.abcEst, draws = 5, rerun = TRUE)
```

10. If you run the `plot.postCheck` command on the simple model a few times, you'll see that the "best values" often fit Experiment 1 pretty poorly. This despite the fact that the model is actually favored over the more complex model when doing ABC model selection (recall `summary(e1.abcSel)`).

    What do you think might be causing this? (Hint: Putting `simple.abcEst$best.values` directly into 'Exp1_Simple.nlogo' and watching it run might help.) What does this suggest about using a set of ABC's 'best values' as a point estimate for an IBM's parameters?

REFERENCES

1. Johnston, A.S.A., et al., *An energy budget agent-based model of earthworm populations and its application to study the effects of pesticides.* Ecological Modelling, 2014. **280**: p. 5-17.
2. Thiele, J.C., W. Kurth, and V. Grimm, *RNetLogo: An R package for running and exploring individual-based models implemented in NetLogo.* Methods in Ecology and Evolution, 2012. **3**: p. 480-483.
3. Gunadi, B., C. Blount, and C.A. Edwards, *The growth and fecundity of Eisenia fetida (Savigny) in cattle solids pre-composted for different periods.* Pedobiologia, 2002. **46**: p. 15-23.
4. Reinecke, A.J. and S.A. Viljoen, *The influence of feeding patterns on growth and reproduction of the vermicomposting earthworm Eisenia fetida (Oligochaeta).* Biology and Fertility of Soils, 1990. **10**: p. 184-187.