Beginner's guide to **rSalvador** *1.6 for Windows, Mac and Linux users*

This guide helps you get started with rSalvador. For more advanced usage, you should consult the accompanying technical manual or use the inbuilt help facility. To get inbuilt help for a function, e.g., `newton.LD`, type the R command `?newton.LD`.

1 Installing rSalvador on the Windows Platform

- Download and install R 3.0.1 or higher version on your computer.
- Make a new folder as your working folder. We recommend naming your working folder `c:/rsalvador`. Copy the rSalvador distribution file `rsalvador_1.6.zip` into that new folder.
- Invoke R. rSalvador requires the R package `hypergeo`, which you can install by the R command

```
install.packages('hypergeo')
```

- rSalvador also requires the R package `gdata`, which in turn requires the `perl` software. However, most Windows computers do not have the software `perl` pre-installed. We recommend `Strawberry Perl`, which is available on the Internet for free download. Choose the stable version to download and accept all defaults during installation. After installing `perl`, install the R package `gdata` with the R command:

```
install.packages('gdata')
```

- Users not interested in the Excel file import capability can skip the installation of both `gdata` and `perl`. In this case, you can ignore the harmless warning messages about the unavailability of `perl` and `gdata` that you may see each time you load rSalvador.
- Use the R command `setwd('c:/rsalvador')` to set the new folder as your working directory. If your working folder has a different name, replace `'c:/rsalvador'` with the actual name. You can verify your working directory with the R command `getwd()`. You can check whether the rSalvador distribution file `rsalvador_1.6.zip` is present in your working folder by executing `list.files()`. Finally, install rSalvador as follows:

```
install.packages('rsalvador_1.6.zip')
```

2 Installation on the Macintosh Platform

The installation procedure is the same as that for the Windows platform, except for the last step. Instead of downloading the file `rsalvador_1.6.zip'`, use the file `rsalvador_1.6.tgz`. That is, from within R, one needs to execute

```
install.packages('rsalvador_1.6.tgz')
```

3 Installation on the Linux Platform

To begin with, install R and the two R packages (`hypergeo`, `gdata`) as described in the above section. Now, follow the following five steps:

1. Make an installation directory and copy the installation file `rSalvadorV1.6.tar` into the installation directory. For example:

   ```
   mkdir myrsalvador
   ```

2. Untar the installation file that is already in your installation directory (e.g., `~/myrsalvador`):

   ```
   tar xvf rSalvadorV1.6.tar
   ```

3. A sub-directory `rsalvador` is now created under the installation directory (e,g., `~/myrsalvador`), but continue to stay in the installation directory. From the installation directory, install rSalvador by executing

   ```
   R CMD INSTALL rsalvador
   ```

4. If installation is not successful, go to subdirectory `~/myrsalvador/rsalvador/src` and delete all files of the form `*.o`, `*.so` and `*.rds`. Then repeat the above steps.

4 Starting rSalvador

Each time you invoke an R session, you need to load rSalvador with the R command `library(rsalvador)`.

5 Your First rSalvador Calculations

Here we use the Demerec data to demonstrate the basic capabilities of rSalvador. To view this data set, type `demerec.data`.

- to find the maximum likelihood estimate of m:

  ```
  newton.LD(demerec.data)
  ```

- to find the 95% confidence interval for m:

  ```
  confint.LD(demerec.data)
  ```

- To view the iteration details:

```
newton.LD(demerec.data,show.iter=TRUE)
```

- to display the log-likelihood function:

```
plot.likelihood.LD(demerec.data)
```

6 Importing and Exporting Data

There are three ways to transfer data into rSalvador.

- creating a sequence of numbers within R:

```
y=c(0, 16, 20, 2, 2, 56, 3, 361, 9)
```

- importing data from a text file. This data file should have just one column of numbers. It can have one or more memo lines. (Use built-in help for details.)

```
y=import.text.data('example1.txt')
```

- Typically, you may have saved your data in an Excel spreadsheet file, like the accompanying example file `example2.xlsx`. (See built-in help for details.) To import data from that file, type

```
y=import.excel.data('example2.xlsx')
```

- Now you can repeat the calculations in the above section by replacing `demerec.data` with the new data variable `y`.
- Occasionally it may be desirable to save your data (say in `y`) into a plain text file for future use. This can be done by

```
export.text.data('mytest.txt',y)
```

  To read this data file back into rSalvador for analysis:

```
import.text.data('mytest.txt')
```

7 Adjusting for Plating efficiency

We use data from experiment #16 of Luria and Delbruck as an example. This experiment has a plating efficiency of 0.4. You can view the data by typing `luria.16.data` and learn more by typing `?luria.16.data`.

- to find the maximum likelihood estimate of m:

```
newton.LD.plating(luria.16.data,e=0.4)
```

- to find the 95% confidence interval for m:

  ```
  confint.LD.plating(luria.16.data,e=0.4)
  ```

- To view the iteration details:

  ```
  confint.LD.plating(luria.16.data,e=0.4,show.iter=TRUE)
  ```

- to view the log-likelihood function graphically:

  ```
  plot.likelihood.LD.plating(luria.16.data,e=0.4)
  ```

8 Adjusting for Relative Fitness

For the sake of illustration, assume that from a fitness assay the experimentalist learns that the relative fitness is $w = 0.21$ for the Demerec data. To get a maximum likelihood estimate of m, execute the R command

```
newton.MK(demerec.data,w=0.21)
```

The output is $\hat{m} = 25.86$. To set up a 95% CI for m, execute

```
confint.MK(demerec.data,w=0.21)
```

which yields a 95% CI for m: $(22.98, 28.81)$.

9 Comparison of mutation rates

Two mutation rates can be compared by checking whether the two 84% confidence intervals overlap.

```
confint.LD.plating(crane.data[[1]],e=0.1,alpha=0.16)/3.6e9
confint.LD.plating(crane.data[[2]],e=0.1,alpha=0.16)/3.9e9
```

The two confidence intervals for the two mutation rates overlap, and hence the difference is not significant. If terminal cell population sizes are the same in the two experiments, `compare.LD` performs a likelihood ratio test. In the Newcombe experiments, the difference in N_t between Experiments F and H are small, so a likelihood ratio test is possible.

```
compare.LD(newcombe.data[[6]],newcombe.data[[8]])
```

The likelihood ratio test statistic is $\Lambda = 0.0119$ and the p-value is 0.9130.

If final cell population sizes differ noticeably, a more flexible approach is to conduct a likelihood ratio test (LRT) that accounts for the difference in final cell population size. rSalvador provides three functions for that purpose, namely, `LRT.LD, LRT.LD.plating` and `LRT.MK`. All these functions require a value of R, the ratio of the two cell population sizes. The function names are intended as a mnemonic aid, e.g., `LRT.LD.plating` indicates that the underlying distribution is the usual Luria-Delbrück distribution with plating efficiency less than perfect. Take the Werngren-Hoffner data for example, Here we have plating efficiency $\epsilon = 0.4$. Immediately prior to plating, the first experiment had final cell density 2.3×10^8 per mL, and the second experiment had final cell density 0.5×10^8 per mL. Therefore, R=0.5/2.3. To compare mutation rates in these two experiments, one executes the following

```
R=0.5/2.3
LRT.LD.plating(wh.data[[1]],wh.data[[2]],R=R,e1=0.4,e2=0.4)
```

which gives $\Lambda = 15.16$ and $p = 9.86 \times 10^{-5}$.

10 Accounting for excessive variation in N_t

When the initial inoculum size N_0 is too small (< 10, say), the final cell count N_t can vary considerably. If N_0 cannot be increased, the gamma mixture model can be used to reduce the potential bias caused by large variation in N_t. An estimate of the coefficient of variation (cv) for N_t is needed. For example, if the cv for N_t in the Demerec experiment were (hypothetically) 0.2, one can proceed as follows.

```
newton.B0(demerec.data, cv=0.2)
```

which produces an ML estimate of 10.96 for the parameter m. Similarly, a 95% likelihood ratio CI can be obtained by

```
confint.B0(demerec.data, cv=0.2)
```

which yields $(8.70, 13.41)$.

11 Estimating m with uncertainty about relative fitness w

The relative fitness parameter w is often measured by a competition/fitness assay in the laboratory. When the sample size is relatively large, the fundamental parameter m can be estimated while w is treated as an unknown parameter of secondary interest. For example, to find the joint ML estimate of m and w for the Demerec data, one executes:

```
newton.joint.MK(demerec.data)
```

The output is a pair of numbers, $(9.852, 1.12)$, representing an ML estimate of the (m, w) duplet.

To find a 95% profile likelihood confidence interval for m, one issues the command

```
confint.profile.m(demerec.data)
```

which yields an interval of the form $(6.98, 13.01)$. The corresponding command for finding a 95% profile CI for the relative fitness w is

```
confint.profile.w(demerec.data)
```

which yields an interval of the form $(0.887, 1.447)$.

12 Computing expected Fisher information

The expected Fisher information for m can be used to determine sample size. By truncating an infinite series, one can get approximate expected Fisher information.

- To calculate $I(3.0)$ under the Mandelbrot-Kock distribution, execute `fisher.MK(m=3.0,w=1.0,n=2000)`, which yields `0.1631`
- To see if we already have 2 good significant digits, execute

  `fisher.MK(m=3.0,w=1.0,n=4000)`, which yields `0.1632`
- To calculate $I(50.0)$ under the standard Luria-Delbrück distribution with plating efficiency $\epsilon = 0.1$, execute `fisher.LD.plating(m=50.0,e=0.2,n=2000)`, which yields `0.00174`