Electronic Supplementary Information 5

Installation and usage of the software package for numerical solution and analytical approximation of the Poisson-Boltzmann equation

Installation instructions

The software package is implemented as a library for R-cran[1], and therefore needs the R-cran program to run. If not already installed, the newest version of R-cran can be obtained from http://cran.r-project.org, along with installation instructions. The R program is a command line calculation environment, similar to Matlab, but with a focus mainly on statistical computing. Very basically, this type of environment allows to type commands at a command prompt, which are then executed when the enter key is hit. R comes with the capacity to install numerous plugins, referred to as libraries, to carry out specific functions beyond the R core commands. The software package for solving the Poisson-Boltzmann equation made available here is such a library, named "poisson.boltzmann.1D" within R.

To use the library, the first step is to install R as outlined under http://cran.r-project.org . Once the R program installed, the library needs to be downloaded from Soft Matter, and more precisely from the Supplementary Info section associated with the paper "Soft nanofluidics governing minority ion exclusion in charged hydrogels", Braschler et al., 2015. The reader can find this paper for instance by navigating to the web page of Soft Matter (http://www.rsc.org/softmatter), and then entering the title of the paper in the Full Text search box.

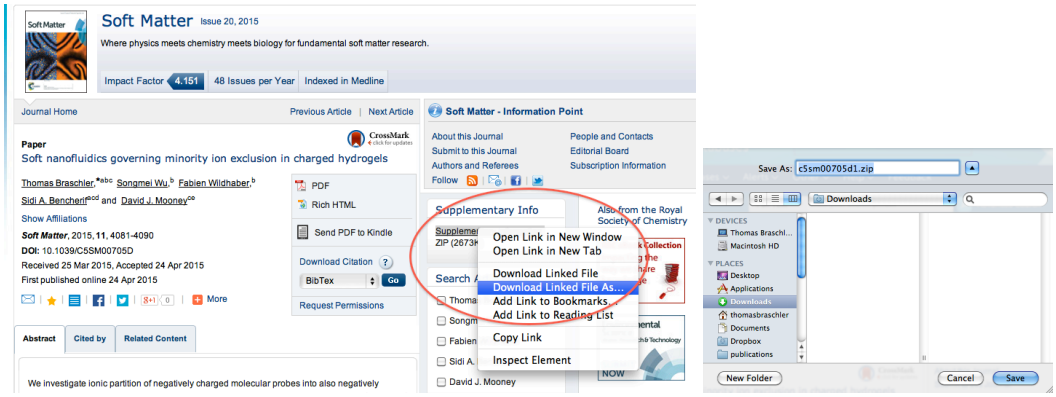
The choice of the exact file to be downloaded is operating-system dependent, specific instructions are given below for MacOSX and Windows.

The software package can then be installed via the standard R package installation mechanism. The exact procedure varies again slightly for different operating systems and R versions, but the general idea is to use the package installer provided by R, specific instructions for MacOSX and Windows being given below.

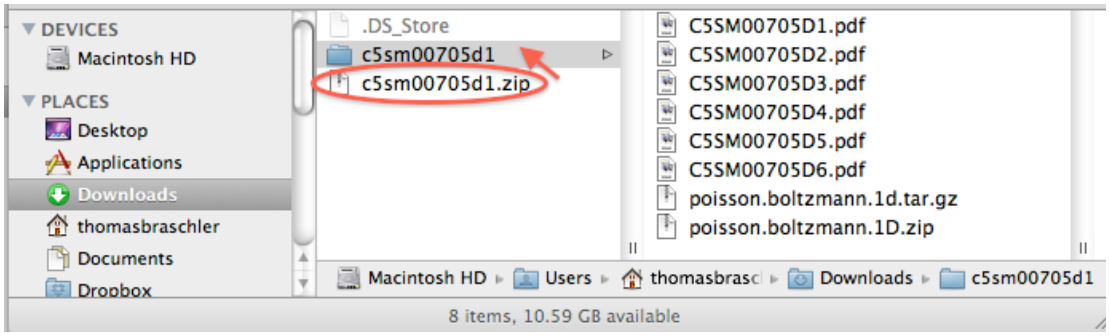
The packages are compiled on R 3.1.3 (MacOSX) and R 3.1.2 (Windows), they do not work with R 2.x.x versions.

MacOSX

The software package can be downloaded from the website of Soft Matter in the Supplementary Info section, as shown in Screenshot 1. The entire Supplementary Info is available as a single zip file (Screenshot 1), which should be unzipped locally (Screenshot 2). This yields the pdf files referred to in the main text as Electronic Supplementary Info 1-6, as well as the R libraries for MacOSX and Windows (Screenshot 2).



Screenshot 1: Download of Supplementary Info on the web page of Soft Matter. The entire Supplementary Info is contained in a single zip file.



Screenshot 2: Unzipping the downloaded file. The zip file from the Soft Matter website ("c5sm00705d1.zip") should be unzipped locally (indicated with arrow). It should contain 6 pdf files (which are the Electronic Supplementary Files 1-6), a file named poisson.boltzmann.1d.tar.gz, which is the desired library for R under MacOSX, and a file named poisson.boltzmann.1D.zip, which is the corresponding Windows version.

The R program should then be launched (in MacOSX, R typically installs as R.app in Applications). The package installer in the MacOSX version of R is available in R under "Packages & Data"->"Package Installer". (Screenshot 3). Within the package installer window, the correct type should be chosen for the package to be installed, it is "Local Source Package" in the dropdown menu for the package types (Screenshot 4). Once the correct package type selected, the "Install …" button needs to be pressed (Screenshot 4); if the button is grayed out, one needs to re-select "Local Source Package" in the dropdown menu. The file "poisson.boltzmann.1D.tar.gz" should then be selected among the unzipped material (Screenshot 5). This launches the automatic installation procedure. During the installation, R prints a series of messages in the command prompt, the line

* DONE (poisson.boltzmann.1D)

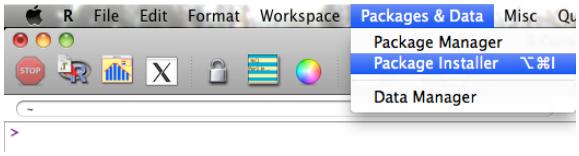
indicates successful installation (Screenshot 6). The library is now installed, but not yet loaded into memory, this needs be done by means of the command:

library(poisson.boltzmann.1D)

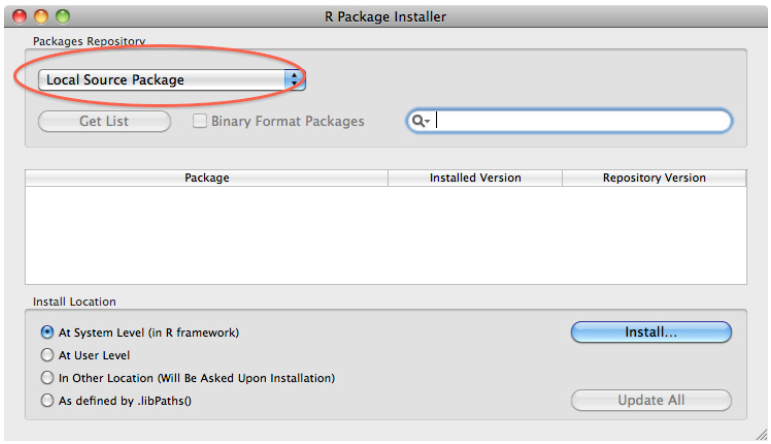
also shown in Screenshot 6. As a first step, and to verify that the package is indeed functional, the package help can be called up by the command:

help(poisson.boltzmann.1D)

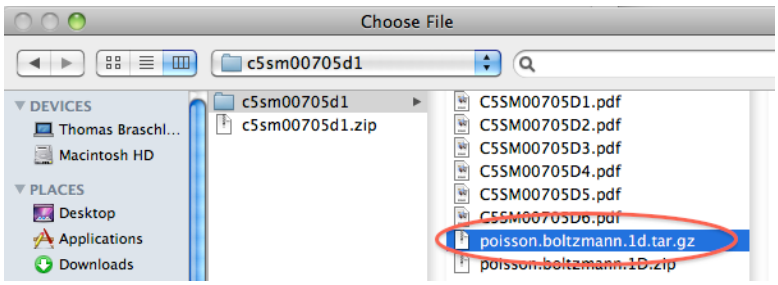
If the installation and package loading went well, the help window shown in Screenshot 12 should pop up (see Package Usage section below).



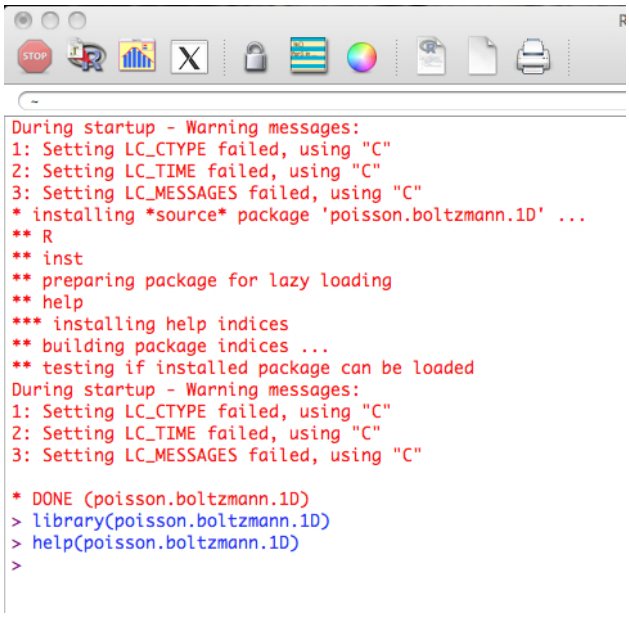
Screenshot 3 (MacOSX): The package installer is launched from the R menu, "Packages & Data"->"Package Installer".



Screenshot 4 (MacOSX): Within the R package installer, the "Local Source Package", encircled in the screenshot, should be selected for the package type.



Screenshot 5 (MacOSX): Selection of the downloaded package. The file to be installed is "poisson.boltzmann.1d.tar.gz", obtained by unzipping the zip file c5sm00705d1.zip downloaded from Soft Matter (Screenshot 1 and 2).

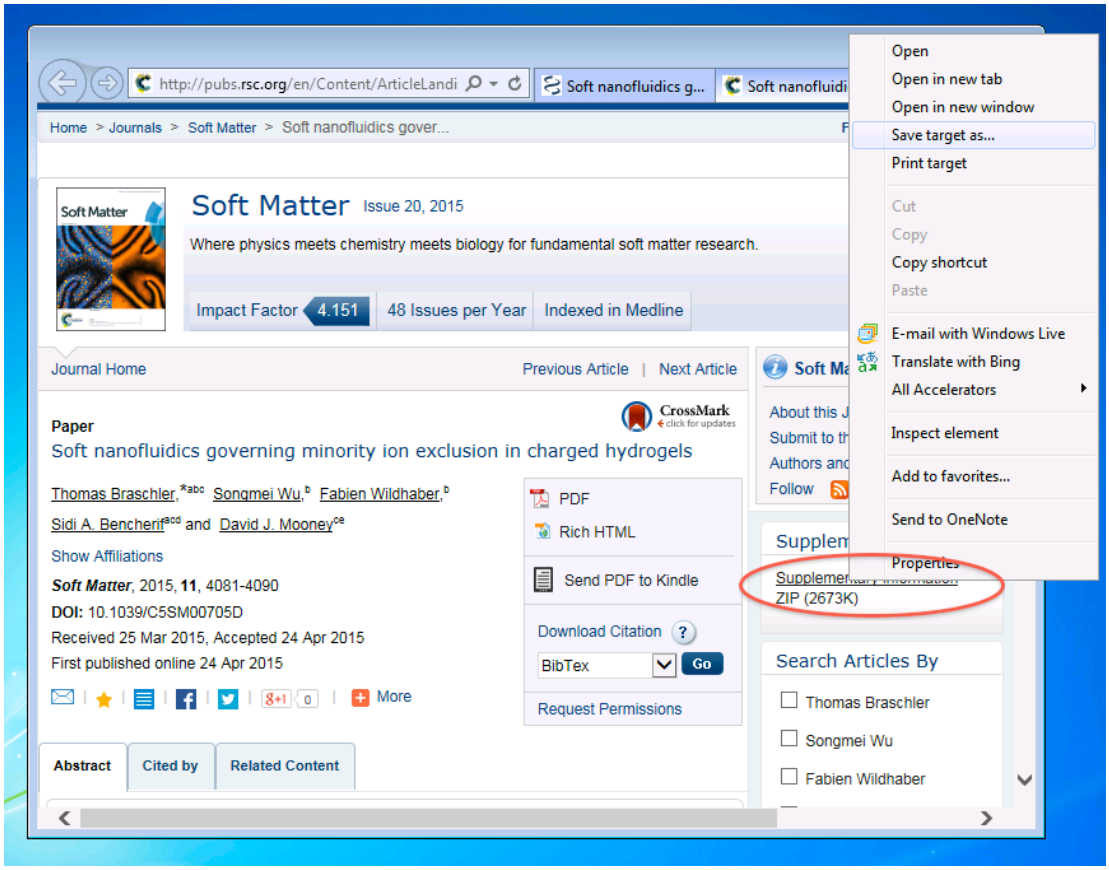


Screenshot 6 (MacOSX): Output in the command window during package installation, followed by package loading and the package help command.

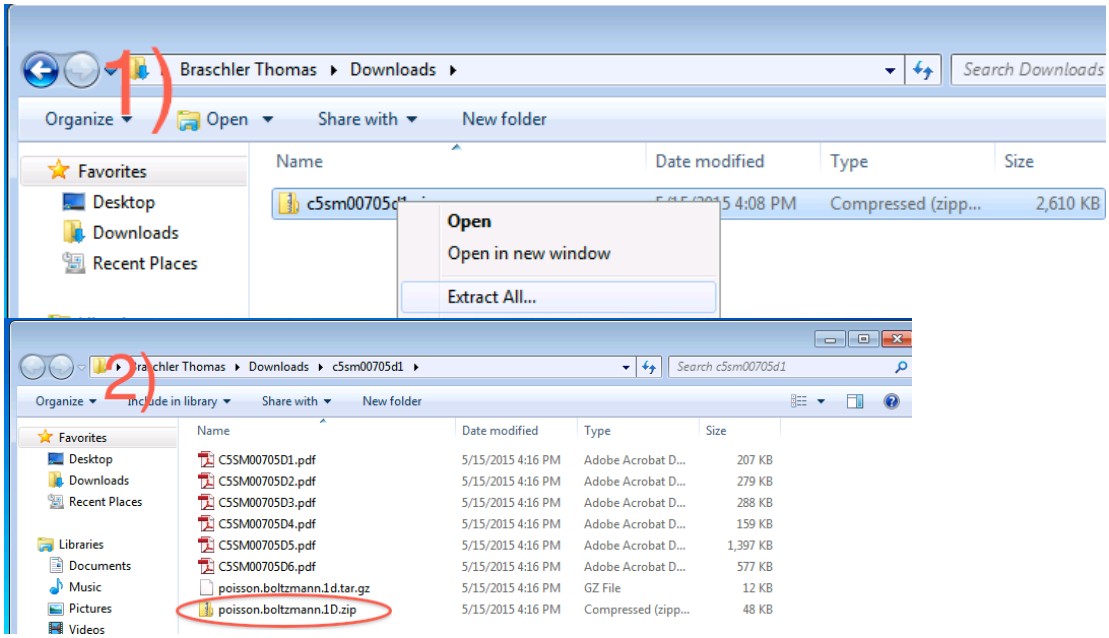
Windows

The software package can be downloaded from the website of Soft Matter in the Supplementary Info section, as shown in Screenshot 7.

The Supplementary Information for the manuscript "Soft nanofluidics governing minority ion exclusion in charged hydrogels", Braschler et al., 2015, is available as a single zip file from the website of Soft Matter, http://www.rsc.org/softmatter (Screenshot 7). Upon download, it should be unzipped locally (Screenshot 8). This yields a total of 8 files, 6 of which are the pdf files referred to as Electronic Supplementary Information 1-6 in the main text, along with a Windows and MacOSX version of the library to be installed. The file of interest for the Windows installation is poisson.boltzmann.1D.zip (Screenshot 8).



Screenshot 7: Download of the package for Windows from the Soft Matter website. The website proposed the automatically generated filename c5sm00705d1.zip, which we shall use here.



Screenshot 8: Unzipping the downloaded file c5sm00705d1.zip yields a total of 8 files. The file of interest for the windows installation is poisson.boltzmann.1D.zip.

After downloading and unzipping of the Supplementary Information, the R program needs then to be launched. The package installer is launched from within R by selecting “Packages” -> “Install packages from local zip files …”. (Screenshot 9). The downloaded file “poisson.boltzmann.1D.zip” should then be selected in the ensuing file selection window (Screenshot 10). The opening of the file in the selection window launches the installation procedure. If successful, the message

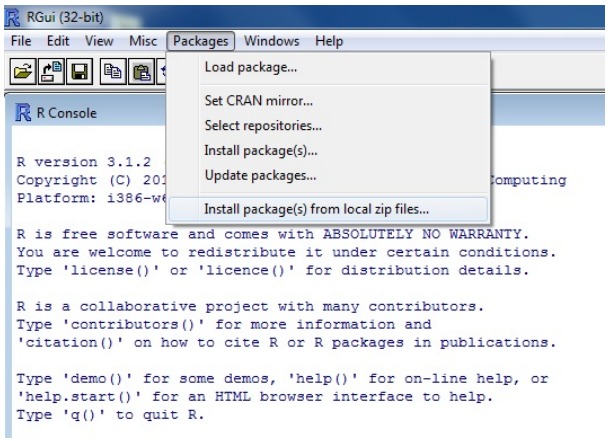
Package “poisson.boltzmann.1D” successfully unpacked and MD5 sums checked

is displayed in the command window (Screenshot 11). Further, executing the loading command:

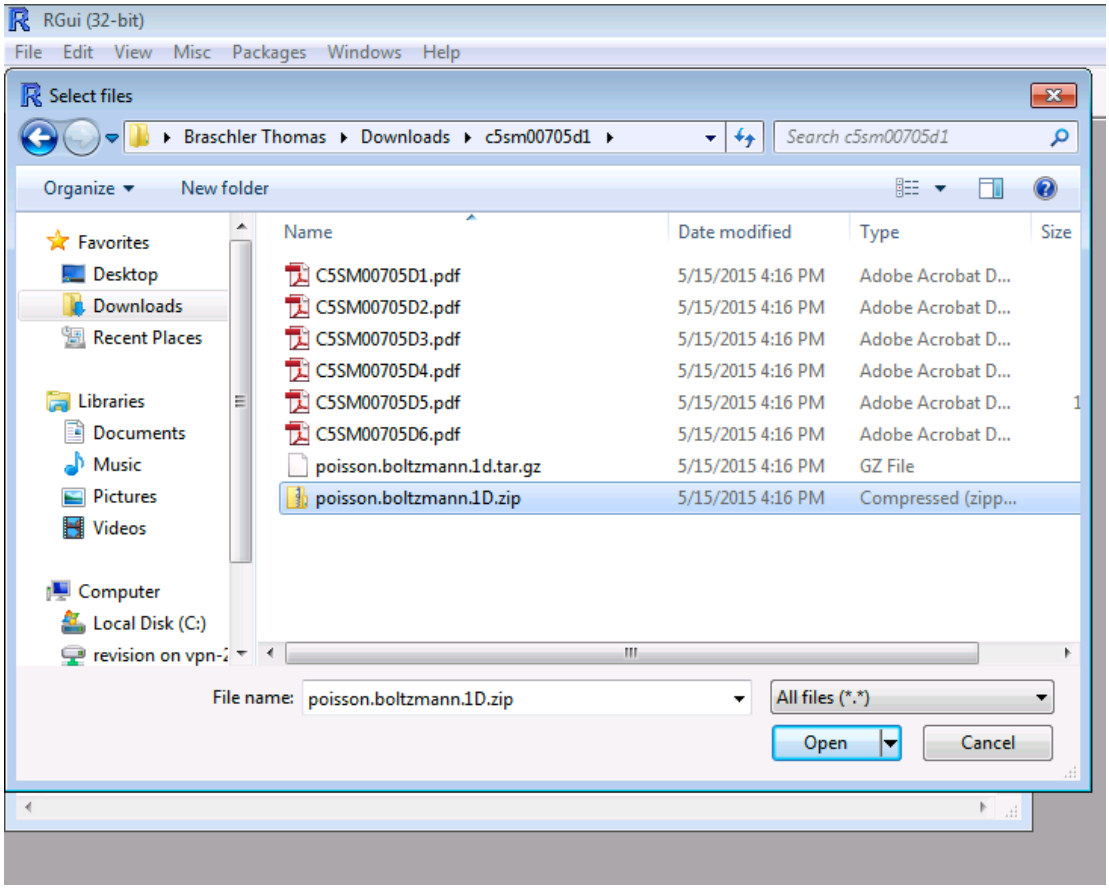
library(poisson.boltzmann.1D)

should produce no error (Screenshot 11), and executing the help command help(poisson.boltzmann.1D)

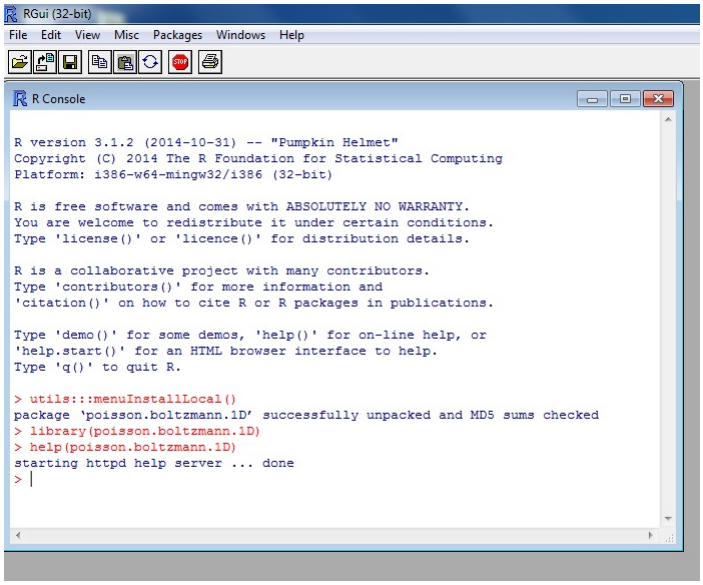
should open the help window shown in Screenshot 12 (see Package Usage section below).



Screenshot 9 (Windows): Launch of the package installer from within R, by using the menu commands Packages -> Install package(s) from local zip files …



Screenshot 10 (Windows): Selection of the package to be installed.



Screenshot 11 (Windows): Command window after package installation.

Package Usage

R for Windows and R for MacOSX use the same commands within the command line window, such that once the package is installed, the usage is identical in Windows and MacOSX.

Very importantly, R separates package installation from package loading. This means that it is imperative to load the package into memory not only immediately after installation, but also every time R is relaunched and one desires to use the package. The specific package loading command is:

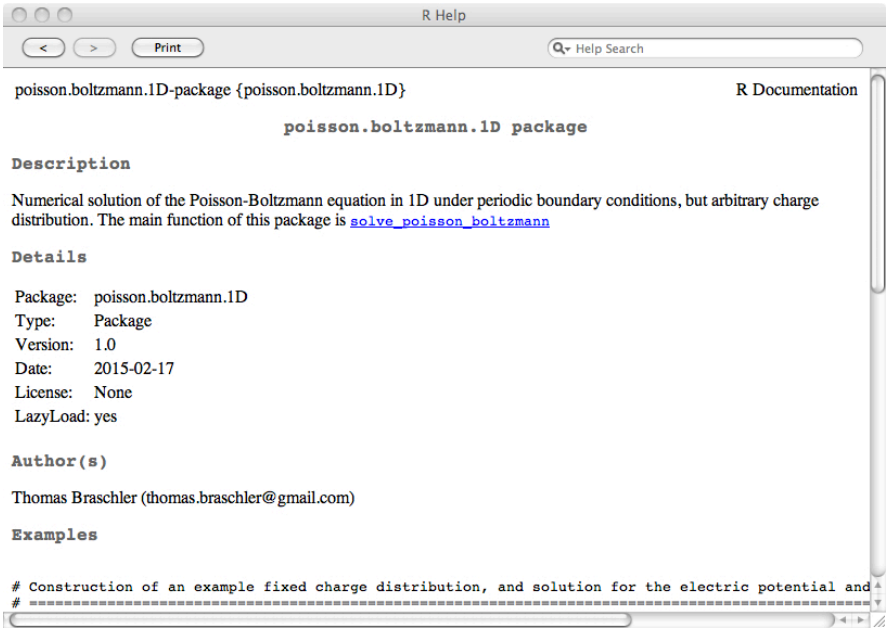
library(poisson.boltzmann.1D)

to be executed on the R command line prompt. Once this is done, the functions defined by the package as well as the associated help can be used.

After package loading, the package-level documentation can be called up with the command:

help(poisson.boltzmann.1D)

Following execution of this command, a pop-up window should open, with the general description of the package and illustrative usage examples. This help window is shown in screenshot 12, the exact layout varies slightly according to operating system and R version.

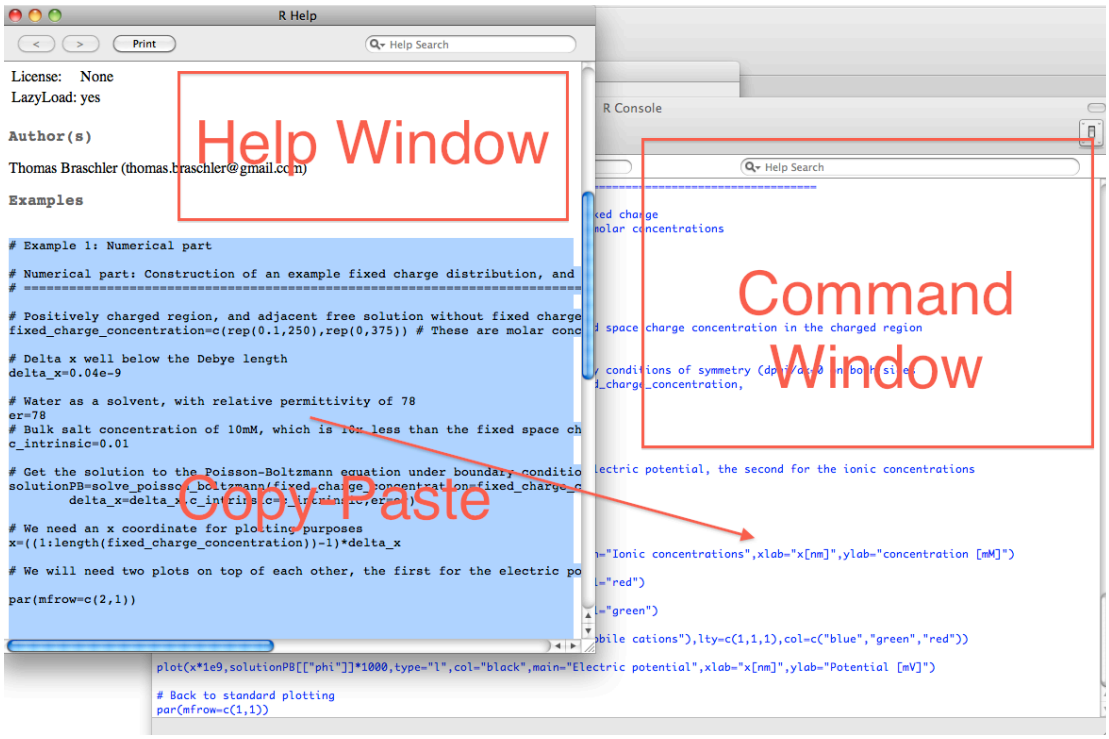


Screenshot 12: Package help file.

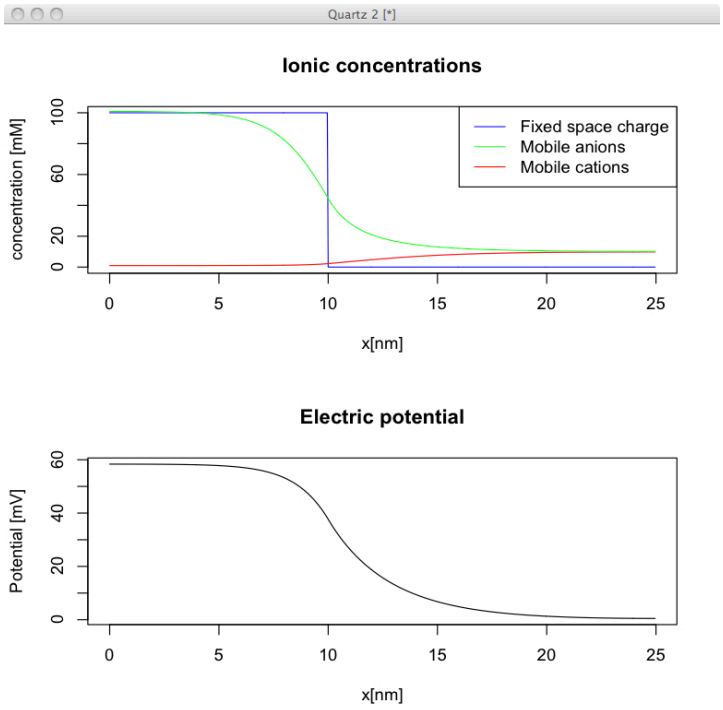
To get a first idea of the package functionality, examples of typical usage have been provided in the "Examples" section of the package-help (Screenshot 12). There are two examples, one for numerical solution of the Poisson-Boltzmann equation, and a second one for analytical approximation.

To run one of the examples, the corresponding code in the Examples section should be copy-pasted into the command window, followed by the "enter" key to execute it (Screenshot 13). Executing the first of the two examples launches the solution of the Poisson-Boltzmann equation for a sample charge distribution. This example also displays the solutions graphically (Screenshot 14). The plots illustrate the mobile ion and electric potential distribution at the edge of a positively charged space charge region.

The second example in the package help section shows analytical estimation of the partition coefficient.



Screenshot 13: Execution of example code by copy pasting from the help window to the command window, followed by the enter key. Here, execution of the first of the two examples available is illustrated.



Screenshot 14: Graphics produced by execution of the package help example #1. The plots show the distribution of mobile ions as well as the electric potential at the edge of a fixed positive space charge region, the fixed space charge concentration being 100mM, the bulk electrolyte concentration being 10mM.

At a technical level, the package contains both functions to numerically solve the Poisson-Boltzmann equation with arbitrary fixed charge distributions, and analytical approximation for the particular case of the solid nanochannel model where the space charge is concentrated in the walls lining a pore containing only electrolyte solution.

Numerical solution of the Poisson-Boltzmann equation

The main function of the numerical Poisson-Boltzmann part of the package is "solve_poisson_boltzmann", used to solve the Poisson-Boltzmann equation in 1D and under symmetry boundary conditions. The specific documentation for this function can be called up using the command

help(solve_poisson_boltzmann)

The help section again contains an example section which can be run in the R command prompt. The solve_poisson_boltzmann internally makes use of 3 helper functions: concentrations_boltzmann, excess_charge_boltzmann, integrate_poisson_boltzmann_1D, for which documentation is also available via the specific help commands:

help(concentrations_boltzmann)

help(excess_charge_boltzmann)

help(integrate_poisson_boltzmann_1D)

The source code of each of the function finally can be displayed by typing the name of the function at the command line, followed by enter, i.e.

solve_poisson_boltzmann

concentrations_boltzmann

excess_charge_boltzmann

integrate_poisson_boltzmann_1D

Analytical approximations for the solid nanochannel case of the Poisson-Boltzmann equation

The package also provides analytical approximations to the partition coefficient along with the potentials at the channel midline and walls for the "solid nanochannel model" as described in the main text. In particular, this allows to evaluate eq. 10 in the main text.

The main function to be used for this purpose is "partition_coefficient_analytical_hydrogel", again with its help page and example accessible by:

help(partition_coefficient_analytical_hydrogel)

The partition_coefficient_analytical_hydrogel function uses a number of sub-routines, each with its own help section, accessible by the syntax given above, with the appropriate function name. These functions are: "partition_coefficient_analytical_nanochannel", "potentials_analytical_nanochannel", "partition_coefficient_analytical_from_potentials". These functions may be useful for specific purposes on their own right. The "partition_coefficient_analytical_nanochannel" function allows to calculate the partition coefficient for a given tracer ion and nanochannel geometry; the "potentials_analytical_nanochannel" provides analytical approximations to the electric potentials in the middle of the channel and at the walls as a function of channel geometry and surface charge; and the "partition_coefficient_analytical_from_potentials" technically evaluates the integrals in eq. 10 in the main text and therefore allows to find the partition coefficient once the channel midline and edge potentials are known.

The exact equations used by these functions are listed in Electronic Supplementary Information 6.

Mathematical Context of the numerical solution

The analytical approximation of the solution to the Poisson-Boltzmann equation, which underpins the analytical part of the package, is discussed in Electronic Supplementary Information 6.

The numerical solution of the Poisson-Boltzmann equation (eq. 1 in the main text), under boundary conditions of symmetry (eq. 2 in the main text) is also not entirely a trivial matter, which is why we provide the "poisson.boltzmann.1D" package described above for the reader's convenience. Below, the mathematical details underlying the algorithm employed are given, they underpin the source code of the function solve_poisson_boltzmann, which is the workhorse of the numerical part of the package.

Basic algorithm

A first problem in the solution of the Poisson-Boltzmann equation (eq. 1 in the main text) arises due to the particular boundary conditions: Eq. 2 in the main text specifies boundary conditions at two different x-coordinates (i.e. $\phi'=0$ at $x=0$, and $\phi'=0$ at $x=r_{pore}$). Standard numerical integration of eq. 1 would however require the knowledge of two boundary conditions at a single point x_0, typically ($\phi=\phi_0$ and $\phi'=0$, both at $x=0$).
Hence, we need an algorithm to find ϕ_0 such that setting $\phi(x=0)=\phi_0$ and $\phi'(x=0)=0$ also satisfies $\phi'(x=r_{pore})=0$. The software package does so by an iterative guessing-and-improvement method. Indeed, the monotonous nature of the right-hand side of eq. 1 as a function of ϕ implies that if a too high value is chosen for ϕ_0, a too high value for $\phi'(x=r_{pore})$ will result, and visa-versa, a too low value for ϕ_0 also implies a too low value for $\phi'(x=r_{pore})$. Hence, the software package proceeds as follow:

0. Choose a starting value for ϕ_0 (the software uses $\phi_0=0$)
1. Perform numerical integration of eq. 1, with the initial values $\phi(x=0)=\phi_0$ and $\phi'(x=0)=0$ and the user-defined fixed charge density distribution $c_{fixed}(x)$. This yields $\phi'(x=r_{pore})$
2. If $\phi'(x=r_{pore})=0$, the value for ϕ_0 is correct. Otherwise, if $\phi'(x=r_{pore})>0$, ϕ_0 should be decreased, whereas if $\phi'(x=r_{pore})<0$, ϕ_0 needs to be increased
3. Return to step 1

The amount by which ϕ_0 is changed needs to be decreased progressively. The software starts with a step size of 1V, and decreases this step size by a factor of two in every cycle where a switch in sign is observed for $\phi'(x=r_{pore})$.

Handling of numerical divergence

A second numerical problem arises due to the exponential terms in eq. 1. Indeed, even for very small deviations from the correct value of ϕ_0, the solution obtained upon integration of eq. 1 rapidly diverges, and due to the finite range of numbers

that can be represented by given floating point arithmetic, one rapidly obtains infinite values respectively floating point overflow.

Depending on the language of implementation, the floating-point overflow has different consequences. R-cran, in which the implementation provided here is written, handles floating-point overflow by using symbolic representations of +Infinity and –Infinity. Obtaining +Infinity for $\phi(x=r_{pore})$ also implies that $\phi'(x=r_{pore})$ has a very large positive value, such that divergence to positive infinity (in terms of available floating point range) indicates in the end simply a positive sign of $\phi'(x=r_{pore})$, whereas negative divergence can be taken as equivalent to $\phi'(x=r_{pore})<0$. In other languages, floating point overflow either leads to a crash or inadvertent sign change of the variable; in this case, explicit error handling routines would have to be implemented.

Limitations of the algorithm

We provide source code for the reader's convenience. The algorithm provided here is voluntarily limited to the needs of the periodic 1D space charge model in the presence of a large excess of a monovalent electrolyte. In particular:

- The Poisson-Boltzmann equation can also be solved in higher dimensionality (2D, 3D). This is not attempted here.
- Only the symmetry boundary condition has been implemented here ($\phi'=0$). Other boundary conditions would obviously also be possible, although this would require some adjustments of the code and would also increase somewhat the complexity of package usage.
- Minor adjustments would also be required to simulate multivalent bulk electrolytes; the algorithm can however be used as is for the estimation of the concentration of a multivalent minority tracer by considering the electric potential to be imposed by the monovalent (majority) electrolyte and then applying equation 8 in the main text for the partition coefficient.
- The numerical solution of the Poisson-Boltzmann equation is very sensitive to the exact value of ϕ_0, used to define the correct solution. Typically, it is not possible to correctly simulate the electric potential and therefore ionic concentrations beyond 10 or 50 Debye lengths from the starting point (x=0).
- The x-points are regularly spaced, and the distance between them has to be specified manually by the user. Typically, this distance should be on the order of 1-10% of the Debye length under the chosen bulk electrolyte concentration.

Finally, it should be said that for legal purposes, it is the user's own responsibility to decide how and to which extent the code is used. We do not claim any correctness or fitness for any particular purpose, and are not responsible for any damage to computer or real-world installations, or physical or moral persons, resulting from potential faultiness, inexactitude,

inappropriateness, or even exactitude, or any other aspects of the algorithm or code.

Bibliography

1. R-Core-Team, R Foundation for Statistical Computing, Vienna, Austria, 2014.