VarGibbs user manual

VarGibbs Version 3.1 user manual

Gerald Weber gweberbh@gmail.com
Departamento de Física, Universidade Federal de Minas Gerais, Brazil

August 17, 2021

This work was supported by Fundação de Amparo a Pesquisa do Estado de Minas Gerais (Fapemig); Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq); and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Contents

1 Introduction

VarGibbs provides the minimization method for calculating nearest-neighbour thermodynamic parameters [1].

What will this software do for you? Given a set of experimental melting temperatures, measured at any species concentration, it will calculate the relevant nearest-neighbour entropies and enthalpies, as well as initiation parameters. Once you have these parameters you may also use them to do simple melting temperature prediction.

The method works by minimizing predicted temperatures for the nearest-neighbour model in a very similar way as used previously by us for the Peyrard-Bishop model [2–4]. This is also the reason why VarGibbs shares many practical aspects with our free software implementation for the Peyrard-Bishop model [5] such as the format of specifying nearest-neighbours and result files.

Evidently, this is work in progress. The method is new and as such there may be aspects of it which are not yet completely understood.

I would find it truly helpful indeed if you would let me know if this software is of any use to you. Showing a list of interested users to funding agencies often helps to secure the necessary resources to keeping such projects running. So, please, if you find this software useful let me know and if you use it for your scientific work please cite the appropriate papers which are listed at the end of this manual.

I wish you all the best in using VarGibbs

Gerald

Belo Horizonte, August 17, 2021

Changes

Version 3.1, August 2021

- New script files for RNA, contributed by Izabela Ferreira.
- You can now do a `-calc=nncheck` (3.1.11) for single sequences.
- New warning and error formats.
- New options to control the amount of output in log files, `-minloglevel` (3.3.8) and `-loginterval` (3.3.9).
- Documentation corrections and improvements.
- Added missing Na^+ concentration information in some parameter files.
- New option `-printusedpar` (3.4.8).

Version 3.0, January 2021

- Extended format for data files, see section 8.3.

Version 2.2, March 2019

- New parameter files for hybrid DNA-RNA [6].
- New option to require log files to be written in gzip format (new option `-zlog` (3.3.7)).
- Redesign of NN calculation taking advantage of pointers, CPU times reduce by a factor of 7 compared to previous versions.
- Added `symmetry_correction_entropy` to some parameter files
- New parameter file `P-SL04.par` from SantaLucia and Hicks [7]
- `boost_iostreams` library is now required for compilation

Version 2.1, October 2018

- New compiler option -std=c++11
- Program now shows elapsed cpu time and memory usage.
- New optimized parameters for RNA with salt dependence.
- New options for performing numerous salt corrections, new parameter `Na+:concentration`, see section 3.5.
- analysis with option `-calc=nncheck` (3.1.11) is now saved into a specific file with extension `.nncheck`
- Removal of `AOP-DRC17.par`, `AOP-LS95.par` and `D-DRC17.dat`, they will be reintroduced after further improvements.

Version 2.0, August 2017

- Modifications for hybrid DNA-RNA sequences with specific naming scheme for parameters. New optimized parameters for DNA-RNA, `AOP-LS95.par` and `AOP-DRC17.par`, and associated datasets `D-LS95.dat` and `D-DRC17.dat`, as well as old unoptimized parameters `P-SG95.par`
- Documentation updates and corrections.
- Corrected parameter files `P-BS86.par` and `P-FR86.par` for RNA.
- Several new parameters from the literature.
- Added new option `-nnpo` (3.1.3) for using NN parameters only.

Version 1.1, December 2015

- Documentation updates.
- Code clean up, removed `gbc_exp` from C++ name space.
- Linking to the library `libboost_regex` is now required.
- Added optimized parameter files at different salt concentrations for D-OW04 datasets.
- Added original parameters from Refs. 8–10.
- Correcting bug which was not generating complementary sequence when `cseq` was not given.
- You can now specify a list of C_t concentrations in option `-ct` (3.1.8)

Version 1.0, July 2014

First public release of VarGibbs.

2 Installation

The easiest way to install is to visit the OpenSUSE Build service (`http://build.opensuse.org`) and search VarGibbs for your Linux distribution, this should take care of obtaining the correct libraries which are needed for VarGibbs to run.

2.1 What will be installed?

Typically, there will be at least a binary executable file

`/usr/bin/vargibbs`

and further files, such as model parameter files, pre-calculated regression parameters and example scripts are to be found in

`/usr/share/vargibbs`

the documentation (which you are reading right now) should be located at

`/usr/share/doc/packages/vargibbs`

2.2 Specific instructions

2.2.1 OpenSUSE

Installing via repository Using the graphical interface Yast2 or the command line zypper add the following repository URL

`http://download.opensuse.org/repositories/home:/drgweber/openSUSE_15.1/`

then search for the package VarGibbs and select install. I you have a different OpenSUSE then change the last numbers to your installed version accordingly. The installation via repository has the advantage that you may simply update for future versions instead of repeating the whole installation procedure.

Command line download/install Download the appropriate package for your system from

`http://download.opensuse.org/repositories/home:/drgweber/openSUSE_15.1/`

for example if your system is 64bits, you may download the package

`http://download.opensuse.org/repositories/home:/drgweber/openSUSE_15.1/x86_64/VarGibbs-3.1-1.21.1.x86_64.rpm`

note: version numbers may vary from this example. Then install

```
zypper install VarGibbs-3.1-1.21.1.x86_64.rpm
```

2.3 Compiling the source files

Please read this section if you are unable to find the packages for your specific Linux distribution.

This software was developed and tested under OpenSUSE Linux 15.1 and depends on some libraries to function properly:

1. `libboost_filesystem1_66_0` `libboost_system1_66_0` `libboost_regex1_66` `http://www.boost.org`
2. `gsl` `http://www.gnu.org/software/gsl/`
3. `libblas3` `http://www.netlib.org/lapack/`

which means that you will need at *least* these specialized packages in addition to the usual gcc and g++ compiler.

Download the source package from the OpenSUSE build service at

`https://build.opensuse.org/package/show?package=VarGibbs&project=home%3Adrgweber`

or from my personal webpage

`https://sites.google.com/site/geraldweberufmg/vargibbs`

Typically the package is called something like `vargibbs-3.1.tar.bz2` After unpacking the `tar` package

```
tar -xvjf vargibbs-3.1.tar.bz2
```

change into the unpacked folder

```
cd vargibbs-3.1
```

if all necessary packages are available you should try to compile using the `make` command

```
make
```

If the compilation is successful, you should see something like

```
g++ -std=c++11 -DBUILDTIME=\"2019-03-22T18:43:15+00:00\" -DSVNVERSION=\"2.2\" -DLINUXSYSTEM=\"\" -g -O3
```

and nothing else, that is it! This generates the binary file `vargibbs`, which you may copy to your main installation at `/usr/bin` (you will need root permission) or into your local folder `/home/user/bin` (replace `user` with you actual user name).

2.4 Additional software

Although not required for vargibbs to run, you may find useful to install GRI which is a scientific graphing language which can be found at

`http://gri.sourceforge.net/`

note that some Linux distributions include GRI already, for OpenSUSE try

```
zypper install gri
```

2.5 If things go wrong

Most problems will come from missing library packages or from erroneous usage of your system. It is not possible for me to cover everything that may go wrong, so please feel free to contact me. Please include a detailed description of error messages, which system you are using and a step by step description of what you tried to do. Please understand that I will need as much information as possible. I can do nothing with messages simply saying "VarGibbs is not working on Ubuntu".

2.5.1 If you don't have a Linux system

You may copy an OpenSUSE Live system on a CD or onto a USB memory stick and reboot your computer with this system. If you do, proceed as you were using a normal OpenSUSE 15.1 system, following the instructions above. See installation instructions for Live OpenSUSE at `http://en.opensuse.org/Live_USB_stick`

2.5.2 License

This software is published under the GNU General Public License version 3 (GPLv3), the complete text of this license can be found in the documentation folder. If you wish to use this software under a different license or wish to make changes to the software without distributing it under the GPLv3 please contact me so that we can arrange for a specific license.

3 Usage

VarGibbs takes all its program option from the command line, therefore you should invoke the binary `vargibbs` with some of the arguments which are described below. Example shell scripts are provided and I recommend you study them as they are the best source to illustrate how to use VarGibbs. The following section details every available program option for VarGibbs.

In this section we provide all options for VarGibbs. Please note that some options will only be used in conjunction with other options. Please see 5 on how to obtain new parameters.

3.1 General options

3.1.1 -o=<basename>

Specifies the output basename, that is, all files which are generated start with `basename`

3.1.2 -par=<filename>

Specifies the input parameter file name. This can be a list of files `-par=file1.par,file2.par,file3.par`. In case of multiple specifications of the same model parameter, the last one supersedes previous parameters. For example if `file1.par` has the parameter `AT_AT:enthalpy -10.6708` and `file3.par` has `AT_AT:enthalpy -10.4256`, the final value for `AT_AT:enthalpy` will be -10.4256.

3.1.3 -nnpo=<0 or 1>

NN parameters only, if set to 1 only NN parameters are used. All initialization, symmetry and terminal parameters will be ignored. This option is useful if one wishes to check energies for specific sequences using only NN parameters. See also section 6.

(default) `-parid=0` use all parameters specified in `-par=<filename>` (3.1.2)

3.1.4 -parid=<identifier>

this is an identification string which is added to newly generated parameter files. Its only function is to help with the organization of your files.

(default) `-parid=vargibbs`

3.1.5 -data=<filename>

specifies the input file containing nucleotide sequences and melting temperatures. VarGibbs was not created with very large datasets in mind, and all sequences are loaded into memory. A safe limit is of the order of 50000 sequences which will require about 1GB of memory to run.

The file format is composed of columns

data/D-OW04-69.dat

```
temperature
ATCAATCATA TAGTTAGTAT 21.3 69 2
TTGTAGTCAT AACATCAGTA 24.7 69 2
GAAATGAAAG CTTTACTTTC 22.1 69 2
```

The first column is the sequence (from $5' \rightarrow 3'$) and the second column is the secondary strand (from $3' \rightarrow 5'$). The third column is the melting temperature in °C, the fourth column is the salt concentration (in mM or mol/L). The last column the species concentration (in μM). The secondary strand does not necessarily need to

be the complementary of the main strand as long as there are parameters for the mismatched pairs. If VarGibbs fails to find parameters for your sequences it will complain loudly.

3.1.6 -duplextype=<DNA or RNA>

Selects the type of duplex we should expect, this is important for selecting the base pair complementarity. Note that IUPAC codes cannot be used since we need to know which nucleotide parameters to use, that is, we cannot use N for example since we would not know which parameters to use.

(default) `-duplextype=DNA` , will expect A, C, G and T base pairs and will consider A complementary to T and C complementary to G.

`-duplextype=RNA` , same as for DNA but considers A and U as complementaries [4].

3.1.7 -rs=<seed>

Sets the seed of the random number generator (C function `srand`) which is used to modify the dataset. If you use the same seed you will get exactly the same random modifications.

3.1.8 -ct=<list of species concentration>

Specifies the species concentration C_t given in 10^{-6} mol/L (or μM). Usually it is not necessary to provide this as this value should be contained in the dataset file, unless of course you would like a different concentration from what is given in the file. For files containing symmetry adjusted species concentration you may prefer to use `-mct` (3.1.9).
For example, for $C_t = 400$ μM you would write

```
-ct=400
```

You may also specify a list of concentrations

```
-ct=100,200,400
```

3.1.9 -mct=<factor>

Multiply the species concentration of all sequences by a certain factor. This is useful for files containing symmetry adjusted melting temperatures. For example for files where the concentration is 100 μM for self-complementary sequences and 400 μM otherwise.

3.1.10 -dict=<list of nucleotides>

When using non-canonical nucleotides, for example inosine, you must tell VarGibbs which characters to use in addition to its usual dictionary of A,T,C,G and U. You should also tell VarGibbs if there is a complementary pair to this new nucleotide. If there is none, simply repeat the character. In the following example we are adding the letter I for inosine

```
-dict=I:I
```

here we are saying: consider I and its complementary I as new letters. If you wish to add two or more new letters simply make a comma-separated list,

```
-dict=I:I,J:J
```

but do not add blank spaces after the comma.

3.1.11 -calc=<type of calculation>

This sets the type of calculation. See section 5 for details on how to perform minimizations.

(default) `-calc=prediction` performs melting temperature predictions for given set of parameters.

`-calc=minlocal` performs one local minimization

`-calc=minglobal` performs several local minimizations by varying either the initial variables or by varying the dataset. This option may take a *very* long time to run depending on the stop.

`-calc=nncheck` no calculations are performed, the sequence data file is thoroughly analysed and each sequence is completely broken down into its constituent base pairs and nearest neighbours. This option works for complete data files `-data` (3.1.5) or with single sequences `-seq` (3.2.1). A file with extension `.nncheck` is created.

3.2 Specific options for temperature prediction (`-calc=prediction`)

Instead of providing a data file containing sequences you may wish to specify a single sequence for predicting its melting temperature. This is achieved with the options `-seq` (3.2.1) and optionally `-cseq` (3.2.2). In this case you may wish to adjust also the value of species concentration with `-ct` (3.1.8).

3.2.1 `-seq=<nucleotide sequence>`

You should give the main strand from 5′ to 3′, the complementary sequence will be worked out automatically (see also `-dict` (3.1.10) for non-canonical nucleotides).

(example 1) `-seq=ACGTTGAATT`

(example 2) `"-seq=d(ACGTTGAATT)"` for hybrid DNA-RNA it is necessary to specially format the sequence. Also, due to the parenthesis, it is necessary to enclose the option in double quotes, otherwise the `bash` command-line interpreter will throw an error. See also 8.2.

3.2.2 `-cseq=<nucleotide sequence>`

You should provide a 3′ → 5′ sequence if your sequence is not perfectly complementary, say like in a sequence with nucleotide mismatches. Note that you also need to provide the necessary parameters for this.

(example 1) `-cseq=TGCTACTTAA` from 3′ → 5′

(example 2) `"-seq=r(ACGUUGAAUU)"` for hybrid DNA-RNA it is necessary to specially format the sequence. Also, due to the parenthesis, it is necessary to enclose the option in double quotes, otherwise the `bash` command-line interpreter will throw an error. See also 8.2.

3.3 Options for minimization (`-calc=minlocal` or `-calc=minglobal`)

3.3.1 `-minmethod=<GSL or GBC>`

This selects which algorithmic implementation of multidimensional minimization to use. Currently GSL (GNU Scientific Library) produces better results.

(default) `-minmethod=GSL` GNU Scientific Library functions

`-minmethod=GBC` C++ textbook-like implementation

3.3.2 `-var=<list of parameters>`

The list of parameters for the minimization is given as in the following example

```
-var=AT_AT:entropy,AT_CG:entropy,AT_GC:entropy,AT_TA:entropy
```

you should use the same notation as in your `.par` file. See section 6 for a full description of available nearest neighbour parameters.

Associate variables You can associate variables, that is, you can ensure that different variables are treated as if they were the same. For this simply assign one variable to the other with an equal sign as in the following example

```
-var=AT_AT:entropy=AT_TA:entropy,AT_CG:entropy=AT_GC:entropy
```

here `AT_AT:entropy` and `AT_TA:entropy` will be take the same values, and so will `AT_CG:entropy` and `AT_GC:entropy`. Effectively you would be reducing the number of variables to minimize from 4 to just 2, which accelerates the calculation.

You can associate as many variables as you like, simply concatenate them with the equal sign. In the following example all variables will be treated as the same

```
-var=AT_AT:entropy=AT_TA:entropy=AT_CG:entropy=AT_GC:entropy
```

effectively reducing them to a single variable.

Trick If your variable list is very long, you can use this `bash` trick to read in your list from a file

```
-var=$(cat my.list)
```

where `my.list` is the file containing your variables.

3.3.3 -limit=<numerical value>

Specifies when to give up minimizing. When your local minimization fails to improve with a value larger than `-limit` the minimization should stop. If you specify a very small value you minimization may take a very long time to complete. Note that the minimization will stop earlier depending on the value of `-me` (3.3.5).

(default) `-limit=0.01`

3.3.4 -cl=<numerical value>

This is the characteristic length used in the multidimensional minimization. Essentially, a larger `-cl` (3.3.4) means that a wider region will be covered for minimization and potentially avoiding local minima. On the other hand, a larger value will also mean longer processing times. The exact value is not critical but should typically be of the order of magnitude of the variables listed in `-var` (3.3.2). For a better understanding of this parameter please see numerical computation text book on downhill simplex methods [11].

(default) `-cl=10`

3.3.5 -me=<integer value>

Maximum consecutive evaluations without improvement of χ^2 per minimization round. If this number is reached before the limit `-limit` (3.3.3), the minimization will stop. In other words, if after `-me` evaluations there was no change in χ^2, VarGibbs stops trying. The purpose of this option is to avoid having VarGibbs run forever.

(default) `-me=<a really large integer>` in other words: do not stop ever. Implemented by C++ method `std::numeric_limits<int>::max()` which provides the largest possible integer for the current system architecture.

3.3.6 -mm=<integer value>

This controls the number of minimization steps, see Figure 5.1. Each round stops when either `-limit` (3.3.3) or `-me` (3.3.5) is reached. The minimization then immediately starts over, using the previous optimized parameters as new initial variables.

(default) `-mm=2` , in other words, do at least one round.

3.3.7 -zlog=<0 or 1>

If set, log files will be written gzip compressed. Other options to control log file size are `-minloglevel` (3.3.8) and `-loginterval` (3.3.9), you may also simply turn off the generation of logfiles with `-gn=0` (3.4.7).

(default) `-zlog=0` , no log compression

`-zlog=1` , all log filed will end in `.gz` and can be handled with common linux commands such as `zcat` (instead of `cat`), `zgrep` (instead of `grep`) etc.

3.3.8 -minloglevel=<progress, full or terse>

(Introduced in version 3.1) Controls the amount of information in log files. During local or global minimization the log files can grow to very large sizes, especially for large values of `-me` (3.3.5).

(default) `-minloglevel=progress` records a new entry in the long file only when there is an improvement in χ^2. This can be further controlled with option `-loginterval` (3.3.9).

`-minloglevel=full` records every minimization attempt, even when there was no change to χ^2. This was the default behaviour up to version 3.0.

`-minloglevel=terse` only records the last χ^2 of each minimization round (rounds are controlled by `-mm` (3.3.6)).

3.3.9 -loginterval=<time in seconds>

(Introduced in version 3.1) Specifies the minimum interval of time between recordings in log files for option `-minloglevel=progress` (3.3.8). For example `-loginterval=3600` will record a new entry roughly every hour.

(default) `-loginterval=0` records continuously.

3.4 Options for global minimization (-calc=minglobal) only

3.4.1 -ee=<experimental error>

This sets the experimental error (in °C) of the melting temperature set. If given, the data set (provided through the `-data` (3.1.5) option) with be modified by small positive or negative amounts such that the standard deviation is close to the experimental error. The purpose of this is to simulate the effect of experimental uncertainty on the parameters. We do not recommend to set this to a very large value as it may start rearranging the order of sequence stability and may give really poor results. A reasonable interval is typically lower than 1 °C. See also `-randomizedataset` (3.4.2).

(example) `-ee=0.5` will modify the dataset to within 0.5 °C

3.4.2 -randomizedataset=<0 or 1>

When `-randomizedataset=1` is selected the randomization of the dataset is activated. The dataset given with `-data=<filename>` will be changed by small amounts such that the standard deviation from the original dataset equals `-ee=<experimental error>` (3.4.1). See section 5 for details.

(default) `-randomizedataset=0` do not modify dataset

3.4.3 -randominitial=<fraction>

When given the initial values of the variables specified in the file given in `-par` (3.1.2) will be randomly modified. For example

```
-randominitial=0.2
```

means that all initial values will be modified within ±20% every time a new local minimization starts. For example if your parameter file contains

```
GC_CG:enthalpy    -11.1
```

and this parameter is listed in `-var` (3.3.2) a value of `-randominitial=0.2` (3.4.3) will randomly chose a new initial value for `GC_CG:enthalpy` within -8.88 and -13.32. This is applied to all variables listed in `-var` (3.3.2), but only to those variables.

3.4.4 -mrr=<integer value>

Controls how many complete rounds of local minimizations are performed.

(default) `-mrr=10` which is a relatively small number and should run in a couple of minutes. A more thorough search should use values larger than 100 which may take several hours to run.

3.4.5 -adl=<integer value>

Stops the minimization if the limit for total average difference given in `-adl` is reached. VarGibbs will compare the results of the current minimization with the previous one and compute de average differences of the parameters. A value such as `-adl=0.1` means that the minimization will stop if on average the differences between the previous parameters and the new parameters is less than 1%.

3.4.6 -asl=<integer value>

Similar to `-adl` (3.4.5) but considers the standard deviation instead of the average difference. In this sense `-asl=0.1` is a much more stricter stop criteria than `-adl=0.1` (3.4.5).

3.4.7 -gn=<0 or 1>

Controls the generation of files for `-calc=minglobal` (3.1.11). Global minimization generates a very large number of files, easily reaching several Gb. If you are not interested in those files you may turn this option off `-gn=0`. Note that if afterwards you change your mind you may simply run again and get exactly the same results as long as you use the same seed for the random number generator `-rs` (3.1.7). Alternatively, if disk space is a problem you may consider using the option `-zlog=1` (3.3.7) instead, to compress log files up to a factor of 10. Other options that control the amount of information written are `-minloglevel` (3.3.8) and `-loginterval` (3.3.9).

(default) `-gn=1` , generate all files.

`-gn=0` , suppress log file generation.

3.4.8 -printusedpar=<0 or 1>

Controls the printing of a file with extension `.usedpar.` that will output all parameters as they were read from files given in `-par` (3.1.2). See section 5.5 for the output format.

(default) `-printusedpar=0` does not generate a file

`-printusedpar=1` generates a file with extension `.usedpar.`, taking the base name from `-o` (3.1.1)

3.5 Options for salt correction

Experimental implementation of salt correction schemes (introduced in version 2.1) for the melting temperatures T_m. Please note that these options may change in the future.

3.5.1 On which T_m will the correction be applied?

Depending on the argument of the `-calc` (3.1.11) option the correction may be applied to the input T_m or to the predicted T_m.

`-calc=nncheck` the salt correction will be applied to the input T_m from the sequence files. The output files will contain the adjusted T_m. The new file will be in the format discussed in chapter 8 and can be used as input file again for other calculations.

`-calc=prediction` **and** `-predsaltcorr=0` first apply the salt correction to the input T_m and place them in the `temperature.adjusted` column, see section 4.1.1. Afterwards calculate the T_m from the given parameters **without** applying any salt correction. This allows you to compare adjusted and predicted temperatures in one file.

`-calc=prediction` **and** `-predsaltcorr=1` first calculate the T_m from the given parameters and **afterwards** apply the salt correction.

3.5.2 -seqsalt=<salt concentration in mM>

In cases when a single sequence was provided through the option `-seq` (3.2.1) you may also provide its salt concentration.

(example) `-seqsalt=121 -seq=CACGGCUC` will set the [Na^+] salt concentration of sequence `CACGGCUC` to 121 mM.

3.5.3 How to specify the salt concentration of the parameter set

In previous versions (before 2.1) it was not possible to specify the salt concentration for the parameter (`-par` (3.1.2)). Now, you may manually edit the parameter file and add the new parameter `Na+:concentration` (in mM), for example, add a new line with the salt concentration for these specific parameters:

```
xia98-original-table4
Na+:concentration 1000
AU_AU:enthalpy    -6.82
AU_AU:entropy     -19.0
AU_UA:enthalpy    -9.38
AU_UA:entropy     -26.7
```

Note: as of version 2.1 all parameter files in the TfReg package were edited to include the `Na+:concentration` parameters.

3.5.4 How to specify the new salt concentration?

The [Na^+] target salt concentration (in mM) that should be calculated.

`-targetsalt=<Na+ salt concentration in mM>`

(example) `-targetsalt=71` will set the [Na^+] target salt concentration at 71 mM.

3.5.5 -saltscheme=<scheme code>

Select specific salt correction schemes, please check the references for the actual equations.

For DNA

`-saltscheme=schildkraut65` from Ref. 12

`-saltscheme=owczarzy04eq19` Eq. (19) from Ref. 13

`-saltscheme=owczarzy04eq20` Eq. (19) from Ref. 13

`-saltscheme=owczarzy04eq21` Eq. (19) from Ref. 13

`-saltscheme=owczarzy04eq22` Eq. (19) from Ref. 13

For RNA

`-saltscheme=chen13eq19` Eq. (19) from Ref. 14

`-saltscheme=chen13eq20` Eq. (20) from Ref. 14

`-saltscheme=chen13eq21` Eq. (21) from Ref. 14

`-saltscheme=chen13eq22` Eq. (22) from Ref. 14

3.5.6 Examples involving salt concentrations

Adjusting all temperatures in a dataset, generating a new dataset

Problem: you have a dataset of melting temperatures for a specific salt concentration and wish to generate a new file with all those sequences but at a different salt concentration.

In the following example, we have a dataset `D-SL96.dat` where all temperatures are at $[Na^+]$ 1000 mM. We wish to convert them to of 71 mM using the salt correction eq. (22) from Ref. 13.

```
vargibbs -data=D-SL96.dat -calc=nncheck -targetsalt=71 -saltscheme=owczarzy04eq22 -o=CORR
```

This generates a new file `CORR.dat` that has exactly the same format and sequences as `D-SL96.dat` but with corrected temperatures to the new salt concentration.

Predict temperatures for a given parameter set and apply the salt concentration afterwards

Problem: you have a parameter set for a specific salt concentration but need to predict temperatures at a different concentration.

In the following example, we calculate the melting temperature prediction at 71 mM for all sequences in file `D-SL96.dat`, using the parameters `P-XIA98.par` (valid for 1000 mM), and then applying the salt correction eq. (22) from Ref. 13.

```
vargibbs -data=D-SL96.dat -calc=prediction -par=P-XIA98.par -predsaltcorr=1 -targetsalt=71 \
-saltscheme=owczarzy04eq22 -o=CORR
```

Note that the file `P-XIA98.par` already has the new parameter `Na+:concentration` that tells the system that those parameters were obtained for a specific salt concentration.

4 Simple melting temperature prediction -calc=prediction

This is the simplest application of VarGibbs. In the following example we show how to obtain the melting temperatures from the SantaLucia paper Ref. [15] with all sequences from the same paper:

examples/D-SL98-P-SL98.sh

```
vargibbs -calc=prediction -o=./D-SL98-P-SL98 -par=../data/P-SL98.par \
-data=../data/D-SL98.dat
```

Instead of a data file with sequences you may simply provide the sequence info directly in the command line

examples/sequence-AOP-CMB.sh

```
vargibbs -o=./sequence-AOP-CMB -par=../data/AOP-CMB.par \
-calc=prediction -v=1 -seq=ACGTTTTAAGGTTC -cseq=TGCAAAATTCCAAG -ct=200
```

All results are collected in the file with extension `.dat` described in the next section.

4.1 Files generated for prediction

4.1.1 Extension .dat sequence information and results

The file with extension `.dat` shows all sequences (format of this file is identical with extension `.ver` of local/global minimization).

examples/verify/D-SL98-P-SL98/D-SL98-P-SL98.dat

```
Main/Complementary alpha salt_concentration species_concentration temperature.measured ...
CCGG/GGCC 1 1000 100 16.6 16.6 12.8709 0 0 -26.4 0 0 -74 0 0 0 0 0
CGCG/GCGC 1 1000 100 23.7 23.7 22.7167 0 0 -30.8 0 0 -85.8 0 0 0 0 0
AGCCG/TCGGC 4 1000 400 35.6 35.6 35.5231 0 0 -33.8 0 0 -91.2 0 0 0 0 0
CACAG/GTGTC 4 1000 400 19.1 19.1 19.6603 0 0 -33 0 0 -94.4 0 0 0 0 0
ACCGCA/TGGCGT 4 1000 400 43.1 43.1 48.0789 0 0 -40.7 0 0 -108.4 0 0 0 0 0
AGTTGC/TCAACG 4 1000 400 37.3 37.3 35.2518 0 0 -40 0 0 -111.4 0 0 0 0 0
ATGCGC/TACGCG 4 1000 400 44.5 44.5 46.4657 0 0 -43.5 0 0 -117.8 0 0 0 0 0
CCAACG/GGTTGC 4 1000 400 37.7 37.7 39.2123 0 0 -43.2 0 0 -120 0 0 0 0 0
CCGCGG/GGCGCC 1 1000 100 55.2 55.2 52.0737 0 0 -46.8 0 0 -125.6 0 0 0 0 0
```

The first column are the sequences, the second column indicates if the sequence is self-complementary (4) or not (1). Then comes the salt concentration (1000) in mM, the species concentration (100 or 400) in μM, the measured melting temperatures (columns 5 and 6), the predicted melting temperature (column 7). The predicted total enthalpy and total entropy variations are given in columns 10 and 13, respectively. The remaining columns (all with null values) are not used in the context of VarGibbs and maintained for compatibility with other programs.

For the case when a single sequence was given this file will look like this

examples/verify/sequence-AOP-CMB/sequence-AOP-CMB.dat

```
Main/Complementary alpha salt_concentration species_concentration temperature.measured ...
ACGTTTTAAGGTTC/TGCAAAATTCCAAG 4 0 200 0 0 65.0278 0 0 -90.868 0 0 -249.021 0 0 0 ...
```

4.1.2 Extension .ver quality of the prediction and verification

examples/verify/D-SL98-P-SL98/D-SL98-P-SL98.ver

```
average diff_deviation sqr_diff sqrt_diff2 relative_sqr_diff N
Tm 1.88066 1.57065 648.413 2.45027 0.345513 108
```

where the first number is the average difference in melting temperature prediction

$$\langle \Delta T \rangle = \frac{1}{N} \sum_{i=1}^{N} |\Delta T_i| \,, \tag{4.1}$$

the second is the standard deviation of ΔT_i,

$$\delta(\Delta T) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\Delta T_i - \langle \Delta T \rangle\right]^2} \tag{4.2}$$

the third is χ^2 given by Eq. (5.1),

$$\chi^2 = \sum_i \Delta T_i^2$$

the fourth is

$$\langle \Delta T \rangle_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\Delta T_i\right]^2} = \sqrt{\frac{\chi^2}{N}}. \tag{4.3}$$

and the last the relative squared difference

$$\left\langle \frac{\Delta T}{T} \right\rangle_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{\left[\Delta T_i\right]^2}{T_i}}, \tag{4.4}$$

4.1.3 Recalculating temperatures for different species concentrations

Sometimes you may need melting temperatures for different species concentration C_t. In this case the options `-ct` (3.1.8) and `-mct` (3.1.9) may be useful. The option `-ct` (3.1.8) defines a new C_t common to all sequences, while `-mct` (3.1.9) multiplies C_t by a certain factor. The second option is useful for files containing symmetry adjusted melting temperatures, where self-complementary sequences are 1/4 of the C_t of other sequences.

5 Obtaining parameters from minimization

5.1 Why would you want to do this?

You may wish to apply VarGibbs in the following practical situations:

1. You have a number of melting temperatures, measured at several species concentrations and you need the best nearest-neighbour enthalpies and entropies.
2. You have a number of melting temperatures, and you do not have the total entropies and enthalpies and you need the best nearest-neighbour enthalpies and entropies.
3. You already have the nearest-neighbour enthalpies and entropies from other methods, but wish to obtain a new set with hopefully better predictive capabilities.
4. You already have the nearest-neighbour enthalpies and entropies from other methods and would like to estimate the uncertainty of these parameters.

5.2 What is the idea?

The basic idea behind the implementation of VarGibbs is to minimize

$$\chi^2 = \sum_i \left[T_i - T_i'(P_k)\right]^2 \tag{5.1}$$

where T_i are the measured melting temperatures for the ith sequence, and T_i' are the predicted temperatures according a set of parameters P_k. We start with an initial set of parameters P_0 and let the parameters P_k vary until we minimize as much as possible χ^2. After N steps of minimizations the resulting set P_N represent the parameters which best reproduce the melting temperatures for model M.

5.3 Global and local minimizations

VarGibbs knows about two types of minimizations, global and local. In the following sections we will describe these in details, but essentially a global minimization is a number of local minimization. We will first describe the local minimization which is easier to understand and introduce the main concepts. Figure 5.1 shows an overview of the two types on minimizations and some of the main options related to each step.

5.4 Local minimization (`-cal=minlocal`)

The local minimization is to use the unmodified T_i experimental melting temperatures and a set of initial parameters P_0 which are read from the parameter file given by `-par` (3.1.2). An example command line would be

```
vargibbs -var=AT_AT:entropy,AT_CG:entropy,AT_GC:entropy,AT_TA:entropy -data=experimental.dat \
-calc=minlocal -o=results -par=test.par
```

The quality of results will depend very much on your initial guess of variables in file `test.par`. The reason for this are local minima, that is the function χ^2 has numerous minima and your minimization may get "stuck" at non-optimal values. One solution, which *may* yield acceptable results is to carefully select generic initial parameters. In the following example this is achieved by a parameter file like this

data/init_s98.par

```
1 init-s98
2 *:entropy     -23
3 *:enthalpy    -8
```

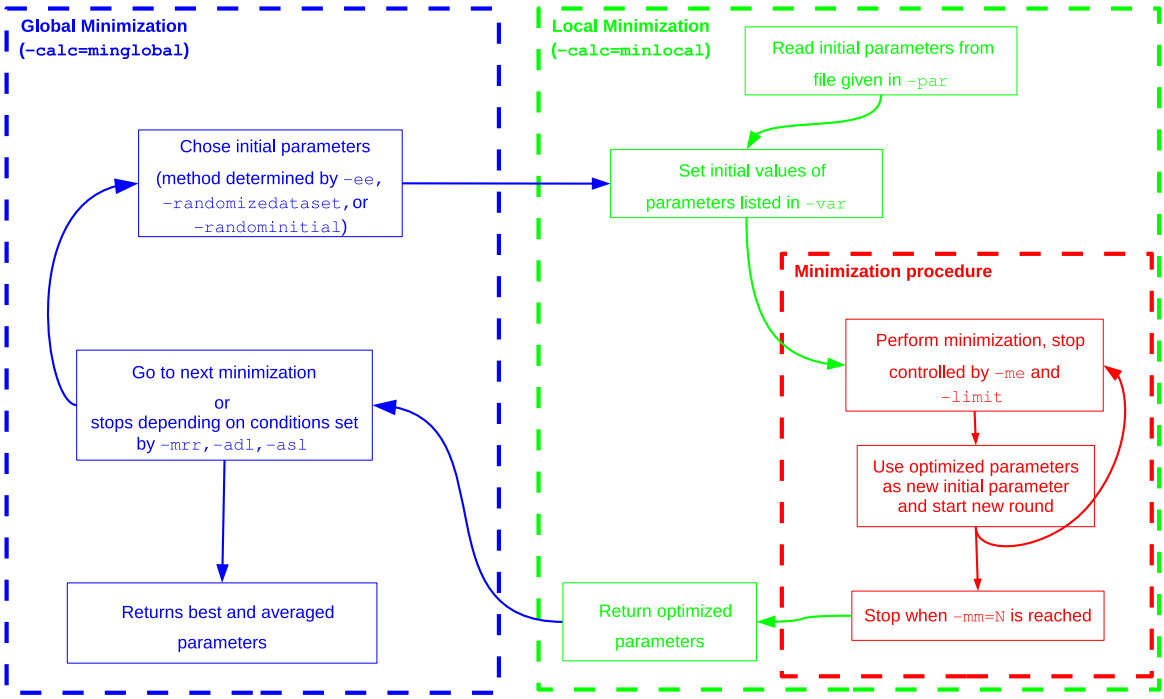

Figure 5.1
Workflow and dependencies of global and local minimizations in VarGibbs.

```
4 terminal_CG_enthalpy 2.0
5 terminal_CG_entropy -5.0
6 terminal_AT_enthalpy 5.0
7 terminal_AT_entropy 10.0
8 symmetry_correction_entropy -1.0
```

where we selected an entropy of -23 and an enthalpy of -8 for any nearest-neighbour context.

The complete example script, which should take about 20 s to run, is

examples/SL98-local.sh

```
1 vargibbs -cl=10 -me=3000 -minmethod=gsl -mm=3 -o=./SL98-local \
2 -par=../data/init_s98.par \
3 -parid=test -calc=minlocal -data=../data/D-SL98.dat \
4 -var=AT_AT:entropy,AT_CG:entropy,AT_GC:entropy,AT_TA:entropy,CG_AT:entropy,CG_CG:entropy,...
5
6 vargibbs -o=./SL98-local-prediction -par=./SL98-local.par \
7 -calc=prediction -data=../data/D-SL98.dat
```

lines 6–7 calculate the prediction from the optimised parameter file `SL98-local.par` which has the following quality of the prediction file

examples/verify/SL98-local/SL98-local-prediction.ver

```
1 average diff_deviation sqr_diff sqrt_diff2 relative_sqr_diff N
2 Tm 1.50292 1.1753 393.129 1.9079 0.219436 108
```

see Sec. 4.1.2 for details about these numbers.

5.4.1 Files generated for local minimization

Extension `.dat`: melting temperatures and melting index results

One of the main result files has the extension `.dat` and contains the optimized parameters. However thes files are not easy to read, you may prefer to inspect the parameter file `.par` instead.

Here is an output of the previous example:

examples/verify/SL98-local/SL98-local.dat

```
1 393.129 -7.74456 -21.9304 -10.2102 -28.7311 -6.27733 -16.4984 -3.29831 -8.39649 ...
```

The first number is the final χ^2, the remaining numbers are the optimized parameters which were given in `-var` (3.3.2) (in lexical order).

Extension .par: reusable parameter file

The optimized parameters are also stored in a file with extension `.par` in addition to initial generic parameters

examples/verify/SL98-local/SL98-local.par

```
test
#parameter map has 27 elements
*:enthalpy -8 [-8.000000:/home/ge/svn/bioinf/distribution/var-gibbs/data/init_s98.par] (10)
*:entropy -23 [-23.000000:/home/ge/svn/bioinf/distribution/var-gibbs/data/init_s98.par] (10)
AT_AT:enthalpy -7.74456 [* -8.000000:new] (18842)
AT_AT:entropy -21.9304 [* -23.000000:new] (18842)
AT_CG:enthalpy -10.2102 [* -8.000000:new] (18842)
AT_CG:entropy -28.7311 [* -23.000000:new] (18842)
AT_GC:enthalpy -6.27733 [* -8.000000:new] (18842)
AT_GC:entropy -16.4984 [* -23.000000:new] (18842)
AT_TA:enthalpy -3.29831 [* -8.000000:new] (18842)
AT_TA:entropy -8.39649 [* -23.000000:new] (18842)
CG_AT:enthalpy -11.1112 [* -8.000000:new] (18842)
CG_AT:entropy -31.3799 [* -23.000000:new] (18842)
CG_CG:enthalpy -8.56919 [* -8.000000:new] (18842)
CG_CG:entropy -21.8905 [* -23.000000:new] (18842)
CG_GC:enthalpy -9.64746 [* -8.000000:new] (18842)
CG_GC:entropy -24.6633 [* -23.000000:new] (18842)
GC_AT:enthalpy -8.04907 [* -8.000000:new] (18842)
GC_AT:entropy -21.867 [* -23.000000:new] (18842)
GC_CG:enthalpy -11.626 [* -8.000000:new] (18842)
GC_CG:entropy -30.7114 [* -23.000000:new] (18842)
TA_AT:enthalpy -3.04064 [* -8.000000:new] (18842)
TA_AT:entropy -8.19177 [* -23.000000:new] (18842)
symmetry_correction_entropy -0.292102 [* -1.000000:/home/ge/svn/bioinf/distribution/var-gibbs/data/init
terminal_AT_enthalpy 2.17672 [* 5.000000:/home/ge/svn/bioinf/distribution/var-gibbs/data/init_s98.par]
terminal_AT_entropy 4.50153 [* 10.000000:/home/ge/svn/bioinf/distribution/var-gibbs/data/init_s98.par]
terminal_CG_enthalpy 1.79407 [* 2.000000:/home/ge/svn/bioinf/distribution/var-gibbs/data/init_s98.par]
terminal_CG_entropy 3.35923 [* -5.000000:/home/ge/svn/bioinf/distribution/var-gibbs/data/init_s98.par]

```

This parameter file can be reused in other minimization rounds or for the global minimizations discussed in the next section.

Extension .log: minimization history

The file `.log` records the complete minimization history and therefore is quite extensive.

examples/verify/SL98-local/SL98-local.log

```
797 567.707 -8.08438 -22.9703 -8.07911 -21.702 -8.22099 -22.2126 -8.05926 -22.9261 ...
818 567.254 -8.08949 -22.9658 -8.07179 -21.6844 -8.21524 -22.2327 -8.06465 -22.9304 ...
837 566.363 -8.09293 -22.9592 -8.07295 -21.6882 -8.21894 -22.23 -8.06101 -22.9301 ...
839 565.206 -8.08707 -22.9587 -8.06474 -21.6733 -8.20445 -22.2494 -8.08262 -22.942 ...
858 564.95 -8.07877 -22.9394 -8.07131 -21.6893 -8.2085 -22.2435 -8.06877 -22.8949 ...
865 563.94 -8.06515 -22.9027 -8.07567 -21.6856 -8.20816 -22.2259 -8.06176 -22.9513 ...
867 563.856 -8.07958 -22.957 -8.07794 -21.6904 -8.20997 -22.2364 -8.0686 -22.9 -8.01153 ...
875 563.229 -8.07775 -22.9298 -8.07224 -21.6908 -8.21368 -22.2364 -8.06419 -22.9185 ...
884 562.996 -8.084 -22.9461 -8.07244 -21.6799 -8.21725 -22.2316 -8.05396 -22.9258 ...
886 562.941 -8.07068 -22.9332 -8.07174 -21.6784 -8.20953 -22.2434 -8.06674 -22.9111 ...
889 562.528 -8.07509 -22.9325 -8.07104 -21.6809 -8.20553 -22.2348 -8.06534 -22.9174 ...
```

The first column is the number of evaluation steps which stops at a maximum defined by `-mm` (3.3.6). The remaining columns are as in the file with extension `.dat`. Please note that the file ending with `.log.best` is always empty.

Extension .ver quality of the prediction and verification

The file with extension `.ver` shows all sequences .

examples/SL98-local/SL98-local.ver

```
TAGATCTA/ATCTAGAT 1 1000 100 33.4 33.4 29.9888 0 0 -33.6789 0 0 -92.7998 0 0 0 0 ...
TAGGCCTA/ATCCGGAT 1 1000 100 51.9 51.9 49.3886 0 0 -43.0468 0 0 -115.162 0 0 0 0 ...
TATGCATA/ATACGTAT 1 1000 100 37.8 37.8 36.4028 0 0 -42.1727 0 0 -117.937 0 0 0 0 ...
TCTATAGA/AGATATCT 1 1000 100 28.1 28.1 29.9888 0 0 -33.6789 0 0 -92.7998 0 0 0 0 ...
TGAGCTCA/ACTCGAGT 1 1000 100 50.1 50.1 50.2665 0 0 -58.1476 0 0 -161.491 0 0 0 0 ...
AAAAAAAAA/TTTTTTTTT 4 1000 400 39.4 39.4 38.6545 0 0 -57.603 0 0 -166.44 0 0 0 0 ...
ATAACTGGC/TATTGACCG 4 1000 400 54 54 51.8572 0 0 -57.9065 0 0 -159.869 0 0 0 0 0
ATCTATCCG/TAGATAGGC 4 1000 400 51.1 51.1 52.1312 0 0 -46.2586 0 0 -123.91 0 0 0 ...
CAAAAAAAG/GTTTTTTTC 4 1000 400 44.2 44.2 42.3168 0 0 -60.2677 0 0 -172.742 0 0 0 ...
CAAACAAAG/GTTTGTTTC 4 1000 400 47.3 47.3 45.7217 0 0 -66.0999 0 0 -188.992 0 0 0 ...
```

The first column are the sequences, the second column indicates if the sequence is self-complementary (4) or not (1). Then comes the salt concentration (1000) in mM, the species concentration (100 or 400) in μM, the measured melting temperatures (columns 5 and 6), the predicted melting temperature (column 7). The predicted total enthalpy and total entropy variations are given in columns 10 and 13, respectively. The remaining columns (all with null values) are not used in the context of VarGibbs and maintained for compatibility with other programs.

5.5 .usedpar summary of parameters read and used

If `-printusedpar=1` (3.4.8) is given, VarGibbs will generate an additional file with extension `.usedpar` which shows which parameters were read from which file, and how many times they were used.

The rules for files of type `.usedpar` are summarized here

```
parameter value [* value:FILE3 <> value:FILE2 = value:FILE1] (N)
```

The files are shown in reverse order from how they were read, that is, they were read in order `FILE1`, `FILE2` and last `FILE3`. The specific value of the parameter is shown in front of the file name `value:FILE3`. The signs indicate if the values were the same as of the previous file (`=`) or different (`<>`), and (`*`) means that the parameter was changed internally, after reading all files. The number of times this parameter was called internally is given by `N`, although this reflects only the number of times it was searched in the database and not the number of times it was employed in equations. However, it is accurate when displaying zero meaning it was really never used.

5.6 Global minimizations (-calc=minglobal)

One way to overcome the problem of local minima is to perform several local minimization. The option `-calc=minglobal` (3.1.11) automates this process, see Figure 5.1 for an overview and how local and global minimization relate to each other. There are two basic global minimization strategies, one which varies the initial parameters randomly and the other varies the experimental data by small random amounts.

5.6.1 Varying the initial parameters (-randominitial)

The idea here is to start the local minimization anew each time with a different set of initial parameters $P_{0,l}$, where l is the index for a given set of initial parameters. VarGibbs will then start the minimization by randomizing the initial parameters. In the next example we modified the example of the previous section by changing `-calc=minglobal` (3.1.11) and `-randominitial=0.2` (3.4.3), which modifies the initial parameters by up to 20%.

```
vargibbs -var=AT_AT:entropy,AT_CG:entropy,AT_GC:entropy,AT_TA:entropy -data=experimental.dat \
-calc=minglobal -o=results -randominitial=0.2
```

VarGibbs will carry on repeating the minimization over and over, every time for a new set of initial parameters $P_{0,l}$ until you tell VarGibbs to stop or until the maximum number of minimizations given by `-mrr` (3.4.4) is reached.

VarGibbs will show a standard output when variables are changed:

```
Warning:replacing AT_AT:enthalpy old value=-9.3522 new value=-8.41043
Warning:replacing AT_AT:entropy old value=-26.487 new value=-18.8472
```

A complete example which should take few minutes to run:

examples/SL96-global.sh

```
vargibbs -calc=minglobal -cl=10 -data=../data/D-SL96.dat -gn=1 \
-me=3000 -minmethod=gsl -mm=6 -mrr=10 \
-o=./BOP-SL96 -zlog=0 -par=../data/init_s96.par \
-parid=BOP-SL96 -randominitial=0.5 -randomizedataset=0 -rs=1234 \
-var=AT_AT:entropy,AT_CG:entropy,AT_GC:entropy,AT_TA:entropy,CG_AT:entropy,\
CG_CG:entropy,CG_GC:entropy,GC_AT:entropy,GC_CG:entropy,TA_AT:entropy,\
AT_AT:enthalpy,AT_CG:enthalpy,AT_GC:enthalpy,AT_TA:enthalpy,CG_AT:enthalpy,\
CG_CG:enthalpy,CG_GC:enthalpy,GC_AT:enthalpy,GC_CG:enthalpy,TA_AT:enthalpy,\
initiation_CG_entropy,initiation_AT_entropy,\
terminal_5TA3_enthalpy,symmetry_correction_entropy
```

5.6.2 Varying the experimental temperatures (-randomizedataset)

This type of minimization should be carried out once you have a fairly good set of optimized parameters, therefore we do not recommend to use this type of minimization as your first attempt. This step is mainly used to evaluate the influence of the reported experimental uncertainty over the calculated parameters.

Here, we use a fixed set of initial parameters P_0, as in the simple minimization, but for every round of minimization we slightly change the experimental melting temperatures T_i by random amounts ΔT_i. The standard deviation between the original set and the new artificial set will be close to the specified experimental error given in `-ee` (3.4.1).

examples/SL96-global.sh

```
vargibbs -asl=0.1 -calc=minglobal -cl=10 -data=../data/D-SL96.dat -gn=1 -ee=0.3 ...
-o=./AOP-SL96 -zlog=0 -par=./BOP-SL96.par.best -parid=AOP-SL96 \
-randomizedataset=1 -rs=1234 \
-var=AT_AT:entropy,AT_CG:entropy,AT_GC:entropy,AT_TA:entropy,CG_AT:entropy,\
CG_CG:entropy,CG_GC:entropy,GC_AT:entropy,GC_CG:entropy,TA_AT:entropy,\
AT_AT:enthalpy,AT_CG:enthalpy,AT_GC:enthalpy,AT_TA:enthalpy,CG_AT:enthalpy,\
CG_CG:enthalpy,CG_GC:enthalpy,GC_AT:enthalpy,GC_CG:enthalpy,TA_AT:enthalpy,\
initiation_CG_entropy,initiation_AT_entropy,\
terminal_5TA3_enthalpy,symmetry_correction_entropy
```

here we are using as initial parameters the best optimization

`AOP-SL96.par.best`

from the example in the previous section.

5.6.3 Files generated during global minimization

For global minimization you will get the same files for local minimizations listed in section 5.4.1 but appended by a number like `.dat.00013`.

list of files for multiple minimzation

```
BOP-SL96.dat.00003  BOP-SL96.par.00003        BOP-SL96.var.00003
BOP-SL96.log.00003  BOP-SL96.reg.00003        BOP-SL96.ver.00003
BOP-SL96.dat.00004  BOP-SL96.par.00004        BOP-SL96.var.00004
BOP-SL96.log.00004  BOP-SL96.reg.00004        BOP-SL96.ver.00004
```

The number indicates the local minimization round, otherwise these files are the same as for the simple minimization files described in section 5.4.1. The multiple minimization produces a very large number of files, therefore is it recommended to run in a separate folder. Alternatively, one may set `-gn=0` (3.4.7) which will suppress the output of the files appended by numbers. Other options to control the size of the log files are `-zlog` (3.3.7), `-minloglevel` (3.3.8) and `-loginterval` (3.3.9).

The additional extension `.best` are the results for the lowest χ^2 found, and extension `.par.average` contains the average parameter values. These files are updated at every round.

One file here is of special importance, the one ended with extension `.dat` will contain all results together.

examples/verify/SL96-global/BOP-SL96.dat

```
529.123 -7.78225 -22.0386 -6.69507 -17.386 -2.52987 -4.87858 -0.30404 1.3981 -8.31749 ...
526.166 -9.85229 -27.997 -9.83689 -26.2417 -5.23629 -12.4947 -3.739 -8.55061 -8.56265 ...
525.7 -8.34932 -23.7104 -8.19589 -21.7873 -3.99421 -9.25101 -3.63315 -8.67596 -7.08778 ...
522.633 -7.67022 -21.7605 -9.56615 -26.0226 -6.01695 -15.4755 -7.06485 -18.9828 ...
```

```
525.7 -9.56528 -27.1325 -10.0257 -26.703 -5.94954 -14.6551 -5.38924 -13.3807 -8.05901 ...
526.799 -9.14595 -26.0312 -8.13573 -21.4327 -3.76715 -8.42463 -2.46223 -5.02819 ...
525.65 -9.84787 -28.0093 -9.37822 -24.8516 -4.95642 -11.7463 -3.838 -8.88293 -7.93385 ...
525.32 -7.23913 -20.438 -8.23796 -21.9436 -5.172 -12.886 -4.8005 -11.9916 -5.97586 ...
526.163 -10.8941 -31.0525 -10.0609 -26.9002 -3.17534 -6.13705 -3.37308 -7.63218 ...
525.294 -9.30893 -26.348 -10.0388 -26.796 -5.78086 -14.1626 -4.97548 -12.1637 -7.78641 ...
```

Where are your best results?

Your best results, that is, those with smallest χ^2 are easily accessible by locating the files ending in `.best`. Please note that the file `.log.best` will be empty as this file is written after the minimization has ended.

example of list of files related to smallest χ^2

```
BOP-SL96.dat.best  BOP-SL96.par.best  BOP-SL96.var.best
BOP-SL96.log.best  BOP-SL96.reg.best  BOP-SL96.ver.best
```

Where are your average results?

It may be more sensible to work with average results as your may be overfitting your data. When using `-calc=minglobal` (3.1.11) the average values are listed in the files ending with `.average`.

5.6.4 How to stop VarGibbs when in `-calc=minglobal` mode?

If you did not specify the maximum rounds of minimization `-mrr` (3.4.4), `-asl` (3.4.6), or `-adl` (3.4.5), then VarGibbs will run non-stop. You may analyse your intermediate results at any time and then stop VarGibbs when you feel that you have gathered enough data. Of course, when this happen you may simply kill the process if you like. But a cleaner way to do this is to set the `.proceed` file to 0. Suppose your results file start all with `test` (`-o=test` (3.1.1))

```
echo 0 > test.proceed
```

before restarting the next minimization VarGibbs will check this file, if it is 0 it will then stop and cleanly close all files.

5.6.5 Work flow suggestion for global minimization

1 Get the melting temperatures either from a paper or format your own data according to section 3.1.5. Make sure you have the correct values for species concentration C_t in 10^{-6} mol/L (μM). You can mix data with different species concentrations, but not with different salt concentrations.

2 Define your initial parameters , ideally start the the most generic values possible, such as in file `data/gibbs-av.par`.

3 Run global minimizations with varying initial parameters . For this step follow the recommendations in section 5.6.1. Let it run for as many times as practically possible, but at least 100 times (2 h). It will take about two days if you let it run 1000. You may also monitor your results continuously as described in section 5.6.4, and then interrupt when you are satisfied.

4 Get the best parameters as described in section 5.6.3.

5 Run global minimizations to obtain error estimates as described in section 5.6.2. Use the best parameters from step 4 as initial parameters.

6 Use the averages obtained in the last step (see 5.6.3 on how to get them). They should represent the data you are looking for.

6 Model parameters

6.1 Naming scheme

6.1.1 Base pairs (BP)

Base pairs (BP) are simply given by two characters like `AT`, `CG` for DNA and `AU`, `CG` for RNA. Note that different non-canonical base pairs may be given such as `GU` or `AA`, this will work as long as there are parameters for them in your parameter file (see 6.3). If you wish to use different characters, use the `-dict` option (see 3.1.10), otherwise anything different from from ACGTU will be silently ignored.

Hybrid DNA-RNA an additional character is required, `d` for the DNA strand and `r` for RNA strand. For example `dCrG` means C on the DNA strand and G on the RNA strand. `VarGibbs` should be able to read a construct such as `dCdG` for DNA-DNA or `rCrG` for RNA-RNA, however this is generally unnecessary.

6.1.2 Nearest neighbours (NN)

Nearest neighbours (NN) are two sequential base pairs (BP) joined with an underscore character `_` as in `BP_BP`. For example AT followed by CG should be written as `AT_CG`. For hybrid DNA-RNA an example would be `dArU_dCrG`.

The convention used by `VargIbbs` is of write `AB_CD`, where `AB` is the first base pair and `CD` the second base pair such that

$$5' - \mathrm{AC} - 3'$$
$$3' - \mathrm{BD} - 5'$$

Symmetry reduction. Note that `DC_BA` is equivalent to `AB_CD`,

$$\begin{matrix} 5' - \mathrm{AC} - 3' \\ 3' - \mathrm{BD} - 5' \end{matrix} \longleftrightarrow \begin{matrix} 5' - \mathrm{DB} - 3' \\ 3' - \mathrm{CA} - 5' \end{matrix}$$

for example, `AT_AT` is the same as `TA_TA`. When specifying a NN sequence always specify in lexical ordering, that is write `AT_AT` and not `TA_TA`. If you do use `TA_TA` `VarGibbs` will silently ignore it.

6.2 Parameter types

The are basically two groups of parameters in the nearest neighbour model: context-dependent parameters and initiation parameters. The context-dependent parameters are the ones that give this model its name, they are dependent on the nearest neighbours. All other parameters are dependent on some specific features of a given sequence, for example a parameter which add some amount to the entropy if there is an AT base pair at the end. These parameters are generally called initiation parameters and largely represent the idea that the melting may be initialized by the presence of one of these features. Table 6.1 summarizes the parameters known to `VarGibbs`.

Parameter	Type	Description
`BP_BP:entropy`	NN	the hybridization entropy for a given NN pair formed by two consecutive base pairs
`BP_BP:enthalpy`	NN	the hybridization enthalpy for a given NN pair formed by two consecutive base pairs
`initiation_CG_entropy`	init.	if the sequence contains a *at least* one CG base pair [16]
`initiation_CG_enthalpy`	init.	if the sequence contains a *at least* one CG base pair
`initiation_AT_entropy`	init.	if the sequence contains *only* AT (or AU) base pairs [16]
`initiation_AT_enthalpy`	init.	if the sequence contains *only* AT (or AU) base pairs
`terminal_CG_enthalpy`	init.	for each terminal CG base pair [15]
`terminal_CG_entropy`	init.	for each terminal CG base pair [15]
`terminal_AT_enthalpy`	init.	for each terminal AT base pair [15]
`terminal_AT_entropy`	init.	for each terminal AT base pair [15]
`terminal_AU_enthalpy`	init.	for each terminal AU base pair
`terminal_AU_entropy`	init.	for each terminal AU base pair
`terminal_5TA3_enthalpy`	init.	for each terminal 5′-TA-3′ base pair [16]
`terminal_dArU_enthalpy`	init.	for each terminal dArU base pair [6]
`terminal_dArU_entropy`	init.	for each terminal dArU base pair [6]
`terminal_dTrA_enthalpy`	init.	for each terminal dArU base pair [6]
`terminal_dTrA_entropy`	init.	for each terminal dArU base pair [6]
`terminal_dCrG_enthalpy`	init.	for each terminal dCrG base pair [6]
`terminal_dCrG_entropy`	init.	for each terminal dCrG base pair [6]
`terminal_dGrC_enthalpy`	init.	for each terminal dGrC base pair [6]
`terminal_dGrC_entropy`	init.	for each terminal dGrC base pair [6]
`Terminal_dCrG_dGrC_enthalpy`	init.	applied once if sequence contains *at least* a terminal dCrG or dGrC base pair [17]
`Terminal_dCrG_dGrC_entropy`	init.	applied once if sequence contains *at least* a terminal dCrG or dGrC base pair [17]
`Terminal_dArU_dTrA_only_enthalpy`	init.	applied once if sequence contains a terminal dArU or dTrA *at both* ends [17]
`Terminal_dArU_dTrA_only_entropy`	init.	applied once if sequence contains a terminal dArU or dTrA *at both* ends [17]
`symmetry_correction_entropy`	init.	correction applied to self-complementary sequences [15, 16]
`symmetry_correction_enthalpy`	init.	correction applied to self-complementary sequences
`initiation_entropy`	init.	a fixed correction applied to all sequences
`initiation_enthalpy`	init.	a fixed correction applied to all sequences
`Na+:concentration`		$[Na^+]$ concentration (in mM) for which the parameter set is valid. Introduced in version 2.1

Table 6.1
Summary of parameters known to `VarGibbs`. Entropies are given in cal/K·mol and enthalpies in kcal/mol. All parameter names are case sensitive. Terminal parameters starting lowercase `terminal` are applied *for each* terminal occurrence, that is either once or twice. Starting uppercase `Terminal` will be applied *just once* per duplex.

6.3 Parameter file specification

This section explains how to write a parameter file. Let's start with the simplest possible parameter file

data/generic-bp.par

```
1 gibbs
2 *:entropy     -23.6
3 *:enthalpy    -8.4
4 initiation_CG_entropy   -5.9
5 initiation_AT_entropy   -20
6 terminal_5TA3_enthalpy   0.4
7 symmetry_correction_entropy     -20
```

Line 1 holds a simple identifier and should not contain spaces. The following lines hold the actual parameters, one parameter per line, first the parameter identification and next the value. Everything else in the line is ignored and can be used for annotation. Since version 3.1, a line having `#` at the first column will be ignored and can be used for documentation or other useful annotations.

Shown in lines 2 and 3 are parameters of type *generic*, see section 6.3.1. Lines 4 to 7 show specific parameters.

6.3.1 Generic * parameters

The generic base `*` specification can be used when a given parameter should be applied to any base pair (see previous example). This will only be used if a specific parameter can not be found, see section 6.3.4 on parameter precedence.

6.3.2 Independent parameters

Independent parameters are those which do not depend on context changes of the sequences. Examples are `initiation_CG_entropy` and `terminal_5TA3_enthalpy`.

6.3.3 NN parameters

`VarGibbs` has only two NN parameters, `enthalpy` and `entropy`. They are specified by `AB_CD` joined by `:` to `enthalpy` or `entropy`, as in the following example.

Examples of NN parameters

```
1 AT_GT:enthalpy 1.0
2 AT_GT:entropy 0.9
```

If several NNs share the same parameters you may write this as in the following example, joining all NNs with semicolons.

Multiple NNs

```
1 AT_GT:AT_TG:enthalpy 1.0
2 AT_GT:AT_TG:entropy 0.9
3 GC_GT:TG_CG:GT_GT:enthalpy 3.3
4 GC_GT:TG_CG:GT_GT:entropy 10.4
```

Note that `GC_GT` is equivalent to `TG_CG`, due to symmetry, and will be ignored as explained in 6.1.2.

6.3.4 Parameter precedence

Since the program can read more than one parameter file, the last parameter read is the final value. There may also be several specification in the same file as well. Consider the following example

Precedence example 1

```
1 *:enthalpy -6.0
2 AT_AT:enthalpy -10.0
```

the first line says that all nearest-neighbours should a `enthalpy` value of -6.0. The second line however says that `AT_AT` nearest-neighbours should use -10.0. In this case `GC_AT` base pairs for instance will use -6.0 since nothing different was specified. If you are unsure how your parameters were read, we suggest using `-printusedpar=1` (3.4.8).

However, a generic nearest-neighbours `*` does not supersedes a specific nearest-neighbours as in the following example

Precedence example 2

```
AT_AT:enthalpy -10.0
*:enthalpy -6.0
```

in this case `AT_AT` nearest-neighbours will continue using -10.0, not -6.0. You should understand the generic nearest-neighbours `*` as: *if nothing else matches, use this value.*

7 Sequence decomposition

Here we will explain, using examples, how `VarGibbs` takes a duplex sequence and breaks it down into its NN components. This is a crucial aspect of the software and understanding it may avoid some common pitfalls. The following examples were generated using the `-calc=nncheck` (3.1.11) option.

7.1 Example for a single sequence GACGTC/CTGCAG

The sequence GACGTC/CTGCAG corresponds to the molecule

$$5' - \text{GACGTC} - 3'$$
$$3' - \text{CTGCAG} - 5'$$

we will pass this to the program as in the following example

examples/sequence-nncheck.sh

```
vargibbs -o=./sequence -seq=GACGTC -cseq=CTGCAG -calc=nncheck -v=1
```

which will generate a file called `sequence.nncheck` which contains everything we could determine about this sequence. Here, we will go through some of the information that is generated. Among the first lines we get

examples/verify/sequence-nncheck/sequence.nncheck

```
Dataset entry: GACGTC/CTGCAG 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
* General characteristics
Salt_concentration_[Na+]: 0
Ct_concentration: 0
Length: 6 Has_CG: 1 Is_AT_only: 0 Self_complementary: 1 CG_content: 4 fCG: 0.666667
```

where it is shown if the file is self-complementary (1 is yes, 0 is no), and its CG-content. Further in the file we will see how it is broken down into NNs as

```
Exact_neighbours : GC_AT AT_CG CG_GC GC_TA TA_CG (5)
```

Then the symmetry reduction is applied (see 6.1.2)

```
Reduced_neighbours : GC_AT AT_CG CG_GC AT_CG GC_AT (5)
```

meaning that actually this sequence has only 3 different types of NN as shown in the summary

```
Reduced_neighbours_set : (3) AT_CG=2 CG_GC=1 GC_AT=2
```

`VarGibbs` also analyses the terminal characteristics

```
* Terminal characteristics
Terminal base pairs at 5: GC
Terminal base pairs at 3: CG
Terminal base pairs (BP symmetry reduced) at 5: CG
Terminal base pairs (BP symmetry reduced) at 3: CG
Terminal_5TA3: 0 Terminal_AU: 0 Terminal_AT: 0 Terminal_CG: 2
Terminal_dArU: 0 Terminal_dTrA: 0 Terminal_dCrG: 0 Terminal_dGrC: 0
```

which is used to determine which of the terminal parameters of Tab. 6.1 to apply to this sequence.

Additional information that is produced, but which is not directly relevant for the NN model, are base-pairs

```
Exact_duplex : GC/AT/CG/GC/TA/CG/ (6)
BP_reduced_duplex : CG/AT/CG/CG/AT/CG/ (6)
```

intra-strand pairs

```
Exact_strandpairs : G>A C<T A>C T<G C>G G<C G>T C<A T>C A<G (10)
Reduced_strandpairs: G>A T>C A>C G>T C>G C>G G>T A>C T>C G>A (10)
```

and trimers

```
Trimers_list : (6) TC_/AG_ GAC/CTG ACG/TGC ACG/TGC GAC/CTG TC_/AG_
Trimers_set : (3) ACG/TGC=2 GAC/CTG=2 TC_/AG_=2
```

where the underscore _ means the end of the duplex.

7.2 Example for data files

For data files passed through the `-data` (3.1.5) option, as in the following example

examples/D-CMB-nncheck.sh

```
vargibbs -o=./D-CMB-nncheck -calc=nncheck -data=../data/D-CMB.dat
```

will produce a file called `D-CMB-nncheck.nncheck` which is of the same format as for single sequences shown in the previous section. For each sequence there will be a full analysis, and as a result this file can be very large. At the end you will a summary for all sequences contain in the data file, among which the number of unique symmetry reduced NNs

examples/verify/D-CMB-nncheck/D-CMB-nncheck.nncheck

```
10 unique nearest-neighbours in dataset (NP+P, NN symmetry reduced)
Neighbours set : AT_AT=511 AT_CG=538 AT_GC=545 AT_TA=295 CG_AT=568 CG_CG=443 CG_GC=287 ...
```

the number of base pairs

```
2 unique base-pairs in dataset (BP symmetry reduced)
Base-pair set : AT=2296 CG=2302
```

while not relevant to the NN model, this information has proven useful to catch typing mistakes. The summary of 5′ and 3′ terminal properties are shown

```
4 unique terminal-5' base pairs
Terminal-5': AT=75 CG=90 GC=72 TA=44

2 unique terminal-5' base pairs (BP symmetry reduced)
Terminal-5' (BP symmetry reduced): AT=119 CG=162

4 unique terminal-3' base pairs
Terminal-3': AT=68 CG=100 GC=74 TA=39

2 unique terminal-3' base pairs (BP symmetry reduced)
Terminal-3' (BP symmetry reduced): AT=107 CG=174
```

In addition, an analysis of the whole data file is performed displaying a number of potentially useful information such as groups o sequence lengths

```
Salt/Length groups: (2)
Salt: 1000 length: 4 (2)
Salt: 1000 length: 5 (2)
Salt: 1000 length: 6 (24)
Salt: 1000 length: 7 (5)
Salt: 1000 length: 8 (49)
Salt: 1000 length: 9 (9)
Salt: 1000 length: 10 (6)
```

where the number in parenthesis shows the number of sequences of each salt/length pair. The groups of different salt concentrations

```
Salt groups: (2)
Salt: 1000 (189)
Salt: 1020 (92)
```

and total strand concentration C_t

```
Ct groups: (45)
Ct: 2 (92)
Ct: 84 (1)
Ct: 91 (1)
Ct: 100 (42)
```

8 Formatting the sequence data file

8.1 DNA-DNA or RNA-RNA duplexes

The data file containing the oligonuceotide sequences and their measured melting temperatures should be given as in the following example

Example data files for DNA

```
temperature
CGTACACATGC GCATGTGTACG 62.5 1000 119
TACTAACATTAACTA ATGATTGTAATTGAT 62.6 1000 151
```

The first line should contain the word `temperature` after which there should be one sequence per line. Make sure you last line terminates in a newline as otherwise the last sequence will be ignored. In the first column place the main strand from 5′ to 3′, in the second column the complementary strand 3′ to 5′. The measured melting temperature should be the third column, followed by the salt concentration in M (mol/L) and the strand concentration C_t in μM (10^{-6} mol/L).

8.2 Hybrid DNA-RNA duplexes

For hybrid DNA-RNA, all sequences should contain the additional `d` and `r` characters to distinguish if they are of DNA or RNA type. The sequence should be enclosed as show in the following example.

Example data files for DNA-RNA

```
temperature
r(UUUGUAUCCAAU) d(AAACATAGGTTA) 45.6 1000 100
d(GTTGGTTGGTTG) r(CAACCAACCAAC) 60.0 1000 8.3
```

8.3 Adding comments to your files

Since version 3.0 it is possible to add comments after the `#` sign for all sequences, as exemplified next

Example data with comments

```
temperature #sequences from schoning00
d(AAAAAAAAAAAAAAAA) t(TTTTTTTTTTTTTTTT) 32 1010 10 #DNA-TNA poly-A and poly-T
d(AAAATTTATATTATTA) t(TTTTAAATATAATAAT) 47 1010 10 #DNA-TNA
```

9 Files included in the package

VarGibbs comes with a number of files included.

9.1 Folder /usr/share/data/

This folder contains the input data such as parameter files (extension `.par`) and melting temperature data (extension `.dat`)

9.1.1 Sequences and melting temperatures

`D-SL96.dat` DNA sequences from Ref. 16

`D-SL98.dat` DNA sequences from Ref. 15 and supplementary data from Ref. 18

`D-OW04-[salt].dat` DNA sequences, UV data from Ref. 13

`D-OW04DSC-[salt].dat` DNA sequences, DSC data from supplementary material of Ref. 13. Note that `D-OW04DSC-1000.dat` uses uncorrected species concentrations.

`D-CMB.dat` DNA sequences, combined dataset D-SL98, D-OW04-1024 and D-OW04DSC-1020.

`D-LS95.dat` DNA-RNA hybrids at 100 mM [Na^+], from Refs. 19.

`D-CH12.dat` RNA sequences containing GU, from Ref. 20

`D-CH13-[salt].dat` RNA sequences from $T_m^{-1} \times \ln C_t$ fitting, `[salt]`=71, 121, 221, 621 and 1021 mM, from Ref. 14

`D-CH13CF-[salt].dat` RNA sequences with curve fitted T_m, `[salt]`=71, 121, 221, 621 and 1021 mM, from Ref. 14

`D-RW-[salt].dat` RNA sequences at various concentrations C_t, `[salt]`=71, 121, 221, 621 and 1021 mM, these were used in Ref. 14

`D-RNAVL-[salt].dat` RNA sequences at various concentrations C_t, with [salt] being [Na^+] concentration close to 71 mM, 121 mM and 1021 mM from Ref. 21–26

`D-DRFT.dat` hybrid DNA-RNA at [Na^+] 1000 mM curve-fitting from data Refs. 27–30

`D-DRVH.dat` hybrid DNA-RNA at [Na^+] 1000 mM van't Hoff regression from data Refs. 27–30

`D-DRLS.dat` hybrid DNA-RNA at [Na^+] 100 mM from 19, 21, 24, 26, 31

9.1.2 Parameters

Generic parameters

Generic parameters which can be used to initialize a minimization if no detailed previous knowledge exists or if you would like to run an unbiased minimization.

`gibbs-av.par` generic set of initial parameters using initiation parameters as in Ref. 16

`init_s96.par` set of initial parameters used to optimize D-SL96 data.

`init_s98.par` set of initial parameters used to optimize D-SL98 data.

`init_ow04.par` set of initial parameters used to optimize D-OW04 data.

Parameters from the literature

Since: version 1.0

`P-BS86.par` for DNA from Ref. 8, very old parameters, only of historic interest

`P-SL96.par` for DNA from Ref. 16

`P-SL98.par` for DNA from Ref. 15

`P-XIA98.par` for RNA from Ref. 9

`P-PY99.par` for AA, CC, GG and TT mismatches in DNA from Ref. 10

Since: version 2.0

`P-AL97.par` for GT mismatches in DNA from Ref. 18

`P-AL98.par` for CT mismatches in DNA from Ref. 32

`P-AL98B.par` for AC mismatches in DNA from Ref. 33

`P-AL98C.par` for GA mismatches in DNA from Ref. 34

`P-CH12.par` for GU mismatches in RNA from Ref. 20

`P-FR86.par` for RNA from Ref. 35, 36, very old parameters, only of historic interest

`P-SG95.par` for hybrid DNA-RNA from Ref. 27

Since: version 2.2

`P-SL04.par` for DNA from Ref. 7

Optimized parameters

Optimized parameters generated by `VarGibbs`.

Since: version 1.0 Ref. 1

`AOP-CMB.par`, `BOP-CMB.par` average optimized parameters (AOP) and best optimized parameters (BOP) generated from dataset D-CMB with parameters.

`AOP-OW04DSC.par` generated from dataset `D-OW04DSC`.

`BOP-OW04DSC.par` generated from dataset `D-OW04DSC`.

`AOP-OW04-[salt].par` generated from dataset `D-OW04-[salt]`.

`AOP-OW04.par` generated from dataset D-OW04-1020. Note that the salt concentration is absent from the file name.

`BOP-OW04.par` generated from dataset D-OW04-1020.

`AOP-SL96.par`, `BOP-SL96.par` generated from dataset D-SL96. See script `SL96-global.sh` for settings.

`AOP-SL98.par`, `BOP-SL98.par` generated from dataset D-SL98. See script `SL98-global.sh` for settings.

Since: version 2.1

`AOP-FIFRW-[salt].par` generated from dataset `D-RW-[salt]`, `[salt]`=71, 121, 221, 621 and 1021 mM, with initiation parameters from `P-XIA98`. [37]

`IP-FIF/P-IP-FIF-Na[salt].par` interpolation of `AOP-FIFRW-[salt].par`, between 10 and 1021 mM. [37]

`AOP-VIFRW-[salt].par` generated from dataset `D-RW-[salt]`, `[salt]`=71, 121, 221, 621 and 1021 mM, with optimized initiation parameters. [37]

`IP-VIF/P-IP-VIF-Na[salt].par` interpolation of `AOP-VIFRW-[salt].par`, between 10 and 1021 mM. [37]

Since: version 2.2

Modification `AOP-OW04DSC.par`, `BOP-OW04DSC.par`, `AOP-OW04-[salt].par`, `AOP-OW04.par`,`BOP-OW04.par` added `symmetry_correction_entropy -0.587328` Reason: if used stand-alone these files may generate incorrect results when used with self-complementary sequences.

`AOP-DRFT.par` optimized parameters for DNA-RNA (DR) hybrids, optimized from curve fitting data [6].

`AOP-DRVH.par` optimized parameters for DNA-RNA (DR) hybrids, optimized from van't Hoff plot data [6].

`AOP-DRLS.par` optimized parameters for DNA-RNA (DR) hybrids, optimized data at low [Na+] concentration [6].

9.2 Folder /usr/share/example/

This folder contains example scripts. Processing times listed here are for an Intel(R) Xeon(R) CPU model E5310 1.60GHz.

To run these scripts on your machine proceed as follows:

1. locate the script, usually you will find them at `/usr/share/VarGibbs/examples`
2. locate the data files, usually you will find them at `/usr/share/VarGibbs/data`
3. run

   ```
   /usr/share/VarGibbs/examples/D-SL98-AOP-SL98.sh /usr/share/VarGibbs/data
   ```

4. alternatively copy the script to your local folder and then run locally

   ```
   cd /usr/share/VarGibbs/examples/D-SL98-AOP-SL98.sh
   ./D-SL98-AOP-SL98.sh ./usr/share/VarGibbs/data
   ```

9.2.1 Local minimizations

`SL96-local.sh` for dataset D-SL96, processing time ≈1 min.

`SL98-local.sh` for dataset D-SL98, processing time: ≈1 min.

`CMB-local.sh` for D-CMB, processing time: ≈8 min.

`XIA98-local.sh` for D-XIA98, contributed by Izabela Ferreira, processing time: ≈5 s.

9.2.2 Global minimizations

The following scripts contain complete global minimization procedures. They will take considerable time to run, CPU times given below were measured on a `i7-5500U`, a high end laptop processor.

`SL96-global.sh` for dataset D-SL96, processing time: ≈2 min.

`SL98-global.sh` for dataset D-SL98, processing time: ≈3 min.

`CMB-global.sh` for dataset D-CMB, processing time: ≈3 min.

`OW04-global.sh` for all salt concentrations of dataset D-OW04, processing time: ≈8 min.

`OW04DSC-global.sh` for all salt concentrations of dataset D-OW04DSC, processing time: ≈2 min.

`XIA98-global.sh` for D-XIA98, contributed by Izabela Ferreira, processing time: ≈2 min.

If you wish to reproduce the AOP and BOP files included in folder `data` then edit the scripts, locate and modify the following to lines to

```
export MRR=1000
export ASL=0.001
```

see options `-mrr` (3.4.4) and `-asl` (3.4.6) for their meaning. Note that as the processing times will increase accordingly you may prefer using a dedicated desktop or server for this purpose.

9.2.3 Temperature prediction

The following scripts use the option `-calc=prediction` (3.1.11), they only read the parameters and apply them to the sequences, each should take less than a second to run.

`D-SL98-P-SL98.sh` self-prediction of dataset D-SL98 from parameters P-SL98.

`D-SL98-AOP-SL98.sh` self-prediction of dataset D-SL98 from parameters AOP-SL98.

`D-CMB-AOP-CMB.sh` self-prediction of dataset D-CMB from parameters AOP-CMB.

`sequence-AOP-CMB.sh` prediction of a single DNA sequence.

`CH12-verify.sh` prediction of dataset D-CH12 with parameters P-XIA98 and P-CH12.

`sequence-XIA98.sh` prediction for single sequence with P-XIA98. Contributed by Izabela Ferreira.

9.3 Folder /usr/share/example/verify

This folder contains all output data of the scripts include in this package. This allows you to verify if your local implementation is running correctly. If you run the scripts described here they should give you identical results to the ones in this folder.

Note that some files are gzipped to save space. To view them unzip as in the following example

```
gunzip AOP-SL96.log.00001.gz
```

Bibliography

[1] Gerald Weber. Optimization method for obtaining nearest-neighbour DNA entropies and enthalpies directly from melting temperatures. *Bioinformatics*, 31(6):871–877, 2015. doi: 10.1093/bioinformatics/btu751. URL `http://bioinformatics.oxfordjournals.org/content/31/6/871`.

[2] Gerald Weber, Jonathan W. Essex, and Cameron Neylon. Probing the microscopic flexibility of DNA from melting temperatures. *Nat. Phys.*, 5:769–773, 2009. doi: 10.1038/nphys1371.

[3] G. Weber. Finite enthalpy model parameters from DNA melting temperatures. *Europhys. Lett.*, 96:68001, 2011. doi: 10.1209/0295-5075/96/68001. URL `http://iopscience.iop.org/0295-5075/96/6/68001`.

[4] Gerald Weber. Mesoscopic model parametrization of hydrogen bonds and stacking interactions of RNA from melting temperatures. *Nucleic Acids Res.*, 41:e30, 2013. doi: 10.1093/nar/gks964. URL `http://nar.oxfordjournals.org/content/41/1/e30`.

[5] Gerald Weber. TfReg: Calculating DNA and RNA melting temperatures and opening profiles with mesoscopic models. *Bioinformatics*, 29:1345–1347, 2013. doi: 10.1093/bioinformatics/btt133. URL `http://bioinformatics.oxfordjournals.org/content/29/10/1345`.

[6] Vivianne Basílio Barbosa, Erik de Oliveira Martins, and Gerald Weber. Nearest-neighbour parameters optimized for melting temperature prediction of DNA/RNA hybrids at high and low salt concentrations. *Biophys. Chem.*, 251C:106189, 2019. doi: 10.1016/j.bpc.2019.106189.

[7] Jr. SantaLucia, John and Donald Hicks. The thermodynamics of DNA structural motifs. *Annu. Rev. Biophys. Biomol. Struct.*, 33:415–440, 2004.

[8] K. J. Breslauer, R Frank, H Blocker, and L. A. Marky. Predicting DNA duplex stability from the base sequence. *Proc. Natl. Acad. Sci. USA*, 83(11):3746–3750, 1986. doi: 10.1073/pnas.83.11.3746.

[9] Tianbing Xia, John SantaLucia, Jr., Mark E. Burkard, Ryszard Kierzek, Susan J. Schroeder, Xiaoqi Jiao, Christopher Cox, and Douglas H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochem.*, 37:14719–14735, 1998. doi: 10.1021/bi9809425.

[10] N Peyret, P A Seneviratne, H T Allawi, and John SantaLucia, Jr. Nearest-neighbour thermodynamics and NMR of DNA sequences with internal A·A, C·C G·G and T·T mismatches. *Biochem.*, 38(12):3468–3477, 1999.

[11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 1988.

[12] Carl Schildkraut and Shneior Lifson. Dependence of the melting temperature of DNA on salt concentration. *Biopoly.*, 3(2):195–208, 1965.

[13] Richard Owczarzy, Yong You, Bernardo G. Moreira, Jeffrey A. Manthey, Lingyan Huang, Mark A. Behlke, and Joseph A. Walder. Effects of sodium ions on DNA duplex oligomers: Improved predictions of melting temperatures. *Biochem.*, 43:3537–3554, 2004. doi: 10.1021/bi034621r.

[14] Zexiang Chen and Brent M Znosko. Effect of sodium ions on RNA duplex stability. *Biochem.*, 52(42): 7477–7485, 2013. doi: 10.1021/bi4008275.

[15] John SantaLucia, Jr. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc. Natl. Acad. Sci. USA*, 95(4):1460–1465, 1998. URL `http://www.pnas.org/cgi/content/abstract/95/4/1460`.

[16] John SantaLucia, Jr., H T Allawi, and P A Seneviratne. Improved nearest-neighbour parameters for predicting DNA duplex stability. *Biochem.*, 35:3555–3562, 1996.

[17] Dipanwita Banerjee, Hisae Tateishi-Karimata, Tatsuya Ohyama, Saptarshi Ghosh, Tamaki Endoh, Shuntaro Takahashi, and Naoki Sugimoto. Improved nearest-neighbor parameters for the stability of RNA/DNA hybrids under a physiological condition. *Nucleic Acids Res.*, 48(21):12042–12054, 2020. doi: 10.1093/nar/gkaa572.

[18] H. T. Allawi and J. SantaLucia Jr. Thermodynamics and NMR of internal G·T mismatches in DNA. *Biochemistry*, 36(34):10581–10594, 1997.

[19] Elena A Lesnik and Susan M Freier. Relative thermodynamic stability of DNA, RNA, and DNA: RNA hybrid duplexes: relationship with base composition and structure. *Biochem.*, 34(34):10807–10815, 1995. doi: 10.1021/bi00034a013.

[20] Jonathan L Chen, Abigael L Dishler, Scott D Kennedy, Ilyas Yildirim, Biao Liu, Douglas H Turner, and Martin J Serra. Testing the nearest neighbor model for canonical RNA base pairs: Revision of GU parameters. *Biochem.*, 51(16):3508–3522, 2012. doi: 10.1021/bi3002709.

[21] Shaohui Wang and Eric T Kool. Origins of the large differences in stability of DNA and RNA helixes: C-5 methyl and 2′-hydroxyl effects. *Biochem.*, 34(12):4125–4132, 1995.

[22] Susan J Schroeder and Douglas H Turner. Factors affecting the thermodynamic stability of small asymmetric internal loops in RNA. *Biochem.*, 39(31):9257–9274, 2000.

[23] A. Pasternak and J. Wengel. Thermodynamics of RNA duplexes modified with unlocked nucleic acid nucleotides. *Nucleic Acids Res.*, 38(19):6697–6706, 2010. doi: 10.1093/nar/gkq561.

[24] S. Nakano, M. Fujimoto, H. Hara, and N. Sugimoto. Nucleic acid duplex stability: influence of base composition on cation effects. *Nucleic Acids Res.*, 27(14):2957, 1999.

[25] Su-Hwi Hung, Qin Yu, Donald M Gray, and Robert L Ratliff. Evidence from CD spectra that d(purine)-r(pyrimidine) and r(purine)-d(pyrimidine) hybrids are in different structural classes. *Nucleic Acids Res.*, 22(20):4326–4334, 1994.

[26] Jeffrey I Gyi, Graeme L Conn, Andrew N Lane, and Tom Brown. Comparison of the thermodynamic stabilities and solution conformations of DNA·RNA hybrids containing purine-rich and pyrimidine-rich strands with DNA and RNA duplexes. *Biochem.*, 35(38):12538–12548, 1996. doi: 10.1021/bi960948z.

[27] Naoki Sugimoto, Shu-ichi Nakano, Misa Katoh, Akiko Matsumura, Hiroyuki Nakamuta, Tatsuo Ohmichi, Mari Yoneyama, and Muneo Sasaki. Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes. *Biochem.*, 34(35):11211–11216, 1995. doi: 10.1021/bi00035a029.

[28] Naoki Sugimoto, Mariko Nakano, and Shu-ichi Nakano. Thermodynamics- structure relationship of single mismatches in RNA/DNA duplexes. *Biochemistry*, 39(37):11270–11281, 2000.

[29] Peng Wu, S. Nakano, and Naoki Sugimoto. Temperature dependence of thermodynamic properties for DNA/DNA and RNA/DNA duplex formation. *Eur. J. Biochem.*, 269(12):2821–2830, 2002. URL `http://www.blackwell-synergy.com/doi/abs/10.1046/j.1432-1033.2002.02970.x`.

[30] Brittany Rauzan, Elizabeth McMichael, Rachel Cave, Lesley R Sevcik, Kara Ostrosky, Elisabeth Whitman, Rachel Stegemann, Audra L Sinclair, Martin J Serra, and Alice A Deckert. Kinetics and thermodynamics of DNA, RNA, and hybrid duplex formation. *Biochem.*, 52(5):765–772, 2013. doi: 10.1021/bi3013005.

[31] Andrew M Kawasaki, Martin D Casper, Susan M Freier, Elena A Lesnik, Maryann C Zounes, Lendell L Cummins, Carolyn Gonzalez, and P Dan Cook. Uniformly modified 2'-deoxy-2'-fluoro-phosphorothioate oligonucleotides as nuclease-resistant antisense compounds with high affinity and specificity for RNA targets. *Journal of Medicinal Chemistry*, 36(7):831–841, 1993.

[32] HT Allawi and Jr SantaLucia, J. Thermodynamics of internal C.T mismatches in DNA. *Nucleic Acids Res.*, 26(11):2694–2701, 1998. URL `http://nar.oupjournals.org/cgi/content/abstract/26/11/2694`.

[33] H. T. Allawi and J. SantaLucia, Jr. Nearest-neighbor thermodynamics of internal A·C mismatches in DNA: Sequence dependence and pH effects. *Biochem.*, 37:9435–9444, 1998.

[34] H.T. Allawi and J. SantaLucia Jr. Nearest neighbor thermodynamic parameters for internal G·A mismatches in DNA. *Biochemistry*, 37(8):2170–2179, 1998.

[35] S. M. Freier, R. Kierzek, J. A. Jaeger, N. Sugimoto, M. H. Caruthers, T. Neilson, and D.H. Turner. Improved free-energy parameters for predictions of RNA duplex stability. *Proc. Natl. Acad. Sci. USA*, 83 (24):9373, 1986.

[36] Douglas H Turner, Naoki Sugimoto, and Susan M Freier. RNA structure prediction. *Annu. Rev. Biophys. Biophys. Chem.*, 17(1):167–192, 1988.

[37] Izabela Ferreira, Elizabeth A. Jolley, Brent M. Znosko, and Gerald Weber. Replacing salt correction factors with optimized RNA nearest-neighbour enthalpy and entropy parameters. *Chem. Phys.*, 521:69–76, may 2019. doi: 10.1016/j.chemphys.2019.01.016. URL `https://www.sciencedirect.com/science/article/abs/pii/S0301010418311200`.