Rabit: Regression analysis with background integration

Peng Jiang (peng.jiang.software@gmail.com)

Rabit is a linear variable selection algorithm. Among all input variables, Rabit first tests the significance of each variable after controlling the effect of background factors. Then, Rabit applies stepwise forward regression to select a subset of variables to optimize the model error. Please contact me for any questions and bugs. Please cite us if you find this tool useful: http://rabit.dfci.harvard.edu/publication/

Installation

The installation of Rabit is very easy on Unix like system, such as Linux or Mac OS. After downloading the package, extract the zip file with "**tar xvf Rabit.tar.gz**" and get into the Rabit folder "**cd Rabit**". Then type the following commands:

```
./configure          # configure according to your system
make                 # build Rabit binary and library dependencies
make check           # test whether Rabit binary produces correct answer
make install         # copy Rabit binary to install directory
```

If you don't have super user privilege, you can install Rabit on a local folder by changing the configure step with "**./configure --prefix=path_to_local_folder**". For example, I do "**./configure --prefix=/Users/peng**" on my laptop. Alternatively, after the step of "**make**", the binary of Rabit is available in "**src**" folder.

During the installation process, Rabit package will search for GSL, BLAS and LAPACK libraries in your system. If they are not found, Rabit will build local libraries with attached packages. So, you don't need to install extra dependencies since Rabit package contains everything. For better time efficiency, you could install ATLAS in your system. The installation of ATLAS could be extremely hard. So try it at your own risk (Appendix A).

Note: For Mac user, if the installation failed, you might need to install gfortran from here: https://gcc.gnu.org/wiki/GFortranBinaries#MacOS

Usage

As an example, get into the "data" directory and type:

```
Rabit  -x RBP_motif.hg19_3UTR.targets.mat \
       -y TCGA.all_cancers.Expression \
       -b hg19.background.PWM.RBP \
       -c CNA:TCGA.all_cancers.CNA \
       -c Methylation:TCGA.all_cancers.DNA_Methylation \
       -o test.output
```

The format of running Rabit is like this:

Rabit -x X -y Y -o output [OPTIONS]

Input Options:

-x	Input variable matrix. Each column represents a variable to be selected. Each row represents a gene with its values on each variable in selection. For example, "data/RBP_motif.hg19_3UTR.targets.mat" lists the predicted binding score of a set of RNA binding protein (RBP) motifs. The RBP motifs are listed on columns and gene names are listed on rows.
-y	Response matrix. Each column represents a response vector and Rabit will run variable selection on each response vector separately. For example, "data/TCGA.all_cancers.Expression" contains the average gene expression profiles across 21 TCGA cancer types. The cancer abbreviation names are listed on columns and gene names are listed on rows.
-b	Background factors. Default: empty. Each column represents a confounding factor to be controlled. For example, "data/hg19.background.PWM.RBP" lists "Binding degree" and "AU content" on target gene 3'UTR as two factors that affect cancer gene expression. Rabit will exclude their effects in linear model.
-c	Background factors for each column of Y. Default: empty. Different from "-b", which inputs background factors shared across all columns of Y, this input is specific for each column of Y. Rabit will search shared column names between Y and input background factors here and control these factors. For example, each TCGA cancer type has its unique copy number alternation ("data/TCGA.all_cancers.CNA") and DNA methylation ("data/TCGA.all_cancers.DNA_Methylation") pattern, and both factors will impact gene expression independently from RBP regulation. You can append a factor title in front of file name separated by colon ':' (e.g., "-c CNA: TCGA.all_cancers.CNA").
-f	FDR threshold. Range: (0,1]. Default: 0.05. Rabit estimates the statistical significance of each variable in X and only keep significant ones in later stepwise forward regression. You can set the threshold as 1 to skip this step.
-t	Transform Y to normal distribution. Options: 1 (yes), 0 (no). Default: 1 (yes). We found transforming response Y to normal distribution will increase the statistical power of linear regression t-test. However, if the Y distribution is very different from normal distribution, the t-test output cannot be trusted. For example, if Y is composed of only 1 and 0, you need to use logistic regression.

The input to Rabit can be divided to three parts: variable matrix (-x), response matrix (-y) and optional background matrices B (-b and -c). Rabit selects a subset of variables among matrix X to optimize the fit of linear regression model Y ~ [B X]. All input files are in matrix form. The first line contains column names separated with tabs. Each following line starts with a row name followed with matrix elements. You don't need to align all input matrices with the same row name order. Rabit will automatically align this for you. Here is a simple example of input matrix:

	ColA	**ColB**
RowA	1	2
RowB	2.5	0

Output:

Please see "data/output" as an example. For each response column in matrix Y, Rabit selects a set of variables to optimize the model error estimated by Mallow's Cp. Then, a linear regression model is fitted over these selected variables. The column Cp represents the Mallow's Cp values in forward selection process, and the rest columns represent regression coefficients (Estimate) and t-test results (Std. Error, t-value, Pr(>|t|)). The Cp column is zero for all background factors.

Note:

There are several input options and output file in Rabit for specific purposes in our project, which may not be useful in general. But we still keep them for your reference (Appendix B).

Appendix A: Install ATLAS

The source code of ATLAS is available here http://math-atlas.sourceforge.net. The ATLAS package will check your computer architecture and automatically tune the linear algebra library for better performance. Some systems may have ATLAS pre-installed, such as Mac OS. However, you still get better performance if you install the latest version by my experience. Before installation, you need to do three things:

1, Turn off CPU throttling: The way of turning off CPU throttling is different for different systems. You need to search on Google for your case. If the CPU throttling is already off, you won't get any error in the configure process. If the CPU throttling is on and you don't have super user privilege, you may not be able to install. However, if you can solve this problem without super user privilege, please email me and share the solution.

2, Get the CPU clock rate: This is different for different systems. In Mac OS, you can do "system_profiler | grep Processor" and translate the processor speed in MHz. My laptop speed is 2.3GHz, so I input 2300 for "-DPentiumCPS" option in configure. In Linux, you can look at "cpu MHz" of file "/proc/cpuinfo".

3, Download LAPACK: http://www.netlib.org/lapack/#_lapack_version_3_4_1

For basic steps of installation, follow this procedure:

```
cd ATLAS                # enter ATLAS directory
mkdir build             # create build directory. You cannot build in root directory.
cd build                # enter build directory
../configure -b 64 -D c -DPentiumCPS=2400 \  # Replace with your CPU clock
   --prefix=/Users/peng \                      # Replace with your install dir
   --with-netlib-lapack-tarfile= /Users/peng/software/lapack-3.4.1.tgz
make build              # tune & build lib
make check              # sanity check correct answer
make ptcheck            # sanity check parallel
make time               # check if lib is fast
make install            # copy libs to install directory
```

Note: If you are installing on a local place instead of system paths (e.g., "/usr/lib", "/usr/local/lib"), you need to modify your ".bash_profile" or ".bashrc" file to guide the gcc compiler to your ATLAS installation. For example, on my ".bashrc" file, I put two lines at the very end of file.

```
export PATH=$HOME/bin:$PATH
export LD_LIBRARY_PATH=$HOME/lib:$LD_LIBRARY_PATH
```

If you are installing on a computer cluster, please don't build ATLAS on head node, which might have different CPU architecture from a computing node. You need to login into a computing node and install ATLAS and build Rabit on that node. For example, on Harvard Odyssey cluster, you can log into a computing node like this "srun -p interact --mem-per-cpu=4000 --pty bash".

After setting up all of these things, you can configure Rabit package and see whether Rabit package can find ATLAS. You may see something like this in configure output:

```
checking for ATL_dgemm in -latlas... yes          # check ATLAS
checking for ATL_dgemm in -lcblas... yes          # check ATLAS cblas interface
checking for APL_dgemm in -latlas... no           # Mac OS ATLAS starts with APL
checking for APL_dgemm in -lcblas... no           # check Mac OS cblas interface
checking for cblas_dgemm in -lcblas... yes        # check cblas interface
checking for dgeqrf_ in -llapack... yes           # check LAPACK function
checking for clapack_dgeqrf in -llapack... yes    # test if LAPACK is from ATLAS
checking for ATL_dgeqrf in -llapack... yes        # test if LAPACK is from ATLAS
```

The installation of ATLAS may fail for no reason. Rabit can still be built and used without ATLAS. GSL will be used under this situation and Rabit will still have reasonable efficiency. So don't be panic if you cannot install ATLAS.

Appendix B: additional input options and output

-s	Select one best variable in X for each category. Options: 1 (yes) or 0 (no). Default: 0 (no). Sometimes, several variables in X come from the same category, such as the same Transcription Factor may have several ChIP-seq profiles available. In matrix X, you need to append the category name in front of each variable name separated with dot "." (e.g., CatA.V1 CatA.V2…). Rabit will select the best variable in the same category, which has the maximum |t-value| in linear regression t-test. The selected variable for each category will be outputted in file with ".selected" postfix.
-r	Run forward stepwise selection. Options: 1 (yes), 0 (no). Default: 1 (yes). If you set this flag as 0, Rabit will only calculate the statistical significance of each variable without forward stepwise selection.
-v	Verbose output such as warnings. Options: 1 (yes), 0 (no). Default: 1 (yes).
output.t	As a beginning step, Rabit will test the statistical significance by t-test for each input variable in **X** across all response vectors in **Y**. We output the t-value in this file.