ApoHP

APOBEC hairpins analysis tool

Table of Contents

---

1. Introduction

ApoHP is a software tool for identifying potential hairpin-forming sequences in genomes, identifying mutations in these hairpins in mutation datasets, computing metric of APOBEC mutation signatures in the samples analyzed, and creating plots of sample characteristics and hairpin mutation rate trends. It formalizes the analyses presented in Langenbucher et al. (2020) *Nat Comm*, and extends and supersedes the analysis published earlier in Buisson et al. (2019) *Science*.

---

2. Installing and running ApoHP

ApoHP is written in MATLAB. It can be run either from the MATLAB command line, or by running a compiled MATLAB executable.

Requirements: MATLAB v2013a or later.

TO INSTALL

The latest version of the source code can be found at:
https://github.com/alangenb/ApoHP

A compiled executable version of ApoHP, as well as zipped files of its reference and input data files, as well as output data files and figures can be downloaded from:
http://www.dropbox.com/sh/8hiyfv542f1i6sy/AABnd-gmXo8y2TA3Z7k-VJN_a?dl=0

Files are organized as follows:

ref	= genome reference data for human genome build hg19
data/init	= mutation data analyzed in the Langenbucher et al. paper
data/processed	= output data files for hairpins and mutations
figs	= output figures in PDF format
src	= MATLAB source code files for ApoHP
bin	= compiled MATLAB version of ApoHP
run.m	= script demonstrating how to run each of the ApoHP commands on the data files included in the distribution package

ApoHP also requires the MATLAB R2013a runtime environment to be installed. It can be downloaded from:
http://www.mathworks.com/supportfiles/MCR_Runtime/R2013a/MCR_R2013a_glnxa64_installer.zip

Installation instructions can be found here:
http://www.mathworks.com/help/compiler/install-the-matlab-runtime.html

This runtime environment should be universally compatible with any recent Linux distribution; we have successfully tested it on 64 bit CentOS 5, RHEL 6, and Debian 8.2. At this time, ApoHP is available for 64 bit Linux systems only.

Once the runtime is successfully installed, you must add it to your LD_LIBRARY_PATH. You will likely want to add the following lines to your .bashrc/.cshrc, so that the MATLAB runtime is always on your path.

For a bash shell:

```
mcr_root=<path to runtime you specified when installing>
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$mcr_root/bin/glnxa64/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$mcr_root/sys/java/jre/glnxa64/jre/lib/amd64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$mcr_root/sys/java/jre/glnxa64/jre/lib/amd64/server
```

export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$mcr_root/sys/java/jre/glnxa64/jre/lib/amd64/native_threads
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$mcr_root/sys/os/glnxa64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$mcr_root/bin/glnxa64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$mcr_root/runtime/glnxa64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$mcr_root/lib

For a C shell:

set mcr_root = <path to runtime you specified when installing>
setenv LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$mcr_root/bin/glnxa64/
setenv LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$mcr_root/sys/java/jre/glnxa64/jre/lib/amd64
setenv LD_LIBRARY_PATH =
$LD_LIBRARY_PATH:$mcr_root/sys/java/jre/glnxa64/jre/lib/amd64/server
setenv LD_LIBRARY_PATH =
$LD_LIBRARY_PATH:$mcr_root/sys/java/jre/glnxa64/jre/lib/amd64/native_threads
setenv LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$mcr_root/sys/os/glnxa64
setenv LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$mcr_root/bin/glnxa64
setenv LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$mcr_root/runtime/glnxa64
setenv LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$mcr_root/lib

If all is well, running test/MCR_test from the mutsig2cv root directory should
output the runtime installation path. If you receive the following error:

test/MCR_test: error while loading shared libraries: libmwlaunchermain.so:
cannot open shared object file: No such file or directory

you have likely not updated your LD_LIBRARY_PATH correctly.

We also provide a Python wrapper "run_matlab.py" that attempts to automatically set up the LD_LIBRARY_PATH.

INPUT FILES

You will need reference files for your genome:

chromInfo.txt	= list of chromosome names and lengths
chr*.fa	= FASTA sequence of each chromosome
refGene.txt	= gene annotations
known_driver_genes.txt	= list of known cancer driver genes (for Fig. 5)

For instance, for human genome build hg19, these files can be downloaded as follows:

wget -r http://hgdownload.cse.ucsc.edu/goldenpath/hg19/chromosomes
wget http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/refGene.txt.gz
wget http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/chromInfo.txt.gz
wget https://cancer.sanger.ac.uk/cosmic/census/all?home=y&name=all&export=tsv

The chromosome files should be unzipped. All reference files should be located in a <refdir> directory whose name will be passed to `ApoHP` as an input parameter.

You will need additional input files:

`<sigfile>` = sigProfiler SBS signatures file with annotated names,
which will be used for assigning names to signatures discovered by NMF.

Provided in distribution package:
sigProfiler_SBS_signatures.WGS.v3.0.mat
sigProfiler_SBS_signatures.WXS.v3.0.mat

`<ttypefile>` = list of tumor types and their full names and assigned colors for plots

Provided in distribution package:
tumortype_longnames_and_colors.txt

`<mutationlist>` = list of mutations in a set of samples to analyze
Format: tab-separated text file with header line containing column names. Columns can be in any order.

Required columns:
`patient` is patient or sample name
`chr` is name of chromosome mutation is on (must match names in `chromInfo.txt`)
`pos` is position of the mutation on the chromsome
`ref` is identity of unmutated nucleotide in reference genome (A, C, G, or T)
`alt` is identity of mutated nucleotide (A, C, G, or T)
Optional column:
`ttype` is tumor type name (must match names in <`ttypefile`>)

---

3. `ApoHP` commands

Once the MATLAB runtime is properly set up, `ApoHP` can be directly invoked from the command line:

```
ApoHP <command> <args>
```

or, it can be run using the python wrapper:

```
python2 run_matlab.py <mccdir> ApoHP <command> <args>
```

where <mccdir> is the directory containing the compiled MATLAB program (ApoHP and ApoHP.ctf)

or, it can be run from directly within Matlab:

```
> addpath(src)
> ApoHP(command,args{1},args{2},...,args{end})
```

where <src> is a path to the MATLAB source files directory from the distribution package. The arguments <args> differ for each of the available ApoHP commands.

3.1. ApoHP survey_hairpins

The survey_hairpins command takes the genomic reference data as input and performs a scan for potential hairpin-forming sequences.

It takes two required input parameters, plus an optional third parameter.

```
ApoHP survey_hairpins <refdir> <outdir> [<blockno>]
```

Required parameters:
<refdir> is the directory containing the genome reference info, as described above.
<outdir> is the directory to write the results of the hairpin survey to.

Optional parameter:
<blockno> is an optional parameter for use when processing each genomic 10Mb block in parallel. If omitted, all blocks will be run as a loop and then gathered. If <blockno> is provided, it should range from 1 to N, where N is the number of 10Mb blocks in the genome. When using this mode, a final "gather" call should be made with <blockno> = -1.

Output:
survey_hairpins outputs a full list of C:G basepairs in the genome, along with their hairpin characteristics, as the file <allCsfile>. It also outputs a list of the optimal APOBEC3A substrates found.
The output files contain the following annotations for each C:G basepair in the genome. All information is relative to the cytosine (information is strand-flipped for G bases on the reference strand). The parameters describe the most stable possible DNA hairpin that can expose this position in a hairpin loop.

looplen is the length (in nucleotides) of the loop.
looppos is the position of C within the loop (ranging from 1 to looplen), or 0 or looplen+1 for bases that are in the final basepair of the stem.
nbp is the number of basepairs in the stem pairing.
ngc is the number of those basepairs that are G:C pairs.
mmp is the position of a mismatch in the stem (or 0 if no mismatch).
bulgepos is the position of a bulged nucleotide in the stem (or 0 if no bulge).
ss is the stem strength of the hairpin stem pairing.
minus2 is the identity of the base three before (5' of) the C (A=1, C=2, G=3, T=4).
minus1 is the identity of the base two before (5' of) the C.
minus0 is the identity of the base directly before (5' of) the C.
plus1 is the identity of the base directly after (3' of) the C.
plus2 is the identity of the base two after (3' of) the C.

3.2. ApoHP analyze_mutations

The analyze_mutations command analyzes mutation data to quantify the Extended APOBEC3A Signature in each sample, and generates plots describing the samples and the mutation rate trends. It performs the main tumor mutation data analyses from the paper, generating tables and figures of its results.

```
ApoHP analyze_mutations <refdir> <mutationfile> <ttypefile>
<sigfile> <k> <allCsfile> <datoutdir> <figoutdir>
[<wxs_mutationfile> <wxs_sigfile> <wxs_k>]
```

Required parameters:
<refdir> is the directory containing the genome reference info.
<mutationfile> is a text file containing the mutations to be analyzed.
<ttypefile> is a text file containing the tumortype longnames and colors.
<sigfile> is a MAT file containing COSMIC sigProfile signature definitions and names.
<k> is a number from 2 to 1000, telling how many signatures to find by NMF.
<allCsfile> is an output of the survey_hairpins command, above.
<datoutdir> is the directory to write output data files to.
<figoutdir> is the directory to write output figures to.

Optional parameters: The following three parameters are used for specifying the WXS dataset to be included in Figure 5.
<wxs_mutationfile> is a text file containing WXS mutations.
<wxs_sigfile> is a MAT file containing COSMIC sigProfile signature definitions and names.
<wxs_k> is a number from 2 to 1000, telling how many signatures to find by NMF.

Output:
`analyze_mutations` generates tables and figures of the main tumor mutation analyses reported in the paper.

3.3. `ApoHP superimpose_data`

The `superimpose_data` command recreates the APOBEC "Bird Plot" from Figure 3 of the paper, with an external mutation dataset superimposed on it. It is useful for comparing a new dataset to the existing pan-cancer datasets analyzed in the paper.

```
ApoHP superimpose_data <refdir> <mutationfile> <ttypefile>
<sigfile> <k> <allCsfile> <fig3file> <datoutdir> <figoutdir>
```

Required parameters:
`<refdir>` is the directory containing the genome reference info.
`<mutationfile>` is a text file containing the mutations to be analyzed.
`<ttypefile>` is a text file containing the tumortype longnames and colors.
`<sigfile>` is a MAT file containing COSMIC sigProfiler signature definitions and names.
`<k>` is a number from 2 to 1000, telling how many signatures to find by NMF.
`<allCsfile>` is an output of the `survey_hairpins` command, above.
`<fig3file>` is an output of the `analyze_mutations` command, above.
`<datoutdir>` is the directory to write output data files to.
`<figoutdir>` is the directory to write output figures to.

Output:
`superimpose_data` generates versions of the APOBEC "Bird Plot" from Figure 3 of the paper, with an external dataset superimposed on it.