**MountainLab PRV System**

**Separation of raw data from record-keeping files**

Files and folders serve two purposes in MountainLab. First, they hold raw data – sometimes the raw data files can be enormous. Second, they help organize studies and experiments. The idea of the PRV system is to separate these two types of files. Among the many advantages, it allows studies to be packaged into relatively small .zip files, to be checked into git or svn repositories, and/or be hosted on servers with limited storage capabilities.

For example, a typical dataset in MountainLab is a directory containing a raw data file (raw.mda.prv), a parameter file (params.json), and a geometry file (geom.csv). All three of these files (including the raw file) are tiny text files. Normally the raw file would be a (potentially enormous) binary file, making it difficult to move the study around, email it to a collaborator or store in a reasonably sized repository. Here is an example raw.mda.prv file entitled raw.mda.prv:

{

"original\_checksum": "9a42d2fb7d0e2cfd8c206c2ddc30640472ffab3d",

"original\_fcs": "head1000-da39a3ee5e6b4b0d3255bfef95601890afd80709",

"original\_path": "20160426\_kanye\_02\_r1.nt16.mda",

"original\_size": 666616580,

"prv\_version": 0.1

}

The fields in this JSON document represent the stats of the file at the time the .prv file was created using the *prv-create* command-line utility as described below. This JSON document is a universal pointer to the original raw data file. The assumption is that the file is uniquely identified by its checksum (sha-1 hash).

Suppose we know that the original file is somewhere on the local disk, or on a particular remote server. The *prv-locate* utility may be used to efficiently find the path or url to the file. As specified in the MountainLab configuration files (mountainlab.default.json and mountainlab.user.json), a set of directories on the local machine (or on the remote server) are searched recursively. For each candidate file we first check whether the size matches the *original\_size* field. If not, the program goes on to the next candidate file. The optional *original\_fcs* field can be used to eliminate the majority of non-matching files that happen to have the correct size. In the example above, this value provides the sha-1 hash of the first 1000 bytes. If the *original\_size* and *original\_fcs* match, then the sha-1 hash of the entire file is computed and compared with *original\_checksum* as the final check. Since this last operation may be computationally demanding, a database of pre-computed checksums is maintained in the /tmp/sumit directory. Here is a description of the above fields:

* **original\_checksum:** The sha-1 hash of the entire raw file.
* **original\_fcs:** (optional) a code that can be used to quickly determine whether a given file is a candidate for a match.
* **original\_path:** Only used for information in case the user needs a reminder on the name and location of the raw file at the time the .prv file was created.
* **original\_size:** The size in bytes of the raw file.
* **prv\_version:** Always 0.1 for now.

A prv file may be created using the command-line *prv-create* utility as follows:

> prv-create /path/to/data.mda raw.mda.prv

Later we can locate the original file using:

> prv-locate raw.mda.prv

This last command prints the path of the found file to standard output. Importantly, the file does not need to be at its original location.

**Provenance tracking**

The PRV system does more than just provide universal pointers to large files. The .prv extension stands for 'provenance', and these files can also contain information about the sequence of operations that were used to create the file. It is assumed that each processing step was accomplished using mountainprocess (described elsewhere). This not only documents processing history, but also allows files that are not on the local machine to be recreated (assuming the necessary input files are present).

Here is an example .prv file encapsulating a preprocessed raw data file called pre.mda.prv:

{

"original\_checksum": "2d38f234d30bc50b7b93aaefda4a8b668f3e5f46",

"original\_path": "0b7a1e3-whiten-timeseries\_out.tmp",

"original\_size": 1333233140,

"processes": [

{

"inputs": {

"timeseries": {

"original\_checksum": "a9812ec96cbbbe94eb647b8d8005bcf4a43fba30",

"original\_path":”8310d11-bandpass\_filter-timeseries\_out.tmp",

"original\_size": 1333233140

}

},

"outputs": {

"timeseries\_out": {

"original\_checksum": "2d38f234d30bc50b7b93aaefda4a8b668f3e5f46",

"original\_path":”0b7a1e3-whiten-timeseries\_out.tmp",

"original\_size": 1333233140

}

},

"parameters": {

},

"processor\_name": "whiten"

},

{

"inputs": {

"timeseries": {

"original\_checksum": "f936e18694a2ea0f3ad595283c7cda53d0423bf6",

"original\_path":”raw.mda",

"original\_size": 218

}

},

"outputs": {

"timeseries\_out": {

"original\_checksum": "a9812ec96cbbbe94eb647b8d8005bcf4a43fba30",

"original\_path":”8310d11-bandpass\_filter-timeseries\_out.tmp",

"original\_size": 1333233140

}

},

"parameters": {

"freq\_max": "8000",

"freq\_min": "300",

"samplerate": "30000"

},

"processor\_name": "bandpass\_filter"

}

]

}

If the raw.mda file is still on the machine, but the intermediate preprocessed files have been deleted (to save disk space), the preprocessed data file may be recovered using the following command:

> prv-recover pre.mda.prv /path/to/recovered/pre.mda

Then *bandpass\_filter* and *whiten* processors will then be run to recreate this file. Note that if the raw.mda file is missing, but the output of *bandpass\_filter* is present, then the preprocessed file can also be recovered.

**Retrieving binary files from a server**

Another valuable use case of the PRV system is to store raw, intermediate, or result files on a remote server. Recall that a .prv file is a universal pointer to the original data file and does not contain any information about which servers the file may reside. This allows tremendous flexibility on how data may be stored or moved around without affecting the ability for researches to access the binary files.