

BDVal User Manual

Version 1.1

(The latest version of this manual can be obtained at <http://campagnelab.org/software/bdval/>)

Fabien Campagne

April 28, 2010

Weill Medical College of Cornell University,

New York, NY 10021

E-mail: fac2003@med.cornell.edu

Contents

1 Downloading BDVal	1
2 Installing BDVal	1
3 Parallel Processing	3
4 Third-party software	3
4.1 R	3
4.2 ROCR	3
4.3 RServe	4
4.4 Configuring the connection to the R server	4
4.5 Apache Ant	5
5 Downloading the prostate cancer dataset	5
6 Getting organized	6
7 Loading the dataset	8
8 Defining an evaluation plan	9
9 Evaluating a feature selection strategy	9
10 Evaluating many feature selection strategies: the BDVal build file	10
10.1 Model evaluation	10
10.2 Final Model Construction	12
10.3 Predicting the validation sets	14
11 Configuring the BDVal project build file	15
11.1 Getting organized	15
11.2 Configuring eval-dataset-root	15
11.3 Configuring parallel processing and memory usage	16
11.4 The project properties file	16
11.5 The project-specific build file	19

This manual presents an example of a biomarker discovery project. In this example, we will analyze a large prostate cancer dataset published recently and available through the Gene Expression Omnibus (GEO). We will conduct the analysis with BDVal, a suite of programs developed in our laboratory and distributed as an open-source project. We routinely use BDVal on Linux, Windows (XP, Vista, 7) and Mac OS X. Please note that BDVal has no graphical user interface, all operations are done through the command line in a terminal or console. We assume that the reader is familiar with these environments.

1 Downloading BDVal

BDVal can be downloaded in binary or source distribution. The binary distribution is recommended for users interested in studying a specific biomarker dataset. The source distribution provides the same functionality as the binary distribution, but must be configured and compiled before use. As such the source distribution is only recommended for scientists interested in implementing new methods in the BDVal framework, or those curious about the implementation of certain features.

The binary distribution can be obtained from <http://campagnelab.org/software/bdval>

The table of contents should indicate how to download the software and point to up to date installation instructions. Always follow the instructions on the web page, which are updated every time a new release of BDVal is released. You should now have downloaded a file called [bdval_latest-bdval.zip](#). Uncompress this file to a directory on your computer. We will refer to this directory as `BDV_UNCOMPRESS_DIR` in the rest of this section. At the time of writing, the latest release is named `bdval_latest-bdval.zip` and uncompressing this archive yields: `bdval_1.1`. This version number also appears in final reports generated by BDVal, so that you can always tell which version of the program you were using to generate specific biomarker models. We will now refer to this location as `BDV_INSTALL_DIR`. This folder contains the following files:

```
README-BDVal.txt
bdval.jar
buildsupport
config
data
```

2 Installing BDVal

This section explains how to install and configure BDVal. We assume that you have downloaded and uncompressed BDVal as described in the previous section.

As a sanity check, try the following:

```
cd $BDV_INSTALL_DIR
java -jar bdval.jar --version
```

You should see a help message starting with a line which indicates which version of BDVal is running, as shown below (we truncate the output):

```
INFO [main ] VersionUtils          - org.bdval.DiscoverAndValidate
Implementation-Version: release bdval_1.0.1
INFO [main ] DiscoverAndValidate - org.bdval.DiscoverAndValidate
Implementation-Version: release bdval_1.0.1
TRACE [main ] TimeLoggingService - TIMING:modelId:no-model-id:MODE:no-
mode-argument:START
  (-m|--mode) <mode>
    Mode of execution, one of: leave-one-out, svm-weights, cox-
regression,
    write-model, ga-wrapper, svm-weights-iterative,
    distribution-difference-by-feature, kendal-tau, distribution-
difference,
    sequence, stats, cross-validation, rserve-status, fold-change, t-
test,
    predict, reformat, define-splits, stats-maqcii, to-ranks, min-max,
execute-
. . .
```

The help message indicates which option BDVal expects. The key option is shown on top: `-m` or `--mode`, which indicates what mode of operation you want to run.

Modes belong to the following categories:

- **Feature selection:** t-test, fold-change, kendal-tau, min-max, smv-weights, svm-weights-iterative, ga-wrapper.
- **Validation protocols:** cross-validation, leave-one-out
- **Embedding feature selection steps within cross-validation:** sequence, define-splits, execute-splits
- **Generate a model:** write-model
- **Predict with a model:** predict

Many BDVal modes of operations share common options. However, each mode may also have its own set of options. You can learn about mode-specific options by running:

```
java -jar bdval.jar --mode <mode> --help
```

These options are also described in the online BDVal documentation, but the help messages in the latest release always contain the information that corresponds to the version of BDVal you are using.

3 Parallel Processing

BDVal will automatically take advantage of as many processors as available on the computer in use. In particular, individual splits of cross-validation are automatically performed in parallel. You may explicitly control the number of processors/threads used by BDVal. Two methods are used:

1. If using the `bdval.jar` file directly, provide the `-Dpj.nt=n` directive to define a property called `pj.nt` (parallel java, number of threads) and set its value to the desired number of threads. For instance, the following will instruct BDVal to run on 5 threads:

```
java -Dpj.nt=5 -jar bdval.jar . . .
```

2. When using an Apache Ant BDVal script, edit the file `config/<project-name>-local.properties` (see detailed instructions in section `Configuring Parallel Processing and Memory Usage`).

4 Third-party software

BDVal relies on third-party open-source software for some operations. You will therefore need to obtain these programs to use all features of BDVal. This section indicates what programs are needed, how they are used by BDVal and how they can be configured.

BDVAL uses R and ROCR in order to perform calculation of some performance measures (i.e., area under the ROC curve, RMSE, MCC and other metrics supported by the optional R package ROCR {{143 Sing,Tobias 2005}}). Rserve is essentially a "bridge" between the BDVAL package and R. Installation of R, Rserve and ROCR is highly recommended since only limited evaluation measures can be produced without these programs.

4.1 R

R is a widely used open source statistical package which provides a user interface, but can also be run in the background {{140 Team,R Development Core 2005}}. R can be downloaded from <http://www.r-project.org/>.

4.2 ROCR

ROCR {{143 Sing,Tobias 2005}} can be conveniently installed as an R add-on package. To install ROCR type the following in the R console:

```
install.packages('ROCR')
```

4.3 RServe

At least one Rserve process must be running for BDVal to evaluate all supported performance metrics. Rserve can be downloaded from <http://www.rforge.net/Rserve/>.

To install Rserve type the following via the R command line

```
install.packages('Rserve',, 'http://www.rforge.net/')
```

Note that if you do not specify the rforge url, you will likely get an older version of the Rserve package.

To start the Rserve process type the following via the R command line

```
library("Rserve")  
Rserve()
```

4.4 Configuring the connection to the R server

The file `BDV_INSTALL_DIR/config/RConnectionPool.xml` indicates to BDVal how to connect to an R server to perform some computations. We provide a template which can be customized. The template is already named `config/RConnectionPool.xml` and is configured to connect to an R server running locally on the same machine as BDVal. Refer to the BDVal web site for details of more advanced configurations (including parallel processing with multiple R servers).

Start the Rserve/R processes and test that BDVal can connect with the following command:

```
java -jar bdval.jar --mode rserve-status
```

If BDVal can successfully connect to R, this command will return:

```
Rserve on localhost:6311 is UP
```

4.5 Apache Ant

While BDVal can be run from the command line and invoked with java (i.e., java -jar bdval.jar) to perform specific steps of the biomarker discovery process, it is often convenient to automate the successive execution of steps. To do so we have developed a variety of scripts which automate most of the biomarker discovery process and provide a consistent organization of the information generated. These scripts are written as Apache *ant* build files. Apache Ant is a tool typically used for software development with Java. Apache Ant is not strictly required for using the binary distribution of BDVal but we highly recommend its use to simplify record keeping and automate many steps of a biomarker discovery project. In this section, we assume that the ant binary is included in the system path (i.e., typing ‘ant’ should find the executable).

Apache Ant can be obtained from <http://ant.apache.org/>.

After Apache Ant is installed successfully, you should be able to type ‘ant’ in a console and see the following message:

```
>ant
Buildfile: build.xml does not exist!
Build failed
```

Apache Ant scripts are included in the BDVal binary distribution (these scripts are located in the data directory, see bdval.xml and prostate-example.xml for instance).

5 Downloading the prostate cancer dataset

As a prerequisite to a biomarker discovery project, one must obtain a dataset suitable for biomarker discovery. Throughout this example, we use the prostate cancer fusion dataset assembled by Setlur and colleagues and made publicly available in the GEO database {{133 Setlur,S.R. 2008}}. The dataset can be downloaded directly from GEO with the accession code GSE8402 (direct URL

ftp://ftp.ncbi.nih.gov/pub/geo/DATA/SOFT/by_series/GSE8402/GSE8402_family.soft.gz).

BDVal can directly read compressed GEO dataset and series soft files.

Most research projects will have data formatted in a tab-delimited file. BDVal supports a strict tab delimited format known as the Columbia Tmm format (because the parser was initially written to read files available in the Tmm database, developed in Paul Pavlidis’ lab at Columbia University). Such files must be named with the .tmm extension and have the following format

Table 1. Microarray tmm format

ID_REF	Sample-Id-1	...	Sample-Id-M
Probeset-Id-1	Signal-value-1-1		Signal-value-1-M
...			
Probeset-Id-N	Signal-value-N-1		Signal-value-N-M

6 Getting organized

Biomarker discovery projects process a variety of information including input datasets, sample label information, gene lists, intermediate feature lists, final model feature lists, models built during evaluation, final models, etc. Our laboratory has found it useful to organize information in a specific and consistent way. This structure is described here because the build scripts expect data and information to be stored in this manner. Following this organization is required if you will use the BDVal apache ant scripts (highly recommended). Other organization schemes could of course work equally well as the one presented here, but will require adapting the bdval.xml script.

We distribute the prostate cancer fusion dataset in the `BDV_INSTALL_DIR/data/bdval/GSE8402/` folder in the binary distribution. This folder contains:

```
BDV_INSTALL_DIR/data/bdval/GSE8402/
    norm-data/
        GSE8402_family.soft.gz
    platforms/
        GPL5474_family.soft.gz
    cids/
        GSE8402-FusionYesNo-TrainingSplit.cids
        GSE8402-FusionYesNo-TestSplit.cids
        GSE8402-FusionYesNo-Validation.cids
    tasks/
        GSE8402-FusionYesNo-TestSplit.tasks
        GSE8402-FusionYesNo-TrainingSplit.tasks
        GSE8402-FusionYesNo-Validation.tasks
    test-sets/
        GSE8402-FusionYesNo-test-samples.txt
        GSE8402-FusionYesNo-validation-samples.txt
    pathways/
        human-IntAct-KEGG-TEPSS-pathways.txt
        GPL5474_gene2probes.txt
    GSE8402-FusionYesNo-TrainingSplit.properties
```

These folders contain:

- The **norm-data** folder contains the biomarker dataset, in GEO soft format, which can be downloaded as described in the previous section, but is also provided in the BDVal distribution.
- The **platforms** folder contains the GEO platform file, in platform soft format, which corresponds to the GSE8402 dataset.
- The **cids** folder contains cids file. Each cids file is a tab delimited file, with two columns which links a sample id to a class label for a specific endpoint. Three cids file are provided in this example, one for each partition of the input file (training set, test set and validation set). The first four lines of the GSE8402-FusionYesNo-TrainingSplit.cids file contain:

```
#fusion GEO-sampleId  
NO      GSM208029  
NO      GSM208030  
YES     GSM208031  
NO      GSM208032
```

The first line is a comment, which provides a reminder of the file content and format. The second line indicates that sample GSM208029 has class label NO. For the prostate cancer fusion endpoint, the symbol NO indicates that no fusion event was detected in the sample. Similarly, the fourth line associates sample GSM208031 with the class label YES, to indicate that the TMPRSS2-ETS gene fusion product was detected in the sample. BDVal supports symbolic class labels. Symbols should be chosen so that they are meaningful in the context of the endpoint. The symbol chosen will be used by BDVal when predicting new samples with the models that it generates, so it is important that the symbol clearly indicate what the model predicts for the sample.

- The **tasks** folder contains tasks files. Each file ending in .task in this folder describes a biomarker classification task. Task files are tab delimited. The file GSE8402-FusionYesNo-TrainingSplit.tasks contains exactly one line:

```
GSE8402-FusionYesNo-TrainingSplit      NO      YES      196      39
```

The first column is a symbol which describes the classification to be performed. BDVal will use this symbol in output reports, so it should be as meaningful as possible. In this case, the classification will use data from GSE8402, predict the fusion endpoint, from training samples only. The second and third columns indicate the symbol of the class labels for which a model should be derived (NO/YES). The next two columns are number

which indicate that there are 196 samples associated with the NO class label, and 39 samples with the YES class label. These numbers are used when reading cids and tasks files together to check that all the samples are accounted for and associated to a class label.

- The **test-sets** folder contains one file per test dataset. Each file has one sample identifier per line. For instance, the first three lines of file GSE8402-FusionYesNo-test-samples.txt contain:

```
GSM208231
GSM208236
GSM208237
```

These lines indicate that samples GSM208231, GSM208236, and GSM208237 belong to the test set.

7 Loading the dataset

Tasks files, cids files, input file and platform files must be specified for most uses of BDVal. A BDVal mode useful to verify that all the information is formatted appropriately is the reformat mode. The following command will load the prostate fusion dataset and output a tab delimited file where samples are listed one per row, and columns are probesets, plus a label column (last column of the file).

```
cd <BDV_INSTALL_DIR>
java -jar bdval.jar -m reformat -p
data/bdval/GSE8402/platforms/GPL5474_family.soft.gz -i
data/bdval/GSE8402/norm-data/GSE8402_family.soft.gz -c
data/bdval/GSE8402/cids/GSE8402-FusionYesNo-TrainingSplit.cids -t
data/bdval/GSE8402/tasks/GSE8402-FusionYesNo-TrainingSplit.tasks -o
reformatted-output.csv
```

Running this command will write a few messages to the console and produce a file named `reformatted-output.csv`. Option `-m` requests the reformat mode, option `-i` indicates which input file should be used. Option `-p` specifies the platform file. The cids file is specified with `-c`, and the task file with `-t`. Output can be redirected to a file named after `-o`. Error messages may indicate the reason(s) why any files cannot be loaded.

8 Defining an evaluation plan

BDVal supports many of the cross-validation approaches. The first step is to decide which evaluation protocol to use for the dataset. For this example, we decide to embed feature selection within cross-validation, use a text book 5 fold cross-validation strategy, with 10 random repeats and with class-label stratification.

To define this evaluation plan, enter the following command:

```
java -jar bdval.jar -m define-splits --folds 5 --stratification true
--cv-repeats 10
-o data/bdval/GSE8402/splits/fusion-cv-5-fs=false.split
-p data/bdval/GSE8402/platforms/GPL5474_family.soft.gz
-i data/bdval/GSE8402/norm-data/GSE8402_family.soft.gz
-c data/bdval/GSE8402/cids/GSE8402-FusionYesNo-TrainingSplit.cids
-t data/bdval/GSE8402/tasks/GSE8402-FusionYesNo-TrainingSplit.tasks
```

The command generates some messages to the console and writes a split-plan file to the specified output: data/bdval/GSE8402/splits/fusion-cv-5-fs=false.split-plan.

The split-plan file indicates how samples in the input file are assigned to cross-validation folds, for each random repeat of cross-validation. The split-plan is saved to a file so that different feature selection strategies can be tested with exactly the same split partitions. The format of this file is defined in the BDVal web documentation. Notice that we recommend writing the split file under data/bdval/GSE8402/splits/. This is also the location where the BDVal ant script will write automatically split plans.

9 Evaluating a feature selection strategy

After choosing an evaluation plan and creating a corresponding split-plan for a dataset, a user can evaluate a feature selection strategy with the BDVal **execute-splits** mode. This mode reads a sequence-file and a split plan and executes the sequence (a feature selection program) for each split of the validation plan. A sequence file is a program that implements a sequence of feature selection steps. Many feature selection programs are provided in the sequence file language and available for use with BDVal. Sequence files are distributed in the data/sequences folder of the distribution. The following example evaluates the baseline sequence file with the validation protocol which we defined in the previous section:

```
java -jar bdval.jar -m execute-splits
--splits data/bdval/GSE8402/splits/fusion-cv-5-fs=false.split
--sequence-file data/sequences/baseline.sequence
--num-features 10
```

```
-p data/bdval/GSE8402/platforms/GPL5474_family.soft.gz
-i data/bdval/GSE8402/norm-data/GSE8402_family.soft.gz
-c data/bdval/GSE8402/cids/GSE8402-FusionYesNo-TrainingSplit.cids
-t data/bdval/GSE8402/tasks/GSE8402-FusionYesNo-TrainingSplit.tasks
```

This command will execute the feature selection steps described in `data/sequences/baseline.sequence` for the 50 evaluation splits of the 5 fold CVx10 defined in `fusion-cv-5-fs=false.split`. At the end of this process, BDVal will calculate performance measures and write these performance statistics to a file called `<YYYYMMDD-TTTT>-results-submission-file.txt`, where `YYYYMMDD-TTTT` encodes the year, month, date and time of job submission. Please note that R, ROCR and Rserve must be correctly configured to run this command successfully. At the end of a successful execution, the file `<YYYYMMDD-TTTT>-results-all-maqcii-submission.txt` will contain two lines. The first line is a header which describes the columns of the file and the second line provides evaluation statistics for the baseline feature selection strategy. Importantly, a file called `model-conditions.txt` is also generated, which logs the parameters provided to BDVal and associates them with a model identifier (`modelId`). The `modelId` also appears in the submission file and makes it possible to track down exactly which parameters yielded which performance measures. New lines are always appended to the submission file and `model-conditions.txt` files. The user should delete these files if their content is no longer required, but in general should archive these results as important documentation about the models generated.

10 Evaluating many feature selection strategies: the BDVal build file

The previous section presented how to evaluate the performance of a single feature selection strategy. In practice, a biomarker discovery project may evaluate tens of feature selection strategies and select the strategy that yields the best cross-validation performance. Varying the number of features used by the model in scans of 5 to 100 features is also frequently done as part of model tuning. BDVal facilitates these activities by offering pre-built Ant scripts to automate these tasks. In this section, we assume that the BDVal project has been customized as described in section “Configuring the BDVal project build file”. The BDVal distribution provides a configured project, which can be used to follow the steps indicated in this section.

10.1 Model evaluation

We recommend evaluating biomarker models following a protocol where feature selection is embedded in the cross-validation loop. BDVal automates such a process, which can be started with the command:

```
cd BDV_INSTALL_DIR/data
ant -f prostate-example.xml
```

After you type this command, you will see:

```
Buildfile: prostate-example.xml
  [echo] Configuration execution for Windows.

prepare-bdval:

tag-output-directory:
  [mkdir] Created dir: d:\dev\tissueinfo_20080903173448\data\20081008-1215-
results
  [echo] Save tag: 20081008-1215-results
  [copy] Copying 4 files to
d:\dev\tissueinfo_20080903173448\data\20081008-1215-results
  [input] Please provide a short description for this run (i.e., condition
tested, summary of parameters).
```

At the prompt, describe the purpose of the run. You may enter any description that will remind you what question the evaluation is designed to answer. The description will be saved together with the results of the evaluation, and is useful for record keeping. If you keep a paper notebook, you may use an identifier that refers back to your notebook, or you may use the description as an electronic record. For this example, you may simply enter ‘Testing BDVal. First ant execution’. After you enter the run description, the ant command will start evaluating the feature selection strategies specified in the file prostate-example.xml. For the sake of this example, the prostate-example.xml distributed in the BDVal distribution is configured to run only the baseline feature selection strategy with 20 features, and CV5x5. With this configuration, the above command may run for an hour or more on a desktop computer (1 processor) or finish in 5-10 minutes on an 8 processor server machine. Actual performance will vary depending on processor and disk speed of the computer used.

At the end of the execution, the BDVal Ant script has generated a result directory and zipped archive of the result directory. The result directory should contain the following:

```

BDV_INSTALL_DIR/data/20081008-1215-results/
    20081008-1215-results-README.txt
    bdval.xml
    prostate-example-local.properties
    prostate-example.properties
    prostate-example.xml

    features/
        GSE8402_FusionYesNo_TrainingSplit/

    models/

        GSE8402_FusionYesNo_TrainingSplit/
    predictions/
        GSE8402_FusionYesNo_TrainingSplit/

20081008-1215-results-all-maqcii-submission.txt

```

The file 20081008-1215-results-README.txt contains the description of the run and describes the parameters which were used during execution. The other four files listed on top are copies of the script and configuration files used during the execution. Copying these files to the result directory makes it possible to link results to the specific conditions used to generate them (the BDVal version number appears in this file, tracking which version of the program was used).

The results directory also contains three sub-folders:

- The folder ‘**features**’ contains one directory per endpoint tested. Each endpoint-specific directory contains the feature lists generated by feature selection, in each split of cross-validation.
- The folder ‘**models**’ contains one directory per endpoint tested. Each endpoint-specific directory lists the models trained in each split of cross-validation. Each model is trained with the list of feature generated in the corresponding split of cross-validation.
- The folder ‘**predictions**’ contains one directory per endpoint tested. Each endpoint-specific directory lists the predictions made by the corresponding model in each split of cross-validation. Prediction files are text files that are human and machine readable. They list the symbol of the class label predicted for each sample of a test fold.

The file 20081008-1215-results-all-maqcii-submission.txt provides the evaluation statistics calculated as average of performance measures obtained across the splits of cross-validation.

10.2 Final Model Construction

After evaluating the baseline feature selection strategy, we proceed to construct a final model for the prostate cancer fusion endpoint. We choose to use the consensus of features identified in each split of cross-validation. We use the consensus list of features to train a model using the entire training set.

With BDVal, this is achieved by executing the following command:

```
ant -f prostate-example.xml generate-final-models
```

The script prompts for a model description file. Accept the default since it names the model-description.txt file which was produced during evaluation.

Next the script prompts for the location of the results directory which contains the features directory to be used to generate a final model. Enter 20081008-1215-results and press Enter.

The script will consider each model id found in the model condition file, look for the corresponding features in 20081008-1215-results/features, determine how many times each feature is found, keep the features which were used most often during cross-validation (breaking ties with the result of a T-test on the entire training set) and keeping only as many features as used in the evaluated model. Consensus features which result from this process are written to the directory 20081008-1215-results/consensus-features, organized by endpoint.

Final models are then trained from the entire training set and written to 20081008-1215-results/final-models. The 20081008-1215-results results directory should now contain:

```
BDV_INSTALL_DIR/data/20081008-1215-results/  
  
    . . .  
    GSE8402_FusionYesNo_TrainingSplit-submission.txt  
  
        features/  
        models/  
        predictions/  
  
        consensus-features/  
        final-models/
```

If you prefer to generate final models by applying the feature selection strategy directly to the entire training set, you can use the command:

```
ant -f prostate-example.xml generate-final-models-direct-method
```

This command will prompt for the same parameters as generate-final-models, and will populate the following directories: consensus-features-direct and final-models-direct. Since models are stored in different directories, it is possible to evaluate models with both approaches and keep the results in the same folder. If you executed both generate-final models and generate-final-models-direct-method, you should now see:

```
BDV_INSTALL_DIR/data/20081008-1215-results/

    . . .
    GSE8402_FusionYesNo_TrainingSplit-submission.txt

    features/
    models/
    predictions/

    consensus-features/
    final-models/

    consensus-features-direct/
    final-models-direct/
```

10.3 Predicting the validation sets

Final models generated in the previous section will now be used to predict samples in a validation set. The Setlur dataset provides two such sets. The first independent set is called test set and contains samples from the same population of patients as the training set. Samples in the test set were set apart randomly at the beginning of the project. The validation set on the other hand contains samples from a different cohort of patients. Let's predict the test set with the model obtained by consensus of features.

We can use the command:

```
ant -f prostate-example.xml evaluate-dataset-statistics
```

This command predicts the samples in the test set and uses the true labels of the samples to estimate performance statistics. The command requests additional information. The first prompt requests the location of the model directory. Indicate 20081008-1215-results/final-models to use consensus of feature models or indicate 20081008-1215-results/final-models-direct to use models derived directly from the feature selection strategy.

The second prompt requests the test set name. Enter ‘test’ (without the quotes) to indicate that the Setlur test set must be used. The name of the test set is used to obtain sample ids, true labels and input file from the project properties file. You would specify ‘validation’ if you wanted to predict the validation dataset, or training if you needed to obtain (over-optimistic) performance on the training set.

The third prompt gives you an opportunity to request sampling with replacement from the test set to estimate standard deviations of the performance measures (1,000 replacement samples are generated in each case for each model).

The last prompt lets you control if performance statistics should be evaluated.

After the evaluate-dataset-statistics command executes (which may take a few seconds or several hours if the model directory contains a large collection of models), the file <endpoint-symbol>-predict-set=<test-set-name>.stats.txt will contain the performance estimates measured on the test/validation set.

If you need to predict class labels for which you do not have labels the evaluate-dataset-statistics command will not have access to true labels, and therefore cannot estimate performance. Instead, this command only generates prediction files, one for each model used to predict the test set. These files are written to the results directory under the predictions/ folder (organized by endpoint). Refer to the BDVal web site for an update to date description of the prediction output format.

11 Configuring the BDVal project build file

This section describes how to configure the BDVal project build file for a new biomarker discovery project. It demonstrates how to create prostate-tutorial files that mimic the prostate-example files used in the previous section.

11.1 Getting organized

The data folder distributed with bdval contains the following files:

```
bdval.properties  
bdval.xml  
prostate-example.properties  
prostate-example.xml
```

The ant build file bdval.xml and its property file (bdval.properties) implement automatic processing for a variety of biomarker projects. These files are designed to be reused as such with a variety of biomarker discovery projects. Because bdval.xml and bdval.properties are designed to be shared, they should not be directly modified. Instead, project-specific files can be created

and customized that leverage resources offered by `bdval.xml`. The `prostate-example.xml` and `prostate-example.properties` files offer an example of project-specific configuration.

11.2 Configuring eval-dataset-root

BDVal defines a variable called *eval-dataset-root* to refer to the root of the directory hierarchy where data files are stored. For this worked example, `eval-dataset-root` should be configured to point to `BDV_INSTALL_DIR/data/GSE8402/`.

This can be accomplished by copying the template configuration file as follows:

```
cp config/prostate-example-local.properties config/prostate-tutorial-  
local.properties
```

The file `config/prostate-tutorial-local.properties` should now contain:

```
eval-dataset-root=bdval/GSE8402
```

The value after `'eval-dataset-root='` indicates where the files are located. The root directory is relative to the data directory in the distribution because this is where the Apache Ant BDVal script should be used. You can change the value to a location not relative to data, but should indicate a full path.

11.3 Configuring parallel processing and memory usage

The file `config/prostate-tutorial-local.properties` also describes the type of computer that BDVal is run on. This property is used to define the number of threads for parallel processing and the total memory available to the java process. The template offers suggestions for each type of computer (desktop or server). We define 8 Gigabyte of memory for a server computer with 8 processors. We define 1.2 Gb of memory for a desktop computer where only one processor should be used by BDVal.

```
# The type of computer BDVal is running on:  
computer.type=desktop  
# The number of parallel threads to use in a server environment:  
server.thread-number=8  
# The amount of memory to use on a server machine (8Gb)  
server.memory=-Xmx8000m  
# The number of parallel threads to use on a desktop machine  
desktop.thread-number=1  
# The amount of memory to use on a desktop machine (1200 Mb)
```

```
desktop.memory=-Xmx1200m
```

11.4 The project properties file

This section explains how to configure the **prostate-tutorial.properties** file. A template is provided in the distribution, which can be copied and customized for each new biomarker discovery project. We start by copying the template to the prostate-example.properties file:

```
cd <BDV_INSTALL_DIR>  
cp data/templates/bdval-template.properties data/prostate-tutorial.properties
```

The configured file is provided with the distribution. This file is called **prostate-example.properties**, you can always refer to it as needed to see what the end result should look like. If you prefer to skip this section, simply copy **prostate-example.properties** to **prostate-tutorial.properties**.

Files ending in .properties are Java properties files and follow a simple syntax: comment lines start with the character # and are ignored. Other lines have the format key=value. Keys may contain dot characters, and by convention, we use dots to introduce a hierarchy of keys.

The BDVal ant scripts uses property files to let users describe commonly used data files and options.

The first key of the file defines a symbol used to refer to the endpoint for which models will be generated:

```
<endpoint-symbol>.dataset-name=<dataset-name>
```

Replace <dataset-name> by something meaningful (e.g., GSE8402).

Choose a symbol and replace every occurrence of <endpoint-symbol> in **prostate-tutorial.properties** by the new symbol (let this symbol be GSE8402_FusionYesNo in this example, to encode the dataset name and the endpoint code (fusion)).

The next line of the template indicates where the dataset file is located:

```
GSE8402_FusionYesNo.dataset-file=${eval-dataset-root}/norm-data/<dataset-filename>.<input-file-extension>
```

Change <dataset-filename>.<input-file-extension> to refer to the prostate dataset GSE8402_family.soft.gz. Notice the reference to *eval-dataset-root*, a variable which indicates the root of the directory hierarchy where input files are located (see section Getting Organized, and

notice that *eval-dataset-root* has value `BDV_INSTALL_DIR/data/GSE8402/`). You should now read:

```
GSE8402_FusionYesNo.dataset-file=${eval-dataset-root}/norm-  
data/GSE8402_family.soft.gz
```

Configuring the rest of the file consists of entering information about each key. We describe the available keys below. Refer to the final `prostate-example.properties` file for the location of each file in the distribution.

- **<endpoint-symbol>.cids-file** Should point to the cids file for the endpoint.
- **<endpoint-symbol>.tasks-file** Should point to the tasks file for the endpoint.
- **<endpoint-symbol>.<test-set-name>.test-samples** Should point to a file with one sample id per line, for the test set named as `<test-set-name>`. Any number of test sets can be defined. The prostate cancer fusion dataset has sets called ‘test’ and ‘validation’.
- **<endpoint-symbol>.<test-set-name>.true-labels** May optionally point to a cids file which provides true labels for the samples defined in the test set. Labels may not be provided if they are unknown.
- **<endpoint-symbol>.<test-set-name>.dataset-file** May optionally point to an input file which contains data for the test set. If `dataset-file` is not filled in, data for the samples in the test set is assumed to be included in the default endpoint input file (property **<endpoint-symbol>.dataset-file**). This is the case for the prostate example dataset, where all the samples are included in the GEO data file.
- **<endpoint-symbol>.platform-file** Should point to the location of the GEO platform file corresponding to the microarray platform used to measure the data. Most platforms are available in GEO. Platform information is required when gene lists are used to restrict the set of features available for analysis. Information to construct platform files is provided on the BDVal web site.
- **<endpoint-symbol>.do-process-gene-lists** Should be true or false. A value of true indicates that feature selection with gene lists should be performed. A value of false performs non-gene list feature selection.
- **<endpoint-symbol>.genelists= Merged HM200 Pomeroy400** provides a space separated lists of gene list names. When specified, only these gene lists will be used for the endpoint. Gene list names must be defined in the `bdval.xml` file. Information about each gene list defined in `bdval.xml` is provided in `BDV_INSTALL_DIR/data/gene-lists`. Each file in this folder describes the genes included in the specific gene list.

- **<endpoint-symbol>.floor=<numerical-threshold>** When a value is provided, it indicates that BDVal should floor the signal value of every probeset on the platform to the threshold. Leaving the property empty disables flooring.
- **<endpoint-symbol>.array-parameters= [--two-color-array] [--logged-array]** Use this property to indicate that the platform is a two-color array (option **--two-color-array**), or that the signal values are the logarithm of raw values (**--logged-array**). These options affect how flooring is performed, so that two color arrays are floored around zero or one. Do not specify these options for single color array.
- **<endpoint-symbol>.pathways-file** Indicate the path to the pathway definition file. The file matching the organism of the data should be used. We provide human, mouse and rat. See the BDVal web site for the syntax of pathway files.
- **<endpoint-symbol>.gene-to-probes-file** Indicate the path to the gene to probe definition file. This file maps each probeset id to a gene id and is required only for pathway runs. See the BDVal web site for the syntax of gene to probe files.

11.5 The project-specific build file

The project-specific build file automates the biomarker discovery process with BDVal and specifies what endpoints should be processed and which feature selection strategies should be used.

This section explains how to configure the **prostate-tutorial.xml** file. A template is provided in the distribution, which can be copied and customized for each new biomarker discovery project. We start by copying the template to the prostate-tutorial.xml file:

```
cd <BDV_INSTALL_DIR>
cp data/templates/bdval-template.xml data/prostate-tutorial.xml
```

The configured file is provided with the distribution. This file is called **prostate-example.xml**, you can always refer to it as needed to see what the end result should look like. If you prefer to skip this section, simply copy **prostate-example.xml** to **prostate-tutorial.xml**.

We reproduce the start of the file below. Replace “bdval-template” on the second line with “prostate-tutorial” to indicate which project name should be used. The project name is used to read various configuration files (e.g., config/project-name-local.properties and data/project-name.properties).

```
<?xml version="1.0" encoding="utf-8"?>
<project name="bdval-template" default="all" basedir="."> <!-- <=== CONFIGURE
THIS-->
```

Define the endpoint symbol to match the information entered in the project properties file. Note that several endpoints can be defined in the same project specific file.

```
<property name="do.<endpoint-symbol>" value="true"/>
```

Also define the list of all valid endpoint symbols. In this case, the configuration should read:

```
<property name="do.GSE8402_FusionYesNo" value="true"/>
<property name="all-endpoints" value="GSE8402_FusionYesNo "/>
```

A few parameters can be controlled by editing the project-specific build file (look for the string “CONFIGURE THIS” as a useful marker). The parameters are described with comment in the build file. For instance:

```
<!-- CONFIGURE THIS
      Number of features to generate models with. This can be a single
      integer, or a list of integers separated by commas. When it is a list,
      models are generated with each number of feature indicated. -->

<property name="num-features" value="5,10,20,30"/>
```

The previous configuration indicates that BDVal should produce models with 5,10, 20 and 30 features.

The type of feature selection task can be configured in the section labeled “all”:

```
<!-- change values to "true" below in order to run the condition listed -->
      <property name="do.baseline" value="true"/>
      <property name="do.naive-bayes" value="false"/>
      . . .
      <property name="do.ttest-svmiterative" value="false"/>
```

Change “false” to “true” to specify that a given feature selection strategy should be used.