DMAP2: User Guide

DMAP2 builds on the original DMAP distribution (Stockwell, *et al*., 2014) by providing scripts to simplify the selection of processing options and running the programs for the various steps.

Overview:
After installation of the various prerequisite software packages and genomic resources, DMAP has a series of steps:

a. Genome preparation to generate files in required form: steps that usually only need doing once for a series of samples. `bismark` mapping needs an indexing step for the genome, whereas `bsmapz` only requires that the genome is in a single file. Information files are created for steps (d) and (e)

b. Adaptor trimming of bisulphite sequence reads for each sample

c. Mapping reads for each sample to the genome with `bismark` or `bsmapz` producing `.bam` or `.sam` files - DMAP2 defaults to `.bam`

d. Comparing mapped data with the DMAP program `diffmeth`

e. Relating genomic fragments or regions of interest to genomic features with the DMAP program `identgeneloc`

DMAP2 groups these steps into a number of scripts each implementing one or more operations. If prior work has already performed some steps (genome preparation, mapping) then those can be omitted and appropriate information put in the appropriate parameter file. While the mapping scripts are based on using the `bismark` or `bsmapz` (formerly `bsmap`) bisulphite aligners, other aligners can be used, in which case alignment files from them can be processed from the `diffmeth` point at step (d).

Script details:

(i) Information common to all scripts is specified in a file like `dmap_basic_params.conf` which defines the following variables:

`verbose` - set to `"yes"` for informational messages
`path_to_dmap` - if `DMAP` executables are not defined on your path, leave blank if they are
`dmap_genome_dir` - directory for the genome fasta file(s)
`dmap_genome_fasta_files` - list of genome fasta files, either one for each chromosome or as a single file
`bisulphite_mapper` - mapper to use, either `bismark` or `bsmapz`
`dmap_bismark_index_location` - where `bismark` index is, defaults to genome location (`dmap_genome_dir`)
`dmap_path_to_bismark_exes` - where `bismark` executables are, blank if already on your path
`bismark_build_options` - options for building `bismark` genomic index
`bismark_run_options` - options for `bismark` at runtime
`bsmapz_exe` - where `bsmapz` executable is, blank if already on your path

`bsmapz_cores` - number of processor cores for `bsmapz` run, determined by your CPU, the more available, the faster the run

`bsmapz_mismatch_val` - if <1 is the proportion of allowable mismatches in a read, if >=1 the actual number of allowable mismatches: setting this too high will reduce the unique mapping and probably increase processing time

`dmap_chr_info_file` - file for `diffmeth` to locate genome fasta files, automatically generated: defaults to `dmap_chr_info.txt`

`feature_annotation_type` - to define annotation format: one of `EMBL`, `Genbank`, `SeqMonk`, `GTF`, `GFF3` or `none` (if no annotation is wanted)

`annotation_file_location` - location of the annotation file(s)

`annotation_files` - list or array of the files: SeqMonk, EMBL or Genbank expect 1 file for each chromosome, GTF & GFF3 use a single file

`dmap_annot_info_file` - name of the information file automatically generated for SeqMonk/EMBL/Genbank annotations: defaults to `dmap_annot_info.txt`

`user_contam_file` - optional name of contaminant adaptor sequences to be trimmed before mapping, leave blank for default file to be created

We have found SeqMonk annotation useful when the locations of TSS (Transcription Start Sites) and CpG Islands are needed, since these are contained in the SeqMonk files. It is also possible to include extra features into SeqMonk files if required. See **Prerequisites** section below for descriptions of how to obtain SeqMonk files or Gencode GTF annotation files.

(ii) `dmap_index_build.sh` – performs the genome preparation (step a) and further sets up details for the annotation step (e). The script takes information from `dmap_basic_params.conf`. For bismark mapping, by default, the indices will be built in the same location as the genome fasta files, but this requires you to have write access to that location. If not then you can specify some other writeable directory in `dmap_bismark_index_location` in `dmap_basic_params.conf`.

Since all the mapping for an experiment would usually be done against the same genome, the index building step should only be needed once.

The annotation preparation depends on the annotation format and produces an information file relating chromosome IDs to files for EMBL, Genbank and SeqMonk formats. This is not needed for GTF/GFF3 formats.

Once the file `dmap_basic_params.conf` has been set up for your environment the script should be run with a command like:

```
./dmap_index_build.sh dmap_basic_params.conf
```

noting that this, for `bismark` mapping, will take some time for the genome indexing process. It may be worth using Linux/Unix tools like `nohup` to keep such jobs running in the background, especially if you need to disconnect from the machine running this operation. Mapping with `bsmapz` does not require an indexing operation, but the `dmap_index_build` step is still required to ensure that genome files are present both as a single file (for `bsmapz`) and multiple files (for `diffmeth`).

The `DMAP` program `diffmeth` needs one genome fasta file per chromosome. In the event that multiple chromosomes or contigs are present in one file they will be made as required by

a script `split_fasta.`awk, automatically generated, which will separate them into individual files, each named by the leading part of the genome file name plus the chromosome fasta ID.

In order to prevent the `bismark` indexing and mapping steps from finding multiple files for each chromosome, when a single file is required, it will be written to a directory (created if needed) `genome_single`. Or if the genome source is a single file it will be split, for `diffmeth` use, into a directory `multi_genome`.

The genome can be `gzip` compressed (`.gz` extension), but will be decompressed by `dmap_index_build.sh` in order to provide appropriate input to `bsmapz` and `diffmeth`. This will also generate `genome_single` and `multi_genome` directories as required.

(iii) `map_bisulphite_reads.sh` performs the steps of adaptor trimming (b) and `bismark` or `bsmapz` mapping (c). Parameters specific to each sample are in a file like `sample_params.conf` (example in Appendix II) where `dmap_run_type` is set to `RRBS` or `WGBS`, needed here since RRBS library preparation adds a CG dinucleotide to the end of each MspI fragment and these require removal during adaptor trimming. In order to avoid issues with overrepresentation of C's central to short reads with paired end mapping, we routinely used read 1 only. See Appendix I for other possible options.

The input fastq files can be `gzip` compressed (`.gz` extension).

`map_bisulphite_reads.sh` will either use an adaptor file specified by `user_contam_file` in `dmap_basic_params.`conf or creates and uses a default adaptor file `contam.fa` for the adaptor trimming step and runs `cleanadaptors` to produce a trimmed file named `<samplename>_at.fastq` where `<samplename>` is taken from the fastq input file name.

For example: the original untrimmed read file `mysample1_R1.fastq` would be adaptor trimmed to `mysample1_R1_at.fastq` then mapped with `bismark` to produce output files like `mysample1_R1_at_bismark_bt2.bam` and `mysample1_R1_at_bismark_bt2_SE_report.txt`. `bsmapz` mapping would generate `mysample1_R1_at.fastq_bsmapz.bam`.

Running the mapping for each sample would be done with a command like:

```
./map_bisulphite_reads.sh dmap_basic_params.conf sample_params.conf
```

again noting that some time will be needed for each sample. The use of `nohup` or some other utility to run these jobs in the background is useful.

Memory requirements for the mapping depends on the genome size and for *Homo sapiens* is typically around 10Gb for `bismark` and 8Gb for `bsmapz`.

(iv) **`dmap_run_diffmeth.sh`** takes mapped data from `bismark` or `bsmapz` and performs the required differential methylation analysis (step d) with parameters from a file which specifies the type of analysis, the input files and the `diffmeth` output file.

diffmeth performs several different types of analysis:

pairwise	Fisher's Exact statistic comparison of two samples
chisquare	Chi Square statistic for differentially methylated regions over 2 or more samples
Anova	Analysis of variance (F statistic) for two or more sets of samples
CpGlist	List methylation for each CpG for one or more samples
binlist	List + and - methylation for each region (bin) for 1 sample
binlist_pc	Append a % methylation column to binlist lines

An example diffmeth parameter file is in Appendix III and, in this case, is for a 3 group Anova run. The important variables are:

dmap_run_type - defines RRBS or WGBS and should match the setting for the map_bisulphite_reads.sh step
diffmeth_run_type - defines the analysis type (Anova, chisquare, pairwise, CpGlist, binlist or binlist_pc) noting that these are case sensitive
file_list1, file_list2, ... each containing a list (array) of bam or sam files for each group.
diffmeth_out_name sets the name of the output file.

diffmeth bases its operation on regions for WGBS or fragments(usually MspI) for RRBS and the term bin applies to those genomic divisions.

file_list requirements for different analyses:

binlist, binlist_pc	a single file in file_list1
CpGlist	one or more files in file_list1
pairwise	two files in file_list1
chisquare	two or more files in file_list1
Anova	two or more sets of files, at least 2 in each in file_list1, file_list2, ...

Anova runs also need anova_group_number and anova_detail to be defined as well as additional lists of file names in file_list2, etc.: one such list for each group. If the variable anova_fold_difference is set to "yes" a column of fold methylation differences will be added to the output.

Further control of diffmeth parameters is *via* the variables:

diffmeth_fragment_min - minimum length of RRBS fragments
diffmeth_fragment_max - maximum length of RRBS fragments
wgbs_window - window length for WGBS runs
min_cpg_hits - minimum number of hits for a CpG to be included in analysis
min_valid_cpgs - number of CpGs meeting min_cpg_hits for a fragment/region to be included
min_sample_count - minimum number of samples fulfilling min_cpg_hits and min_valid_cpgs for a fragment/region to be included
map_init_CpG_to_prev - for 3' matching reads, the initial C is counted in the adjacent RRBS fragment
pr_threshold - reject fragments/regions with Pr above this threshold

For SAM mapping files it may be necessary to define `max_sam_readlength` if the reads exceed 150bp. This is not required for BAM files.

Run the `diffmeth` script with a command like:

```
./dmap_run_diffmeth.sh dmap_basic_params.conf my_diffmeth_params.conf
```

noting that the execution time can range from 20 minutes to some hours, depending on the amount of data and the computer performance. `diffmeth` runs will typically require some 5Gb of RAM or more depending on the genome size, the number of samples and the coverage.

(v) **`dmap_run_genloc.sh`** implements step (e) on the output from `diffmeth` runs or other appropriately formatted input in which `identgeneloc` looks upstream (in the sense of the feature) for the nearest feature of interest to each region/fragment. The minimum input is 3 columns (chromosome ID, start, stop) but more usually the other columns generated by `diffmeth` would be included.

`dmap_run_genloc.sh` requires a parameter file (example in Appendix IV) to specify the source of annotation data and to control other aspects of the run. Variables are:

`genloc_input_file` - name of input file with 3 or more columns
`genloc_output_file` - name of output file, default is `stdout`, but more usefully set to a named file
`genloc_column_number` - number of columns of input file to include, leave as 0 for automatically including all columns
`internal_to_genes` - control whether regions can be located internal to genes, a value of `internal` will permit this, `exon_intron` will further relate regions to intron/exon boundaries
`show_TSS` - for SeqMonk annotation, show nearest upstream transcription start sites (TSS)
`TSS_limit` - if non-zero will limit how far TSS will be scanned from regions, zero means no limit
`show_CpGi` - for SeqMonk annotation, show nearest upstream CpG island
`show_CpGi_internal` - allow CpG islands to be internal to regions
`show_TSS_CpGi_ranges` - show ranges for CpG islands and TSS
`genloc_seqmonk_biotypes` - for SeqMonk annotation restrict features of interest to specific biotypes (e.g. `protein_coding`): multiple values accepted as a list or array
`genloc_GTF_attributes` - for GTF annotation, use the specified attributes, e.g. `transcript_name, transcript_id, gene_name`; multiples allowed as a list or array
`genloc_GTF_featuretypes` - for GTF annotation, specify the desired features as in column 3 of the GTF file, e.g. `gene`, multiples allowed as a list or array

Run the `identgeneloc` script with a command like:

```
./dmap_run_genloc.sh dmap_basic_params.conf geneloc_params.conf
```

`identgeneloc` runs will typically require some 350Mb of memory, depending on the size of the annotation file(s) and would take some 12 minutes or so for input files of 500,000 regions.

Prerequisites:

In order to run DMAP the following prerequisites should be obtained and installed.

(a) Genomes

DMAP2 requires genomes in FASTA format either a single whole genome file or with each chromosome in an individual file: files can be `gzip` compressed (`.gz` externsion). Various sources exist, but one of the most convenient is `https://www.ensembl.org/info/data/ftp/index.html` following the 'DNA (FASTA)' link. Three different versions of each chromosome are offered: `.dna.`, `.dna.rm.` and `.dna.sm.` respectively being complete genome, genome with repeat regions masked as 'N' and genome with repeats in lower case. The first (`.dna.`) is probably the most useful.

(i) As a single file (e.g. for Homo sapiens):

Select the file `Homo_sapiens.GRCh38.dna.primary_assembly.fasta.gz` for download or execute the following at the command line:

```
curl -O \
ftp.ensembl.org/pub/current_fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
```

The complete assembly contains all chromosomes, including mitochondrial, as well as a series of scaffolds with IDs of the form `KI270728.1`. Mapping reads to these may be inconvenient in which case downloading the required multiple files as below may be better. DMAP2 is capable of using compressed genome files so they do not need to be decompressed.

In `dmap_basic_params.conf` set `dmap_genome_file_location` to the directory where this file has been downloaded and fill `dmap_genome_fasta_files` with the complete genome file name.

(ii) As multiple files (e.g. for Homo sapiens):

Files can be downloaded by selecting each individually with the browser or as a group from the command line using a command like:

```
curl -O ftp.ensembl.org/pub/current_fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.chromosome.[1-22].fa.gz
```

followed by

```
curl -O ftp.ensembl.org/pub/current_fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.chromosome.[X-Y].fa.gz
```

In `dmap_basic_params.conf` set `dmap_genome_file_location` to the directory where these files have been downloaded and fill `dmap_genome_fasta_files` with the chromosome file names.

(iii) Decompressing downloaded files:

Not actually necessary since DMAP2 will generate these as required during the `dmap_index_build.sh` phase, however it may be helpful to check the file contents which can be done with commands like:

```
gzip -dc Homo_sapiens.GRCh38.dna.chromosome.1.fa.gz | more
```

or

```
gunzip Homo_sapiens.GRCh38.dna.chromosome.1.fa.gz
```

which will replace the compressed file with the uncompressed version. Compressed fasta files are about 1/3 the size of the uncompressed versions.

(iv) Other genomes:

Various releases are available at `ftp.ensembl.org/pub/` noting that the `fasta` links will return files in the required formats for DMAP2 mapping.

(b) Annotation files in one of the following formats:

(i) SeqMonk - obtained through the SeqMonk mapped sequence data analyser, available from

`https://www.bioinformatics.babraham.ac.uk/projects/seqmonk/`

by starting the application and choosing `File > New Project` then '`Import Genome From Server`' and selecting then downloading the genome of choice. This should create a directory `seqmonk_genomes`, probably in your top level directory, containing the organism and genome version which contain a series of files `1.dat`, `2.dat`,... - these contain the annotations for each chromosome. Set the location of these files in `dmap_basic_params.conf` as `annotation_file_location` and fill `annotation_files` with the list of chromosome files. Set `feature_annotation_type` to `SeqMonk`.

SeqMonk annotation has proven particularly useful when transcription start sites and CpG islands are needed and allows the choice of specific biotypes for the features of interest. Further, it is possible to append other feature information (e.g. enhancers) to each chromosome: Appendix V contains examples of added features.

(ii) GTF - obtained from `https://www.gencodegenes.org/` for mouse or human, by choosing the organism and genome version, then downloading the comprehensive gene annotation file as GTF which will download a file with a name like: `gencode.v39.annotation.gtf.gz` which can be moved to an appropriate location. The `dmap_index_build.sh` step will decompress the file, or you can do this with:

```
gunzip gencode.v39.annotation.gtf.gz
```

producing `gencode.v39.annotation.gtf`. Set the location in `dmap_basic_params.conf` as `annotation_file_location` and `annotation_files` to the filename. Set `feature_annotation_type` to `GTF`.

(iii) GFF3 - obtained as for GTF, with similar basic parameter settings.

(iv) EMBL - obtained from `https://www.ensembl.org/info/data/ftp/index.html` as 'Annotated sequence (EMBL)' or consider using utilities like `ftp` or `curl` to transfer the files at the command line, with a command (for Homo sapiens) like:

```
curl -f -O "ftp.ensembl.org/pub/current_embl/homo_sapiens/Homo_sapiens.GRCh38.[108-120].chromosome.[X-Y].dat.gz"
```

where `[108-120]` will find the latest version. It is also likely to produce some error messages for the versions it can't find, but these can be ignored. X and Y chromosomes will need `[1-22]` replaced by `[X-Y]`.

The `dmap_index_build.sh` step will decompress the files or you can decompress them with a command like:

```
gunzip Homo_sapiens.GRCh38.105.chromosome.*.dat.gz
```

Set the location of these files in `dmap_basic_params.conf` as `annotation_file_location` and fill `annotation_files` with the list of chromosome files. Set `feature_annotation_type` to `EMBL`.

Note that EMBL (and Genbank) feature tables obtained from ENSEMBL only return ENSEMBL gene IDs with DMAP2.

(v) Genbank - available from NCBI, though hard to find there and comes with awkward chromosome IDs. Better obtained similarly to EMBL from Ensembl as 'Annotated sequence (GenBank)' or at the command line with something like:

```
curl -f -O "ftp.ensembl.org/pub/current_genbank/homo_sapiens/Homo_sapiens.GRCh38.[108-120].chromosome.[1-22].dat.gz"
```

as above. Set `feature_annotation_type` to `Genbank`.

For SeqMonk, EMBL and Genbank annotation formats the list of file names and chromosome IDs are written to the file set in `dmap_annot_info_file`, which defaults to `dmap_annot_info.txt`.

(c) Bismark: we have generally found `bismark` to be a good performer for vertebrate genomes, available from `https://github.com/FelixKrueger/Bismark` or from `https://www.bioinformatics.babraham.ac.uk/projects/bismark/`. Installation is as per instructions, which amount to putting the perl executables into an appropriate directory: either system ones (`/usr/local/bin` - system privilege required) or your own top level `bin`.

You can set the variable `bismark_run_options` in `dmap_basic_params.conf` to reflect your needs: we have found that '`-N 1`' restricts the permitted seed mismatches to 1 (defaults to 2) works well for our data and gives significantly faster mapping. The option '`--bowtie2`' is not now necessary for recent bismark versions, but remains for completeness.

(d) bowtie2: recent versions of `bismark` require the aligner bowtie2. If not already installed in your systems it is available from `https://github.com/BenLangmead/bowtie2` or `https://sourceforge.net/projects/bowtie-bio/files/bowtie2/`, the latter providing precompiled binaries which can be unzipped and installed in an appropriate directory.

(e) samtools: required by `bismark` in order to generate bam rather than sam files. (bam files are about 1/3 the size of the corresponding sam files.) If not already installed, it is available from `https://sourceforge.net/projects/samtools/` as source code which will need unpacking and compiling using instructions like:

```
bzip2 -dc samtools-1.16.tar.bz2 | tar -xvf -
cd samtools-1.16
./configure
make
make install
```

The latter instruction may require system privilege (e.g. sudo). Alternatively you can copy the executable to a location on your PATH with something like:

```
cp samtools ~/bin/
```

(f) bsmapz: replaces the original `bsmap` and is available as source code from `https://github.com/zyndagj/BSMAPz`. `bsmapz` runs much faster than earlier versions of `bsmap` and works well with a wide range of genomes. Note that the methods of resolving multiply-mapped reads differ between `bsmapz` and `bismark` so they do not produce identical results - it is your choice which you prefer. Building `bsmapz` requires `C` and `C++` compilers and the install process is described in the distribution.

(g) DMAP: available as source code from `https://github.com/peterstockwell/DMAP` either by downloading as a zip archive from 'Code->Download ZIP' and using unzip or from the command line with:

```
git clone https://github.com/peterstockwell/DMAP
```

producing either `DMAP-master` or `DMAP` which contains instructions for building in `README.md`.

The source code should compile with any reasonable C compiler, but note that the library `zlib` is needed for full functionality with .bam files. The main executables needed are `diffmeth` and `identgeneloc` which should be copied to an appropriate directory on your PATH or you can set `path_to_dmap` in `dmap_basic_params.sh` to the actual location.

(g) DMAP2: available from:
`https://github.com/peterstockwell/DMAP2`

with a web browser or from the command line with:

```
git clone https://github.com/peterstockwell/DMAP2
```

which will produce a directory `DMAP2_master` or `DMAP2` containing the following:

`README.md` - distribution notes
`DMAP2_UserGuide.pdf` - this document. Further documentation is present in the script files as commments
`Scripts` - directory containing the main DMAP2 scripts:
- `dmap_index_build.sh` - to generate bismark indices and organise the genomic annotation
- `map_bisulphite_reads.sh` - to adaptor trim reads and map with `bismark` or `bsmapz`
- `dmap_run_diffmeth.sh` - to calculate differential methylation with the DMAP program `diffmeth`
- `dmap_run_genloc.sh` - to relate sequence regions or fragments to genomic features

`Example_params` - directory containing example parameter scripts for the various steps:
- `dmap_basic_params.conf` - to provide basic project details for all DMAP2 scripts
- `diffmeth_run_params.conf` - `diffmeth` configuration file in Appendix III
- `sample_params.conf` - typical sample parameters for the mapping step for map_bisulphite_reads.sh
- `dmap_anova2_params.conf` - parameters for `diffmeth` with 2 group Anova statistic
- `dmap_anova3_params.conf` - for 3 group Anova statistic
- `dmap_chisq_params.conf` - for differential methylation analysis by Chi Square statistic
- `dmap_pairrwise_params.conf –` for Fisher's Exact pairwise analysis
- `dmap_cpglist_params.conf` - list CpG counts and positions for a single sample for each fragment or region
- `dmap_binlist_params.conf` - list methylation counts for a single sample for each fragment or bin
- `genloc_params.conf` - parameters for `identgeneloc` to relate fragments/regions to genomic features

parameters for SRA examples:

`test_dmap_basic_params.conf`	- basic parameters for test data runs
`srr18283145_sample_params.conf`	- mapping parameters for SRR18283145.
`srr_145_146_diffmeth_pairwise.conf`	- for `diffmeth` pairwise run
`srr_145_146_genloc_params.conf`	- for `identgenloc` run on pairwise data

You may find it helpful to copy the scripts into the working directory for your project, especially `dmap_basic_params.conf` and the example parameter scripts which will need editing to suit your data. The example run commands shown above assume that these files are all in your project working directory.

References:

Stockwell, P.A., Chatterjee, A., Rodger, E.J. and Morison, I.M. "DMAP: Differential Methylation Analysis Package for RRBS and WGBS data" *Bioinformatics* (2014) DOI: 10.1093/bioinformatics/btu126.

Krueger, F., and Andrews, S.R., Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications, *Bioinformatics*, (2011), https://doi.org/10.1093/bioinformatics/btr167

Yuanxin Xi and Wei Li, "BSMAP: whole genome bisulfite sequence MAPping program" (2009) BMC Bioinformatics 2009, 10:232.

Appendix I: Options for mapping paired end reads.

The authors of `bismark` (Felix Krueger and Simon Andrews, 2011) advise that the use of paired end reads with RRBS fragments will result in over-representation of the overlapping portion of short fragments, given that the 40-220 bp size selection will generate many reads shorter than read lengths presently available. A possible solution is to use tools like `FLASH` (`http://www.cbcb.umd.edu/software/flash`) to generate longer reads where paired end reads clearly overlap. This process generates 3 fastq output files: longer overlapped reads, R1 non-overlapped and R2 non-overlapped. Since `bismark` is only configured to accept either single ended files (or a list of them) or paired end files for R1 & R2 (or a list for each), a strategy would be for two `bismark` runs, one single-ended for joined reads, the other paired-ended for unjoined reads, then use `samtools merge` to combine the results before continuing with `DMAP` analysis. Our experience was that this did not improve the mapping efficiency sufficiently to justify the extra work, hence our decision to work with single-ended data.

Appendix II: An example sample parameter file for mapping:

```
# sample_params.conf: parameters to describe a single bisulphite
# sequence sample, including the raw fastq file(s) and any
# trimming that might be justified by the read qualities.

# identify if this was RRBS or WGBS: this determines a number
# of downstream operations, including adaptor trimming and
# differential methylation calculations.  Values are RRBS or WGBS.

dmap_run_type="RRBS";
# dmap_run_type="WGBS";

# name of adaptor-trimmed output dir:

adtrimmed_out_dir="./";

# array of sample file name(s): we usually only map read 1, so
# read 2 name would be omitted.


dmap_sample_files=(
"NC-P-20_1.fastq"
)

# hard trim length, leave 0 for no hard trimming.

dmap_sample_trim_length=0;

# minimum read length for retention: usually set this to 20

dmap_min_read_length=20;

# mapping output directory

mapping_out_dir="bismark_out/";
```

Appendix III: Example parameter file for `diffmeth` runs

In this case an Anova run with three groups.

```
# dmap_anova3_params.conf: parameters to define differential
# methylation run type and mapped files for the DMAP diffmeth
# program.

# These parameters are for a 3 group Anova analysis.

# identify if this was RRBS or WGBS: this determines a number
# of downstream operations, including adaptor trimming and
# differential methylation calculations.  Values are "RRBS" or "WGBS"
# where:
# "RRBS" compares methylation on fragments (usually MspI) in size range
# "WGBS" compares methylation on defined windows or regions
#
# this is previously specified for the adaptor trimming and mapping
step
# and obviously the values here should be consistent with those.

dmap_run_type="RRBS";
# dmap_run_type="WGBS";

# type of diffmeth run: values are one of "pairwise" "chisquare"
"Anova" "CpGlist" "binlist" or "binlist_pc"
# where:
# "pairwise"  runs Fisher's Exact statistic on two samples
# "chisquare" runs a Chi Square test on multiple samples to identify
fragments/regions with greatest variation
# "Anova"    run Analysis of Variance on two sets of multiple samples,
producing F statistic
# "CpGlist"   lists methylation counts for each CpG
# "binlist"   lists methylation information for each fragment/region
# "binlist_pc" same as binlist but appends % methylation to each line

diffmeth_run_type="Anova";

# Anova requires a group number to define how many groups are involved
# and a list of at least 2 mapping output files for each group
# a further parameter defines the amount of information returned, being
# one of: "anova_simple" "anova_medium" or "anova_full"
# where
# "anova_simple" returns F statistic and its probability
# "anova_medium" adds sample counts and greater methylated group
# "anova_full"   further adds methylation figures for each CpG in
fragment/region

anova_group_number=3;

file_list1=(
 "./rundata/comb_D1.bam"
 "./rundata/comb_D2.bam"
 "./rundata/comb_D3.bam"
 "./rundata/comb_D4.bam"
);

file_list2=(
 "./rundata/comb_D5.bam"
 "./rundata/comb_D6.bam"
 "./rundata/comb_D7.bam"
```

```
);

file_list3=(
 "./rundata/comb_D8.bam"
 "./rundata/comb_D9.bam"
 "./rundata/comb_D10.bam"
)

anova_detail="anova_full";

# fold difference is not relevant for > 2 anova groups

anova_fold_difference="no";

# diffmeth requires a fragment size specification, even for WGBS
(ignored then)

diffmeth_fragment_min=40;
diffmeth_fragment_max=220;

# "WGBS" requires a window (tile) length

wgbs_window=1000;

# output file name for diffmeth

diffmeth_out_name="diffmeth_output_anova3.txt";

# threshold criteria for including fragment/regions

# No. of hits per CpG for fragment/region to be valid
# default is 1

min_cpg_hits=10;

# No. of CpG of fragment/region that must meet min_cpg_hits criterion
# 0 indicates no check for this

min_valid_cpgs=0;

# Minimum No. samples reaching criteria for a fragment/region to be
included
# defaults to 2

min_sample_count=6;

# For RRBS 3' mapping reads, count initial CpG to previous fragment.

map_init_CpG_to_prev="yes";

# Probability threshold: omit fragments/regions with Pr greater.
# default (1.0) implies no filtering

#pr_threshold=1.0;

# Max read length for .SAM files (not needed for .BAM, though)
# defaults to 150bp, for 0

#max_sam_read_length=150;
```

Appendix IV: Example parameter file for `dmap_run_genloc.sh`

```
# genloc_run_params.conf: parameters to define gene location
# operation with identgeneloc.  Some relevant parameters
# are already set with the basic parameters, this file
# relates to the specific settings required for a project.

# file of chromosome & coordinate data (min 3 columns) as:
# chrid start stop
#

genloc_input_file="diffmeth_output_anova_3grp.txt";

# name of output file: identgeneloc writes to stdout, we need
# to direct that somewhere, else will still got to stdout

genloc_output_file="genloc_anova_3grp.txt";

# the number of columns can be identified automatically
# putting a non-zero value here will override the automatically
# chosen value

genloc_column_number=0;

# Control whether regions internal to genes are allowed
# and whether they are to be related to exon/intron
# boundaries.

# values for the parameter internal_to_genes are:

# "" = disallow
# "internal" = allow
# "exon_intron" = relate to exon/intron boundaries

internal_to_genes="exon_intron";

# SeqMonk annotations define TSS and CpG islands
# ignored for other annotation formats

show_TSS="yes";

# limit for TSS ranges, zero=nolimit

TSS_limit=0;

show_CpGi="yes";

# CpG islands can be shown within regions: "yes" permits this

show_CpGi_internal="yes";

show_TSS_CpGi_ranges="yes";

# For SeqMonk annotations, can restrict to particular biotypes;
# valid values are: 3prime_overlapping_ncrna IG_C_gene IG_C_pseudogene
IG_D_gene IG_J_gene IG_J_pseudogene
# IG_V_gene IG_V_pseudogene Mt_rRNA Mt_tRNA Mt_tRNA_pseudogene TEC
TR_C_gene TR_D_gene TR_J_gene TR_J_pseudogene
```

```
# TR_V_gene TR_V_pseudogene ambiguous_orf antisense lincRNA miRNA
miRNA_pseudogene misc_RNA misc_RNA_pseudogene
# ncrna_host non_coding non_stop_decay nonsense_mediated_decay
polymorphic_pseudogene processed_transcript
# protein_coding pseudogene rRNA rRNA_pseudogene retained_intron
scRNA_pseudogene sense_intronic sense_overlapping
# snRNA snRNA_pseudogene snoRNA snoRNA_pseudogene tRNA_pseudogene
transcribed_unprocessed_pseudogene

# protein_coding is probably one of the most useful.

genloc_seqmonk_biotypes=(
"protein_coding"
)

# GTF & GFF3 annotations allow selection of attributes and feature
types
#
# Attributes default to "transcript_name,transcript_id,gene_name" if
# not specified.  Actual attributes can vary widely for GTF format


genloc_GTF_attributes=(
"transcript_name"
"transcript_id"
"gene_name"
)

# feature types default to "exon,CDS,start_codon,stop_codon,transcript"
if
# not specified.  Valid values for GTF are are any of:
#
CDS,start_codon,stop_codon,exon,intron_CNS,intron,5UTR,3UTR,CNS,inter,t
ranscript,gene
# GFF3 values are far more diverse


genloc_GTF_featuretypes=(
"exon"
"CDS"
"start_codon"
"stop_codon"
"transcript"
)
```

Appendix V: Example of additional SeqMonk annotation appended to a chromosome feature file

```
FT   gene            856539..856757
FT                   /description="enhancer chr1:856539-856757"
FT                   /name="enh_856539"
FT                   /score="2"
FT                   /biotype="enhancer"
FT   gene            858256..858648
FT                   /description="enhancer chr1:858256-858648"
FT                   /name="enh_858256"
FT                   /score="5"
FT                   /biotype="enhancer"
FT   gene            868565..868684
FT                   /description="enhancer chr1:868565-868684"
FT                   /name="enh_868565"
FT                   /score="2"
FT                   /biotype="enhancer"
```