The GroopM manual

Version 0.2.10.14 Last updated 20th December 2013

GroopM was developed to be used in conjunction with a specific experimental design pattern. Before you try GroopM please ensure:

- You are using a MODERN sequencing platform. Preferably Illumina based
- You have sampled your metagenomic community at at least 3 time points / spatial positions

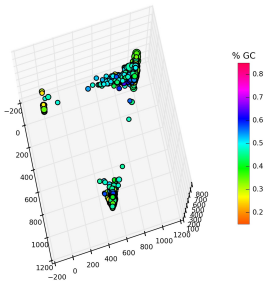
Please be aware. GroopM will not work well (or at all) in the following situations:

- For sets of unrelated metagenomes
- For fewer than 3 samples
- For low (very) coverage data sets
- For (very) bad assemblies

How do you know if one of these (unfortunate) things describes your data?

GroopM needs at least 3 samples at reasonable depth and a pretty decent collection of long contigs to bootstrap itself into action. Not all your contigs need to be super long, just a decent chunk. If your longest contig is only 3 Kbp long then GroopM may not do very well. Perhaps you need to review your sampling / filtering / assembly / post assembly workflows. GroopM requires that your samples come from *related metagenomes*. By this we mean that you have sampled a collection of similar habitats, for example 20 human gut microbiomes OR a set of replicated reactors OR the same habitat multiple times. GroopM assumes that a substantial number of reads from each sample will map to most organisms in all samples. If your environments differ too greatly GroopM will probably choke. To see how you'll go you can parse in the data and then run:

```
$ groopm explore -m allcontigs <db>
```



If you see only clusters of contigs at the extremities of the resulting plot (as shown here) then GroopM probably won't work for your data.

GroopM has been designed to be user friendly. So far we've used it to get good results from a wide variety of habitats with highly varying sequencing approaches.

There's no need to be scared. Chances are, GroopM will work with your data.

For impatient people

1) Make a co-assembly of ALL of your data using Velvet or similar
2) Map each of your read sets to these contigs using BWA or similar
3) $ groopm parse db.gm contigs.fa *.bam
4) $ groopm core db.gm
5) $ groopm refine db.gm
6) $ groopm recruit db.gm
7) $groopm extract db.gm contigs.fa

Table of Contents

Data storage

GroopM stores all of it's data in one database. A single databse provides two advantages. 1) Each data type used by GroopM is stored as a table or group of tables within the database. Thus users do not need to worry about the format and location of several csv files. 2) Other programs can be made aware of this database structure and can add to or modify its contents.

Currently we use hdf5 for this but we are shortly going to move over to using plain old sqllite3. The database used for groopm should have the suffix ".gm" or ".sm". Technically you can make it whatever you like, but this is the convention we have adopted. The current implementation uses standard hdf5 layout, so you can view and modify the contents of the database with third party tools such as hdfview. ALL GroopM commands expect to be given the database as a parameter. We will denote it here as <db>. We are working on publication about the standardization of this data structure as well as programming APIs to access and modify its structure. If this is something you think would interest you, please contact me at mike@mikeimelfort.com.

GroopM command line overview

GroopM is executed from the command line and can be run in several different "modes". Some modes are used to build or modify the .gm database, some produce bins and others print or visualise data. This is a brief overview of GroopM commands. Detailed information for each command is given in the sections below.

Typical work flow commands:

groopm parse	→ Load the raw data (contigs /mappings) and create database
groopm core	→ Create core bins
groopm refine	→ Interactively refine core bins (merge / split etc.)
groopm recruit	→ Add unbinned contigs to the cores / enlarge cores
groopm extract	→ Extract binned contigs or read identifiers

Manual bin editing utilities:

groopm merge	→ Merge two or more bins
groopm split	→ Split a bin into N parts
groopm delete	→ Delete a bin

Printing, visualisation and data extraction:

groopm explore	→ Methods for visualising bins and contigs
groopm plot	→ Plot bins
groopm highlight	→ Highlight individual bins and apply labels
groopm print	→ Print summary statistics about bins
groopm dump	→ Write database fields to csv

Typical work flow overview and corresponding commands

Before you can use GroopM you'll need to assemble and map your reads. The general recipe is to make a co-assembly of ALL of your data using Velvet or similar. Take these contigs and map each of your read sets to them using BWA or similar. If you have N sampling points then your aim is to produce N sorted-indexed BAM files. samtools can help with this.

The typical workflow for GroopM is as follows:

- **parse** → Load then contigs and coverage information and create database
- **core** → Produce a set of 'core' bins
- **refine** → Make sure these bins are ok. Fix any errors you see
- **recruit** → Recruit smaller contigs into your existing bins
- **extract** → Extract binned contigs

GroopM was designed to be as parameter-free as possible. For more information on any of these steps type:

```
$ groopm OPTION -h
```

See below for more detailed information on each of these steps.

Loading your data into GroopM

GroopM parse

Description:

Load the raw data (contigs /mappings) and create database

Required arguments:

dbname	name of the database being created
reference	fasta file containing bam reference sequences (your contigs)
bamfiles	bam files to parse

Optional arguments:

-f --force	[False]	overwrite existing DB file without prompting

Example usage:

```
$ groopm parse database.gm contigs.fa sample1.bam sample2.bam sample3.bam …
```

This will parse your contigs (contigs.fa) and corresponding bam files (Sample1.bam, ...) and store the information in database.gm. You will use this database in all remaining GroopM steps.

Creating “core” bins

GroopM core

Description:

Create a set of high quality “trusted” bins

Required arguments:

dbname	name of the database to open	

Optional arguments:

-c --cutoff	[1500]	cutoff contig size for core creation
-s --size	[10]	minimum number of contigs which define a core
-b --bp	[1000000]	cumulative size of contigs which define a core regardless of number of contigs
-f --force	[False]	overwrite existing DB file without prompting
-g –graphfile		output graph of micro bin mergers
-p --plot	[False]	create plots of bins after basic refinement
-m --multiplot	[0]	create plots during core creation – (0-3) MAKES MANY IMAGES!

Example usage:

```
$ groopm core database.gm
```

This will bin all contigs longer than the cutoff length. You can modify this using the -c option. Bins will be stored in the database.

Refining bins

GroopM refine

Description:

Examine your bins and try fix any errors you find

Required arguments:

dbname	name of the database to open	

Optional arguments:

-a --auto	[False]	automatically refine bins
-r --no_transform	[False]	skip data transformation (3 stoits only)
-p --plot	[False]	create plots of bins after refinement

Example usage:

```
$ groopm refine database.gm
```

This will start GroopM's interactive bin editing workflow. MORE about this soon!

Recruiting unbinned contigs

GroopM recruit

Required arguments:

dbname	name of the database to open	

Optional arguments:

-c --cutoff	[500]	cutoff contig size
-f --force	[False]	overwrite existing db file without prompting
-s --step	[200]	step size for iterative recruitment
-i --inclusivity	[2.5]	make recruitment more or less inclusive

Example usage:

```
$ groopm recruit database.gm
```

This will recruit unbinned contigs into your core bins. By default it tries to recruit all unbinned contigs. You can specify a lower length cutoff using the '-c' option. Increasing the '-i' option makes your bins bigger, but not necessarily better.

Extracting your bins

GroopM extract

Required arguments:

dbname	name of the database to open
data	data file(s) to extract from bam or fasta

Optional arguments:

-b --bids	[]	bin ids to use (None for all)
-c --cutoff	[0]	cutoff size (ignored for reads, 0 = no cutoff)
-m --mode	[contigs]	what to extract [reads contigs]
-o –outfolder	[]	write to this folder (None for current dir)
-B --no_separate_bams	[False]	use one file for all stoits

Example usage:

```
$ groopm extract database.gm contigs.fa
$ groopm extract database.gm -m reads Sample1.bam Sample2.bam
```

The first usage will create a collection of multiple FASTA files, one for each bin, that contain binned contigs. The second usage will extract read headers from the bam files provided and create one file of headers for each bin. If the user specifies the '-B' option, then GroopM will produce B * M output files where B is the number of bins and M is the number of BAM files provided. Each file will contain read headers from one bin and from one sample.

Manual bin editing utilities

GroopM merge

Required arguments:

dbname	name of the database to open
bids	bin ids to merge.

Optional arguments:

-f --force	[False]	merge without prompting

Example usage:

```
$ groopm merge database.gm 5 10
$ groopm merge database.gm 5 10 11
```

The first usage will merge bins 5 and 10. You will be given the chance to review the merge before confirming (GroopM will produce some plots). If you don't want this then use the '-f' option. The contigs from bin 10 will be placed into bin 5 and bin 10 will be deleted. The second usage will merge bin 10 with bin 5 and then bin 11 with bin 5.

GroopM split

Required arguments:

dbname	name of the database to open
bid	bin id to split
parts	number of parts to split the bin into

Optional arguments:

-m --mode	[kmer]	profile to split on [kmer, cov, length]
-f --force	[False]	split without prompting

Example usage:

```
$ groopm split database.gm 7 3
$ groopm split database.gm -m cov 7 3
```

The first usage will split bin 7 into three parts using k-means clustering based on tetranucleotide signatures. You will be given the chance to review the split before confirming (GroopM will produce some plots). If you don't want this then use the '-f' option. The second usage will do the same type of split but use coverage profiles instead.

GroopM delete

Required arguments:

dbname	name of the database to open
bids	bin ids to delete

Optional arguments:

-f –force	[False]	delete without prompting

Example usage:

```
$ groopm delete database.gm 87
$ groopm delete database.gm 87 88 89
```

The first usage will delete bin 87. The second usage will delete bins 87, 88 and 89.

Printing, visualisation and data extraction

Extracting data from the database

groopm print

Description:

Print summary statistics about bins.

Required arguments:

dbname	name of the database to open

Optional arguments:

-b --bids	[]	bin ids to print (None for all)
-o --outfile	[]	print to file not STDOUT
-f --format	[bins]	output format [bins contigs]
-u --unbinned	[False]	print unbinned contig IDs too

Example usage:

```
$ groopm print database.gm
$ groopm print database.gm -f contigs -b 7 4
```

The first usage will print some summary statistics about all the bins including the number of contigs, average GC content etc. The second usage will print detailed information about bins 7 and 4 on a per contig basis.

GroopM dump

Required arguments:

dbname	name of the database to open

Optional arguments:

-f --fields	[names,bins]	fields to extract*
-o --outfile	[GMdump.csv]	write data to this file
-s --separator	[,]	data separator
--no_headers	[False]	don't add headers

*Build a comma separated list from [names mers gc coverage tcoverage ncoverage lengths bins] or just use all

Example usage:

```
$ groopm dump database.gm
$ groopm dump database.gm -f names,coverage,bins,gc -o summary.csv
```

More soon

Visualising your bins

groopm plot

Required arguments:

dbname	name of the database to open

Optional arguments:

-b --bids	[]	bin ids to plot (None for all)
-t --tag	[BIN]	tag to add to output filename
-f --folder	[]	save plots in folder
-p --points	[False]	ignore contig lengths when plotting
-C --cm	[HSV]	set colormap*

*[HSV Accent Blues Spectral Grayscale Discrete DiscretePaired]

Example usage:

```
$ groopm plot database.gm
```

More soon

groopm highlight

Required arguments:

dbname	name of the database to open	

Optional arguments:

-L --binlabels	[]	replace bin IDs with user specified labels (use 'none' to force no labels)
-C --contigcolors	[]	specify contig colors (use 'none' to force no colors)
-r --radius	[False]	draw placement radius to help with label moving
-c --cutoff	[1000]	cutoff contig size
-e --elevation	[25.0]	elevation in printed image
-a --azimuth	[-45.0]	azimuth in printed image
-f --file	[gmview]	name of image file to produce
-t --filetype	[jpg]	type of file to produce
-d --dpi	[300]	image resolution
-s --show	[False]	load image in viewer only
-p --points	[False]	ignore contig lengths when plotting
-b --bids	[]	bin ids to plot (None for all)

Example usage:

```
$ groopm highlight database.gm
$ groopm highlight database.gm -s -L bin_labels.csv -t png -f myBins
$ groopm highlight database.gm -L bin_labels.csv -f myBins -d 600 -a 20 -e 95
```

More soon

GroopM explore

Required arguments:

dbname	name of the database to open	

Optional arguments:

-b --bids	[]	bin ids to plot (None for all)
-c --cutoff	[1000]	cutoff contig size
-m --mode	[binids]	Exploration mode*
-r --no_transform	[False]	skip data transformation (3 stoits only)
-k --kmers	[False]	include kmers in figure (only used when mode == together)
-p --points	[False]	ignore contig lengths when plotting
-C --cm	[HSV]	set colormap**

*[binpoints binids allcontigs unbinnedcontigs binnedcontigs binassignments compare sidebyside together]

**[HSV Accent Blues Spectral Grayscale Discrete DiscretePaired]

Example usage:

```
$ groopm explore database.gm
$ groopm explore database.gm -m allcontigs
```

More soon

Appendix 1 .gm (.sm) database layout

This is the structure of the PyTables implementation of the data storage used in GroopM.

METADATA

Group: 'meta'

Description: Information about the database, bins and contigs

Metadata

Table name: 'meta'

Description: Information about the data set and DB

Fields:

'stoitColNames'	: String (512) :	Names of the BAM files supplied to parse
'numStoits'	: Int :	Number of samples
'merColNames'	: String (4096) :	Concatenated Nucl strings of kmers used
'merSize'	: Int :	Length of the kmer used in the signature
'numMers'	: Int :	Number of kmers used to make signatures
'numCons'	: Int :	Number of contigs stored in DB
'numBins'	: Int :	Number of bins stored in DB
'clustered'	: Bool :	Set to true after clustering is complete
'complete'	: Bool :	Unused
'formatVersion'	: Int :	GroopM file version

PC variance

Table name: 'kpca_variance'

Description: Variance of kmer signature PCAs

Fields:

'pc1_var'	: Float :	Variance of 1st kmer principal component
'pc2_var'	: Float :	Variance of 2nd kmer principal component
'pc3_var'	: Float :	
...		

Contigs

Table name: 'contigs'

Description: Information about the contigs being worked with

Fields:

'cid'	: String (512) :	Fasta header of contig

'bid'	: Int :	Bin ID for this contig
'length'	: Int :	Contig length
'gc'	: Float :	GC percentage of this contig

Bins

Table name: 'bins'

Description: Information about the bins

Fields:

'bid'	: Int :	ID of the bin
'numMembers'	: Int :	Number of contigs in thebin
'isLikelyChimeric'	: Bool :	Has the bin been flagged as chimeric

Transformed coverage corners

Table name: 'transCoverageCorners'

Description: Coordinates of the 'corners' in transformed coverage space

Fields:

'x'	: Float :	X-coordinate of the corner point
'y'	: Float :	Y-coordinate of the corner point
'z'	: Float :	Z-coordinate of the corner point

PROFILES

Group: 'profile'

Description: The profile group stores all the information about contigs. All tables in the profile group have the same number of rows. Identical rows in different tables describe the same contig.

Kmer Signature

Table name: 'kms'

Description: Tetranucleotide signatures

Fields:

'mer1'	: Float :	Frequency of first kmer
'mer2'	: Float :	Frequency of second kmer
'mer3'	: Float :	
...		

Kmer Vals

Table name: 'kpca'

Description: Principal components of tetranucleotide signatures

Fields:

'pc1'	: Float :	First principal component
'pc2'	: Float :	Second Principal component
'pc3'	: Float :	
...		

Coverage profile

Table name:	'coverage'
Description:	Raw coverage profiles (coverage in samples)
Fields:	

'stoit1'	: Float :	Coverage in first sample
'stoit2'	: Float :	Coverage in second sample
'stoit3'	: Float :	
...		

Transformed coverage profile

Table name:	'transCoverage'
Description:	Coverage profiles in GroopM transformed coverage space
Fields:	

'x'	: Float :	X-coordinate in transformed space
'y'	: Float :	Y-coordinate in transformed space
'z'	: Float :	Z coordinate in transformed space

Normalised coverage profile

Table name:	'normCoverage'
Description:	Normalised coverage (Euclidean distance from origin to raw coverage profile point)
Fields:	

'normCov'	: Float :	Value

LINKS

Group:	'links'
Description:	Storage of paired read links between contigs (unused in this version)

Links

Table name:	'links'
Description:	Storage of paired read links between contigs
Fields:	

'contig1'	: Int :	Reference to index in meta/contigs
'contig2'	: Int :	Rreference to index in meta/contigs

'numReads'	: Int :	Number of reads supporting this link
'linkType'	: Int :	The type of the link (SS, SE, ES, EE)
'gap'	: Int :	The estimated gap between the contigs