



Version 0.3.0 Last updated 12th November 2014

This document is released under the GNU Free documentation license version 1.3 (<https://www.gnu.org/copyleft/fdl.html>)

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

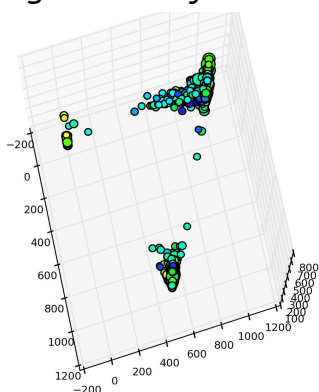
- You are using a MODERN sequencing platform. Preferably Illumina based
- You have sampled your metagenomic community at at least 3 time points / spatial positions

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

For sets of unrelated metagenomes	For (very) low coverage data sets
For fewer than 3 samples	For (very) bad assemblies
For (very) large assemblies (Think Gigabases of contigs)	

How do you know if one of these (unfortunate) descriptions applies to 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 work flows. 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 and die. To see how you'll go you can parse in the data and then run:



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

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`

*BAM files must be indexed and sorted. GroopM has been known to have difficulty parsing Picard-generated BAM files. If this occurs then re-sorting the BAM file with [samtools](#) usually fixes the problem.

Table of Contents

Data storage	4
GroopM command line overview	4
Typical work flow overview and corresponding commands	5
Loading your data into GroopM	6
GroopM parse	6
Creating “core” bins	6
GroopM core	6
Refining bins	7
GroopM refine	7
Recruiting unbinned contigs	8
GroopM recruit	8
Extracting contigs and reads from bins	8
GroopM extract	8
Manual bin editing utilities	9
Merging bins	9
GroopM merge	9
Splitting bins into multiple parts	10
GroopM split	10
GroopM delete	11
Printing, visualisation and data extraction	11
Extracting data from the database	11
GroopM print	11
GroopM dump	12
Visualising your bins	12
GroopM plot	12
GroopM explore	13
GroopM highlight	15
Appendix 1 .gm (.sm) database layout	17

Data storage

GroopM stores all of its data in one database. A single database provides two awesome advantages. 1) One project, one file. 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. ALL GroopM commands expect to be given the database name as a parameter.

We currently use [HDF5](#) for this purpose but we are shortly going to move over to using plain old sqlite3. The name of this project is called StoreM. It's recommended that databases generated using 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 a standard HDF5 layout, so you can view and modify the contents of a GroopM database with third party tools such as [hdfview](#). We are working on the standardization of this data structure, as well as developing programming APIs to access and modify its contents. 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:

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

Manual bin editing utilities:

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

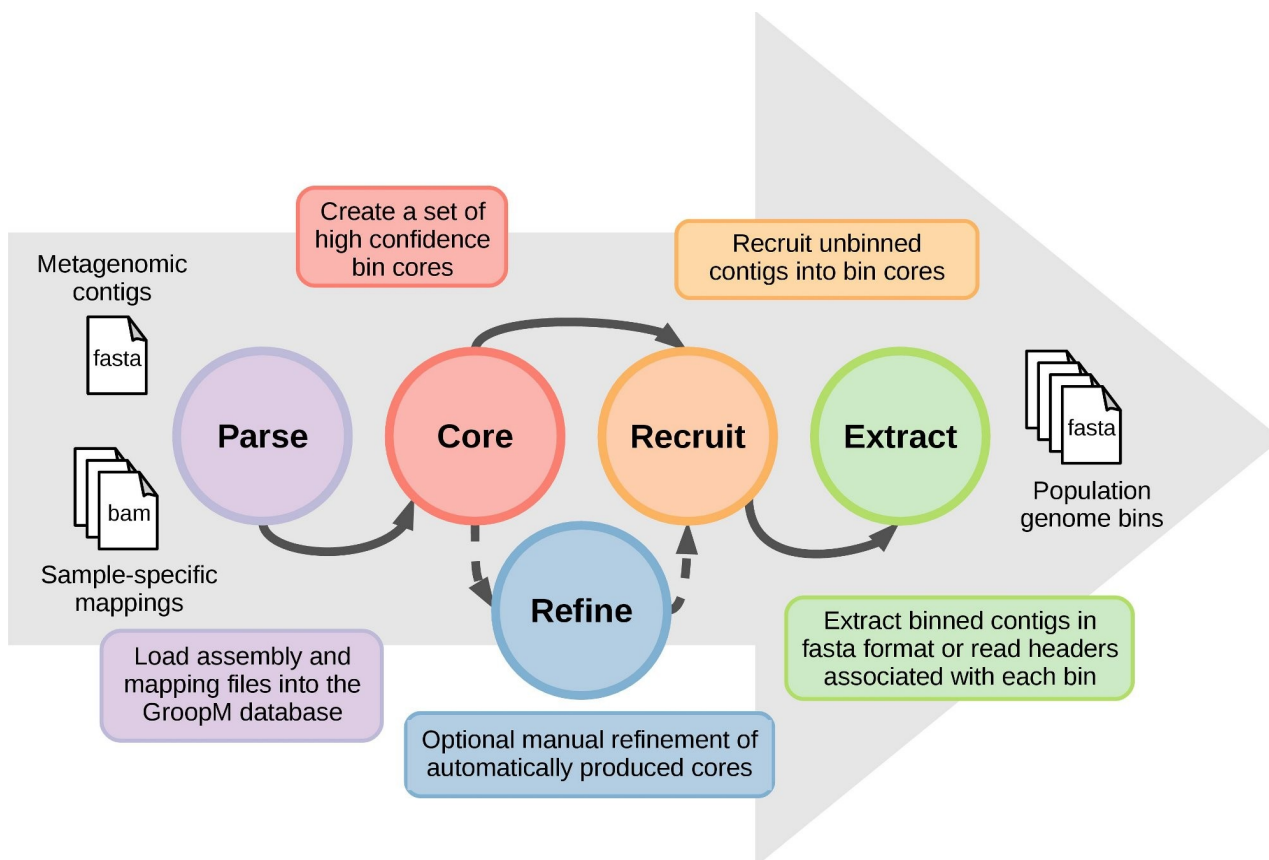
Printing, visualisation and data extraction:

explore	→	Methods for visualising bins and contigs
plot	→	Plotting bins
highlight	→	Highlight individual bins and apply labels
flyover	→	Create a movie of your data
print	→	Print summary statistics about bins
dump	→	Write database fields to csv

NOTE: There are currently four different ways to generate images of your data using GroopM. We know this is far far less than perfect and we're working to fix it!

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](#), [Ray](#) 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:



Please note that this is the core work flow. GroopM also contains a number of visual exploratory tools and some bin editing tools including the optional 'refine' step. This step can be run after 'core' or after 'recruit' or after both. GroopM was designed to be as parameter-free as possible. For more information on any of these steps simply type:

```
$ groopm OPTION -h
```

The rest of this manual provides detailed information on and useful examples of all of the GroopM command modes.

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 (for making coverage profiles)

Optional arguments:

-f --force	[False]	overwrite existing DB file without prompting
-c --cutoff	[500]	only include contigs this length or longer
-t --threads	[1]	number of threads to use when parsing

Example usage:

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

This example will parse your contigs (contigs.fa) and the three 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. These will be refined and expanded in later stages.

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	[False]	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
$ groopm core -c 500 database.gm
```

The first example will attempt to bin all contigs at least as long as the default cut off length (1500 bp). You can modify this using the '-c' option as demonstrated in the second example which tries to bin all contigs that are at least 500 bp long. All bins created using this command are stored in the database.gm.

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:

-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 work flow. The main menu contains the following options:

r	→	plot entire space using bin ids
p	→	plot entire space with bins as points
g	→	plot entire space for bins within a specific GC range (e.g., 0.5-0.6)
u	→	plot all contigs in untransformed coverage space (first 3 stoits only)
b	→	plot one or more bins
v	→	plot all contigs in vicinity of bin
m	→	merge two or more bins
s	→	split a bin into multiple pieces
c	→	change colormap
e	→	toggle ellipses (default = on)
x	→	toggle chimeric bins (default = hidden)
q	→	quit

Most of these options are reasonably self-explanatory. The basic usage pattern is that you type an option and then press 'enter'. If there are any other options to enter (e.g. GC range) you will be prompted for them then. The 'r' option places the bin ID at the average of the transformed coverage positions of all of it's contigs (bin centroid). The colour of the text is the average GC of all it's included contigs. The 'p' option is just like the 'r' option except GroopM places dots at the bin centroid instead of text.

Recruiting unbinned contigs

GroopM recruit

Description:

Recruit unbinned contigs using the core bins as seeds.

Required arguments:

dbname	name of the database to open
--------	------------------------------

Optional arguments:

-c --cutoff	[500]	cut off contig size
-f --force	[False]	overwrite existing database 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 example tries to recruit unbinned contigs into your core bins. By default it tries to recruit ALL unbinned contigs. You can specify a lower length cut off using the '-c' option. Increasing the '-i' option makes your bins bigger, but not necessarily better.

Extracting contigs and reads from bins

GroopM extract

Description:

Extract reads or contigs from one or more bins and write results to disk.

Required arguments:

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

Optional arguments:

NOTE: Some options are only available in read mode {RMO} or contig mode {CMO}

-b --bids	[]	bin ids to use (None for all)
-m --mode	[contigs]	what to extract [reads contigs]
-o --outfolder	[]	write to this folder (None for current dir)
-p --prefix	"	prefix to apply to output files
-c --cutoff	0	{CMO} cutoff contig size (0 for no cutoff)
--mix_bams	[False]	{RMO} use the same file for multiple bam files
--mix_groups	[False]	{RMO} use the same files for multiple groups
--mix_reads	[False]	{RMO} use same files for paired/unpaired reads
--interleave	[False]	{RMO} interleave paired reads in output files
--headers_only	[False]	{RMO} extract only (unique) headers
--no_gzip	[False]	do not gzip output files
--mapping_quality	0	{RMO} mapping quality threshold
--use_secondary	[False]	{RMO} use reads with the secondary flag
--use_supplementary	[False]	{RMO} use reads with the supplementary flag
--max_distance	1000	{RMO} maximum allowable edit distance
-v, --verbose	[False]	{RMO} be verbose
-t --threads	1	{RMO} maximum number of threads to use

Example usage:

```
$ groopm extract database.gm contigs.fa
```

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

The first usage pattern will create a collection of multiple FASTA files, one for each bin, that contain binned contigs. The second usage pattern 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

Merging bins

GroopM merge

Description:

Pairwise merge two or more bins.

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 pattern 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 pattern will merge bin 10 with bin 5 and then bin 11 with bin 5.

Splitting bins into multiple parts

GroopM split

Description:

Split a bin into two or more parts based on kmer-signature, coverage or contig length distributions.

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

Description:

Delete one or more bins from the database

Required arguments:

dbname	name of the database to open
bid	bin id 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 pattern will delete bin 87. The second usage pattern 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 pattern will print some summary statistics about all the bins including the number of contigs, average GC content etc. For example:

"bin id"	"Likely chimeric"	"Coverage 1 std"	"Coverage 2 mean"	"length (bp)"	"# seqs"	"GC mean"	"GC std"	"Coverage 1 mean"	"Coverage 2 std"	"Coverage 3 mean"	"Coverage 3 std"
1	False	2666950	50	0.4200	0.0260	66.1525	47.6605	23.5423	16.0933	65.8442	11.9635
2	False	2722587	110	0.4200	0.0169	1.5232	1.3288	1.9121	0.7005	28.0646	3.6266

The second usage pattern will print detailed information about bins 7 and 4 on a per contig basis.. For example:

#"bid"	"cid"	"length"	"GC"
7	filter1.paired.qc_contig_10007	1539	0.4800
7	filter1.paired.qc_contig_105074	2045	0.5200
7	filter1.paired.qc_contig_10701	21587	0.5300
7	filter1.paired.qc_contig_11399	16662	0.5300

GroopM dump

Description:

Dump information from the tables in the gm file to csv.

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

The first usage pattern writes the complete set of columns stored in all tables in the database (excluding links) to a csv file. The second usage pattern dumps only four columns from the database to a csv file.

Visualising your bins

GroopM plot

Description:

Create plots for each of the bins made by GroopM.

Required arguments:

dbname name of the database to open

Optional arguments:

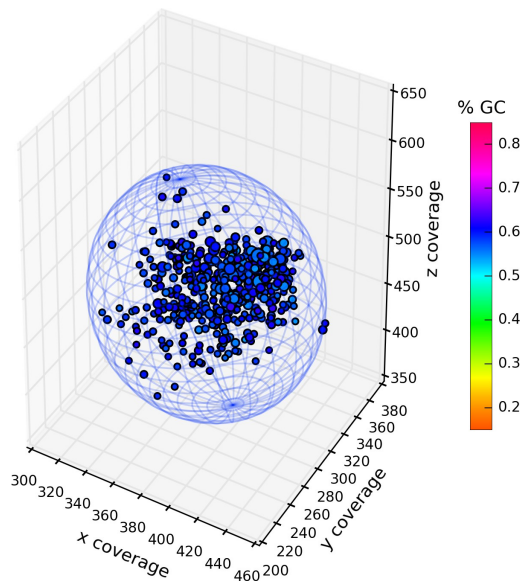
-b --bids	[]	bin ids to plot (None for all)
-t --tag	[BIN]	tag to add to output filename
-f --folder	[]	save plots in folder
-p --points	[False]	ignore contig lengths when plotting
-C --cm	[HSV]	set colormap*

*[HSV Accent Blues Spectral Grayscale Discrete DiscretePaired]

Example usage:

\$ groopm plot database.gm

Bin: 22 : 461 contigs : 2,279,583 BP
Coverage centroid: 392 293 [432 -> 543]
GC: median: 0.5885 stdev: 0.0211



This command will create individual plots of all your bins and write the images to file. It's a useful command for first assessments of the binning quality. Each image will look something like the one shown here. The relatively high GC-stdev indicates that this bin is probably contaminated, although you would have to do a little more work to confirm this. However, if the plot looks like someone went totes crazy with rainbow spray paint then you can be assured that the bin is contaminated.

GroopM explore

Description:

Very very useful utilities for visualising contigs and bins.

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]

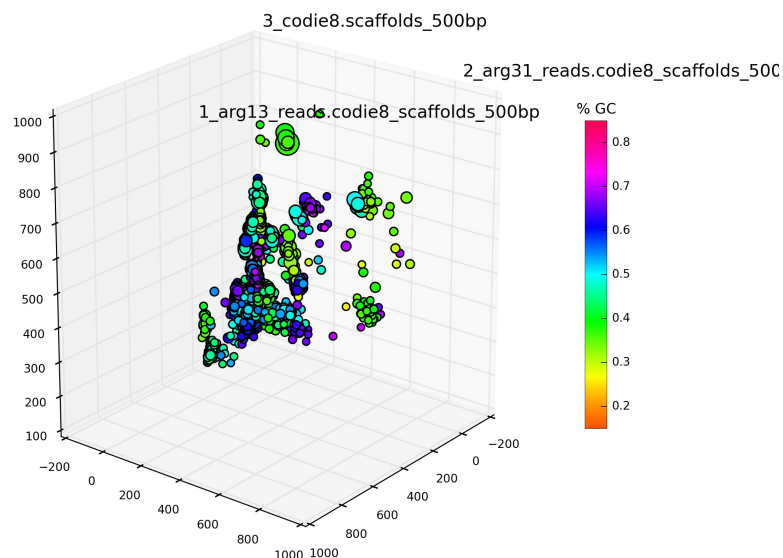
Example usage:

```
$ groopm explore database.gm
```

The first usage pattern will plot bin Ids in GroopM space at each bin's centroid position. The explore command can be run in several modes which generally have self explanatory names however it's worth highlighting a few of these modes specifically.

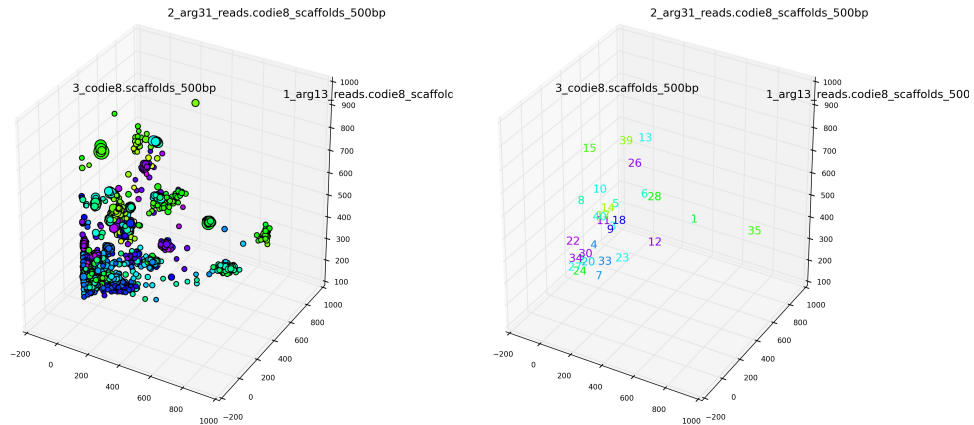
allcontigs

Plots *all* contigs. This is a handy mode because it doesn't require any bins to be made so it can be run straight after GroopM parse. Plots look like this:



compare

Makes two side by side plots, one with bin centroids and one with contigs (see below). It allows you to pinpoint which bin lies where.



GroopM highlight

Description:

Highlight contigs in a GroopM plot.

Required arguments:

dbname name of the database to open

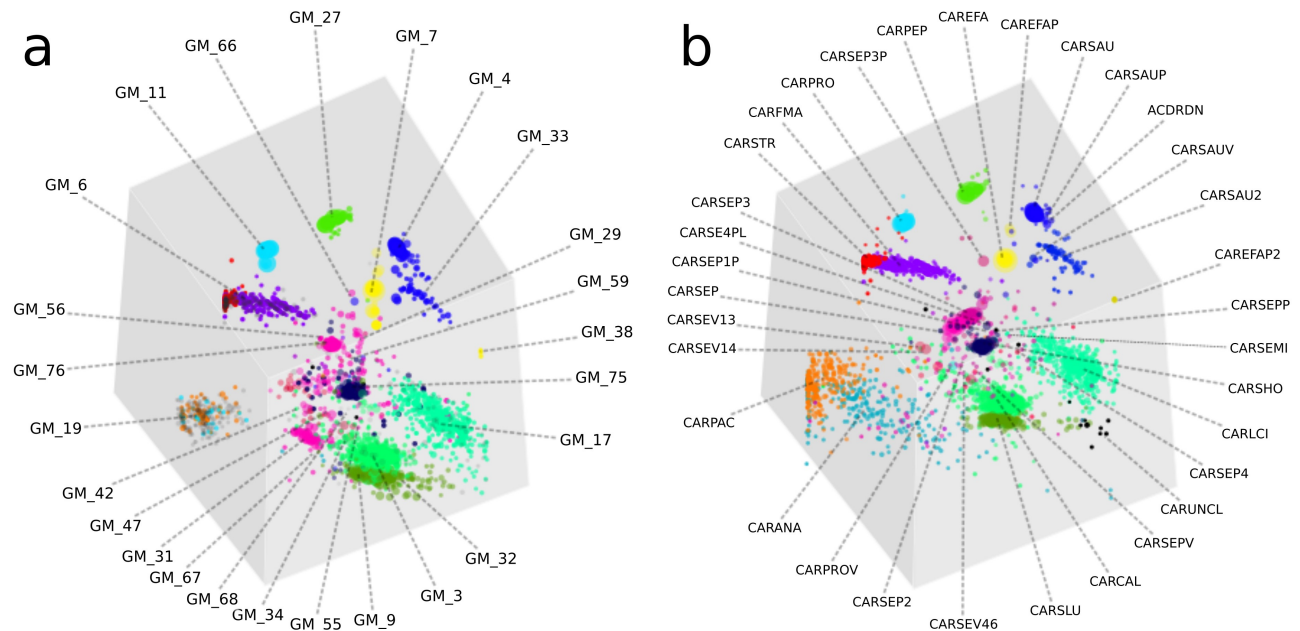
Optional arguments:

-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 -C colours.tsv -f myBins -d 600 -a 20 -e 95
```

This command is really useful for creating plots that highlight contigs or even particular bins. I made these pretty plots for the GroopM paper using this command.



Colours.tsv is a two column file where the first column is the contig name (as written in the fasta file) and the second column is a colour code written as #RRGGBB. You can direct GroopM to make bin labels by supplying a "labels.tsv" which is also a two column file where the first column contains the bin's GroopM Id and the second column contains the label you'd like to use. You can set the azimuth and elevation of the plot using the **-a** and **-e** flags. Use GroopM explore to mess around and find a view that looks good.

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 1 st kmer principal component
'pc2_var'	: Float :	Variance of 2 nd 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 the bin
'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