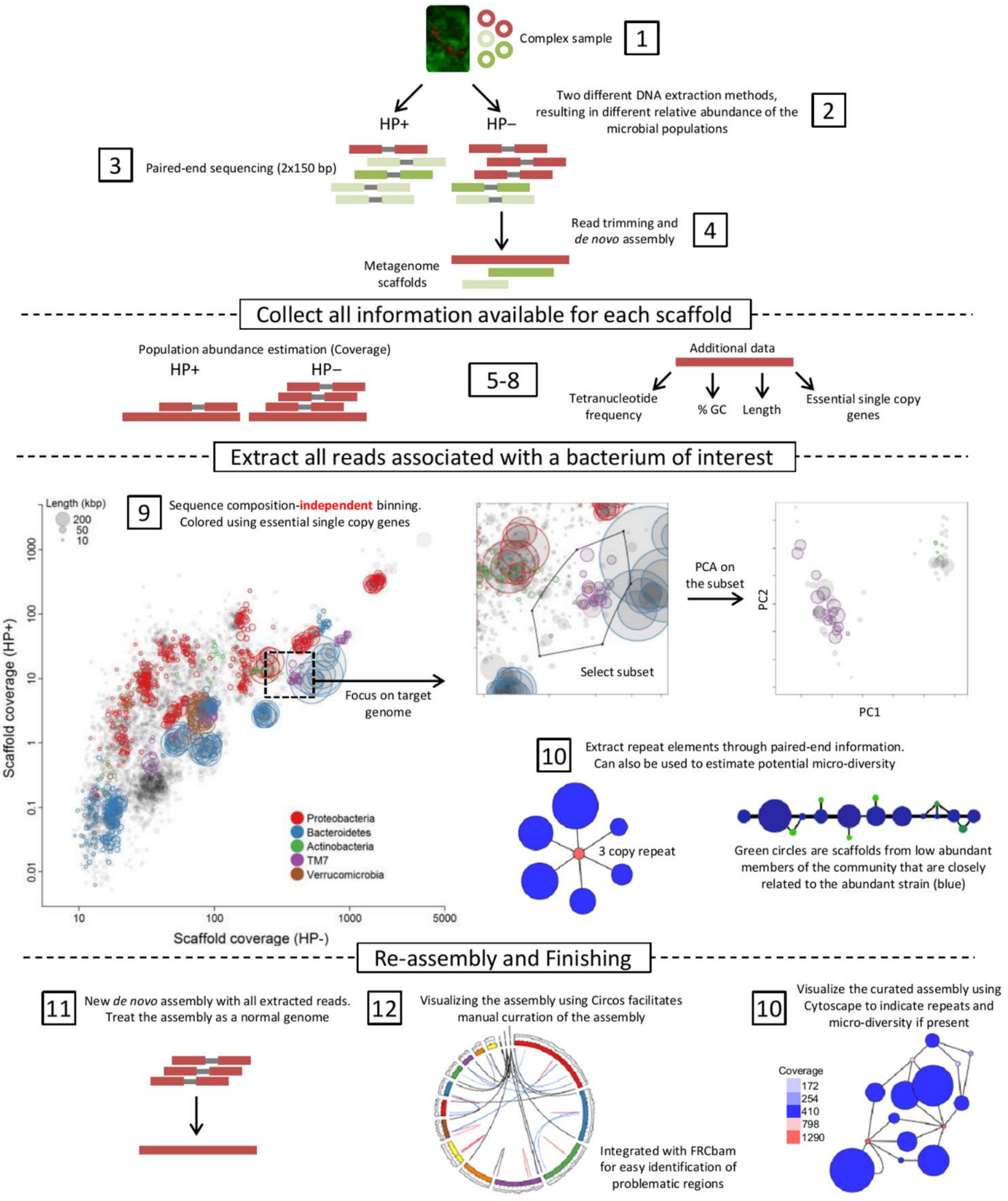
Metagenome Assembly Guide


Complex sample 1
HP+
HP−
Two different DNA extraction methods, resulting in different relative abundance of the microbial populations 2
3 Paired-end sequencing (2x150 bp)
Read trimming and de novo assembly 4
Metagenome scaffolds
Collect all information available for each scaffold
Population abundance estimation (Coverage)
HP+
HP−
5-8
Additional data
Tetranucleotide frequency
% GC
Length
Essential single copy genes
Extract all reads associated with a bacterium of interest
9 Sequence composition-independent binning. Colored using essential single copy genes
Length (kbp)
200
50
10
Scaffold coverage (HP+)
Scaffold coverage (HP-)
Focus on target genome
Proteobacteria
Bacteroidetes
Actinobacteria
TM7
Verrucomicrobia
Select subset
PCA on the subset
PC1
PC2
10 Extract repeat elements through paired-end information. Can also be used to estimate potential micro-diversity
3 copy repeat
Green circles are scaffolds from low abundant members of the community that are closely related to the abundant strain (blue)
Re-assembly and Finishing
11 New de novo assembly with all extracted reads. Treat the assembly as a normal genome
12 Visualizing the assembly using Circos facilitates manual curration of the assembly
Integrated with FRCbam for easy identification of problematic regions
10 Visualize the curated assembly using Cytoscape to indicate repeats and micro-diversity if present
Coverage
172
254
410
798
1290

Introduction

This guide details the process used in Albertsen *et al.*, 2013 to extract high-quality genomes from metagenomes. Much of the data exploration is done through Rstudio (http://www.rstudio.com/) a very powerful IDE for R (http://www.r-project.org/). Links to a few basic R guides can be found here http://www.rstudio.com/ide/docs/help_with_r.

The described pipeline have a small number of dependencies that needs to be installed:

MEGAN (Huson *et al.*, 2011)

Prodigal (Hyatt *et al.*, 2012)

HMMER 3.0 (http://hmmer.org)

BLAST+

Perl

All scripts described in the approach can be accessed from: https://github.com/MadsAlbertsen/multi-metagenome or downloaded directly:

```
git clone git://github.com/MadsAlbertsen/multi-metagenome.git
```

Albertsen *et al.*, 2013 data:

The raw reads can be obtained from the NCBI Short Read Archive under accession numbers:

SRX247688 (HP–) and SRX206471 (HP+).

Scaffolds from the initial *de novo* assembly can be obtained from NCBI GenBank under accession number:

APMI01000000

To try out the approach the complete dataset used in R can be downloaded from: https://www.dropbox.com/s/tffw0yj01rg2wqx/metagenome.workflow.rar

References

Albertsen, M. *et al.*, Recovery of genomes from rare, uncultured bacteria by differential coverage binning of multiple deep metagenomes. *Nature Biotechnology*, (2013).

Huson, D. H., Mitra, S., Ruscheweyh, H.-J., Weber, N. & Schuster, S. C. Integrative analysis of environmental sequences using MEGAN4. *Genome Res.* **21**, 1552–60 (2011).

Hyatt, D., LoCascio, P. F., Hauser, L. J. & Uberbacher, E. C. Gene and translation initiation site prediction in metagenomic sequences. *Bioinformatics* **28**, 2223–30 (2012).

Step 1: Sample complexity

In order to obtain decent genome assemblies two criteria must be met:

1. At least 30-50 X coverage is needed for a decent genome assembly and 100+ for high quality genomes using Illumina data (100bp+ paired-end reads). Hence, if the target genome is at 1% metagenome abundance and assuming a 3 Mbp genome, a rough estimate of the required sequencing effort can be calculated as: 3Mbp * 100 X coverage / 1% abundance = 30 Gbp.
2. If there is substantial micro-diversity of the target genome it will not be possible to obtain a high-quality genome. Micro-diversity is poison for *de novo* assembly.

Step 2: Multiple related samples

To obtain high-resolution sequence-independent binning at least two metagenomes are needed from related samples (more is better...). An easy way to obtain this is using two very different DNA extraction methods on the same sample. However, anything that changes the relative abundance of the microorganisms can be used, e.g. time-series.

Step 3: Paired-end sequencing

We normally sequence 2x150 bp paired-end reads using the Illumina HiSeq2000. However, additional mate-pair libraries would improve the final assemblies.

Step 4: Read trimming and *de novo* assembly

1. Read trimming: The reads are trimmed to remove low quality bases and sequencing adapters. This can be done using a number of different open source tools e.g. the fastx_toolkit. However, we use the commercial package CLC. Our standard trim settings are:
    - A minimum phred score of 20
    - Allowing no ambiguous nucleotides (N's)
    - Remove Illumina sequencing adapters if found
    - A minimum read length of 50 bp after trimming

2. *De novo* assembly of metagenomes is a field in development. We use CLC's *de novo* assembly algorithm as it is able to assemble datasets with very uneven coverage profiles (as metagenomes) and at the same time can handle extremely large datasets. In other projects we have successfully assembled over 1 billion 150 bp paired-end reads on a 40 core server with 256 Gb of RAM in less than a day. We use standard settings, except a kmer value of 63.
    - If CLC is not an option for *de novo* assembly we recommend the open source digital normalization tool khmer (https://khmer.readthedocs.org/en/latest/) to reduce the dataset size and even out the coverage profiles. The data can afterwards be assembled with standard open source *de novo* assemblers.

Step 5: Scaffold coverage and length

Scaffold coverage is estimated by mapping the reads from the two samples independently to the assembled scaffolds. Again we use CLC, but any other open source tool could be used. The result of this step is a list of all scaffolds and their coverage in either sample. The following files contain the scaffold coverage information for the two metagenome samples. The files also contain the scaffold length.

HPminus.scaffold.coverage.csv

HPplus.scaffold.coverage.csv

(Red = used later in R)

Step 6: Tetranucleotide frequency

Tetranucleotide frequencies patterns are generated from the assembled scaffolds.

```
perl multi-metagenome/R.data.generation/calc.kmerfreq.pl -i assembly.fa -o assembly.kmer.tab
```

(Blue = applied script or program)

Step 7: GC content

The GC content of each scaffold is calculated.

```
perl multi-metagenome/R.data.generation/calc.gc.pl -i assembly.fa -o assembly.gc.tab
```

Step 8: Identification of conserved marker proteins

A set of 100+ HMM models (essential.hmm) of conserved marker proteins is used to evaluate completeness and contamination in the genome bins and also to guide the initial selection of genome bins.

1. As the HMM models are on protein level it is initially needed to call open reading frames in the assembled scaffolds. Here the metagenome version of prodigal (Hyatt *et al.*, 2012) is used.

```
prodigal -a temp.orfs.faa -i assembly.fa -m -o temp.txt -p meta -q
```

2. Reformatting the header of output proteins.

```
cut -f1 -f “ “ temp.orfs.faa > assembly.orfs.faa
```

3. After predicting open reading frames essential proteins are identified using HMM models and HMMER3.0.

```
hmmsearch --tblout assembly.hmm.orfs.txt --cut_tc --notextw
multi-metagenome/R.data.generation/essential.hmm assembly.orfs.faa
```

4. The output is reformatted for later use in R.

```
tail -n+4  assembly.hmm.orfs.txt | sed 's/ * / /g' | cut -f1,4 -d " " > assembly.orfs.hmm.id.txt
```

5. The txt file assembly.hmm.orfs.txt contains information of all HMM positive ORFs. In order to get a fast overview of their taxonomic affiliation the a) names of the HMM positive ORFs are extracted b) the sequence of the positive ORFs are extracted c) and then blasted against the NCBI refseq_protein database.

```
tail -n+4  assembly.hmm.orfs.txt | cut -f1 -d " " > list.of.positive.orfs.txt
```

```
perl multi-metagenome/R.data.generation/extract.using.header.list.pl -l list.of.positive.orfs.txt -s
assembly.orfs.faa -o assembly.orfs.hmm.faa
```

```
blastp -query assembly.orfs.hmm.faa -db refseq_protein -evalue 1e-5 -num_threads 60 -
max_target_seqs 5 -outfmt 5 -out assembly.orfs.hmm.blast.xml
```

6. To get a reasonable taxonomic classification MEGANs LCA algorithm (Huson *et al.*, 2011) is utilized and the taxonomic assignment of each protein exported.

```
MEGAN +g -x "import blastfile= assembly.orfs.hmm.blast.xml meganfile=temp.rma;recompute
toppercent=5;recompute minsupport=1;update;collapse rank=Species;update;select nodes=all;export
what=CSV format=readname_taxonpath separator=tab
file=assembly.orfs.hmm.blast.tax.txt;update;close"
```

7. The MEGAN output is reformatted for later use in R.

```
sed 's/\t/;/' assembly.orfs.hmm.blast.tax.txt | cut -f1,5 -d ";" | sed 's/;/\t/' | sed 's/_/\t/'  >
assembly.orfs.hmm.blast.tax.tab
```

8. As the marker proteins are also used for guiding the binning process it is needed to aggregate the protein results to scaffold level. However, as large scaffolds often contain multiple marker proteins a small perl script is used to find the consensus taxonomic assignment on scaffold level.

```
perl multi-metagenome/R.data.generation/hmm.majority.vote.pl -i assembly.orfs.hmm.blast.tax.txt -o
assembly.tax.consensus.txt -n
```

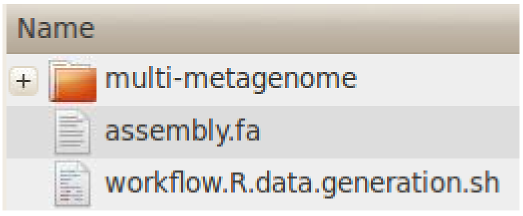
Shell script wrapper for the steps 5-8

To automate the data generation process steps 5-8 are combined in a small shell script "workflow.R.data.generation.sh" which can be found in the "multi-metagenome/R.data.generation/ folder".

1. Download the multi-metagenome repository from github.

```
git clone git://github.com/MadsAlbertsen/multi-metagenome.git
```

2. Add the assembled scaffolds to the folder (assembly.fa). Do not use fancy scaffold naming as it might interfere with subsequent scripts (If in doubt just name the scaffolds 1, 2, 3 ... etc.).



```
Name
multi-metagenome
assembly.fa
workflow.R.data.generation.sh
```

3. Copy the "workflow.R.data.generation.sh" script to the above folder.

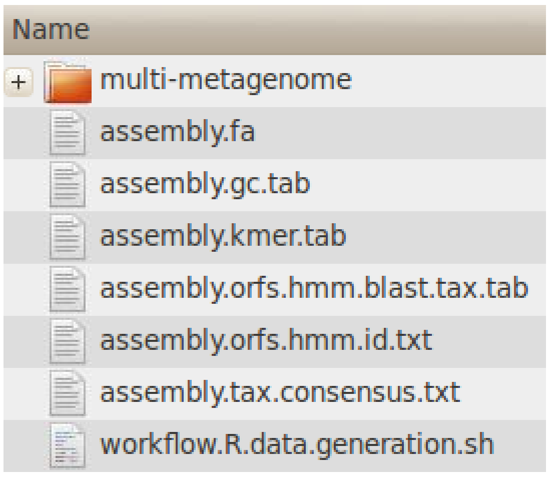
```
cp multi-metagenome/R.data.generation/workflow.R.data.generation.sh .
```

4. Enable execution of the shell script.

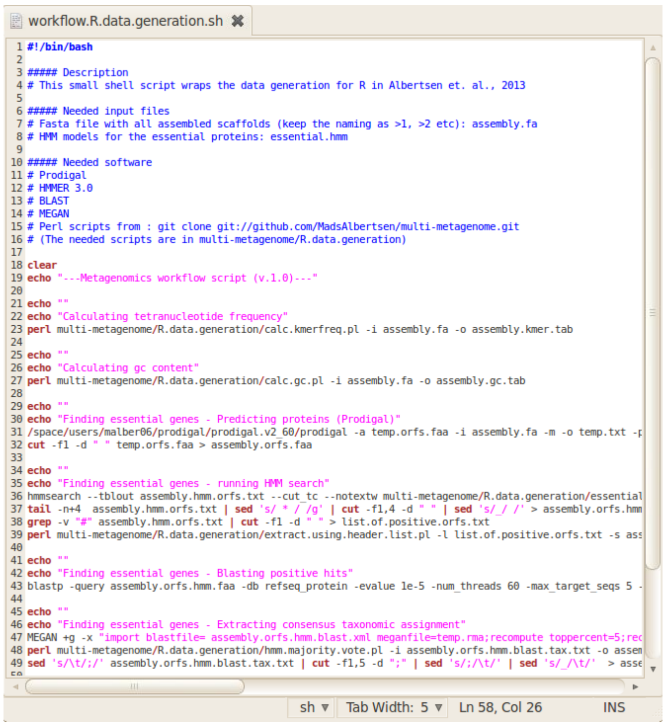
```
chmod +x workflow.R.data.generation.sh
```

5. Execute the script. This should generate the files needed for R.

```
./workflow.R.data.generation.sh
```



```
Name
multi-metagenome
assembly.fa
assembly.gc.tab
assembly.kmer.tab
assembly.orfs.hmm.blast.tax.tab
assembly.orfs.hmm.id.txt
assembly.tax.consensus.txt
workflow.R.data.generation.sh
```



```
#!/bin/bash

##### Description
# This small shell script wraps the data generation for R in Albertsen et. al., 2013

##### Needed input files
# Fasta file with all assembled scaffolds (keep the naming as >1, >2 etc): assembly.fa
# HMM models for the essential proteins: essential.hmm

##### Needed software
# Prodigal
# HMMER 3.0
# BLAST
# MEGAN
# Perl scripts from : git clone git://github.com/MadsAlbertsen/multi-metagenome.git
# (The needed scripts are in multi-metagenome/R.data.generation)

clear
echo "---Metagenomics workflow script (v.1.0)---"

echo ""
echo "Calculating tetranucleotide frequency"
perl multi-metagenome/R.data.generation/calc.kmerfreq.pl -i assembly.fa -o assembly.kmer.tab

echo ""
echo "Calculating gc content"
perl multi-metagenome/R.data.generation/calc.gc.pl -i assembly.fa -o assembly.gc.tab

echo ""
echo "Finding essential genes - Predicting proteins (Prodigal)"
/space/users/malber06/prodigal/prodigal.v2_60/prodigal -a temp.orfs.faa -i assembly.fa -m -o temp.txt -p
cut -f1 -d " " temp.orfs.faa > assembly.orfs.faa

echo ""
echo "Finding essential genes - running HMM search"
hmmsearch --tblout assembly.hmm.orfs.txt --cut_tc --notextw multi-metagenome/R.data.generation/essential
tail -n+4  assembly.hmm.orfs.txt | sed 's/ * / /g' | cut -f1,4 -d " " | sed 's/_/ /' > assembly.orfs.hmm
grep -v "#" assembly.hmm.orfs.txt | cut -f1 -d " " > list.of.positive.orfs.txt
perl multi-metagenome/R.data.generation/extract.using.header.list.pl -l list.of.positive.orfs.txt -s ass

echo ""
echo "Finding essential genes - Blasting positive hits"
blastp -query assembly.orfs.hmm.faa -db refseq_protein -evalue 1e-5 -num_threads 60 -max_target_seqs 5 -

echo ""
echo "Finding essential genes - Extracting consensus taxonomic assignment"
MEGAN +g -x "import blastfile= assembly.orfs.hmm.blast.xml meganfile=temp.rma;recompute toppercent=5;rec
perl multi-metagenome/R.data.generation/hmm.majority.vote.pl -i assembly.orfs.hmm.blast.tax.txt -o assen
sed 's/\t/;/' assembly.orfs.hmm.blast.tax.txt | cut -f1,5 -d ";" | sed 's/;/\t/' | sed 's/_/\t/'  > asse
```

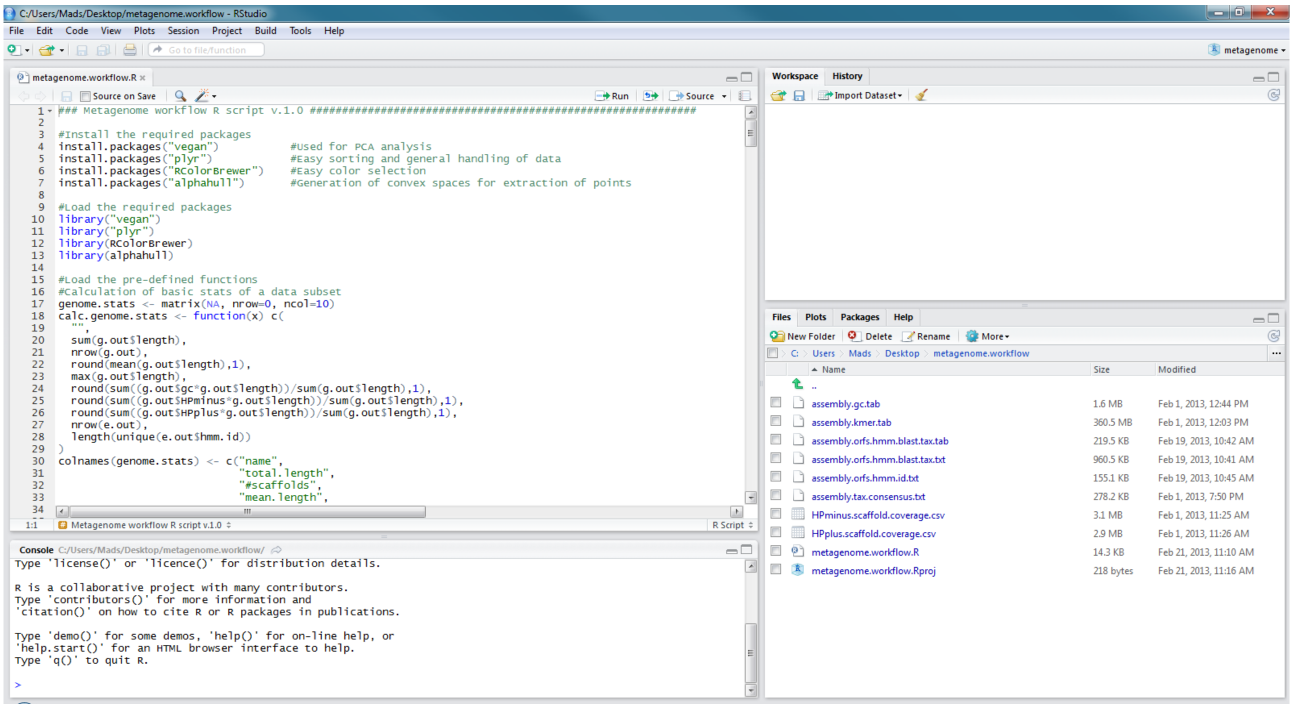
Step 9: Defining initial genome bins using R

R is used to combine all the data generated in steps 5-8 and extract initial genome bins. The complete dataset used in R can be obtained from: https://www.dropbox.com/s/tffw0yj01rg2wqx/metagenome.workflow.rar.

Install R and Rstudio. We recommend to make a Rproject in the folder where all your data is stored. All steps are also described in detail in the R script:

“multi-metagenome/R.data.generation/metagenome.workflow.R”

After installation and project generation your Rstudio start screen should look something like this:

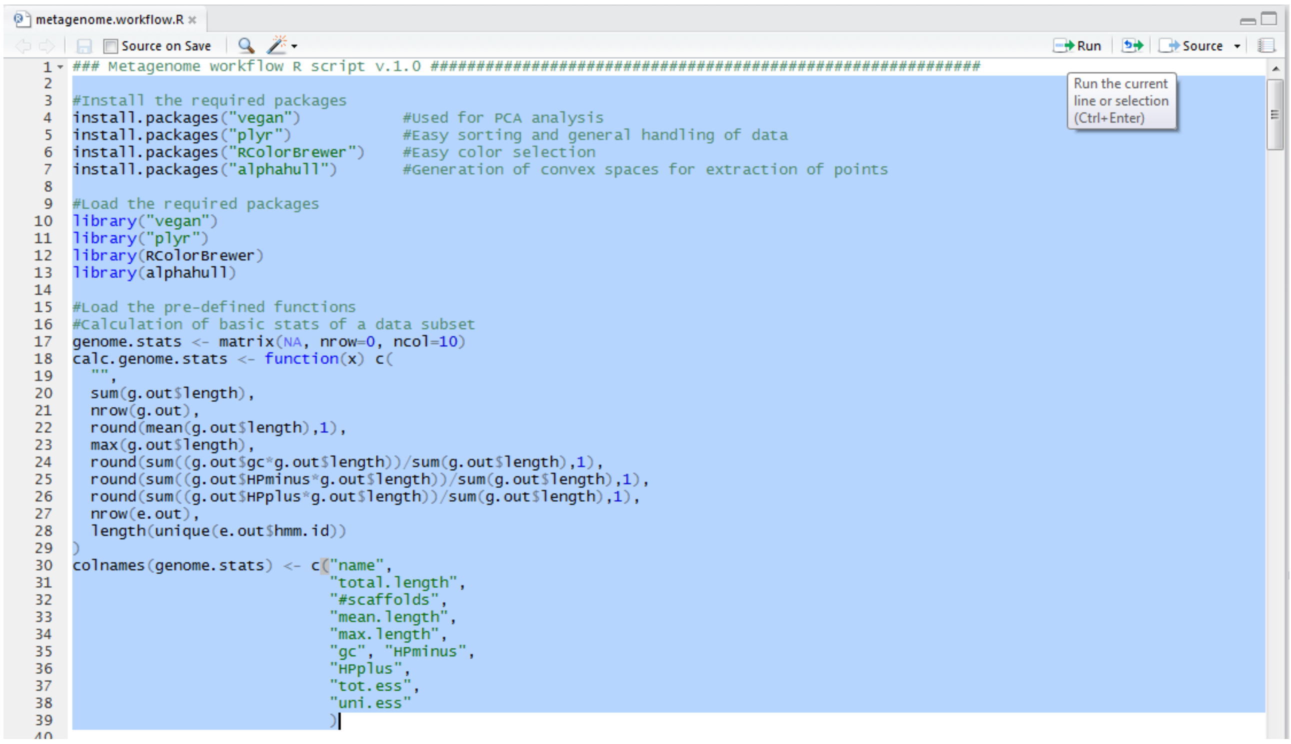

C:/Users/Mads/Desktop/metagenome.workflow - RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
metagenome
metagenome.workflow.R
Source on Save
Run
Source
Workspace History
Import Dataset
Files Plots Packages Help
New Folder Delete Rename More
C: Users Mads Desktop metagenome.workflow
1:1 Metagenome workflow R script v.1.0
R Script
Console C:/Users/Mads/Desktop/metagenome.workflow/

```
### Metagenome workflow R script v.1.0 ############################################################

#Install the required packages
install.packages("vegan")           #Used for PCA analysis
install.packages("plyr")            #Easy sorting and general handling of data
install.packages("RColorBrewer")    #Easy color selection
install.packages("alphahull")       #Generation of convex spaces for extraction of points

#Load the required packages
library("vegan")
library("plyr")
library(RColorBrewer)
library(alphahull)

#Load the pre-defined functions
#Calculation of basic stats of a data subset
genome.stats <- matrix(NA, nrow=0, ncol=10)
calc.genome.stats <- function(x) c(
  "",
  sum(g.out$length),
  nrow(g.out),
  round(mean(g.out$length),1),
  max(g.out$length),
  round(sum((g.out$gc*g.out$length))/sum(g.out$length),1),
  round(sum((g.out$HPminus*g.out$length))/sum(g.out$length),1),
  round(sum((g.out$HPplus*g.out$length))/sum(g.out$length),1),
  nrow(e.out),
  length(unique(e.out$hmm.id))
)
colnames(genome.stats) <- c("name",
                            "total.length",
                            "#scaffolds",
                            "mean.length",
```

```
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

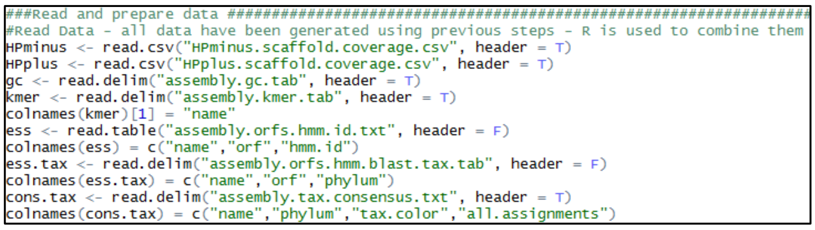
Name	Size	Modified
..		
assembly.gc.tab	1.6 MB	Feb 1, 2013, 12:44 PM
assembly.kmer.tab	360.5 MB	Feb 1, 2013, 12:03 PM
assembly.orfs.hmm.blast.tax.tab	219.5 KB	Feb 19, 2013, 10:42 AM
assembly.orfs.hmm.blast.tax.txt	960.5 KB	Feb 19, 2013, 10:41 AM
assembly.orfs.hmm.id.txt	155.1 KB	Feb 19, 2013, 10:45 AM
assembly.tax.consensus.txt	278.2 KB	Feb 1, 2013, 7:50 PM
HPminus.scaffold.coverage.csv	3.1 MB	Feb 1, 2013, 11:25 AM
HPplus.scaffold.coverage.csv	2.9 MB	Feb 1, 2013, 11:26 AM
metagenome.workflow.R	14.3 KB	Feb 21, 2013, 11:10 AM
metagenome.workflow.Rproj	218 bytes	Feb 21, 2013, 11:16 AM

1. Install and load the dependencies – to run a command or several simply mark the lines and press "Run". This loads 4 packages and a small custom function for calculating basic statistics for the extracted bins



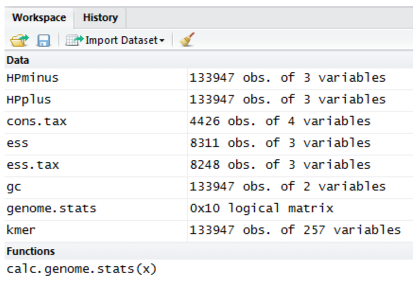
```
### Metagenome workflow R script v.1.0 ##############################################################

#Install the required packages
install.packages("vegan")           #Used for PCA analysis
install.packages("plyr")            #Easy sorting and general handling of data
install.packages("RColorBrewer")    #Easy color selection
install.packages("alphahull")       #Generation of convex spaces for extraction of points

#Load the required packages
library("vegan")
library("plyr")
library(RColorBrewer)
library(alphahull)

#Load the pre-defined functions
#Calculation of basic stats of a data subset
genome.stats <- matrix(NA, nrow=0, ncol=10)
calc.genome.stats <- function(x) c(
  "",
  sum(g.out$length),
  nrow(g.out),
  round(mean(g.out$length),1),
  max(g.out$length),
  round(sum((g.out$gc*g.out$length))/sum(g.out$length),1),
  round(sum((g.out$HPminus*g.out$length))/sum(g.out$length),1),
  round(sum((g.out$HPplus*g.out$length))/sum(g.out$length),1),
  nrow(e.out),
  length(unique(e.out$hmm.id))
)
colnames(genome.stats) <- c("name",
                           "total.length",
                           "#scaffolds",
                           "mean.length",
                           "max.length",
                           "gc", "HPminus",
                           "HPplus",
                           "tot.ess",
                           "uni.ess"
                           )
```

2. Read in all the data generated in the previous steps and rename the column names for some of the datasets.

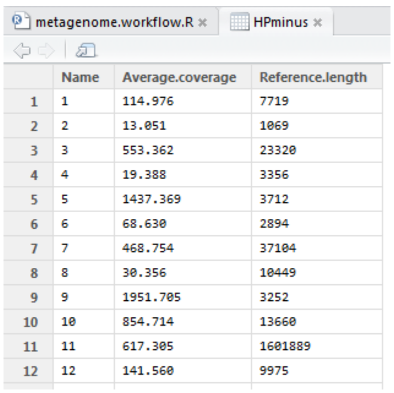


```
###Read and prepare data #############################################################
#Read Data - all data have been generated using previous steps - R is used to combine them
HPminus <- read.csv("HPminus.scaffold.coverage.csv", header = T)
HPplus <- read.csv("HPplus.scaffold.coverage.csv", header = T)
gc <- read.delim("assembly.gc.tab", header = T)
kmer <- read.delim("assembly.kmer.tab", header = T)
colnames(kmer)[1] = "name"
ess <- read.table("assembly.orfs.hmm.id.txt", header = F)
colnames(ess) = c("name","orf","hmm.id")
ess.tax <- read.delim("assembly.orfs.hmm.blast.tax.tab", header = F)
colnames(ess.tax) = c("name","orf","phylum")
cons.tax <- read.delim("assembly.tax.consensus.txt", header = T)
colnames(cons.tax) = c("name","phylum","tax.color","all.assignments")
```

3. The loaded data can be seen under "workspace". Clicking on e.g. "HPminus" show the imported data.



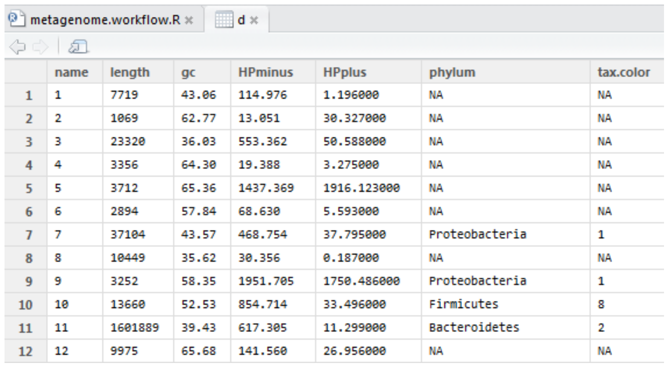
```
Data
HPminus        133947 obs. of 3 variables
HPplus         133947 obs. of 3 variables
cons.tax       4426 obs. of 4 variables
ess            8311 obs. of 3 variables
ess.tax        8248 obs. of 3 variables
gc             133947 obs. of 2 variables
genome.stats   0x10 logical matrix
kmer           133947 obs. of 257 variables
Functions
calc.genome.stats(x)
```



	Name	Average.coverage	Reference.length
1	1	114.976	7719
2	2	13.051	1069
3	3	553.362	23320
4	4	19.388	3356
5	5	1437.369	3712
6	6	68.630	2894
7	7	468.754	37104
8	8	30.356	10449
9	9	1951.705	3252
10	10	854.714	13660
11	11	617.305	1601889
12	12	141.560	9975

4. To make the subsequent steps easy we combine all scaffold data into the data frame “d”.

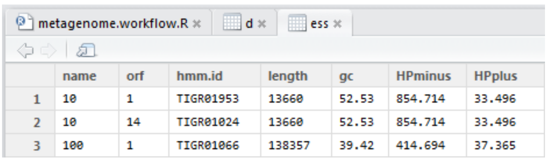
```
#Combine all data to one data matrix "d"
d <- as.data.frame(cbind(HPminus$Name,
                         HPplus$Reference.length,
                         gc$gc,
                         HPminus$Average.coverage,
                         HPplus$Average.coverage
                         )
                   ,row.names=F
                   )

colnames(d) = c("name",
                "length",
                "gc",
                "HPminus",
                "HPplus"
                )

d <- merge(d,cons.tax, by = "name", all = T)
```



	name	length	gc	HPminus	HPplus	phylum	tax.color
1	1	7719	43.06	114.976	1.196000	NA	NA
2	2	1069	62.77	13.051	30.327000	NA	NA
3	3	23320	36.03	553.362	50.588000	NA	NA
4	4	3356	64.30	19.388	3.275000	NA	NA
5	5	3712	65.36	1437.369	1916.123000	NA	NA
6	6	2894	57.84	68.630	5.593000	NA	NA
7	7	37104	43.57	468.754	37.795000	Proteobacteria	1
8	8	10449	35.62	30.356	0.187000	NA	NA
9	9	3252	58.35	1951.705	1750.486000	Proteobacteria	1
10	10	13660	52.53	854.714	33.496000	Firmicutes	8
11	11	1601889	39.43	617.305	11.299000	Bacteroidetes	2
12	12	9975	65.68	141.560	26.956000	NA	NA

5. In addition, we combine all data on the essential proteins into the data frame “ess”.

```
#Combine data on essential genes
ess<- merge(ess, d, by = "name", all.x = T)
ess<- merge(ess, ess.tax, by = c("name","orf"), all.x = T)
```



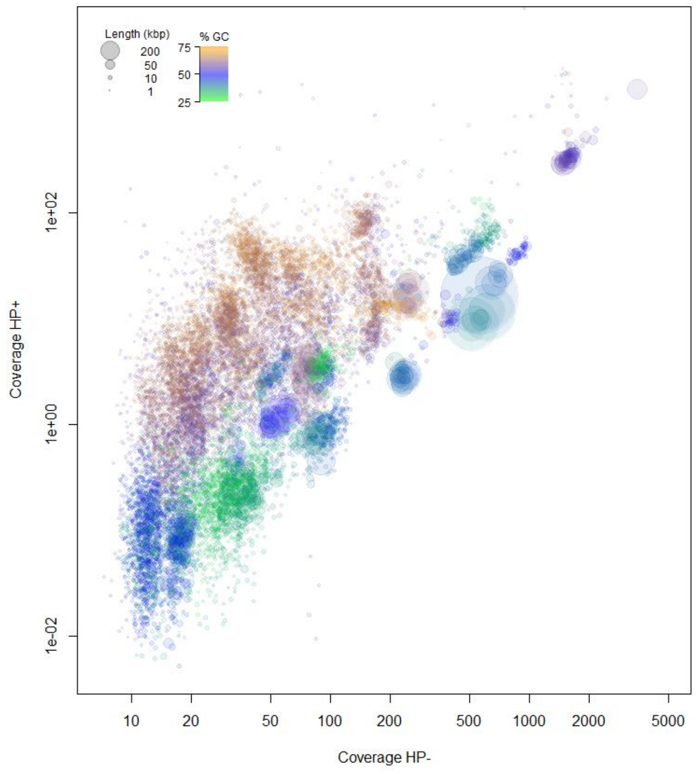
	name	orf	hmm.id	length	gc	HPminus	HPplus
1	10	1	TIGR01953	13660	52.53	854.714	33.496
2	10	14	TIGR01024	13660	52.53	854.714	33.496
3	100	1	TIGR01066	138357	39.42	414.694	37.365

6. To get an initial overview of the data, we subset the data to all scaffolds > 5kbp.

```
#Subsetting the data by scaffold length is nice for an initial overview of the data
d <- subset(d,length > 5000)
```

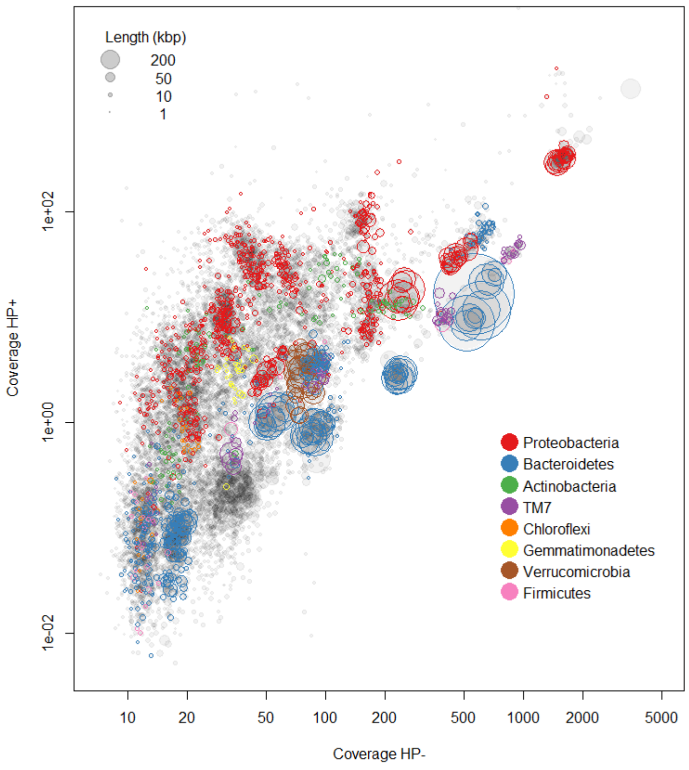
7. In order to easily see the structure of the data we color all scaffolds by their GC content and scale the size of the points by the scaffold length. As much data is located in a tight space we make the color palette transparent. Clusters of scaffolds with similar coverage and GC content represent potential genomes.

```
### Coverage plots - Colored by GC ####################
gbr<-colorRampPalette(c("green","blue","orange","red"))
palette(adjustcolor(gbr(70), alpha.f = 0.1))

plot(x = d$HPminus,
     y = d$HPplus,
     log="xy",
     cex = sqrt(d$length)/100,
     pch=20,
     col=d$gc-25,
     xlim = c(7,5000),
     ylim = c(0.005,5000),
     xlab = "Coverage HP-",
     ylab = "Coverage HP+"
     )
```

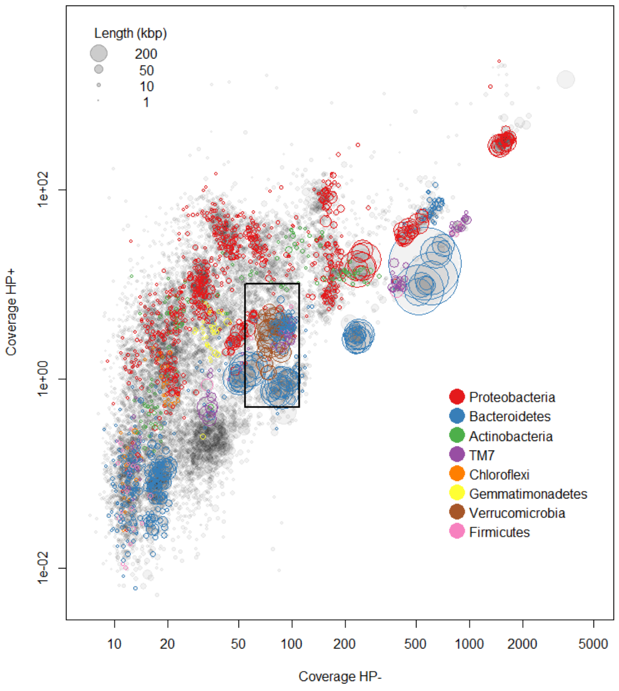


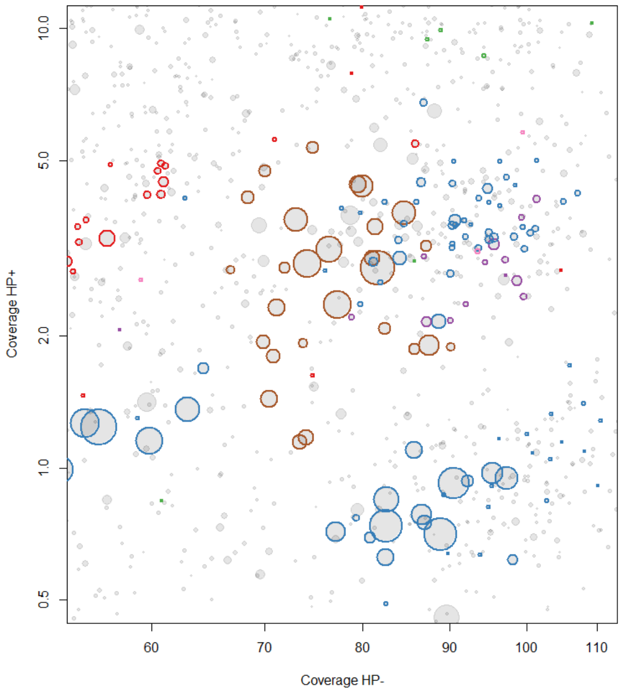
8. Using the phylum level assignments of the essential proteins the genome clusters appear even clearer. First a greyscale plot is made and then the 8 most abundant phyla are added (The d$tax.color variable is arranged by decreasing number of essential genes assigned at phylum level). In this case the most abundant phylum is Proteobacteria and the second Bacteroidetes.

```
### Coverage plots - Colored by phylum affiliation ##
dev.off()
plot(x = d$HPminus,
     y = d$HPplus,
     log="xy",
     cex = sqrt(d$length)/100,
     pch=20,
     col=rgb(0,0,0,0.05),
     xlim = c(7,5000),
     ylim = c(0.005,5000),
     xlab = "Coverage HP-",
     ylab = "Coverage HP+"
)

# Add phylum level classificiation

palette(brewer.pal(8,"Set1"))

points(x = d$HPminus[d$tax.color<9],
       y = d$HPplus[d$tax.color<9],
       cex = sqrt(d$length[d$tax.color<9])/100*0.7,
       col=d$tax.color[d$tax.color<9]
)
```



9. To extract potential genome bins, first zoom in on the area of interest. In this case a genome from the phylum Verrucomicrobia is targeted. This is done by modifying the xlim and ylim parameters.

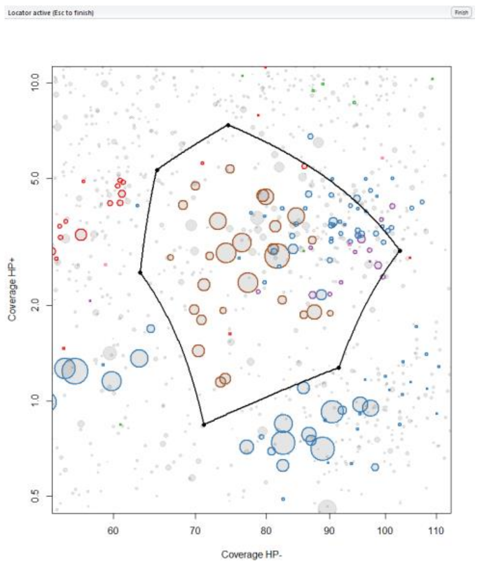




```
#Zoom on target genome
plot(x = d$HPminus,
     y = d$HPplus,
     log="xy",
     cex = sqrt(d$length)/100,
     pch=20, col=rgb(0,0,0,0.1),
     xlim = c(55,110),
     ylim = c(0.5,10),
     xlab = "Coverage HP-",
     ylab = "Coverage HP+"
     )
```

10. Narrow down the genome using the essential genes as a rough guide. Using the locater function you interactively chose 6 points on the plot. The space defined by the 6 chosen points is stored in the variable “z.def”. The subsequent command plots the defined space on the figure.

```
z.def <- ahull(locator(6, type="p", pch=20), alpha=100000)
plot(z.def,add=T, col="black")
```



11. The scaffolds in the selection are stored in the variable g.out (genome.out) and the essential genes are stored in the variable e.out (essential.out).

```
g.out <- {}
for (i in 1:nrow(d)) { if (inahull(z.def, c(d$HPminus[i],d$HPplus[i]))) g.out <- rbind(g.out,d[i,])}
e.out<-{}
for (i in 1:nrow(ess)) { if (inahull(z.def, c(ess$HPminus[i],ess$HPplus[i]))) e.out <- rbind(e.out,ess[i,])}
```

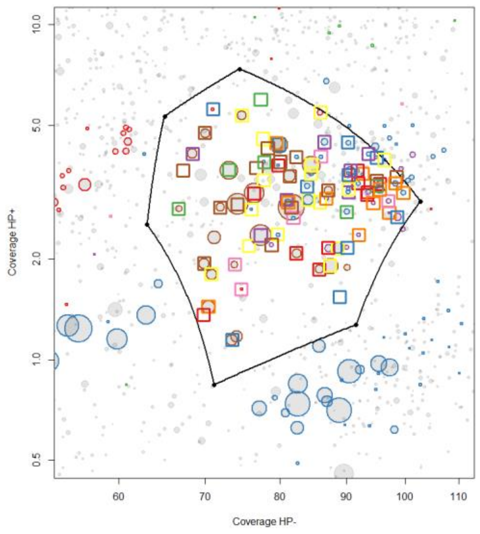
12. An easy way to investigate if it is a complete and clean genome is to look at the number of unique single copy genes in the bin and the number of duplicated single copy genes (Note: the initial selection includes many other bacteria. Hence, we do expect many "single copy genes" in multiple copies). The information of duplicated single copy genes in the variable d.out. Next we calculate some basic statistics for the extracted subset using the function "calc.genome.stats()". 104 unique single copy genes (uni.ess) could indicate a complete genome, however a total of 220 single copy genes show that we have multiple genomes in the initial bin (as expected). Note that the genes TIGR00436, PF01795 and PF00750 often is seen in multiple copies and that some bacteria may miss essential genes – or they are simply not picked up by the HMM models or gene caller.

```
d.out<-e.out[which(duplicated(e.out$hmm.id) | duplicated(e.out$hmm.id, fromLast=TRUE)),]
d.out[,c(1,3,8)]
cbind(colnames(genome.stats),calc.genome.stats())
```

```
"name"          ""
"total.length"  "10522004"
"#scaffolds"    "250"
"mean.length"   "42088"
"max.length"    "701203"
"gc"            "53.4"
"HPminus"       "80.7"
"HPplus"        "3.1"
"tot.ess"       "220"
"uni.ess"       "104"
```

13. To investigate where the duplicated single copy genes are located, they are simply added to the plot as small boxes. The duplicated genes are all over the place in this selection – therefore, it is not possible to make a clean genome using just the coverage. Therefore we use tetranucleotide frequencies in the next step on the subset of scaffolds we have extracted in the previous steps.

```
points(d.out$HPminus,
        d.out$HPplus,
        col=d.out$hmm.id,
        cex=3,
        pch = 22,
        lwd = 3
        )
```

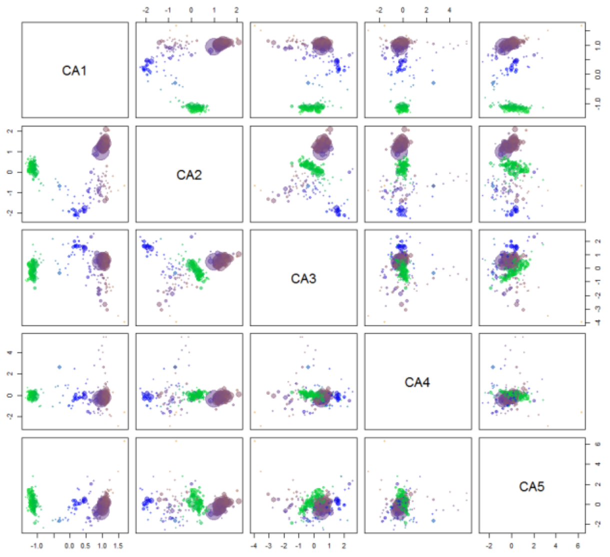


14. To separate the target genome from the contaminating genomes the Vegan package is used to do a Correspondence Analysis (CA) on tetranucleotide frequencies in the extracted subset of scaffolds. The CA scores are extracted and added to the scaffolds. More sophisticated methods could be used – but it is rarely necessary. Lastly, the initial selection is stored in for documentation of the extraction procedure.

```
ca<-scores(cca(kmer[g.out$name,2:ncol(kmer)]), choices=1:5)$sites
g.out<-cbind(g.out,ca)
e.out<-merge(e.out,g.out[,c(1,9:13)],all.x=T,by="name")
d.out<-merge(d.out,g.out[,c(1,9:13)],all.x=T,by="name")

g2g.a <- g.out
g2d.a <- z.def
g2e.a <- e.out
```

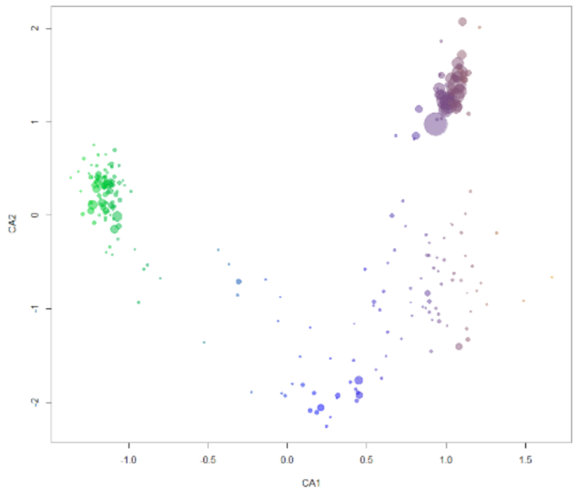
15. To get an overview of which components to use the first 5 are plotted using a pairs plot.

```
palette(adjustcolor(gbr(70), alpha.f = 0.5))
pairs(g.out[,9:13],
      cex = sqrt(g.out$length)/100,
      pch=20,
      col=g.out$gc-25
      )
```



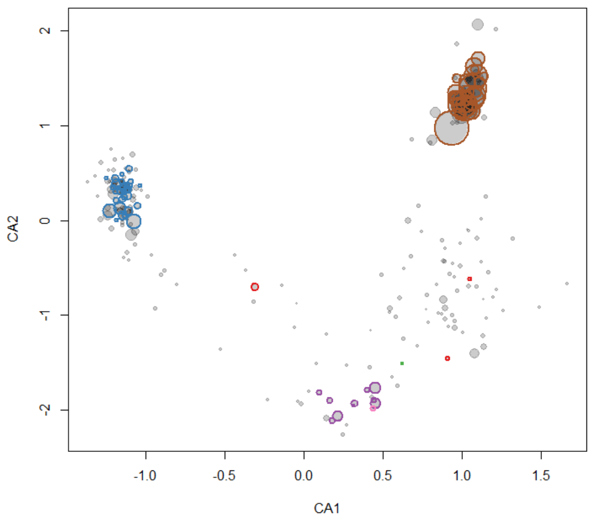
16. In this case the first 2 components seem promising.

```
plot(x = g.out$CA1,
     y = g.out$CA2,
     cex = sqrt(g.out$length)/100,
     pch = 20,
     col = g.out$gc-25,
     xlab = "CA1",
     ylab = "CA2"
     )
```



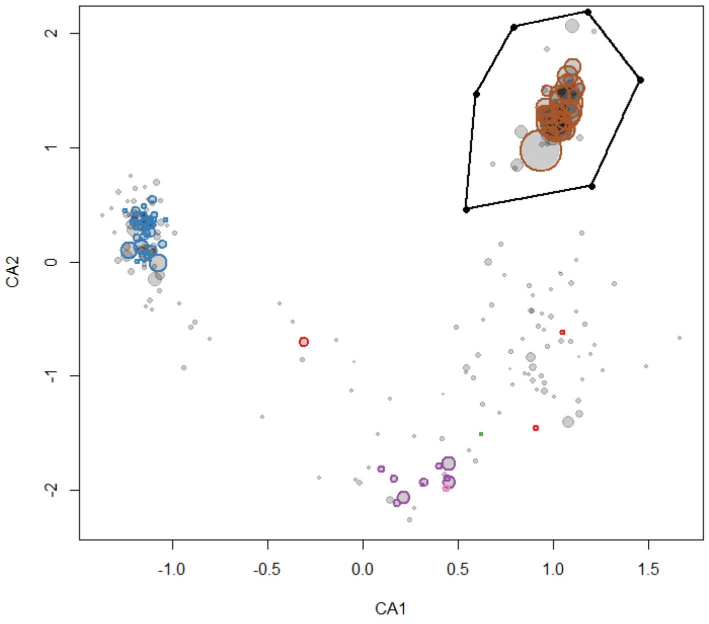
17. In order to confirm the clusters, the essential genes are added to the plot. All scaffolds with essential genes classified to Verrucomicrobia are clustering.

```
palette(brewer.pal(8,"Set1"))
points(x = e.out$CA1,
       y = e.out$CA2,
       col = e.out$tax.color,
       cex = sqrt(e.out$length)/100*0.7,
       lwd = 2
       )
```



18. The locater tool is used again to extract the bin of interest. And a new subset is extracted from the first subset.

```
z.def <- ahull(locator(6, type = "p", pch=20), alpha = 10000)
plot(z.def,add=T, col = "black")
g.out <- {}
for (i in 1:nrow(g2g.a)) { if (inahull(z.def, c(g2g.a$CA1[i],g2g.a$CA2[i]))) g.out <- rbind(g.out,g2g.a[i,])}
e.out<-{}
for (i in 1:nrow(g2e.a)) { if (inahull(z.def, c(g2e.a$CA1[i],g2e.a$CA2[i]))) e.out <- rbind(e.out,g2e.a[i,])}
```

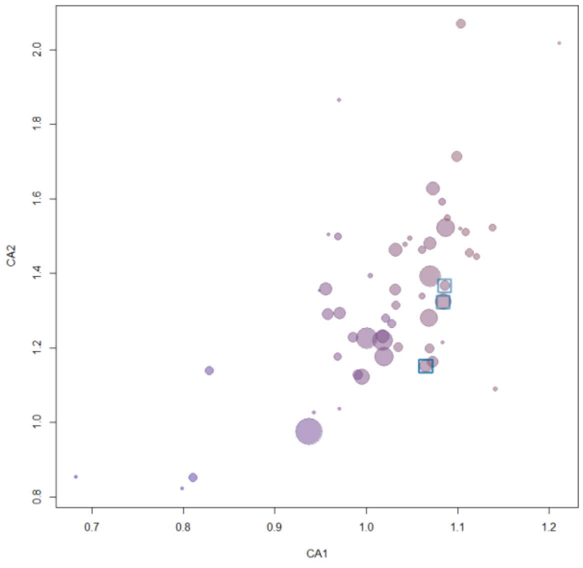


19. The final extraction is saved, plotted, examined for duplicated single copy genes, and finally the genome statistics are calculated. It can be seen that there seem to be 3 single copy genes in multiple copies (103 total vs. 100 unique). However, closer inspection reveals that it is the PF01795 which often is found in multiple copies.

```
g2g.b <- g.out
g2d.b <- z.def
g2e.b <- e.out

cbind(colnames(genome.stats),calc.genome.stats())

d.out<-e.out[which(duplicated(e.out$hmm.id) | duplicated(e.out$hmm.id, fromLast=TRUE)),]

plot(x = g.out$CA1,
     y = g.out$CA2,
     cex = sqrt(g.out$length)/100,
     pch = 20,
     col = g.out$gc-25,
     xlab = "CA1",
     ylab = "CA2"
)

points(x = d.out$CA1,
       y = d.out$CA2,
       col = d.out$hmm.id,
       cex = 3,
       pch = 22,
       lwd = 3
)
```

```
"name"           ""
"total.length"  "6736286"
"#scaffolds"    "55"
"mean.length"   "122477.9"
"max.length"    "701203"
"gc"            "61"
"HPminus"       "77.4"
"HPplus"        "3"
"tot.ess"       "103"
"uni.ess"       "100"
```

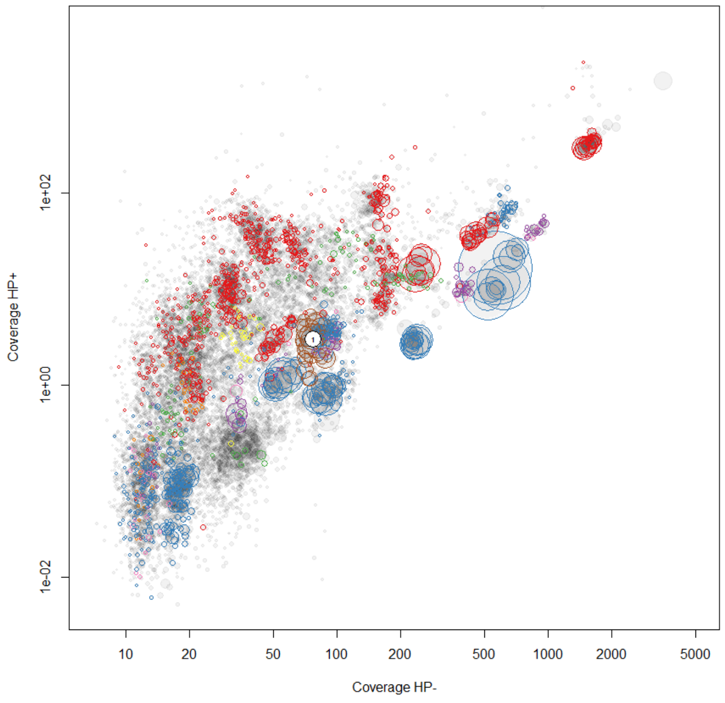
```
> d.out[,c(1,3,8)]
     name      hmm.id           phylum.x
68    735 PF01795.14 Verrucomicrobia group
109  2546 PF01795.14 Verrucomicrobia group
110  2546 PF01795.14 Verrucomicrobia group
178  5388 PF01795.14 Verrucomicrobia group
```



20. The calculated genome statistics can be stored in the variable "genome.stats" for convenient plotting on the original dataset.

```
genome.stats<-rbind(genome.stats,calc.genome.stats())
genome.stats[nrow(genome.stats),1] <- "g2"
```

```
palette(brewer.pal(8,"Set1"))
plot(x = d$HPminus,
     y = d$HPplus,
     log="xy", cex = sqrt(d$length)/100,
     pch=20,
     col=rgb(0,0,0,0.05),
     xlim = c(7,5000),
     ylim = c(0.005,5000),
     xlab = "Coverage HP-",
     ylab = "Coverage HP+"
     )

points(x = d$HPminus,
       y = d$HPplus,
       cex = sqrt(d$length)/100*0.7,
       col=d$tax.color
       )

points(genome.stats[,7],
       genome.stats[,8],
       pch = 20, cex=4,
       col="white"
       )

points(genome.stats[,7],
       genome.stats[,8],
       cex=4*0.7
       )

text(as.numeric(genome.stats[,7]),
     as.numeric(genome.stats[,8]),
     cex=0.5,
     font=2
     )
```



21. Finally the names of the scaffolds in the bin are exported to a text file.

```
write.table(g2g.b$name,file="g2.txt",quote=F,row.names=F,col.names=F)
```

Step 10: Tracking and visualization of paired-end connections between scaffolds

Tracking and visualizing paired-end connections between scaffolds is a simple but effective method to identify repeat elements and refine the assemblies. Metagenome assemblies are often fragmented due to a high degree of strain-heterogeneity (closely related strains). Hence, the fragmentation is often not due to lack of data in a particular region, but due to the assemblers having several options at the branch point. This can also be showed by simply visualizing scaffolds that are connected to one another. The data used in Cytoscape can be found here: "\multi-metagenome\cytoscapeviz\full.albertsen.connection.network".

In this step we are mainly interested in reads mapping in ends of scaffolds as this indicate that there exists a proper link between the two scaffolds.

To track the location of all paired-end reads, the mapping from Step 5 is exported in SAM format. The SAM format is a community standard for representing reads mapped to reference sequences.

The script \multi-metagenome\cytoscapeviz\cytpscapeviz.pl searches through the SAM file and identifies all the paired-end reads that fulfill the user-defined criteria.

```
perl \multi-metagenome\cytoscapeviz\cytpscapeviz.pl -i mapping.sam -f 2 -a 125 -e 500 -m 3000 -c
```

- i: the SAM file of the mapping. Note the script use the read header to identify PE reads. The assumed header structure is:

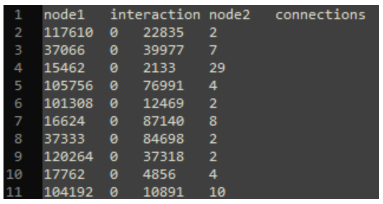
```
read1_1
read1_2
read2_1
read2_2
read3_1
read3_2
```

where "readX" designates a specific read-pair and "_X" specifies forward or reverse reads. The script strips "_X" after which the names of the forward and reverse reads are identical. "readX" could be anything as long as the name doesn't contain "_".

- f: the minimum number of connections to link two scaffolds.
- a: the average read length of the reads.
- e: the maximum distance from the start or end of the scaffold to be considered an "end match".
- m: The minimum scaffold length to allow reads to match start and end of the same scaffold.
- c: flag to indicated that a condensed output is made. By default scaffolds are represented by a start and end node. However, it is often easier to look at each scaffold as a single node.

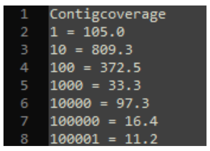
The script outputs several files that can be used directly for visualization in the open source tool Cytoscape.

condensed.cytoscape.connections.tab contains the links between scaffolds that can be visualized as a network in Cytoscape. Node 1 and 2 is the names of the connected scaffolds and the number of connections is found in column 4 (connections). The number of connections can be loaded as an edge attribute and thereby be displayed in the network.



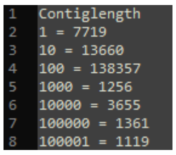
```
node1   interaction node2   connections
117610  0   22835   2
37066   0   39977   7
15462   0   2133    29
105756  0   76991   4
101308  0   12469   2
16624   0   87140   8
37333   0   84698   2
120264  0   37318   2
17762   0   4856    4
104192  0   10891   10
```

condensed.cytoscape.coverage.tab contains the coverage of each scaffold. This can be loaded as a node attribute and displayed in the network. 1 = 105.0 means that scaffold 1 has an average coverage of 105.0.



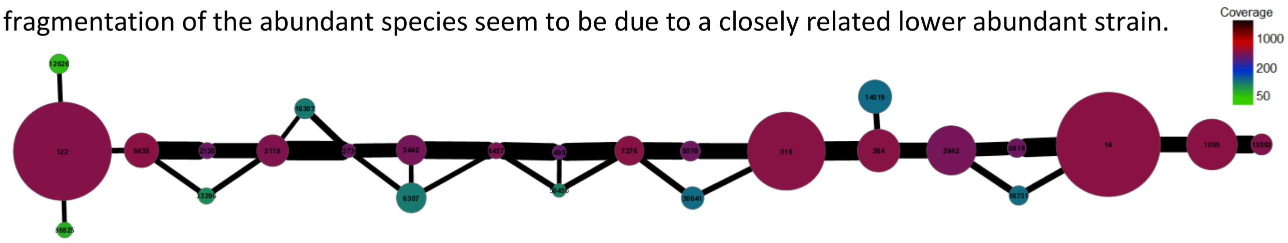
```
Contigcoverage
1 = 105.0
10 = 809.3
100 = 372.5
1000 = 33.3
10000 = 97.3
100000 = 16.4
100001 = 11.2
```

condensed.cytoscape.length.tab contains the length of each scaffold. This can be loaded as a node attribute and displayed in the network. 1 = 7719 means that scaffold 1 has a length of 7719 bp.

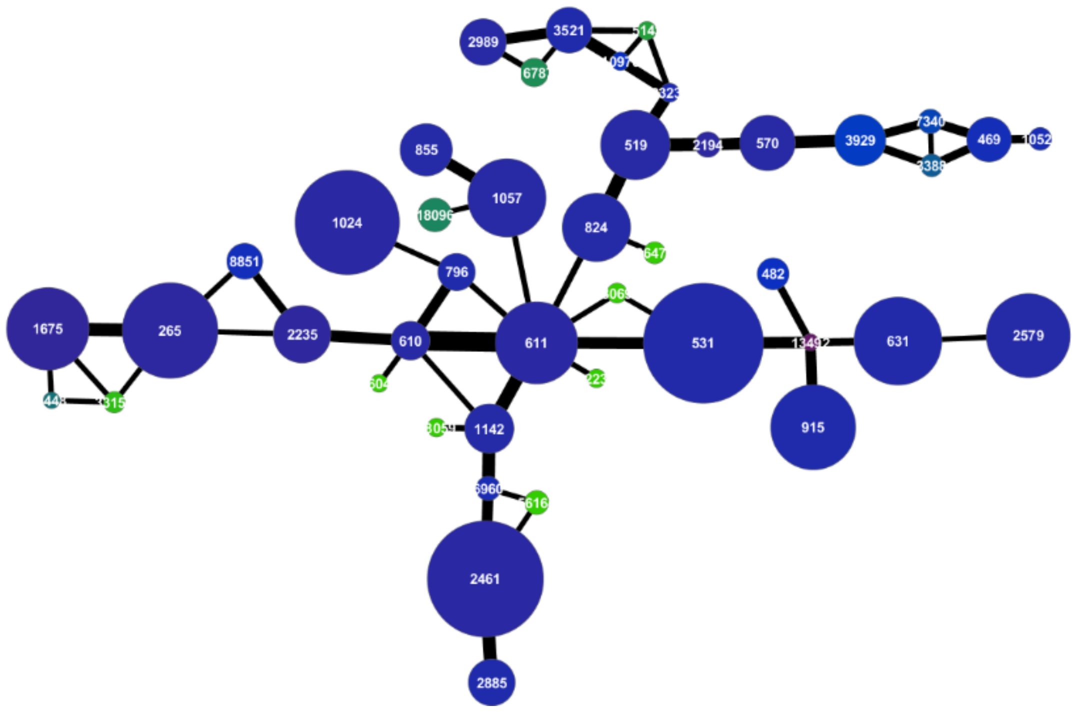


```
Contiglength
1 = 7719
10 = 13660
100 = 138357
1000 = 1256
10000 = 3655
100000 = 1361
100001 = 1119
```

Below is an example of micro-heterogeneity. Scaffolds are scaled by length, colored by coverage and the number of connections is indicated by the thickness of the lines between scaffolds. It can be seen that the fragmentation of the abundant species seem to be due to a closely related lower abundant strain.



A more complex example is seen below. Here micro-heterogeneity is present as in the other example but also repeats. Both as correct repeats (scaffold 13492), but also in many cases integrated in ends of large scaffolds – the clearest example being scaffold 611. Each normal scaffold should only have 2 connections.



The aim of the network analysis is to look at the bins defined in R and investigate the 3 following questions:

1. Any scaffolds wrongly included in the R bin? The "wrong" scaffolds can often be identified as linking to scaffolds that are not included in the R bin.
2. Any correct scaffolds that were missed in the initial R bin? This is mainly to pick up repeat elements that were not included in the initial coverage bin. In addition small scaffolds can have significant variation in coverage even though they originated from the same genome.
3. Is there any integration of repeat elements in scaffolds that needs to be corrected?

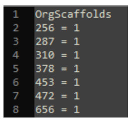
To extract all the sub-network of scaffolds associated with a particular R bin a small perl script is provided. The network associated to the Verrucomicrobia in the R example is extracted by using the names of the scaffolds in the R defined bin:

```
perl \multi-metagenome\cytoscapeviz\cytoscape.extract.sub.graph.using.list.pl -l g2.txt -c condensed.cytoscape.connections.tab
```

The output files are:

condensed.cytoscape.connections.tab.sub.txt the sub-network.

g2.txt.orginal.paint.cyto.txt a node attribute file that can be used to mark the original scaffolds in the sub-network directly in Cytoscape.



```
OrgScaffolds
256 = 1
287 = 1
310 = 1
378 = 1
453 = 1
472 = 1
656 = 1
```

Step 11: Reassembly

If the bin selection after Cytoscape is fragmented - but in high coverage (> 100X) reassembly of the reads included in the selected scaffolds might improve the assembly considerable. This also mean that the assembly can be treated as a "normal" single genome assembly and thereby allow optimization of the assembly parameters.

However, to improve the assembly it is important that all relevant reads are included. Areas in the assembly with no coverage (e.g. gaps, N's) can never be resolved if only reads from the initial scaffold mapping is included. Hence, if read_1 of a read-pair is mapped to the selected scaffold, this read can be used to extract read_2. In practice it is done using a few simple steps:

1. All reads mapping to a subset of scaffolds is extracted using a list of the scaffold names and a SAM file of the mapping.

```
perl \multi-metagenome\reassembly\extract.fasta.from.sam.using.list.pl -s mapping.sam -l
list.of.scaffols.txt -o subset.fa
```

   - The output is a fasta file with all reads (subset.fa).

2. Next all missing PE reads are extracted from the original reads.

```
perl \multi-metagenome\reassembly\extract.fasta.pe.reads.using.single.pl -p paired.fa -s subset.fa
```

   - p: is all the original fasta reads used for the assembly.
   - s: is the subset of reads extracted in "step 1".
   - The output is two fasta files with read1 (p1.fa) and read2 (p2.fa) for all paired-end reads.

3. The genome can now be reassembled with any standard genome assembler where the assembly parameters can be optimized. After reassembly Cytoscape (step 11) and Circos (step 12) can used to inspect the assembly.

Step 12: Assembly inspection using Circos

No assemblies are perfect. Hence, manual refinement it is often needed to improve the assemblies. The script multi-metagenome/circosviz/circosviz.pl generates data that enable visual inspection of the assemblies through the software Circos.

While Circos has a steep learning curve excellent tutorials are available at http://circos.ca/ and it is highly recommended to start there.

1. Data generation using multi-metagenome/circosviz/circosviz.pl. The script generates an overview of all non-canonical paired-end connections in a given assembly. As input the script needs a SAM file of the assembly and a fasta file of the scaffolds.

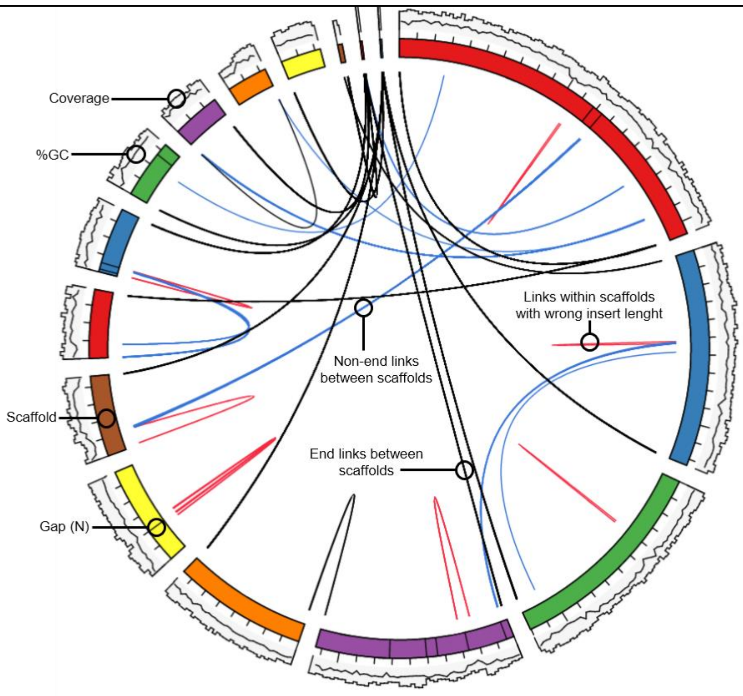
```
perl \multi-metagenome\circosviz\circosviz.pl -i mapping.sam -f scaffolds.fa -e 500 -m 3000 -a 125 -b 10000 -p 1000
```

- i: SAM file of the mapping of PE reads to the assembled scaffolds. The read name structure is the same as required in the Cytoscape script.
- f: a fasta file of the scaffolds. This is used to calculate GC content and identify gaps (N's) in scaffolds.
- e: the maximum distance from the start or end of the scaffold to be considered an "end match".
- m: the minimum scaffold length to allow reads to match start and end of the same scaffold.
- a: the average read length of the reads.
- b: the bin size used to display coverage and GC content on the Circos plots.
- p: the minimum distance to be considered a split PE read. Only split PE reads are visualized.
- g: add FRCbam Features using the Feature.gff output of FRCbam

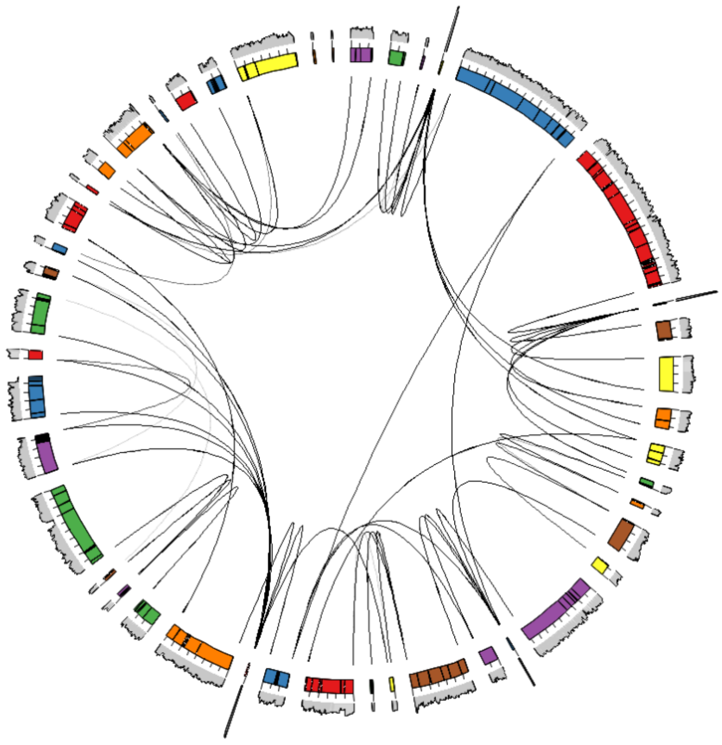
2. A number of files are produced that are needed for visualization in Circos.
    a. circos.karyotype.txt specifies the default order of scaffolds, their length and color. By default they are arranged by decreasing length.
    b. circos.coverage.txt specifies the coverage of each scaffold in the given bin size (-b).
    c. circos.gc.txt specifies the coverage of each scaffold in the given bin size (-b).
    d. circos.ends.txt describes all PE reads mapping to ends of different scaffolds.
    e. circos.dcontigs.txt describes all PE reads mapping to different scaffolds, where at least one of the reads do not map to a scaffold end.
    f. circos.scontigs.txt describes all PE reads mapping to the same scaffold but violating the normal PE length constrain (-p)
    g. circos.rules.txt defines rules for coloring reads between scaffolds the same color as the parent scaffold.
    h. circos.count.dcontigs.txt: Contig coverage file of reads mapping on different contigs. The 3 "count" files are usefull for locating regions of the genome where the number of reads mapping are over a given threshold.
    i. circos.count.ends.txt: Contig coverage file of reads mapping to ends of different contigs.
    j. circos.count.scontig.txt: Contig coverage file of reads mapping within the same contig.
    k. frc.tracks.txt: Reference free validation statistics generated by FRCbam (optional).

3. To illustrate how to generate a default Circos plot an example dataset is supplied in \multi-metagenome\circosviz\circosexample. It contains two folders \data and \etc. All generated data should go in the \data folder. The \etc folder contains the Circos configuration file that is needed to generate the plot seen below.

```
perl circos.pl -conf etc\denovoviz.conf
```



4. A more complex example is shown below. Only links between ends are shown and only coverage on the outer axis. It can be seen that several scaffolds have local low coverage regions that needs inspection. In addition, several scaffolds seem to have repeats integrated in the end that needs to be split from the scaffold. More dense lines indicate a high number of connections (this is only visible in the .png output). Circos offers full customization and zoom, which is very useful to investigate selected structural problems in detail.



Step 13: Identification of rRNA genes

There are many ways of identifying rRNA genes. We usually take a low-tech local blast approach to get an initial overview of rRNA genes in the assembled scaffolds:

1. Download the current greengenes 16S database (or any other 16S/23S database you like): http://greengenes.secondgenome.com/downloads
2. To make the search substantial faster closely related sequences are clustered using usearch. 90% similarity usually does the trick:

```
usearch --sort current_GREENGENES_gg16S_unaligned.fasta --output 16Ssorted.fa
```

```
usearch --cluster 16Ssorted.fa --id 0.90 --seedsout gg_90.fa
```

3. The 16S clustered greengenes database is blasted against database of our assembled scaffolds:

```
makeblastdb -in assembly.fa -dbtype nucl
```

```
blastn -query gg_90.fa -db assembly.fa  -num_threads 10 -max_target_seqs 20 -outfmt 6 -evalue 1e-10 -
out gg_blast_results.txt
```

4. Finally, the longest match for each scaffold is extracted using a small perl script. The script returns a fasta file with the identified sequences. "m" denotes the minimum length of sequences to return.

```
perl \multi-metagenome\misc.scripts\extract.long.hits.from.blast.pl -b gg_blast_results.txt -d
assembly.fa -m 500 -o assembly.16S.fa
```

5. The sequence header is used to store information on where the sequence originates from. The extracted sequences can now be classified using e.g. RDP or SINA online.

```
>68.890.56.834.2446
ATCGATCGATCGATCG…
```

- Scaffold name
- Start position on the scaffold
- End position on the scaffold
- 16S gene length
- Total length of parent scaffold