Analysis of baboon miRNA

INFBIO5120/9120 Autumn 2012

1. Preparations

Background:

In the case of baboon (*Papio Hamadryas*) there is no annotated genome available so we will be using sequences from miRBase for the alignment. MiRBase has two types of miRNA sequences available, hairpin (precursor) sequences and mature sequences. Both of them can be used. We will download and use the mature sequences as our reference. This will make it easier to separate the mature and mature star sequence reads into two categories, and we do not need to consider reads other than those from miRNA that may align to the hairpin. Baboons and humans are closely related, so we will use only the set from *Homo Sapiens* (hsa). Our reads are already quality checked, adapter is removed and they are collapsed into fasta files, so we can use them as they are.

Practical:

Before starting, make a mirna directory and use it as your working directory for this exercice.
Tip: Use the mkdir and cd commands, check with pwd

1. Downloading the miRNA sequence file

   wget ftp://mirbase.org/pub/mirbase/CURRENT/mature.fa.gz
   This will download the mature sequences of all known miRNAs in miRBase. The file is in gzipped fasta format.

2. Unzipping the .gz file

   gunzip mature.fa.gz
   Unzips the file to get the regular fasta format.

Fasta format example

```
>hsa-miR-20a-5p MIMAT0000075 Homo sapiens miR-20a-5p
UAAAGUGCUUAUAGUGCAGGUAG
>hsa-miR-20a-3p MIMAT0004493 Homo sapiens miR-20a-3p
ACUGCAUUAUGAGCACUUAAAG
>mmu-miR-101a-5p MIMAT0004526 Mus musculus miR-101a-5p
UCAGUUAUCACAGUGCUGAUGC
>mmu-miR-101a-3p MIMAT0000133 Mus musculus miR-101a-3p
UACAGUACUGUGAUAACUGAA
>hsa-miR-22-5p MIMAT0004495 Homo sapiens miR-22-5p
AGUUCUUCAGUGGCAAGCUUUA
```

3. Getting a file with only the sequences from human miRNAs

grep -A 1 "hsa" mature.fa | grep -v "^--" > hsa_mature_U.fa

The grep command looks for all lines in the file mature.fa containing the phrase “hsa”, which is the identifier for human miRNAs. The option –A 1 adds the line after the hit to the output. We need to add this to get both the identifier and the sequence. This option adds -- some places in the output, we remove those by using grep –v “^--“. The –v option prints out all non-matching lines. The ^ sign means start of the line. So, all lines that do not start with -- will be kept. The outputfile is named hsa_mature_U.fa.

4. Changing the Us into Ts

cat hsa_mature_U.fa | tr "U" "T" > hsa_mature_T.fa

As you may have noticed, the sequences have Us instead of Ts (because they are RNA not DNA sequences). Since our reads are sequenced from cDNA and contain Ts, not Us, we have to change this by using cat to read the file line by line and pipe the output to the tr command that will change all occurrences of U to T.

5. Inspecting results

head mature.fa

Look at the top lines in the first file.

head hsa_mature_U.fa

Look at the top lines in the file with Us to see that it has only hsa-mirs.

head hsa_mature_T.fa

Look at the top lines in the result file to see that it has Ts instead of Us.

6. Copying the read files and an R file to your mirna directory

cp /data/mirna/bab_1_1_HighLDL_HCHF.fa .
cp /data/mirna/bab_1_2_HighLDL_Chow.fa .
cp /data/mirna/bab_2_1_HighLDL_HCHF.fa .
cp /data/mirna/bab_2_2_HighLDL_Chow.fa .
cp /data/mirna/bab_3_1_HighLDL_HCHF.fa .
cp /data/mirna/bab_3_2_HighLDL_Chow.fa .

cp /data/mirna/maplot-before-and-after-normalization.R .

2. Alignment

Background:

There are a number of different aligners available, all applying different methods to get the reads correctly placed within the reference sequences of choice. The results will be somewhat different based on which aligner you choose and how you set the options. The options (parameters) you can choose from will be different from aligner to aligner, so taking a look at what each of them has to offer in that respect might be a good idea when deciding which one to use. For example, Novoalign has something called miRNA mode. When this is turned on it searches for a possible matching sequence (that makes hairpin formation possible) in an appropriate distance from where the read aligns. If it finds such a matching sequence, the probability that this is a correct alignment location is estimated to be higher than it would be if the matching sequence was not found. This is an advantage when aligning miRNA reads to whole genomes or hairpin sequences. In our case it serves no purpose, since we will align to the mature sequences only.
Depending on what you will be doing with the data after alignment, you might also want to check what output formats are offered. Some downstream analysis programs require specific formats.
In this exercise we will be using Novoalign, which is said to be a very accurate aligner. The downside is that it is not the quickest one, but since we have collapsed fasta reads and a very small genome that is not a problem. Novoalign reference manual (http://www.novocraft.com/userfiles/file/NovoCraftV2.07.pdf).

After aligning the miRNA sequences of choice, you may want to align to sequences from the Rfam database (a collection of RNA families, http://rfam.sanger.ac.uk/) to get an idea of what the unaligned sequences are. We are not doing this today, but it is considered good practice to do so.

Practical:

1. Making the index

> ***novoindex hsa_mature.nix hsa_mature_T.fa***
> The file hsa_mature.nix is the index we are creating from the fasta file hsa_mature_T.fa.

2. Aligning our reads to Homo Sapiens miRNA with novoalign

> ***novoalign -d hsa_mature.nix -f bab_1_1_HighLDL_HCHF.fa -F FA -r All > bab_1_1_HighLDL_HCHF.nov***
> ***novoalign -d hsa_mature.nix -f bab_1_2_HighLDL_Chow.fa -F FA -r All >bab_1_2_HighLDL_Chow.nov***
>
> -d is our index (database)
> –f is our read file
> -F is for file format, FA is for fasta file
> -r All means we want novoalign to print all the hits
> In our outfile we set the extension to be .nov. Several output formats are available from Novoalign, we use the default novoalign format.

3. Inspecting the output

head -30 bab_1_1_HighLDL_HCHF.nov
head -30 bab_1_2_HighLDL_Chow.nov

This command will give you the first 30 lines of the output. After some informational lines starting with #, you will see the alignments. The columns are described below. For a more detailed description, see the Novoalign Reference Manual.

Column description

1. Read id
2. S for single end, L (1st file) or R (2nd file) if paired end
3. Read sequence
4. Base qualities (. if fasta files)
5. Status (U single alignment, R multiple alignment, QL alignment below quality threshold, NM no alignment found, QC read quality to low or homopolymer read)
6. Alignment score
7. Alignment quality
8. Alignment sequence (fasta header)
9. Aligned offset (1-based position of the alignment in the sequence)
10. Strand (F forward, R reverse)
11. Pair sequence (. when single end)
12. Pair offset (. when single end)
13. Pair strand (. when single end)
14-> Mismatches (list of mismatches and bases inserted or deleted)

tail bab_1_1_HighLDL_HCHF.nov
tail bab_1_2_HighLDL_Chow.nov

The tail command will give you the last ten lines of the output. These lines contain info about the novoalign run, such as how many reads you have and how many of them aligned.

Write down the number of aligned sequences for bab_1_1 and bab_1_2 in the table below, and find the average number of aligned reads for the two samples. You will need this info later.

bab_1_1_HighLDL_HCHF	bab_1_2_HighLDL_Chow	Average

Do steps 2 and 3 for the remaining 4 fasta files.

Tip:

Use the up and down arrow keys and change the previous commands rather than re-writing the whole thing or use the tab key to easily change file names.

3. Annotation and read counting

Background:

The next step after alignment is often annotation. We need to know not only where the sequences align, but also what they are. When aligning to a full genome, one can use the genomic positions of the aligned reads and compare with genomic positions of for example known miRNAs. A way to do this is making .bed files and using the program intersectBed from the package BEDTools. In our case this part is unnecessary. Since we have aligned our reads to the sequences of interest only, we know what we have immediately after aligning.

To be able to compare expression of miRNAs between samples, we must start by counting the number of reads aligned to the miRNAs in each sample. As you remember, several of the reads will align to more than one miRNA, and it is not obvious how this should be handled when counting.
In the case of miRNAs, keeping only the uniquely mapping reads will lead to loss of data because of the many similar miRNAs. Dividing 1 by the number of places a read map, and give for example 1/3 as count to each of three miRNAs for one read, will for the same reason possibly give a false low count for several miRNAs. As calling differential expression requires a high enough count, this is not the best choice for miRNAs. One possible solution is to divide the read count based on the number of uniquely mapped reads for each of the miRNAs sharing a group of reads. Another possible solution is give count 1 to all the miRNAs that share the read. In the latter case, one can keep track of shared reads for each miRNA, and check for each of the differentially expressed miRNAs if they have shared reads. That is what we will do in this exercise, with a python script.

Practical:

1. Find read counts for each miRNA

    count_miRs_course.py bab_1_1_HighLDL_HCHF.nov 1
    count_miRs_course.py bab_1_2_HighLDL_HCHF.nov 1

    The first parameter to this program is the file name, the second is the number of mismatches to allow. We filter out all alignments with more than one mismatch. Prints one file with info about uniquely mapped reads reads and reads shared between several miRNAs, and one count file for use when doing expression analysis. Extensions _mirhits.txt, _mircounts.txt.

Do the same for the remaining 4 .nov files.

2. Inspecting the output

more bab_1_1_HighLDL_HCHF_mircounts.txt
This is the countfile where column 1 is the name of the miRNA and column 2 is the number of reads that aligned to it

more bab_1_1_HighLDL_HCHF_mirhits.txt
This file will for each miRNA show you how the reads aligning to this also align to other miRNAs. We will use these files later. Take a look at the example output in the box beneath. First is the miRNA name, then its groups and the number of reads aligning to each group. Group and read number are separated by :, and the groups are separated by ***.
hsa-miR-4448 has 15 reads aligned to just it.
hsa-miR-548aa has 3 reads aligned to itself and hsa-miR548t-3p.
hsa-miR-200b-3p has 1 read aligned to itself and hsa-miR-200c-3p and 75 reads aligned to itself only.

_mirhits.txt example output

```
>hsa-miR-4448 GROUPS: ***>hsa-miR-4448:15
>hsa-miR-548aa GROUPS: ***>hsa-miR-548aa_>hsa-miR-548t-3p:3
>hsa-miR-652-3p GROUPS: ***>hsa-miR-652-3p:30
>hsa-miR-200b-3p GROUPS: ***>hsa-miR-200b-3p_>hsa-miR-200c-3p:1***>hsa-miR-200b-3p:75
>hsa-miR-25-3p GROUPS: ***>hsa-miR-25-3p:665
>hsa-miR-19a-5p GROUPS: ***>hsa-miR-19a-5p:3
```

4. Normalization

Background:

The last step before calling differential expression is normalization. Normalization based on read counts has been widely used, but is considered not to be an optimal method. We will try it out anyway, to be able to easily see how normalization affects the data. Often, the programs/packages that offer differential expression analysis also have one or more normalization method(s) available. This is the case for our package of choice, edgeR.

Suggested litterature:
Garmire & Subramaniam, 2012. Evaluation of normalization methods in mammalian microRNA-Seq data. RNA **18**: 1279-1288

Practical:

1. Open R

R
Opens R. R is an open source programming language and software environment for statistical computing and graphics. To get back to the regular unix window, you must type q() and enter.

2. Read in the data and merge them into one table with miRNAs as rownames

bab1_HCHF = read.table("bab_1_1_HighLDL_HCHF_mircounts.txt")
read.table is used to read in a file. The argument in quotes is the file and bab1_HCHF is the name of the variable the resulting table is stored in.

head(bab1_HCHF)
Lets you look at the top of the table you have made. Notice that R has added column and row names.

bab1_Chow = read.table("bab_1_2_HighLDL_Chow_mircounts.txt")
Reads in the next file

bab1_table = merge(bab1_HCHF, bab1_Chow, all=T, by="V1")
Merges the two tables based on the value of column V1 (the miRNA)

head(bab1_table)
Look at the resulting table

colnames(bab1_table) = c("miRNA","count1_HCHF","count1_Chow")
Changes the column names

head(bab1_table)
Look at the resulting table

bab1_table[is.na(bab1_table)]=0
If a miRNA is present in only some of the samples, in the others it will be given the value NA when merging the two tables into one. With this command we set it to 0 instead.

rownames(bab1_table) <- bab1_table[,1]
Here, we make the miRNAs the rownames.

head(bab1_table)
Look at the resulting table

bab1_table <- bab1_table[,-1]
Remove the first column

head(bab1_table)
Look at the resulting table

3. Find the normalization factors and apply them

We will normalize to the average of aligned reads in the two samples, that is we will take the average divided by the number of aligned reads in each sample to get the normalization factor for that sample, and then the counts for each miRNA will be multiplied by this factor. You wrote down the numbers for the avarage number of aligned reads and the number of aligned reads for the two samples on page 4.

nfac_bab1_HCHF = average number of aligned reads/number of aligned reads for bab_1_1
nfac_bab1_Chow = average number of aligned reads/number of aligned reads for bab_1_2
Making the variables containing the normalization factors. You need to find the numbers to put in in the table on page 4.

nfac_bab1_HCHF
You can check the value of the normalization factors by writing the variable name and press enter.
nfac_bab1_Chow
Checking the other.

bab1_table$norm1_HCHF = bab1_table$count1_HCHF*nfac_bab1_HCHF
bab1_table$norm1_Chow = bab1_table$count1_Chow*nfac_bab1_Chow
Here we are adding two new columns to the table. The new columns contains the normalized numbers. The $ sign is used to specify that it is a column in the table we are talking about.

head(bab1_table)
Look at the resulting table

4. Plot the data

plot(log2(bab1_table$count1_HCHF+1), log2(bab1_table$count1_Chow+1), pch=16, main="Normalization plot", xlab="bab1_HCHF", ylab="bab1_Chow",xlim=c(0,13), ylim=c(0,13), col="lightskyblue")
Plots the count values for the two samples against each other. This is supposed to be one line when you write it into R.

abline(0,1)
Adds a diagonal line through the plot. miRNAs with the same expression should fall on this line. Notice how the blue dots seem to be a bit skewed to the left of the line. Why is that?

points(log2(bab1_table$norm1_HCHF+1), log2(bab1_table$norm1_Chow+1), pch=16, col="sienna4")
Adds the normalized values for the two samples plotted againts each other to the plot. You can see that the brown (normalized) dots seem to center better around the diagonal line.

q()
Quit R. Answer n to the question about saving workspace.

5. Expression analysis

Background:

There are several statistical packages and programs that offers differential expression analysis. They often include several methods of both normalization and differential expression testing. These methods can differ in which miRNA they call differentially expressed. The difference between two packages/programs can be even greater. In this exercise we will use the R package edgeR.

Practical:

1. Start R again and call the installed edgeR library

R
Starts R

library(edgeR)
Imports the installed library. The message "loading required package limma" should appear.

2. Read in the data and place the counts of all samples in the same table with miRNAs as rownames

bab1_HCHF = read.table("bab_1_1_HighLDL_HCHF_mircounts.txt")
colnames(bab1_HCHF)=c("miRNA", "count1_HCHF")
bab2_HCHF = read.table("bab_2_1_HighLDL_HCHF_mircounts.txt")
colnames(bab2_HCHF)=c("miRNA", "count2_HCHF")
bab3_HCHF = read.table("bab_3_1_HighLDL_HCHF_mircounts.txt")
colnames(bab3_HCHF)=c("miRNA", "count3_HCHF")
bab1_Chow = read.table("bab_1_2_HighLDL_Chow_mircounts.txt")
colnames(bab1_Chow)=c("miRNA", "count1_Chow")
bab2_Chow = read.table("bab_2_2_HighLDL_Chow_mircounts.txt")
colnames(bab2_Chow)=c("miRNA", "count2_Chow")
bab3_Chow = read.table("bab_3_2_HighLDL_Chow_mircounts.txt")
colnames(bab3_Chow)=c("miRNA", "count3_Chow")
Reads the data from files, puts them into variables and renames the columns.

bab_table = merge(bab1_HCHF,bab2_HCHF, all=T, by="miRNA")
bab_table = merge(bab_table, bab3_HCHF, all=T, by="miRNA")
bab_table = merge(bab_table, bab1_Chow, all=T, by="miRNA")
bab_table = merge(bab_table, bab2_Chow, all=T, by="miRNA")
bab_table = merge(bab_table, bab3_Chow, all=T, by="miRNA")
Merges the tables into one table.

bab_table[is.na(bab_table)]=0
rownames(bab_table) <- bab_table[,1]
bab_table <- bab_table[,-1]
Sets the NA values to 0, sets the rownames to be the miRNA names and cuts out the miRNA column.

head(bab_table)
Look at the table

3. Run differential expression analysis with edgeR

d <- DGEList(counts=bab_table, group=c("HCHF", "HCHF", "HCHF", "Chow", "Chow", "Chow"), lib.size=colSums(bab_table))
This edgeR command makes a DGE-object to be used when looking for differentially expressed miRNAs. The columns are parted into groups (HCHF/Chow) and the library sizes are set to be the counts of each library (sample).

keep <- rowSums(cpm(d) > 10) >= 3
d <- d[keep,]
These commands filter out miRNA with low counts. This is a precaution, to make sure we have enough material to make conclusions from. If there are only a few reads in the miRNAs, conclutions based on their expression values might not be trustworthy because these counts may be chance occurrences.

d$samples$lib.size <- colSums(d$counts)
After removing low count miRNAs, the library sizes are reset with this command

d <-calcNormFactors(d)
This commands finds the normalization factors

d$samples
Look at the numbers

source("maplot-before-and-after-normalization.R")
The source command reads in a file with R commands and executes them. In this case, the file called "maplot-before-and-after-normalization.R" contains a function that creates a MA-plot with original values as light blue dots and normalized values as brown dots. MA-plots have the sum of the log expression values in the two samples on the x-axis (log2(s1)+log2(s2))) and on the y-axis is the difference (log2(s1)-log2(s2)).

par(mfrow=c(3,2))
Make a 3 times 2 plot window

par(cex=0.3)
Change font and point size

maplot_before_and_after_norm(d,c(1,2))
maplot_before_and_after_norm(d,c(4,5))
maplot_before_and_after_norm(d,c(2,3))
maplot_before_and_after_norm(d,c(5,6))
maplot_before_and_after_norm(d,c(1,3))
maplot_before_and_after_norm(d,c(4,6))
Using the maplot function. The first argument in the maplot function is a DGE-object (with counts etc), the second argument specifies which columns (samples) in the count matrix that are compared in the maplot.

par(cex=1)
Change font and point size back to default size

d <- estimateCommonDisp(d, verbose=TRUE)
This command estimates the dispersion (variation)

de <- exactTest(d)
This command does the differential expression testing.

topres <- topTags(de)
With topTags, we get out the 10 best results.

write.table(topres, "bab_expression_analysis_top_results.txt", quote=FALSE, sep="\t")
Write the results to the file bab_expression_analysis_top_results.txt. sep="\t" means the file will be tab delimited (columns separated by tabs). quote=FALSE means we do not want R to add quotes when printing.

de_counts <- rownames(topres)
de_cpm <- cpm(d)[de_counts,]
These commands will give you the counts per million for the 10 most differentially expressed miRNAs for all samples.

write.table(de_cpm, "bab_expression_analysis_top_counts.txt", quote=FALSE, sep="\t")
Writes de_cpm to a file called bab_expression_analysis_top_counts.txt.

q()
Quit R

4. Inspecting the results

more bab_expression_analysis_top_results.txt
more bab_expression_analysis_top_counts.txt
Look at the files and note the miRNAs with an adjusted p-value (FDR) lower than 0.05. This is a very common threshold to use in these kind of experiments.

Now that we have some miRNAs that we have found to be significantly differentially expressed, we should check if they are sharing a lot of their read with other miRNAs. For each of your DE miRNAs do:

grep –w "miRNA" bab_*_mirhits.txt

Example: grep -w ">hsa-miR-1" bab_*_mirhits.txt

This looks for the miRNA name you are interested in in all the _mirhits.txt files. The –w option makes it look only for separate words. If we do not use this option, for >hsa-miR-1 for example, we will get a lot of irrelevant results. (Feel free to try it out!)
From the results you will see if this miRNA shares many reads with others.
What does these numbers tell you about hsa-miR-133a and hsa-miR-133b?

You will find some of the results from the original experiment in the file /doc/mirna/Baboon_miRNA.pdf. Take a look at the table at page 11 in the pdf. In this table they have presented the results from some miRNAs associated with metabolic disorder risk factors. They have not presented the ones that are the most differentially expressed, but we can still try to compare. Look at the miRNA names and the associated p-values in the High-LDL-C column. If you also look at your top ten results and their p-values, does their results seem to be consistent with your results? If not, what could be possible reasons for this?