01/12/2019

Manual for k-mers-based GWAS

This manual details how to build a *k*-mers presence/absence table and how to use it to run *k*-mers-based GWAS.

The **first part** lists 4 steps to build a k-mers presence/absence table from raw sequencing reads. The **second part** lists the different functionalities in this library. And the **third part** explain how to use the python script that wrap the k-mers GWAS.

Building k-mers presence/absence table from raw sequencing reads:

First you need to decide what is the *k*-mer length to use. The length can not exceed 31 in the current implementation, and should be the same for all individuals. We usually used a *k*-mer length of 31.

Step 1: download KMC and *k*-mers GWAS package

Download KMC which will be used for *k*-mers counting.
Download and unzip/compile the *k*-mers GWAS library.

Step 2: Count *k*-mers and sort for each individual separately

This step will be run separately for each individual. We will combine the output of all individuals in the next steps.

Create a directory for the individual:

```
user@server:kmers_proj$ mkdir individualX
user@server:kmers_proj$ cd individualX
```

Create a textual file within the individual folder with the list of sequencing files, a path for each file in a new line. For example:

```
user@server:individualX$ cat input_files.txt
/server/data/my_project/individualX_run1/seq_R1.fastq.gz
/server/data/my_project/individualX_run1/seq_R2.fastq.gz
/server/data/my_project/individualX_run3/seq_R1.fastq.gz
```

Note: as we count *k*-mers the coupling of paired-end sequencing files are not needed.

Run KMC for the first time, with canonized count of *k*-mers:

```
user@server:individualX$ kmc -t2 -k31 -ci2 @input_files.txt
output_kmc_canon ./ 1> kmc_canon.1 2> kmc_canon.2
```

```
Parameters used (more information in KMC's manual):
t2   - can use two threads
k31  - k-mer length of 31
ci2  - a k-mer has to appear at least 2 times to be counted.
This parameter depends on your coverage, but should be the same
for all the individuals.
```

Run KMC for the second time, counting all *k*-mers with no canonization:

```
user@server:individualX$ kmc -t2 -k31 -ci0 -b @input_files.txt
output_kmc_all ./ 1> kmc_all.1 2> kmc_all.2
```

```
Parameters used (more information in KMC's manual):
b    - do not canonized k-mers
ci0  - this time we count all k-mers
```

Combine the information from the two KMC runs to one list of k-mers:

```
user@server:individualX$
<KMERS-GWAS-PATH>/bin/kmers_add_strand_information
output_kmc_canon output_kmc_all 31 kmers_with_strand
```

```
Parameters used:
31   - k-mer length
```

This process should be repeated for each individual we wish to use in our GWAS analysis.

The KMC files, especially from the second run, can be large. Therefore, it is recommended to remove the KMC files once the information was combined[1]:

```
user@server:individualX$ rm *.kmc*
```

Useful statistics to collect: collect in one table for all individuals: (i) the number of reads, (ii) the number of unique canonized *k*-mers, (iii) the number of non-canonized *k*-mers, and (iv) the number of *k*-mers with each type of flag (1,2, or 3). These numbers are found in the log files created by the KMC's runs or by the log from *kmers_add_strand_information*. Make sure the numbers makes sense. For example, plot the number of unique *k*-mers as a function of the number of reads. You expect to see a correlation between the two, with number of canonized *k*-mers showing signs of saturation (weaker slope for higher values). It is also important to notice if there are individuals which deviate significantly from the general trend, which can be due to technical issues. These specific samples are the ones you might want to omit from further analysis.

Step 3: List all *k*-mers to be used in GWAS from all individuals

In this step we will define the set *k*-mers used for our *k*-mers table. We will go over the k-mers lists from each individual and combine them to one list. The *k*-mers will be filtered according to two criteria: First, a *k*-mer needs to appear in at least X (e.g. X=5) individuals. Second, a *k*-mer will have to appear in each canonized/non-canonized form in at least *P* percent (e.g. *P*=20%) of individuals from which it appeared in.

For example: X=5, P=20%, and we have 1000 individuals

- *k*-mer **x** was found in 4 individuals - **x** is filtered as 4<X.
- *k*-mer **x** was found in 98 individuals: 10 individuals in canonized and non-canonized form, in 7 only in canonized form, and in 81 only in non-canonized form - **x** will be filtered as it appeared only in 17 (7+10) individuals in non-canonized form. However, 17 < 98*0.20=19.6.

We will create a list of all individuals *k*-mers list files. Each line in the file will have the full path of the file with the k-mers list, followed by a **<tab>** and the individual name. For example:

```
user@server:kmers_proj$ cat kmers_list_paths.txt
/server/kmers_proj/individualX/kmers_with_strand  X
/server/kmers_proj/individualY/kmers_with_strand  Y
/server/kmers_proj/individualZ/kmers_with_strand  Z
/server/kmers_proj/individualW/kmers_with_strand  W
…
```

[1] You can save the statistic of k-mers appearances (see second part) before deleting the KMC files

For example, you can use the following command to create this file (if you want to include all subdirectories):

```
ll /server/kmers_proj/ | tail -n +2 | awk '{printf
"/server/kmers_proj/%s/kmers_with_strand\t%s\n", $NF,$NF}' >
kmers_list_paths.txt
```

Combine lists of k-mers to one file:

```
user@server:kmers_proj$
<KMERS-GWAS-PATH>/bin/list_kmers_found_in_multiple_samples
kmers_list_paths.txt kmers_to_use 5 31 1000000 0.2
```

Parameters used:
Kmers_list_paths.txt - list of k-mers lists files
Kmers_to_use - output file
5 - minimum number of individuals a k-mer needs to appear in
31 - *k*-mer length
1000000 - initial size of hash table used
0.2 - minimum percent a k-mer needs to appear in non-/canonized form

Files outputted:

1. kmers_to_use - binary file with the filtered list of k-mers to use.
2. kmers_to_use.shareness - Textual file with counts of how many k-mers appeared in any number of individuals. Only take into account k-mers which were not filtered out.
3. kmers_to_use.stats.both / kmers_to_use.stats.only_canonical / kmers_to_use.stats.only_non_canonical - Textual files containing a matrix of size (N+1)x(N+1), where N is the number of individuals. In row **i** and column **j**, the number indicates the number of *k*-mers which appeared (i-1) times in total and (j-1) times in the files specific form, that is in both forms, only in canonical form, or only in non-canonical form.

Step 4: Create the *k*-mers table

Now we can create the *k*-mers table, containing the presence/absence pattern of each *k*-mer in the population.

```
user@server:kmers_proj$ <KMERS-GWAS-PATH>/bin/build_kmers_table
kmers_list_paths.txt kmers_to_use kmers_table 31
```

```
Parameters used:
kmers_list_paths.txt      - list of k-mers lists files
kmers_to_use              - list of all k-mers to use
kmers_table               - output base filename
31                        - k-mer length
```

Files outputted:

1. kmers_table.table - Binary file with all the *k*-mers presence/absence information.
2. kmers_table.names - Textual file with name of all individual in seperate lines.

After creating the *k*-mers table, there is no longer the need for the separate *k*-mers list from each individual. Therefore, to save space, all these files (...*kmers_with_strand*) can be removed.

Functionalities of the library

Calculation of kinship matrix

To calculate the kinship matrix from the *k*-mers table run this:

```
user@server:kmers_proj$ <KMERS-GWAS-PATH>/bin/emma_kinship_kmers
kmers_table 31 0.01 > kmers_table.kinship
```

```
Parameters used:
kmers_table     - output base filename
31              - k-mer length
0.01            - minimum minor allele frequency of k-mers to use
```

Files outputted:

kmers_table.kinship - a textual file with a matrix of size NxN, where N is the number of individuals. In position i,j in the matrix there is the relationship between individual i and j. Relatedness are calculated as in EMMA[2], where 0 is the minimum relatedness and 1 is the maximal. The order of the samples is the same as in the kmers_list_paths.txt file.

Note on performance: This command might take a few days to run, according to the number of variants included in the analysis (the MAF threshold affects the running time). For example, for MAF of 0.01 on ~1000 *A. thaliana* individuals it took around 5 days to run. This command can

[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3386377/

be significantly optimized if needed, however, as it has to run only once for a project, I see no reason to do so for now.

k-mers table conversion to PLINK binary format

The k-mers table can be converted to PLINK binary file, coding k-mers presence/absence as two homozygous variants. This option is useful if you want to run GWAS on all *k*-mers using an external GWAS software, and it can be used for other purposes.
To convert the *k*-mers table, run the following:

```
user@server:kmers_proj$ <KMERS-GWAS-PATH>/bin/mers_table_to_bed
-t kmers_table -k 31 -p phenotypes.pheno --maf 0.05 --mac 5 -b
10000000 -o output_file
```

This procedure condenses the k-mers table only to the individuals specified in the *phenotypes.pheno* file. It will also filter for the minor allele count (MAC) and minor allele frequency (MAF) provided by the user. The output matrix can be separated into smaller matrices of the defined maximum batch size. This is done so one can run association mapping in parallel on each such matrix.

```
Parameters used:
-o output_file - output base filename
-b 10000000 - maximal variants in each PLINK binary file
-u - (optional) output only unique presence/absence patterns
```

Format of phenotype file: the phenotype file should be with two columns separated by tabs (\t), with the header “accession_id” and “phenotype_value”. The first column lists the name of the individuals, as in the kmers_table, and the phenotype_value the phenotypic values in numerical form.

This PLINK binary files can then be used as input to SNP-based GWAS program. Notice that in this case you will also need to condense the kinship matrix (*k*-mers- or SNP- based) to the same individuals in the phenotype file, and in the same order.

Run *k*-mers-based GWAS with permutation based-threshold

To run k-mers based GWAS on your phenotype you need to run the python wrapper script. For example:

```
user@server:kmers_proj$ python2.7
<KMERS-GWAS-PATH>/src/py/pipeline.py --pheno phenotype.pheno
--kmers_table kmers_table -p 8 -l 31 --outdir output_dir
--gemma_path /.../gemma
```

Parameters used:

--pheno **phenotype.pheno** - a file with numerical phenotypic information (format described above)

--kmers_table **kmers_table** - path to k-mers table

-l **31** - k-mers length

-p **8** - maximum number of threads to use

--outdir **output_dir** - path to a directory for output

--gemma_path **/.../gemma** - path to GEMMA executable

In the second step of the k-mers based GWAS, we use GEMMA. We used and tested version 0.96, but other versions should work as well, however, small modifications might be needed.

Output files inside the output directory:

The output directory contains many intermediate files of the pipeline. The following files are the most relevant for the user:

- kmers/pass_threshold_5per - the GEMMA results passing the 5% threshold
- kmers/pass_threshold_10per - the GEMMA results passing the 10% threshold
- kmers/output/phenotype_value.assoc.txt.gz - all the GEMMA results
- kmers/threshold_5per - the (-log10) threshold for 5% family-wise error rate
- kmers/threshold_10per - the (-log10) threshold for 10% family-wise error rate
- kmers/pheno.pattern_counter - the number of unique presence/absence patterns of *k*-mers used in the GWA