umi4cPackage

Omer Schwartzman

2016-05-17

# umi4cPackage

#### *Omer Schwartzman*

#### *2016-05-17*

UMI-4C Package is a processing and analysis pipeline for UMI-4C experiments.

## Background

This package is intended for processing and analysis of UMI-4C experiments. The UMI-4C protocol exploits sonication of 3C template material to facilitate molecule counting and quantitative analysis of chromatin contacts of one or several genomic loci at high.

The two parts of the package are:

* **Processing of of raw reads (FASTQ format) to genomic tracks:** **Paired-end** FASTQ files are parsed, splitted according to the restriction enzyme sequence (Usually DpnII) and summarized in adjacency table (*adj* file). This part uses as a bunch of *Perl* scripts wrapped in R.
* **Analyzing UMI-4C genomic tracks and producing nearcis figures:** Genomic tracks are dynamically processed to produce nearcis profiles at the desired genomic region specified by the user.

## Installation

### Requirements:

The pipeline is designed to run on a standard linux machine. The basic requirements are:

* *Perl*
* *Bowtie2*: <http://bowtie-bio.sourceforge.net/bowtie2>.
* Reference index for *Bowtie2*.
* R packages:
  + *devtools*.
  + *misha*.
  + *zoo*.

#### Installing misha package:

Since *misha* package is not in CRAN, we will need to install it directly as follow:  
Download the package from: <http://compgenomics.weizmann.ac.il/tanay/?page_id=617>.  
Install the package:

install.packages("misha\_3.2.7.tar.gz", repos=NULL) #Installs misha package from file

#### Importing UMI-4C package:

Download and install *umi4cPackage*:

devtools::install\_bitbucket("tanaylab/umi4cpackage")  
library(umi4cPackage)

##### Preparing genomic database

The pipeline requires a special database called *trackdb* which contains binary genomic tracks and other data, as well as restriction site database in text format called *redb*.

Downloading the data (~2.5Gb) is done from inside the package.  
We will first create a new project directory:

mkdir my\_u4c\_proj

From *R* we will download the data to the provided folder:

library("umi4cPackage")  
# Use build 'hg19' for DpnII enzyme  
p4cBuildRequirements(output\_dir = "my\_u4c\_proj", genome = "hg19", reseq = "GATC")

Two directory will be generated after downloading the extracting the data:

* **my\_u4c\_proj/hg19/trackdb**
* **my\_u4c\_proj/hg19/redb**

### Configuration files

The next step is to set up the *conf* files. To do so we will first need to dump templates of conf files to a conf directory:

p4cDumpConfFiles(conf\_dir = "my\_u4c\_proj/conf")

Now we will need to go to the supplied directory (my\_u4c\_proj/conf in our case) and complete the *conf* files.  
**Make sure that all newlines are linux compatible (\n) and not windows (\r\n)!**

* **paths.conf**: Set the paths to the directories and dependencies.
* **samples.txt**: Samples table. One row for each experiment.
* **baits.txt**: Baits table. One row for each bait.
* **pipeline.conf**: Pipeline settings. The defaults should work fine for **DpnII**.
* **p4c.conf**: Analysis settings. The defaults should work fine.

## Importing and analyzing UMI-4C files.

### Initialization:

We need to load the configuration files to our envirnoment by:

p4cLoadConfFiles(conf\_dir = "my\_u4c\_proj/conf")

#### Working example:

Download example fastqs from: <http://compgenomics.weizmann.ac.il/tanay/?page_id=617>. These files include three UMI-4C experiments done on three cell lines on the *ANK1* promoter.  
Extract the files to a directory.  
In **samples.txt** - change *fastqs\_dir* to the working directory with the downloaded example files. Notice that the pipeline detect the files associated files by the regular expression pattern defined in *fastqs\_regex*. See that *Bait\_IDs* of the samples is 1.  
In **baits.txt** - Inspect that indeed the *Bait\_ID* 1 is correctly filled.

##### Importing genomic tracks

Now we will run the pipeline on a single sample:

p4cCreate4CseqTrack(sample\_ids = 1)

All intermediate files will be saved in the workdir that was defined in *TG3C.workdir* parameter.

The pipeline uses the samples and baits tables to retrieve the relevant information on experiment ID#1, and transform the raw fastq files to UMI counts (see more details on intermediate steps below). This might take some time, depending on the size of the fastq file and the machine being used. Then UMI counts are imported and saved in a special data structure we refer to as “genomic tracks”. For each sample-bait combination a genomic track will generated. In our example, a track named umi4C\_example\_CMK\_ANK1\_TSS will be generated. This track contains data for sample CMK and bait *ANK1\_TSS*.  
After successfully importing the data, the pipeline will report some essential statistics.  
These statistics can also be accessed from 4CQC.txt in the sample workdir directory.

Once imported the genomic tracks are saved in our trackdb, and can be listed by:

gtrack.ls()  
#> [1] "redb.AAGCTT\_flen" "redb.AAGCTT\_gc"   
#> [3] "redb.AAGCTT\_map" "redb.feAAGCTT\_flen"   
#> [5] "redb.feAAGCTT\_gc" "redb.feAAGCTT\_map"   
#> [7] "redb.feGATC\_flen" "redb.feGATC\_gc"   
#> [9] "redb.feGATC\_id" "redb.feGATC\_map"   
#> [11] "redb.GATC\_flen" "redb.GATC\_gc"   
#> [13] "redb.GATC\_map" "umi4C\_example\_CMK\_ANK1\_TSS"   
#> [15] "umi4C\_example\_DND41\_ANK1\_TSS"

#### Analyzing and plotting

Now we can analyze the data:  
Suppose we want to create an UMI-4C profile 200kb upstream and downstream the bait in our exmaple CMK experiment. The first step is to create p4cProfile object:

CMK\_fc <- p4cNewProfile("umi4C\_example\_CMK\_ANK1\_TSS", scope\_5=200000, scope\_3=200000)  
#> will use stat\_type linear  
#> done extract, contacts in window: 4042

Plot nearcis profile:

plot(CMK\_fc)  
#> will create smoothed trend for track  
#> plotting from41454693 to 41854693  
#> range of values in dgram: from 0.1 to 25.4  
#> transform to log2(fraction of max): from -7.98868468677217 to 0

![](data:image/png;base64;base64,) This command generates a plot of the 4C profile with a smoothed trend and a domainogram. It accepts some more parameters which are described in the function documentation.  
For example, if we want to save the domainogram to a png file we can do:

plot(CMK\_fc, png\_fn="figs/CMK\_ANK1\_TSS.png")

#### Compare the profile to another profile

To compare two 4C experiments, we first import two other profiles (notice, this time we will do it in batch):

p4cCreate4CseqTrack(sample\_ids = c(2,3))

Two new tracks were created. To list the tracks again by gtrack.ls().

After the tracks were imported we can produce comparative plots. For example, we will compare CMK which express *ANK1* gene to *DND41* which does not.

# Generate a second p4cProfile on the same scope  
DND41\_fc <- p4cNewProfile("umi4C\_example\_DND41\_ANK1\_TSS", scope\_5=200000, scope\_3=200000)  
  
# plot a comparative plot  
plot(CMK\_fc, DND41\_fc)

![](data:image/png;base64;base64,)

Additional feature is the ability to derive mean contact intensity for defined genomic intervals. The following command will return normalized contact intensities of the two profiles, fold change, and p-value (Chi-square test)

fold\_change <- p4cIntervalsMean(CMK\_fc, DND41\_fc, start=41665000, end=41675000)  
knitr::kable(fold\_change, align = 'l')

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| start | end | umi4C\_example\_CMK\_ANK1\_TSS | umi4C\_example\_DND41\_ANK1\_TSS | mean\_scale | log2\_foldChange | p.value |
| 41665093 | 41674794 | 5.468199 | 3.732738 | 10 | 0.5508315 | 0 |