PBGL CNVkit Analysis v1.0

A Laboratory Manual

Anibal E. Morales-Zambrana

Plant Breeding and Genetics Laboratory
FAO/IAEA Joint Division
Seibersdorf, Austria

Created: April, 2021
Last updated: 21 April 2021

Contents

1 Background

[DRAFT]

Copy number variation (CNV) analysis using CNVkit, R, Jupyter Notebooks, Miniconda3, Git, along other packages.

---

2 Installations

Before installing any necessary software, it is recommended to check if the computer is running 32-bit or 64-bit for downloading Miniconda3. Run the following to verify the system:

```
$ uname -m
```

2.1 Miniconda3 (conda)

Download the Miniconda3, or simply "conda", installer:

- Miniconda3 installer for Linux

Run the downloaded installer (for a 64-bit system):

```
$ bash Miniconda3-latest-Linux-x86_64.sh
```

Open a new terminal window for conda to take effect. Verify the installation in new terminal window and update conda:

```
$ conda list
$ conda update --all
$ conda upgrade --all
```

2.2 Git with conda

Git will be installed first to clone locally (download a copy to your local computer) the **pbgl-cnvkit** repository from GitHub. To do so, run the following:

```
$ conda install -c anaconda git
```

2.2.1 Cloning the Necessary Repositories

After the installation, clone the **pbgl-cnvkit** repository to the local computer in the desired directory.

```
$ git clone https://github.com/amora197/pbgl-cnvkit.git
```

A folder called **pbgl-cnvkit** should be listed in the directory. Navigate into it and inspect its items.

```
$ cd pbgl-cnvkit
$ ls -l
```

The **pbgl-cnvkit** directory should contain:

- 3 folders:
  - docs
  - envs
  - output
- 3 files:
  - cnvkit-analysis.ipynb

- config-cnvkit.yml
- README.rst

Once inside the **pbgl-cnvkit** directory, clone **etal/cnvkit** repository that contains the workflow and source code for analyzing copy number variations/alterations.

```
$ git clone --branch v0.9.7 --single-branch https://github.com/etal/cnvkit.git
$ ls -l
```

A new directory **cnvkit** should be present.

2.3 Required Libraries with conda

cnvkit has multiple dependencies, listed below:

- Git
- cnvkit
- Jupyter Notebook

There are two ways to install the rest of the necessary libraries to run cnvkit: automatically or manually. The former is slower, providing a long coffee break (sometimes overnight durations) while the conda installations run. The latter proves a faster way to get the tool up-and-running.

2.3.1 Automatically (slower)

One YAML file, **environment.yml**, is provided inside the **envs/** directory to automatically create a conda environment and install the dependent libraries. This creates the conda environment, along all necessary packages to run cnvkit. Run **environment.yml**:

```
$ conda env create --file envs/environment.yml
```

Once done, the created environment can be verified running:

```
$ conda env list
```

Activate the created environment (**cnvkit**):

```
$ conda activate cnvkit
```

Once done, all the necessary packages should be installed. This can be verified with:

```
$ conda list
```

2.3.2 Manually (faster)

To manually create and activate an environment, run:

```
$ conda create --name cnvkit
```

Once done, the created environment can be verified running:

```
$ conda env list
```

Activate the virtual environment with:

```
$ conda activate cnvkit
```

Start running the installations of the necessary libraries, paying attention to the prompts for each one:

```
$ conda install pyyaml
$ conda install cnvkit
$ conda install notebook
```

Once done, all the necessary packages should be installed. This can be verified with:

```
$ conda list
```

3 Running a Jupyter Notebook

To access the Jupyter Notebooks, run the following command inside the **pbgl-cnvkit** directory:

```
$ jupyter notebook
```

This command will start a Jupyter Notebook session inside the directory the command is run. The user can navigate between directories, visualize files, and edit files in the browser by clicking on directories or files, respectively.

Look for **cnvkit-analysis.ipynb** and click on it to open the Jupyter Notebook and run the analysis.

Note: Jupyter lets the user duplicate, rename, move, download, view, or edit files in a web browser. This can be done by clicking the box next to a file and choosing accordingly.

---

4 Editing the Configuration File

In order to run the CNVkit Jupyter Notebook, the user needs to feed it with a configuration file (**config-cnvkit.yml**) that specifies the paths to the bam files, comparisons to be done, chromosomes to analyze, and parameter definitions for calculating and plotting CNVs.

The configuration file **config-cnvkit.yml** can be found in the same directory as the Jupyter Notebook.

Note: The user needs to edit **config-cnvkit.yml** to point towards bam/bed/fasta files; specify comparisons and chromosomes to analyze; and define the output path.

The configuration file **config-cnvkit.yml** contains multiple parameters to be defined by the user:

- *paths*:
  - sample names and their respective paths to **.bam** files
  - samples can be named as desired but the sample name must be repeated after the colon and prefixed with a *&* sign
  - the *&* prefix sign is used to reference the sample's path in different places of the same configuration file
  - example use:

```
paths:
  mysample: &mysample /home/john/bam_files/mysample.bam
  XYZ-123: &XYZ-123 /home/john/bam_files/XYZ-123.bam
  potato95: &potato95 /home/john/bam_files/potato95.bam
```

- *bed_path*:
  - path to bed file if using varying window sizes
- *fasta_path*:
  - path to fasta file
- *output_path*:
  - path of output files (references, plots, CNVs) to the **pbgl-cnvkit/output** directory
- *references*:
  - references to use for making comparisons
  - a reference can be built from multiple "normal" files, which are in turn listed under *files_for_ref*
  - an output reference name needs to be defined
  - example use:

```
references:
  first_reference:
    output_ref: &first_reference my_first_reference.cnn
    files_for_ref:
      - *first_bam
      - *second_bam
      - *third_bam
```

- *comparisons*:
  - comparison names with respective reference and mutant samples per comparison

  - each comparison can be named as desired
  - the sample names to be used as *control* and *mutant* need to be prefixed by a * sign
  - the * prefixed sign is used to extract the sample's path defined in the *paths* section
  - example:

```
comparisons:
  variety-x:
    comparison-1:
      reference: *reference-one
      mutant: *potato95
    a-different-comparison-278asd:
      reference: *another-reference
      mutant: *XYZ-123
```

- *chromosomes*:
  - list of chromosome names to analyze
  - chromosome names can be extracted from a bam file's header
- *cores*:
  - a digit, specifying the number of cores to parallelize the workflow

5 Running the cnvkit-analysis Jupyter Notebook

Note: It is recommended to duplicate the **cnvkit-analysis.ipynb** notebook and then renaming the copy before doing any edits to the notebook.

Click on **cnvkit-analysis.ipynb** and a new tab will open the notebook.

The notebook contains cells that are populated by text or code. Information about each command is provided in the notebook to guide the user. It consists of four parts:

1. Setup and Configuration File Extraction
2. Reference Creation
3. Comparisons
4. Plotting

5.1 Setup and Configuration File Extraction

A configuration file config.cnvkit.yml in the config/ directory is provided for specifying file paths, references to build, comparisons to analyze, chromosomes to plot, and cores for parallelization.

All the analyses are done by extracting parameters from the configuration file, looping with Python, and running bash system commands through Python's os library.

5.2 Reference Creation

Compiling a copy-number reference from given files or directory (containing normal samples). The reference can be constructed from zero, one or multiple control samples. If given a reference genome, also calculate the GC content and repeat-masked proportion of each region. Files needed:

- bam files of normal/control sample(s)
- fasta file
- bed file with target regions

There are two ways to run the command:

5.2.1 Option 1

Using wildcard * to specify all normal/control files to use for reference building.

```
cnvkit/cnvkit.py batch --normal normalFile*.bam \
--output-reference /output/path/nameOfReferenceToCreate.cnn \
--fasta /path/fastaFile.fna \
--targets /path/bedFile.bed \
--output-dir /output/path \
-p numberOfCoresToUseForParallelization
```

5.2.2 Option 2

Listing each normal/control file separately if wildcard cannot be applied.

```
cnvkit/cnvkit.py batch --normal normalFile1.bam normalFile2.bam normalFileN.bam \
--output-reference /output/path/nameOfReferenceToCreate.cnn \
--fasta /path/fastaFile.fna \
--targets /path/bedFile.bed \
--output-dir /output/path \
-p numberOfCoresToUseForParallelization
```

5.3 Comparisons

Using a reference for calculating coverage in the given regions from BAM read depths. Command:

```
cnvkit/cnvkit.py batch mutantFile.bam \
-r /output/reference/path/referenceFile.cnn \
-d /output/path
-p numberOfCoresToUseForParallelization
```

5.4 Plotting

Plot bin-level log2 coverages and segmentation calls together. Without any further arguments, this plots the genome-wide copy number in a form familiar to those who have used array comparative genomic hybridization (aCGH). The options –chromosome or -c focuses the plot on the specified region. Command:

```
cnvkit/cnvkit.py scatter /output/path/mutantFileName.cnr \
-s /output/path/mutantFileName.cns \
-c chromosomeName
-o /output/path/nameOfPlot.png
-p numberOfCoresToUseForParallelization
```

To run a cell, click on the corresponding cell and press **Ctrl + Enter** or **Shift + Enter**.

6 References

BMC Bioinformatics Publication:

- Talevich, E., Shain, A. H., Botton, T., & Bastian, B. C. (2014). CNVkit: Genome-wide copy number detection and visualization from targeted sequencing. PLOS Computational Biology 12(4): e1004873. doi: 10.1371/journal.pcbi.1004873

GitHub repositories:

- etal/cnvkit
- amora197/cnvkit