# FocalScan manual 1.1

# Contents

# 1. Introduction

FocalScan identifies genomic regions where many tumors show simultaneous increases in DNA copy-number amplitude (CNA) and RNA expression (or conversely for DNA deletions). Empirically, many important oncogenes show this pattern of alteration. The FocalScan score is based on the dot product of CNA and RNA changes. This puts equal weight to the two variables, but requires coordinated changes in both to achieve a positive score.

Examples (for a given genomic position):

Some tumors show both elevated CNA and RNA levels -> medium score
Many tumors show both elevated CNA and RNA levels -> high score
Many tumors show elevated CNA and highly increased RNA -> very high score
Many tumors show elevated CNA, but RNA is unchanged -> neutral score

Regions with coordinated CNA and RNA reduction will also score favorably.

FocalScan computes two basic statistics: One is calculated as described above. The other is based on 'high-pass filtered' CNA data, where large (>10 Mbp) segments, such as arm-level events, are effectively subtracted, leaving only the focal/small alterations. Apart from that, the same scoring method is used in both cases. The second method can be very sensitive at identifying genes of interest in focally altered regions.

Importantly, FocalScan can also be run in a "non gene-centric" fashion: The genome is scanned at high (500 nt) resolution by dividing chromosomes into small (1000 nt) overlapping tiles. As such, it does not care about preconceptions about gene locations. RNA-seq data is used to quantify transcription and scores are computed for each tile. This makes FocalScan suitable for identifying e.g. novel non-coding RNAs that are altered in tumors.

## 1.1 Requirements

FocalScan is a command-line based tool and requires either an installation of MATLAB (at least version R2015b), or and installation of the MATLAB runtime if the pre-compiled executable will be used (only available on Mac and Linux, installer included). In addition, the following software is likely to be useful for pre-processing the data.

- bedtools (2.21.0) [1]

- HTseq (0.6.1) [2]
- samtools (1.1) [3]

Version numbers in parenthesis indicates the versions tested.

Additionally, Integrative Genomics Viewer (IGV) [4] is recommended for visualizing some of the results.

**Note:** Tile-level analysis will require a large amount (>30 GB, or more depending on the number of samples analyzed) of RAM, so it is advisable to run this on a powerful server. Gene-level analysis should work fine on a laptop or desktop computer with at least 8 GB RAM, however.

## 1.2 Installation

There are two main ways to run this program. Either from within MATLAB, or directly from the Unix/Linux command line with or without the compiled executable (does not require a MATLAB installation). If MATLAB is not installed or if an older version is installed (and updating is not an option), use the compiled executable.

Download either the MATLAB scripts or the compiled executable from `github.com/jowkar/focalscan` and place the files in any suitable directory. If usage via the compiled executable is intended, run the "Installer" file to install the MATLAB runtime environment. Otherwise, from within MATLAB, add the path to the program with `addpath(genpath('path_to_focalscan'))`. To run the program, refer to the examples in section 2.1.

**Note:** There are two different shell scripts available to execute the program: `focalscan_compiled.sh` and `focalscan.sh`. The first one is used to execute the compiled executable and the second one to run the MATLAB scripts (without a compiled executable).

### 1.2.1 Detailed instructions

To run this software, it is important that all necessary environment variables are set up the right way (and that the correct version of the MATLAB runtime is installed, if using the compiled executable). This section will detail this process for the alternative ways of running the program.

#### Mac/Linux, compiled executable

1. Download the file titled `FocalScan_compiled_Linux.zip` or `FocalScan_compiled_Mac.zip`.
2. Open the terminal application and unarchive the file:

```
1 unzip FocalScan_compiled_Linux.zip
```

3. Install the MATLAB runtime (it is important that the runtime is the correct version, in this case v901):

```
cd FocalScan_compiled_Linux
./Installer.install
```

or

```
cd FocalScan_compiled_Mac
./Installer.app
```

This should bring up a window for downloading and installing the runtime. Make a note of where FocalScan is installed and where the runtime is installed (such as `/Applications/FocalScan/application` and `~/bin/MCR/v901`, respectively, or any other directories chosen). The path to the runtime directory (from now on referred to as "`MCR_root`") and the path to FocalScan (from now on referred to as "`path_to_focalscan`") will have to be specified later when running the program. Note also that the program files will be located in a subdirectory named "application". I should also be noted that remote installation on a Linux server was observed to occassionally fail if X forwarding was not used (a bug in the MATLAB installer program).

4. Setup environment variables (in the following, substitute `path_to_focalscan` for the directory where the files were downloaded (for instance, `~/focalscan`):

```
export PATH=$PATH:path_to_focalscan
```

Example:

```
export PATH=$PATH:/Applications/FocalScan/application
```

5. Download and unzip example data to test the installation with, available at the following links:

Annotation files:

```
https://drive.google.com/open?id=0B_52viSz8FLNeUlPZ0c3akU1ZlE
```

Test data:

```
https://drive.google.com/open?id=0B_52viSz8FLNWFRvaUNyMGhCbHM
```

6. Test the installation (gene-level analysis on breast cancer data from TCGA). In the following command, substitute "`MCR_root`" for the installation directory of the MATLAB compiler runtime and `path_to_focalscan` for the installation directory of FocalScan (obtained from step 3 above) (all in one line):

```
path_to_focalscan/focalscan_compiled.sh MCR_root expr_csv ...
    ./example_data/BRCA_expr.csv seg_file ...
    ./example_data/BRCA_cna.seg annot_file ...
    ./annotation/gencode17_symbols.bed reportdir test_gene
```

This might take up to 40 minutes or slightly longer, depending on processor speed and available memory.

7. Inspect the output:

```
cat test_gene/peaks.txt
```

This should list the top ranking genes:

```
Id  Score   Sum_CNA_HP  Chr Start   Stop
ERBB2   942.124267578125    262.793273925781    chr17   ...
    37844167    37886679
CCND1   447.170532226562    220.362243652344    chr11   ...
    69455855    69469242
WHSC1L1 440.08837890625 202.27294921875 chr8    38132544    ...
    38239790
EGFR    129.50341796875 21.137996673584 chr7    55086714    ...
    55324313
TRAF4   117.803070068359    80.5899887084961    chr17   ...
    27071002    27077974
IGF1R   116.806274414062    26.7439022064209    chr15   ...
    99192200    99507759
FGFR2   96.7242736816406    24.9825992584229    chr10   ...
    123237848   123357972
CCNE1   83.940673828125 27.0290393829346    chr19   ...
    30302805    30315215
PHGDH   80.3051528930664    20.5632972717285    chr1    ...
    120202421   120286838
```

Also available in the in the output are a full report with detailed statistics and .wig files that can be visualized with IGV, containing full tracks with scores, copy number amplitudes and mean expression levels of all genes.

### Mac/Linux, MATLAB installed

This assumed that the user has a full MATLAB installation, version 2015b or newer (since some functions only available in this version and later are used by the program).

1. Open the terminal application and download the scripts:

```
git clone git@github.com:jowkar/focalscan.git
```

2. Setup environment variables (in the following, substitute `path_to_focalscan` for the directory where the files were downloaded (for instance, `~/focalscan`) and `path_to_matlab` for the directory where MATLAB is installed (for instance, `~/bin/MATLAB_2015b/bin/`)):

```
export PATH=$PATH:path_to_focalscan
export PATH=$PATH:path_to_matlab
```

Example:

```
export PATH=$PATH:~/focalscan
export PATH=$PATH:~/bin/MATLAB_R2016a/bin
```

The above paths will differ depending on system and where the respective applications were installed by the user.

3. **Optional:** To avoid having to type the commands in the previous step every time the tool is run, add those commands to the ~/.bashrc or ~/.bash_profile file (depending on the operating system).

4. Add permissions to execute the shell scripts:

```
chmod +x *.sh
```

5. Download and unzip example data to test the installation with, available at the following links:

   Annotation files:

```
https://drive.google.com/open?id=0B_52viSz8FLNeUlPZ0c3akU1ZlE
```

   Test data:

```
https://drive.google.com/open?id=0B_52viSz8FLNWFRvaUNyMGhCbHM
```

6. Test the installation (gene-level analysis on breast cancer data from TCGA) (all in one line):

```
path_to_focalscan/focalscan.sh expr_csv ...
    ./example_data/BRCA_expr.csv seg_file ...
    ./example_data/BRCA_cna.seg annot_file ...
    ./annotation/gencode17_symbols.bed reportdir test_gene
```

   This might take up to 40 minutes or slightly longer, depending on processor speed and available memory.

7. Inspect the output:

```
cat test_gene/peaks.txt
```

### Any platform, usage from within the MATLAB environment

1. Open the terminal application and download the scripts (assuming that git is installed, otherwise just download the zip file):

```
git clone git@github.com:jowkar/focalscan.git
```

2. Download and unzip example data to test the installation with, available at the following links:

   Annotation files:

```
https://drive.google.com/open?id=0B_52viSz8FLNeUlPZ0c3akU1ZlE
```

   Test data:

```
https://drive.google.com/open?id=0B_52viSz8FLNWFRvaUNyMGhCbHM
```

3. Open MATLAB and add the scripts to the MATLAB path (will have to repeated each time MATLAB is opened, unless the startup settings are also changed):

```
addpath(genpath('path_to_focalscan'))
```

   Example:

```
addpath(genpath('~/focalscan'))
```

   (If focalscan was downloaded to the home directory on Linux/Mac)

4. Test the installation (gene-level analysis on breast cancer data from TCGA):

```
FocalScan.sh('expr_csv','example_data/BRCA_expr.csv',...
'seg_file','example_data/BRCA_cna.seg','annot_file',...
'annotation/gencode17_symbols.bed','reportdir','test_gene')
```

   This might take up to 40 minutes or slightly longer, depending on processor speed and available memory. The results will be saved to the directory "test_gene".

# 2. Input options

## 2.1 Running FocalScan

The minimal input required to run the program is a path to expression data files, a path to a file with copy number data and the path to an annotation file. Different input parameters are available for this purpose, depending on the format of the input files.

FocalScan can be run either from within the MATLAB environment or directly from the Unix/Linux command line (using the compiled executable and the included wrapper script **`focalscan_compiled.sh`** or **`focalscan.sh`** if an appropriate version of MATLAB is installed). Input parameters are specified as pairs of parameter name and parameter value.

**Note:** There are two separate shell scripts available to run the program on Mac/Linux, depending on whether the compiled executable is used (**`focalscan_compiled.sh`**) or not (**`focalscan.sh`**).

**Note:** Make sure that all environment variables are set up correctly (as detailed in section 1.2) and that the correct version of the MATLAB runtime is installed (if using the compiled executable).

**Example 2.1.1.** Running FocalScan from the Mac/Linux command line (all in one line)

```
1 focalscan_compiled.sh MCR_root expr_csv ...
      example_data/BRCA_expr.csv seg_file ...
      example_data/BRCA_cna.seg annot_file ...
      annotation/gencode17_symbols.bed reportdir test_gene ...
      reportdir test_dir_MCR
```

`MCR_root` refers to the directory where MATLAB is installed.

**Example 2.1.2.** Running FocalScan from the Mac/Linux command line (no compiled executable)

```
1 focalscan.sh expr_path expr_csv example_data/BRCA_expr.csv ...
      seg_file example_data/BRCA_cna.seg annot_file ...
      annotation/gencode17_symbols.bed reportdir test_gene ...
      reportdir test_dir
```

**Example 2.1.3.** Running FocalScan from within MATLAB

```
addpath(genpath('path_to_focalscan'))
FocalScan('expr_csv','example_data/BRCA_expr.csv','seg_file',...
'example_data/BRCA_cna.seg','annot_file',...
'annotation/gencode17_symbols.bed','reportdir','test_gene',...
'reportdir','test_dir')
```

Note that there are two ways to provide expression data. In addition to the CSV format, as shown in the above examples, it is also possible to use a directory containing separate of read count files for each sample (example 2.1.4). See section 2.3 for more details. For copy number data, the SEG format is preferred, but it is also possible to use CSV input.

**Example 2.1.4.** Separate expression read count files as input

```
focalscan_compiled.sh MCR_root expr_path ...
    example_data/read_count_files index_file ...
    example_data/index.txt file_extension .gene_counts seg_file ...
    example_data/BRCA_cna.seg annot_file annotation/gencode17.bed
```

When separate read count files are used, it is necessary to also provide an index file connecting the sample names in the copy number data to the sample (file) names in the expression data (see section 2.3), in addition to the file extension used for the read count files.

## 2.2 List of parameters

Below, all the available parameters are listed, with default values in brackets.

The minimum required input is the path to expression data, copy number data and an annotation file.

The expression data can be given either as a CSV file (for gene-level analysis) or as a directory of separate read count files (gene-level and tile-level analysis). When the second option is used, the file extension of these files must also be given, and additionally an index file. The index file should contain two columns, where the first one includes the file names of the read count files (excluding the extension) and the second one includes the sample names used in the copy number data.

The copy number data needs to be provided in segmented format, in a file following the SEG format (https://www.broadinstitute.org/igv/SEG)

To perform a tile-level analysis, use the included file hg18_hg19_1kb_tiles.bed as annotation

For tile-level analysis, also remember to specify the parameter optional_gene_annot

and give the path to a standard gene annotation file (BED format) in order to report which genes overlap each tile in the final peak report.

**Input**

`expr_csv`: Path to a CSV file containing unnormalized expression data. Columns are expected to correspond to samples and rows to genes. The columns should be titled with sample IDs. (Only for gene-level analysis)

`seg_file`: File containing segmented copy number data for all samples

`annot_file`: Gene annotation or tile definition file in .bed format.

`expr_path`: Path to directory containing files with gene or tile level count data for all samples (given in separate files)

`index_file`: File that links expression data files to sample IDs

`file_extension`: The file extension of the gene or tile-level expression files. {`expr_path`, `index_file`, `file_extension`}: Need to be specified together.

`optional_gene_annot`: When tile-level analysis is performed, providing a gene annotation via this option will enable annotation of the reported peak tiles with respect to overlapping genes

`fast_read`: When set to 1 and separate read count files are used, will assume that all files have identical first columns (gene IDs) in order to speed up reading of these. [0]

**Normalization**

`normalization`: The normalization mode to employ (percentile_expressed is the same as percentile, but only considering genes with reads. Might be useful if several samples have many genes without expression). [percentile] {percentile, percentile_expressed, library_size, none}

`percentile`: The percentile to use when percentile normalization is employed. For instance, "95" will normalize to the median of the top 5 percent most highly expressed genes in each sample [95]

**Score calculation**

`window_size`: Window size used by the focality filter [10e6]

`neutral_thresh`: Absolute copy number amplitude threshold for defining neutral samples [0.1]

`min_neutral`: Demand at least this many neutral samples to examine a given gene/tile (will ignore genes/tiles not meeting this threshold). [20]

`pseudo_expr`: Pseudo expression value to add (needed to avoid division with zero when calculating ratios)

`pseudo_expr_relative`: The pseudo expression value can be specified relative to the median of all non-zero expression values. This parameter defined the relation between the pseudo count and this median. For instance, a value of 10 sets the pseudo count to 10 times the median. [10]

`max_nan`: Maximum proportion of missing values to accept for a given gene/tile [0.1].

**Output and peak detection**

`reportdir`: Directory in which to store output files [.]

`scorefield`: The metric to use as basis for peak detection [fs_hp] {fs, fs_hp, cna, cna_hp}

`only_focal`: When set to 1, will avoid additional calculation of scores without the focality filter (will speed up execution). [0]

`peak_level`: Sets the granularity of the peak detection method. A high value will cause only the most prominent peaks to be reported. A low value will cause additional, less prominent, peaks to be reported [0.6] {0.1 - 1.0}

## 2.3 Data formats

### 2.3.1 Gene-level analysis

**Expression data**

Expression data should be provided in either of the two ways listed below. If the read counts are provided in separate files for each sample (example 3.1.1 shows the expected structure of these files), then place them all in the same directory and specify this directory together with their file extension and an index file, as in example 2.3.1.

**Example 2.3.1.** Expression data as separate read count files

```
1 focalscan.sh expr_path example_data/read_count_files ...
2     index_file example_data/index.txt ...
3     file_extension .gene_counts ...
4     seg_file example_data/BRCA_cna.seg ...
5     annot_file annotation/gencode17.bed ...
6     reportdir example_data/example_output/test_gene
```

The index file links the file names of the read count files to the sample names found in the copy number data SEG file. The file has two columns: The first contains the names of the read count files and the second column lists the corresponding sample names (example 2.3.2).

**Example 2.3.2.** Example of index file

```
1  976cc6d7-7c97-4fd1-8228-661fcd521a21    TCGA-A1-A0SB
2  2fbb8fee-6bf0-4aba-9f9f-fe197470c52b    TCGA-A1-A0SD
3  984e0398-7f92-409e-9fc4-58939d23b1d7    TCGA-A1-A0SE
4  76ec52fc-b274-4fa3-9cad-e3bf12bf7d26    TCGA-A1-A0SF
5  490de977-d563-4867-a721-56ba8e2a2665    TCGA-A1-A0SG
6  f5cbb8c7-bf34-4d42-8042-711e97653ded    TCGA-A1-A0SH
7  b0c31927-2b29-44bb-a626-f90edcf91228    TCGA-A1-A0SI
8  889d4bfc-11a0-4756-8ba9-253ec225bdcb    TCGA-A1-A0SJ
9  1a2bf6fd-af4a-4b72-a13d-4da162a9e57d    TCGA-A1-A0SK
10 4559bf70-dc6e-45b2-a64c-aaf237dd8296    TCGA-A1-A0SM
```

[file name, sample ID]

**Note:** It is required that the CSV file contains one header line with sample IDs. These have to exactly match the sample IDs in the copy number data.

**Note:** The CSV file does not have to include gene IDs. However, then the rows must correspond exactly to the rows (gene IDs) in the provided genome annotation file.

#### Copy number data

Copy number data should be provided as a SEG file. For this, use the parameter "***expr_seg***".

#### Annotation file

In all cases, it is mandatory to also provide a genome annotation in four column BED format (example 2.3.3) (the parameter "***annot_file***"). In the case of a tile-level analysis, the included tile annotation `hg18_hg19_1kb_tiles.bed` may be used. The annotation file needs to have gene/tile IDs that correspond exactly to the rows in the expression data file(s). For tile level analysis, it is recommended to also provide standard gene annotation file (for instance any of the included files "`gencode17_symbols.bed`" or "`gencode17.bed`") so that genes overlapping peak tiles can be written to the output.

**Example 2.3.3.** Genome annotation file

```
1  chrX    99883667    99894988    ENSG00000000003.10
2  chrX    99839799    99854882    ENSG00000000005.5
3  chr20   49551404    49575092    ENSG00000000419.8
4  chr1    169821804   169863408   ENSG00000000457.8
5  chr1    169631245   169823221   ENSG00000000460.12
6  chr1    27938575    27961788    ENSG00000000938.8
7  chr1    196621008   196716634   ENSG00000000971.11
8  chr6    143816614   143832827   ENSG00000001036.8
9  chr6    53362139    53481768    ENSG00000001084.6
10 chr6    41040684    41067715    ENSG00000001167.10
```

[chromosome name, gene start, gene end, gene ID]

### 2.3.2 Tile-level analysis

For tile-level analysis, expression data should be provided as a directory of read count files (one for each sample), together with their file extension and an index file. CSV input is not available for tile-level analysis. Also specify either the included tile definition file (`annotation/hg18_hg19_1kb_tiles.bed`, suitable for the human genomes hg18 and hg19) or a custom one.

**Example 2.3.4.** Expression data input

```
focalscan.sh expr_path example_data/read_count_files ...
    index_file example_data/index.txt ...
    file_extension .tile_counts ...
    seg_file example_data/BRCA_cna.seg ...
    annot_file annotation/hg18_hg19_1kb_tiles.bed ...
    optional_gene_annot annotation/gencode17_symbols.bed ...
    reportdir example_data/example_output/test_tile
```

**Example 2.3.5.** Structure of the tile definition (annotation) file

```
chr1    1   1000    1
chr1    501 1500    2
chr1    1001    2000    3
chr1    1501    2500    4
chr1    2001    3000    5
chr1    2501    3500    6
chr1    3001    4000    7
chr1    3501    4500    8
chr1    4001    5000    9
chr1    4501    5500    10
```

[chromosome name, tile start, tile end, tile ID]

For tile-level analysis it is not mandatory to provide a standard gene annotation, but highly recommended in order to find genes overlapping peak tiles. Use the ***optional_gene_annot*** parameter for this.

## 2.4 Peak detection

The purpose of the peak detection algorithm is to select genome-wide peaks of highly scoring genes or tiles. As described in the paper, the peak detection is performed iteratively across multiple levels of granularity. The parameter ***peak_level*** (a number between 0 and 1) can be used to set the desired level of granularity from which peaks will be returned. A high number causes only the most prominent peaks to be reported, whereas a low number causes additional, less prominent peaks to be detected. By default 0.6 is used. See examples 2.4.4 and 2.4.5 for an illustration of the effect of peak level choice.

### 2.4.1 Standalone peak detection

After a completed FocalScan run, it is possible to run peak detection separately, using the "report.txt" file as input. This could be useful if one desires to investigate either additional, less prominent, peaks or to remove noise by restricting the analysis to more prominent peaks. For this purpose, use either the MATLAB function **`standalone_peakdetection`**, the Mac/Linux command line script **`standalone_peakdetection.sh`** or the compiled executable with **`run_standalone_peakdetection.sh`** (if MATLAB is not installed (only available on Mac/Linux)). The output file is described in example 4.0.2.

General usage:

```
standalone_peakdetection.sh report_file_path annot_file_path ...
    peak_level scorefield out_file
```

Valid options for the "***scorefield***" parameter (the metric to use as basis for peak detection) are:

`fs_hp`: the standard FocalScan score (with focality filter)

`fs`: FocalScan score without focality filter

`sum_cna_hp`: summed copy number amplitudes, with focality filter

`sum_cna`: summed copy number amplitudes, without focality filter

`pearson_corr`: spearman correlation coefficient

(Assuming that all of the above scores are present in the report.txt file, as is the case by default.) Either use the shell script:

**Example 2.4.1.** Standalone peak detection from the Mac/Linux command line

```
standalone_peakdetection.sh example_data/test_CSV/report.txt \
    annotation/gencode17_symbols.bed 0.7 fs_hp ./new_peaks.txt
```

Or the MATLAB function:

**Example 2.4.2.** Standalone peak detection from the MATLAB command line

```
standalone_peakdetection('example_data/test_CSV/report.txt',...
    'annotation/gencode17_symbols.bed',0.7,'fs_hp','./new_peaks.txt')
```

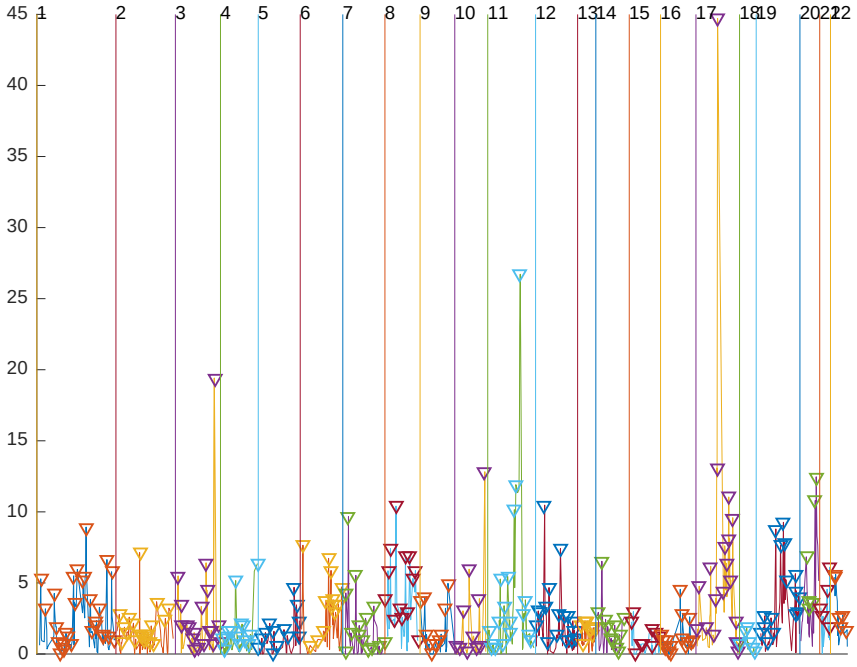
Or, for the compiled executable:

**Example 2.4.3.** Standalone peak detection from the Mac/Linux command line

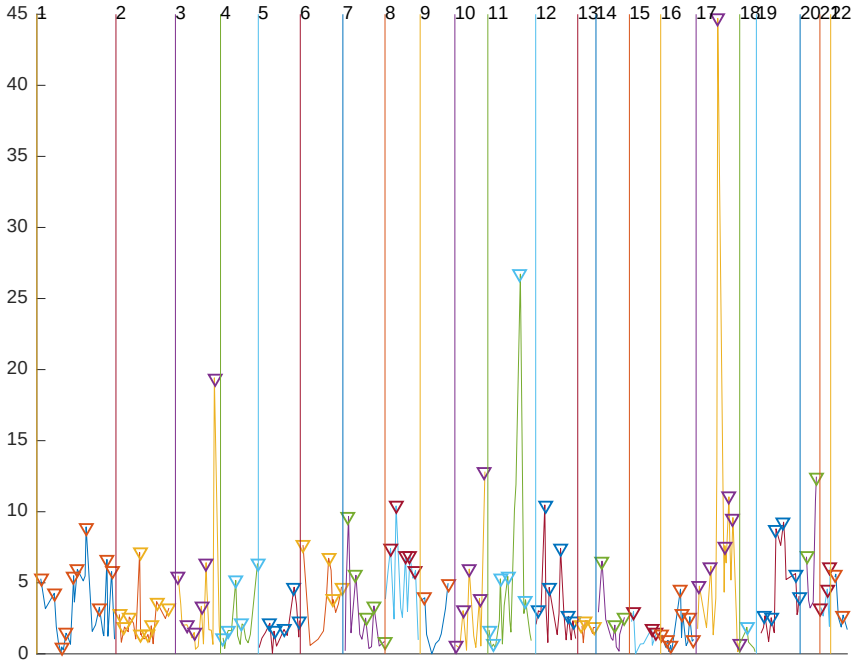
```
standalone_peakdetection_compiled.sh MCR_root \
    example_data/test_CSV/report.txt \
    annotation/gencode17_symbols.bed 0.7 fs_hp ./new_peaks.txt
```

Remember to specify the location of the MATLAB runtime environment (`/Applications/MATLAB/MATLAB_Runtime/v90` in the above example)

**Example 2.4.4.** Amplification peaks detected at level 0.6




**Example 2.4.5.** Amplification peaks detected at level 0.7




### 2.4.2 Annotate peaks

If tile-level analysis was performed and no genome annotation was provided via the option "***optional_gene_annot***" (in addition to the standard tile annotation file), it

is possible to add the IDs of genes overlapping each tile in the peak table afterwards. To do this, use the script ***annotate_peaks.sh*** (or the function `annotate_peaks` if running running the software from inside the MATLAB environment). The following are the parameters required to run this function:

`peak_file_path`: path to main report file created by running FocalScan (ie. peaks.txt).

`annot_file_path`: path to annotation file (ie. annotation/gencode17.bed)

`out_file`: name of output file (ie. peaks_annotated.txt)

Usage:

```
annotate_peaks.sh peaks_file_path annot_file_path out_file
```

**Example 2.4.6.** Usage of annotate_peaks.sh:

```
annotate_peaks.sh peaks.txt annotation/gencode17.bed ...
    ./peaks_annotated.txt
```

# 3. Preparing the input data

## 3.1 Gene-based analysis

This analysis mode requires that the expression level of each gene has been quantified beforehand. This can be done with, for instance, HTSeq. Unnormalized read counts are expected (although, pre-normalized values can also be provided if one sets the FocalScan ***normalization*** option to "none"). Gene expression data can be input either as a directory of separate read count files for each sample (each structured as in example 3.1.1) or as one single CSV-file. The gene IDs in these files need to correspond to those in the annotation file provided to FocalScan (for instance, the included "gencode17.bed" file). In the case of CSV-input, the genes must also be in the same order as those in the annotation. In addition, the CSV file should contain sample IDs as column titles and no row names (thus, gene IDs should not be included in this file).

Copy number data should be provided in SEG format (a description of this format can be found at *https://www.broadinstitute.org/igv/SEG*) (example 3.2). Data for all samples needs to be in the same file.

**Example 3.1.1.** Structure of gene read count file

```
1  ENSG00000000003.10  2315
2  ENSG00000000005.5   23
3  ENSG00000000419.8   1570
4  ENSG00000000457.8   588
5  ENSG00000000460.12  293
6  ENSG00000000938.8   487
7  ENSG00000000971.11  5193
8  ENSG00000001036.8   3289
9  ENSG00000001084.6   1482
10 ENSG00000001167.10  2498
```

[gene ID, read count]

## 3.2 Tile-based analysis

The tile-based approach enables studying coordinated expression and copy number amplitude changes both in annotated and non-annotated genomic regions. Tiles are short (1000bp) overlapping (by 500 bp) sequences ("tiles") of the genome. Using the coverageBed tool of the bedtools software package and the included tile definition file ("hg18_hg19_1kb_tiles.bed', made to suit both data aligned to and annotated

with the hg18 and hg19 versions of the human genome), tile-based read counts can be estimated.

To perform such a quantification, one may either choose to specify the parameters to coverageBed directly or use the included wrapper script (**quantify_tiles.sh**) as in example 3.2.1:

**Example 3.2.1.** Quantifying tiles for a sample with the included script

```
./quantify_tiles.sh sampleXYZ.bam annotation/hg18_hg19_1kb_tiles.bed
```

This will create an output file called `sampleXYZ.tile_counts` (structured as in example 3.2.2).

**Example 3.2.2.** Tile read count file

```
134217  7
134218  17
268435  0
268436  0
402653  0
402654  0
16777   2
16778   0
33554   14
33555   7
```

[tile ID, read count]

This needs to be done for each .bam file, preferably in parallell to speed up processing. A file with RNA-seq read counts (*.tile_counts) for each genomic tile will be generated.

NOTE:

- You may consider pre-filtering your .bam files to only consider uniquely mapped/high quality reads (e.g. quality 255 only for TopHat alignments, by running 'samtools view -b -q255 in.bam >out.bam').
- FocalScan does not take RNA-seq strand information into account. E.g. TCGA RNA-seq datasets are not strand specific, but this could be useful in other cases. Strand-specific analysis can be accomplished by first splitting .bam files into '+' and '-' fractions using samtools, and running FocalScan on each fraction.

After quantifying tile-level expression data for all samples, run FocalScan as shown in example , specifying the directory containing the tile expression files of all samples. When performing a tile-level analysis, copy number data should be provided in SEG format.

## 3.3 Creating a tile annotation file

To create a tile annotation file, such as the one included with this tool (downloadable from the link referenced in the README file), one might use the script "make_tile_annotation.sh":

1. Download a tile with chromosome information from UCSC (hg19 in this case):

```
wget ...
ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/...
chromInfo.txt.gz
```

2. Make the tile annotation (requires MATLAB):

```
make_tile_annotation.sh chromInfo.txt
```

3. Two output files are created: "1kb_tiles.bed" and "1kb_tiles_nochr.bed", where the first one has chromosome names that include "chr".

## 3.4 Example: Walkthrough for a tile-level analysis on TCGA data

This section outlines the steps required for a typical tile-level analysis of a TCGA cancer dataset. The additional scripts referenced here are available in the "TCGA_scripts" folder (at https://github.com/jowkar/focalscan).

1. Download RNA-seq data (BAM format) corresponding to primary tumors from Cancer Genomics Hub (https://cghub.ucsc.edu). Add the relevant samples to the cart and download the data using GeneTorrent. Also download the summary.tsv file from the cart page.

2. Download segmented copy number data (SEG format) for all samples corresponding to the downloaded expression data files (available at http://gdac.broadinstitute.org). Choose files with germline variants excluded (titled "minus_germline").

3. Quantify tile-level expression using the included script (for each sample, preferably done in a loop):

```
quantify_tiles.sh sample.bam annotation/hg18_hg19_1kb_tiles.bed
```

   Then move the resulting files to the same directory (from now on referred to as "read_count_files").

4. Create an index file from the cghub summary.tsv file. Either cut out the columns corresponding to the TCGA barcode and BAM filename, respectively, or use the script create_index.sh (which also removes the last characters from each barcode to facilitate matching with the copy number sample names):

```
create_index.sh summary.tsv index.txt
```

Note: It might be necessary to remove the last few elements from the TCGA barcode in order for the sample name to match those of the same patient in the SEG file.

5. Reformat the sample names in the SEG file to match those in the expression data (removes the last part of the barcodes of each sample, also removes non-primary tumors):

```
reformat_seg.sh cancer.seg
```

A file will be created titled "cancer.reformatted.seg".

6. Run FocalScan

```
focalscan.sh expr_path read_count_files
index_file index.txt
file_extension .tile_counts
seg_file cancer.reformatted.seg
annot_file annotation/hg18_hg19_1kb_tiles.bed
optional_gene_annot annotation/gencode17_symbols.bed
```

Note: The "optional_gene_annot" parameter needs to be given in order to for genes overlapping each tile to be written to the output peak report.

7. The resulting top-ranking tiles are listed in "peaks.txt". Intergenic peaks lack any name in the "Id" field.

8. **Optional:** If the "optional_gene_annot" parameter was given and the "peaks.txt" file lacks information about overlapping genes, use the "annotate_peaks.sh" script:

```
annotate_peaks.sh peaks.txt ...
    annotation/gencode17_symbols.bed peaks_annotated.txt
```

9. **Optional:** If a more or less sensitive detection of peak tiles/genes is desired, use the "standalone_peakdetection.sh" script:

```
standalone_peakdetection.sh report.txt ...
    annotation/hg18_hg19_1kb_tiles.bed 0.8 fs_hp peaks08.txt
```

Note: Overlapping gene IDs can be added to the resulting file as in the previous step.

# 4. Output files

The following output files are produced after running FocalScan:

`report.txt`: Comprehensive report with statistics for each gene/tile (example 4.0.1)

`peaks.txt`: Ranked list of most prominent peak genes/tiles (example 4.0.2)

`score_hp.wig`: The main score calculated for each gene/teils, with focality filter

`score.wig`: The main score calculated for each gene/teils, without focality filter

`sum_cna.wig`: Summed copy number amplitudes for each gene/tile

`sum_cna_hp.wig`: Summed copy number amplitudes for each gene/tile, with focality filter

`rna.wig`: Mean expression level for each gene/tile

`log.txt`: log file that lists parameter choices and other program output

The .wig-files can be used to visualize the results together with IGV [4] as in example 4.0.3.

**Example 4.0.1.** report.txt

```
1 gene_id fs  sum_cna fs_hp   sum_cna_hp  mean_expr   num_neutral ...
     num_amplified   num_deleted pearson_corr    pearson_p_val
2 ENSG00000000003.10  NaN NaN NaN NaN 0.445249050855637   0   0   ...
     0   NaN NaN
3 ENSG00000000005.5   NaN NaN NaN NaN 0.00780816376209259 0   0   ...
     0   NaN NaN
4 ENSG00000000419.8   NaN 11.6571989059448    2.14084219932556    ...
     1.48330008983612    0.298378676176071   43  4   1   ...
     0.132765218615532   0.368349492549896
5 ENSG00000000457.8   NaN 11.9454650878906    0.23913137614727    ...
     -0.291880905628204  0.147126391530037   46  0   1   ...
     -0.0801341384649277 0.591266751289368
6 ENSG00000000460.12  NaN 11.9287166595459    -0.151118218898773  ...
     -0.308629840612411  0.101743817329407   46  0   1   ...
     -0.170675307512283  0.250511586666107
```

This file is a full report of the statistics calculated for each gene or tile. `fs`: coordination score calculated without focality filter; `fs_hp`: standard score calculated with focality filter; `sum_cna`: summed copy number amplitudes across all samples; `sum_cna_hp`: summed focality filtered copy number amplitudes; `num_neutral`:

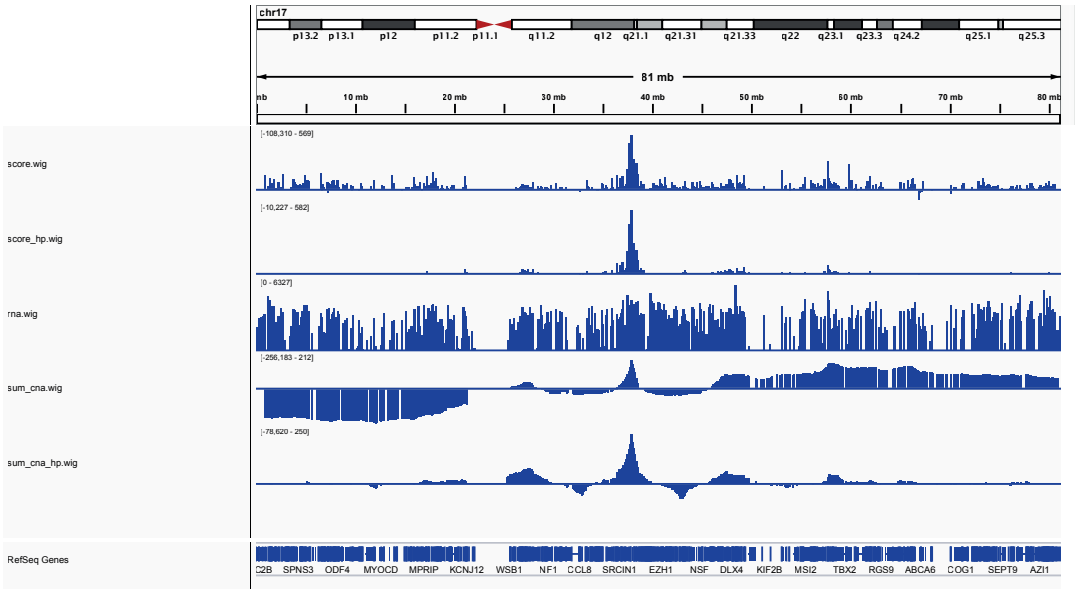
number of copy number neutral samples; `num_amplified`: number of copy number amplified samples; `num_deleted`: number of samples with deletions.

**Example 4.0.2.** peaks.txt

```
1  Id  Score   Sum_CNA_HP  Chr Start   Stop
2  ENSG00000141736.9   44.7113952636719    13.6427001953125    ...
       chr17   37844167    37886679
3  ENSG00000118369.8   26.7186393737793    9.87020111083984    ...
       chr11   77899858    77925757
4  ENSG00000136161.8   21.0623111724854    -10.0967998504639   ...
       chr13   49063095    49107369
5  ENSG00000121879.3   19.4183578491211    3.04330015182495    ...
       chr3    178865902   178957881
6  ENSG00000017373.11  13.0788612365723    4.09929990768433    ...
       chr17   36686251    36762183
7  ENSG00000066468.16  12.8228664398193    4.13399982452393    ...
       chr10   123237848   123357972
8  ENSG00000101132.5   12.465539932251 4.62510013580322    chr20   ...
       52824386    52844591
9  ENSG00000172927.3   11.9149017333984    8.785400390625  chr11   ...
       69061605    69182494
10 ENSG00000172893.11  11.8471593856812    -0.594799995422363  ...
       chr11   71139239    71163914
```

This file lists the top ranked genes or tiles. `Score` refers to the chosen metric on which peak detection is performed (the standard FocalScan score is used by default). To fine-tune the peak detection algorithm used for this ranking, see section 2.4.

**Example 4.0.3.** Visualization of WIG files using IGV



# 5. Causes of common errors and warnings

For the most part, the error messages output by FocalScan should be descriptive enough to decipher the cause and solutions to any errors that may arise (such as the wrong format of input files). This section details some errors that may require additional explanation:

## 5.1 Errors

- **Message:**

```
Requested 6203915x1044 (24.1GB) array exceeds maximum array ...
    size preference. Creation of arrays greater than this ...
    limit may take a long time and cause MATLAB to become ...
    unresponsive.
```

**Cause:** Tile-level analysis (or gene-level analysis with very many samples) was attempted on a system with too little RAM available.

**Solution:** Be aware that tile-level analysis demands a lot of memory (generally over 30 GB). Perform this analysis on a powerful server. (A workaround using temporary files is in development and should enable this type of analysis also on less powerful systems.)

- **Message:**

```
error while loading shared libraries: libmwmclmcrrt.so.8.4: ...
    cannot open shared object file: No such file or directory
```

**Cause:** The script for the complied exeutable (**`focalscan_compiled.sh`**) was used and either the MATLAB runtime is not installed or the wrong version is installed. This error may also arise if an older version of the compiled FocalScan application is installed, but a later runtime version is installed.

**Solution:** Make sure that the latest version of both the "Installer" program and the FocalScan application is downloaded and run the installer, making a note of where the runtime is installed.

- **Message:**

```
Usage:
/Applications/FocalScan/application/focalscan_compiled.sh ...
    <deployedMCRroot> args
```

**Cause:** The script for the complied exeutable (**`focalscan_compiled.sh`**) was used and the path to the MATLAB runtime was not specified.

**Solution:** When using the compiled executable, the path to the MATLAB runtime will have to be given as the first argument following the name of the program (and after that all the other program parameters, such as the path to the gene expression data, etc.

- **Message:**

```
Error using FocalScan/read_data (line 343)
No gene IDs were present in the expression data and the ...
    number of rows did not correspond to the number of genes ...
    in the annotation.
```

**Cause:** When CSV input is used for expression data, rows may or may not contain gene IDs. When gene IDs are not included in the first column, it is assumed that rows correspond exactly to rows (gene IDs) in the specified annotation file. This error indicates that the annotation and the read count CSV file do not match in this fashion. If using the example data, note that the file "`gencode17_symbols.bed`" contains fewer genes than "`gencode17.bed`". Thus, when running the program with the file "BRCA_expr.csv" as input, one must use the "`gencode17_symbols.bed`" file (and, conversely, the "`gencode17.bed`" file when using the separate read files in the directory `example_data/read_count_files`).

- **Message:**

```
The necessary files were not found in the current ...
    directory. Please manually add the directory containing ...
    the FocalScan files to PATH by typing: export ...
    PATH=$PATH:path_to/focalscan
```

**Cause:** To run the program, the main file needs to be able to automatically detect where the remaining scripts are located. This is done via the PATH environment variable, where the path to the directory containing the program files must be added (as shown in the message).

- **Message:**

```
error while loading shared libraries: libmwlaunchermain.so: ...
    cannot open shared object file: No such file or directory
```

**Cause:** This error occurs when the path to the MATLAB runtime was not provided as the first argument when running one of the compiled executables.

## 5.2 Warnings

- **Message:**

```
1 Warning: Tile-level analysis chosen, but no gene annotation ...
      has been specified. The peak report will not contain ...
      information about genes overlapping each tile.
```

**Cause:** Tile-level analysis calculates scores for 500 bp regions across the genome, however, most often it is also of interest which genes these tiles overlap. To get this information, one should also remember to include an additional (optional) genome annotation file.

**Solution:** Use the "***optional_gene_annot'***" parameter to specify a genome annotation file. This is the same file that one would use for a gene level analysis. For instance, the "`gencode17_symbols.bed`" file provided in the example data.

- **Message:**

```
1 Warning: Duplicate entries detected in annotation. Ignoring ...
      these.
```

**Cause:** This warning may be printed when running a gene-level analysis with an annotation that uses gene symbols rather than for instance Ensembl IDs, since not all gene symbols are unique. This is generally nothing to worry about.

# 6. References


[1] Aaron R. Quinlan and Ira M. Hall. BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26:841–842, 2010.

[2] S. Anders, P. T. Pyl, and W. Huber. HTSeq A Python framework to work with high-throughput sequencing data. Technical report, 2014.

[3] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25:2078–2079, 2009.

[4] Helga Thorvaldsdóttir, James T. Robinson, and Jill P. Mesirov. Integrative Genomics Viewer (IGV): High-performance genomics data visualization and exploration. *Briefings in Bioinformatics*, 14:178–192, 2013.