FocalScan 1.0 manual

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 can be run either with or without (using the standalone executable on Mac or Linux) an installation of MATLAB (R2015b) [1]. In addition, the following software is likely to be useful for pre-processing the data.

- bedtools (2.21.0) [2]
- HTseq (0.6.1) [3]

- samtools (1.1) [4]

Numbers in parenthesis indicate tested versions.

Additionally, Integrative Genomics Viewer (IGV) [5] is recommended for visualizing some of the results.

Also, note that the non-gene centric ("tile-level") analysis will likely require a large amount (>30 GB) 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. In the latter case, it is also possible to run the program without a MATLAB installation, using the compiled executable.

First download either the scripts or the compiled executable from *github.com/jowkar/focalscan* and place the files in any suitable directory.

1.2.1 With MATLAB installed

After downloading the files, either execute them from that same directory or set the path to where the scripts were saved (on Mac/Linux):

```
export PATH=$PATH:path_to_dir_containing/FocalScan/
```

Otherwise, from within MATLAB, just add the path to the program with `addpath(genpath('path_to_FocalScan'))`. To run the program, refer to the examples in section 2.1.

Standard MATLAB use of this software is possible on both Windows, Linux and Mac, whereas the standalone executable and the shell scripts require Linux or Mac.

1.2.2 Without MATLAB installed

If usage via the standalone executable is intended, run the included file titled **`Installer`** to install the MATLAB runtime environment and the executable. Make a note of the directory in which this is installed (needs to be specified when running the program, see example 2.1.3).

2. Input options

2.1 Running FocalScan

FocalScan can be run either from within the MATLAB environment (example 2.1.2) or directly from the Mac/Linux command line (example 2.1.1). In the latter case, it is also possible to run the program without a MATLAB installation by using the compiled executable (example 2.1.3). Input parameters are specified as pairs of parameter name and parameter value.

Example 2.1.1. Running FocalScan from the Mac/Linux command line

```
./focalscan.sh 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 \
    reportdir example_data/test_gene
```

`/Applications/MATLAB/MATLAB_Runtime/v90` refers to the directory where the MATLAB runtime is installed (using the included installer or otherwise). If MATLAB is installed (and not just the runtime environment, it is also possible to give the path to the MATLAB installation directory, for instance `/Applications/MATLAB_R2015b.app/`.

Example 2.1.2. Running FocalScan from within MATLAB

```
FocalScan('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',...
    'reportdir','example_data/test_gene');
```

Example 2.1.3. Running FocalScan using the compiled executable

```
./run_FocalScan.sh /Applications/MATLAB/MATLAB_Runtime/v90 \
    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 \
    reportdir example_data/test_gene
```

`/Applications/MATLAB/MATLAB_Runtime/v90` refers to the directory where the MATLAB runtime is installed (using the included installer or otherwise). If MATLAB is installed (and not just the runtime environment, it is also possible to give the path to the MATLAB installation directory, for instance `/Applications/MATLAB_R2015b.app/`.

2.2 List of parameters

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

Note that there are two ways to provided expression data (either as a directory of read count files or a single CSV file). See section 2.3 for more details. Segmented copy number data should be given as a single SEG file with data for all samples.

The parameters listed under "Additional options" are non-mandatory. It is, however, strongly recommended to provide a genome annotation file also when performing a tile level analysis, so that genes overlapping peak tiles can be written to the output report (use the ***optional_gene_annot*** option for this).

Basic options

`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.

Alternative input options for expression data

`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.

`expr_ratio_csv`: Path to a CSV file containing log2 ratios of normalized expression relative to diploid reference samples. (Only for gene-level analysis)

Additional options

`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]

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

`normalization`: The normalization mode to employ [percentile] {percentile, 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]

`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.

`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}

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

`scorefield`: The metric to use as basis for peak detection [fs_hp] {fs, fs_hp, sum_cna, sum_cna_hp, spearman_corr}

2.3 Data formats

2.3.1 Gene-level analysis

Expression data should be provided in either of the two ways listed below (examples 2.3.1 and 2.3.3). If read counts are provided in separate files for each sample, then place them all in the same directory and specify this directory together with their file extension and an index file.

Example 2.3.1. Expression data as separate read count files

```
FocalScan('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',...
    '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

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

[file name, sample ID]

Expression data may also be provided as a single CSV file:

Example 2.3.3. Expression data as a single CSV file

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

Copy number data should be provided as a SEG file (paired with either of the above expression data input methods):

Example 2.3.4. SEG file

```
Sample  Chromosome  Start   End Num_Probes  Segment_Mean
TCGA-A1-A0SB    1   3218610 247813706   129072  0.0034
TCGA-A1-A0SB    2   484222  174313755   92446   0.0014
TCGA-A1-A0SB    2   174314142   174314161   2   -1.268
TCGA-A1-A0SB    2   174315778   194887369   10897   0.0019
TCGA-A1-A0SB    2   194888052   194892814   4   -0.9361
TCGA-A1-A0SB    2   194898700   242476062   27869   0.0013
TCGA-A1-A0SB    3   2212571 197538677   106304  0.0019
TCGA-A1-A0SB    4   1053934 188763651   102677  0.0014
TCGA-A1-A0SB    5   914233  31655645    18310   -0.0041
```

In all cases, it is mandatory to also provide a genome annotation in four column BED format (example 2.3.5). The annotation needs to have identical gene IDs as the one that was used to quantify gene/tile read counts.

Example 2.3.5. Genome annotation file

```
chrX    99883667    99894988    ENSG00000000003.10
chrX    99839799    99854882    ENSG00000000005.5
chr20   49551404    49575092    ENSG00000000419.8
chr1    169821804   169863408   ENSG00000000457.8
chr1    169631245   169823221   ENSG00000000460.12
chr1    27938575    27961788    ENSG00000000938.8
chr1    196621008   196716634   ENSG00000000971.11
chr6    143816614   143832827   ENSG00000001036.8
chr6    53362139    53481768    ENSG00000001084.6
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 2.3.6. Expression data input

```
FocalScan('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 2.3.7. 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 4.0.4.

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

`spearman_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')
```

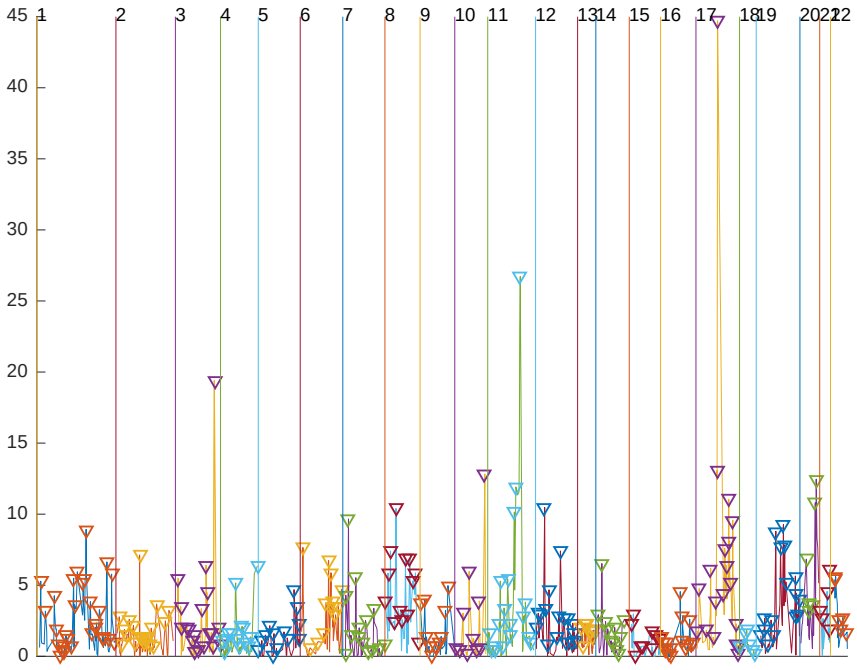
Or, for the compiled executable:

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

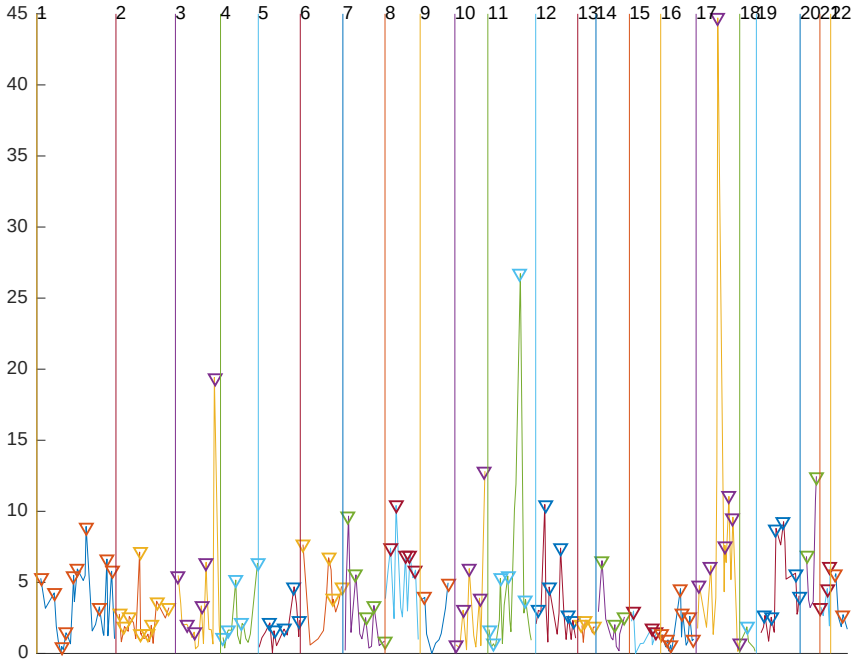
```
./run_standalone_peakdetection.sh ...
    /Applications/MATLAB/MATLAB_Runtime/v90 \
     example_data/test_CSV/report.txt \
     annotation/gencode17_symbols.bed 0.7 fs_hp ./new_peaks.txt
```

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


45
40
35
30
25
20
15
10
5
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

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:

- The .bam files may use chromosome names formatted as e.g. 'chr1' or simply '1'. In the latter case, instead use the hg18_hg19_1kb_tiles_nochr.bed file.
- 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.

4. Output files

The following output files are produced after running FocalScan:

`report.txt`: Comprehensive report with statistics for each gene/tile (example 4.0.3)

`peaks.txt`: Ranked list of most prominent peak genes/tiles (example 4.0.4)

`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 [5] as in example 4.0.5.

Example 4.0.3. report.txt

```
1 gene_id fs  sum_cna fs_hp   sum_cna_hp  mean_expr   num_neutral ...
     num_amplified   num_deleted spearman_corr   spearman_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`:

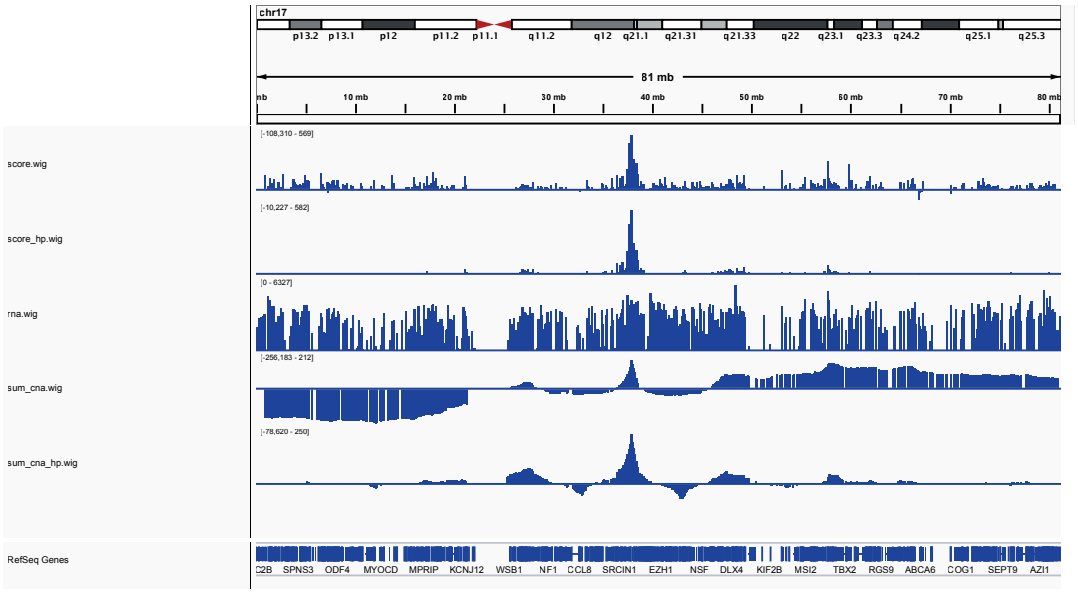
number of copy number neutral samples; `num_amplified`: number of copy number amplified samples; `num_deleted`: number of samples with deletions.

Example 4.0.4. 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 4.0.5. Visualization of WIG files using IGV



5. References

[1] The MathWorks Inc. MATLAB (R2015b). *The MathWorks Inc.*, 2015.

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

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

[4] 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.

[5] 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.