

Documentation for

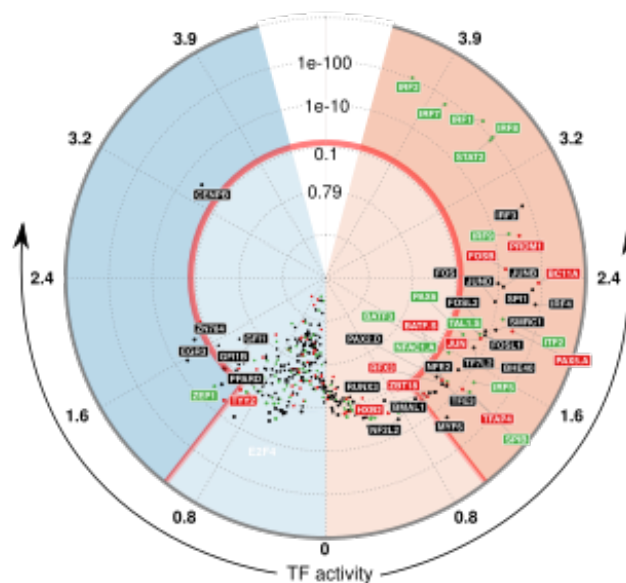
Genome-wide quantification of differential transcription factor activity: *diffTF*

Ivan Berest¹, Christian Arnold¹, Armando Reyes-Palomares¹, Kasper Rassmussen² &
Judith B. Zaugg¹

¹ EMBL-European Molecular Biology Laboratory, Heidelberg, Germany.

² Biotech Research and Innovation Centre (BRIC), University of Copenhagen

Correspondence should be addressed to J.B.Z. (judith.zaugg@embl.de)



Last update: 11/21/17

This document provides documentation and additional information for the *diffTF* pipeline.

If you have questions or comments, feel free to contact us. We will be happy to answer any questions related to this project as well as questions related to the software implementation. For method-related questions, contact Judith B. Zaugg (judith.zaugg@embl.de) or Ivan Berest (berest@embl.de). For technical questions, contact Christian Arnold (christian.arnold@embl.de).

If you use this software, please cite the following reference:

Ivan Berest*, Christian Arnold*, Armando Reyes-Palomares, Kasper Rassmussen & Judith B. Zaugg. Genome-wide quantification of differential transcription factor activity: diffTF. 2017. submitted.

Table of Contents

- 1. Quick Start.....4
- 2. Prerequisites.....5
 - 2.1. Snakemake.....5
 - 2.2. R and R packages.....5
 - 2.3. samtools and bedtools.....5
- 3. Running your own analysis.....6
- 4. Pipeline Details.....7
 - 4.1. General configuration file for the analysis (config.json).....7
 - 4.2. Input metadata.....11
 - 4.3. Output Folders and Files.....12
 - 4.4. Working with the pipeline.....16
 - 4.5. Handling errors.....16

1. Quick Start

The following quick start briefly summarizes the necessary steps to use our pipelines.

1. Install the necessary tools (Snakemake, samtools, and bedtools; see Section 2). We recommend installing them via conda, in which case the installation is as easy as

```
$ conda install -c bioconda snakemake bedtools samtools
```

If conda is not yet installed, follow the instructions in <https://conda.io/docs/user-guide/install/index.html>. If you want to install the tools manually and outside of the conda framework, see Section 2.

2. Clone the Git repository:

```
$ git clone https://git.embl.de/grp-zaugg/diffTF
```

3. To run the example analysis, simply perform the following steps::

- a) Change into the example/input directory within the Git repository

```
$ cd diffTF/example/input
```

- b) Download the data via the download script

```
$ sh downloadAllData.sh
```

- c) To test if the setup is correct, start a dryrun via the helper script *startAnalysisDryRun.sh*

```
$ sh startAnalysisDryRun.sh
```

- d) Once the dryrun is successful, start the analysis via the helper script *startAnalysis.sh*

```
$ sh startAnalysis.sh
```

4. To run your own analysis, modify the files *config.json* and *sampleData.tsv*. See the instructions in Section 3 for more details.

If you run into troubles or questions, make sure to carefully read this document and Section 4 in particular.

2. Prerequisites

2.1. Snakemake

Please ensure that you have at least version 4.3 installed. Principally, there are multiple ways to install Snakemake, see <https://snakemake.readthedocs.io>. As outlined in Section 1, we recommend installing Snakemake via conda

2.2. R and R packages

A working R installation is needed and a number of packages from either CRAN or Bioconductor have to be installed. Type the following in R to install them:

```
$ install.packages(c("checkmate", "futile.logger", "tidyverse",  
"reshape2", "gridExtra", "scales", "jsonlite", "RcolorBrewer", "rlist",  
"ggrepel", "lsr", "modeest", "locfdr", "boot"))  
  
$ source("https://bioconductor.org/biocLite.R")  
  
$ biocLite(c("limma", "vsn", "csaw", "DESeq2", "DiffBind",  
"geneplotter"))
```

2.3. samtools and bedtools

In addition, samtools (<http://www.htslib.org/download/>) and bedtools (<http://bedtools.readthedocs.io>) are needed to run *diffTF*. As outlined in Section 1, we recommend installing them via conda.

3. Running your own analysis

Running your own analysis is almost as easy as running the example analysis. Carefully read and follow the following steps and notes:

1. Copy the files *config.json* and *startAnalysis.sh* to a directory of your choice.
2. Modify the file *config.json* accordingly. See Section 4 for details about the parameters. We suggest to convert all relative paths to absolute ones. Do not delete or rename any parameters or sections.
3. Create a tab-separated file that defines the input data, in analogy to the file *sampleData.tsv* from the example analysis, and refer to that in the *config.json* (parameter *summaryFile*)
4. Adapt the file *startAnalysis.sh* if necessary (Snakemake parameters)

Important notes:

- Since running the pipeline might be computationally demanding, make sure you have enough space left on your device. As a guideline, analysis with 8 samples need around 12 GB of disk space, while a large analysis with 84 samples needs around 45 GB. Also, adjust the number of available cores accordingly. The pipeline can be invoked in a highly parallelized manner, so the more cores are available, the better!
- The pipeline is written in Snakemake, so for a deeper understanding and troubleshooting errors, some knowledge of Snakemake is invaluable. The same holds true for running the pipeline in a cluster setting. We recommend using a proper cluster configuration file in addition. For guidance and user convenience, we provide different cluster configuration files for a small (up to 10-15 samples) and large (>15 samples) analysis. See the folder *src/clusterConfigurationTemplates* for examples. Note that the sample number guidelines above are very rough estimates only. See the Snakemake documentation for details for how to use cluster configuration files.

4. Pipeline Details

4.1. General configuration file for the analysis (config.json)

A configuration file that defines various parameters of the pipeline is required. In the following, we explain the parameters in detail. Note that neither section nor parameter names must be changed.

Section “par_general”

outdir

String. Default “output”. Root output directory.

The root output directory where all output is stored. We recommend specifying an absolute path. There should be NO trailing slash!

regionExtension

Integer > 0. Default 100. Target region extension in base pairs.

This parameter specifies the number of base pairs each target region (from the peaks file) should be extended in both 5’ and 3’ direction.

comparisonType

String. Default ‘’.

This parameter helps to organize complex analysis for which multiple different types of comparisons should be done. Set it to a short but descriptive name that summarizes the type of comparison you are making or the types of cells you compare. The value of this parameter appears as prefix in most output files created by the pipeline. It may also be empty.

designContrast

String. Default “~ Treatment + Condition”. Design formula for the differential accessibility analysis in DESeq2.

The variable corresponding to the two conditions comes typically last (here: Condition).

designVariableTypes

String. Default "Treatment:factor, Condition:factor".

The data types of all elements listed in *designContrast*. Names must be separated by commas, spaces are allowed and will be eliminated automatically. The data type must be specified with a “:”, followed by either “numeric”, “integer”, “logical”, or “factor”.

Importantly, if the variable of interest is continuous-valued (i.e., marked as being integer or numeric), then the reported log2 fold change is per unit of change of that variable. That is, in the final circular plot, TFs displayed in the left side have a negative slope per unit of change of that variable, while TFs at the right side have a positive one.

nPermutations

Integer ≥ 0 . Default 0. The number of random sample permutations.

If set to a value > 0 , in addition to the real and non-permuted data, the sample conditions as specified in the sample table and in the parameter *conditionComparison* will be randomly permuted *nPermutations* times. Specifically, for the condition with fewer samples, 50% will be randomly chosen and switched with the same number of samples from the other condition. This procedure maximizes randomized conditions. Note that the running time of the pipeline will be increased when using permutations because parts of the pipeline are run for each permutation.

TFs

String. Default "all". Either “all” or a comma-separated list of TF names of TFs to include.

If set to “all”, all TFs that are found in the directory as specified in *dir_PWMScan* will be used. If the analysis should be restricted to a subset of TFs, list the names of the TF to include in a comma-separated manner here. Note that for each TF {name}, a file “{name}_pwmScan.bed” needs to be present in the directory *dir_PWMScan*.

We strongly recommend running diffTF with as many TF as possible due to our statistical model that we use that compares against a background model.

dir_scripts

String. Default “”. The path to the directory where the R scripts for running the pipeline are stored. We recommend specifying an absolute path.

RNASeqIntegration

Logical. true or false. Default false. Should RNA-Seq data be integrated into the pipeline?

If set to true, RNA-Seq counts as specified in the parameter *RNASeqCounts* will be used to classify each TF into either “activator”, “repressor”, “unknown”, or “not-expressed” for the final circular visualization and the summary table. For methodological details, see the Methods details of the publication. Note that RNA-Seq integration is only included in the very last step of the pipeline, so it can also be easily integrated later.

Section "samples"

summaryFile

String. Default ‘samples.tsv’. Path to the sample metadata file.

Path to a tab-separated file that summarizes the input data. See the next section and the example file for how this file should look like. We recommend specifying an absolute path.

Section "peaks"

consensusPeaks

String. Default “ (empty). Path to the consensus peak file. If set to the empty string “”, the pipeline will generate a consensus peaks out of the peak files from each individual sample. We recommend specifying an absolute path.

If no file is provided, the pipeline computes consensus peaks based on all individual peak files. For this, you need to provide the following two things:

- a peak file for each sample in the metadata file in the column “peaks”, see the next section for details.
- The format of the peak files, as specified in the parameter *peakType* (see below)

If a file is provided, it has to match exactly(!) the following specifications:

- tab-separated columns
- no column names in the first row
- five columns in total:
 1. Chromosome
 2. Start position
 3. End position

4. Identifier
5. Line number (starting from 1)

peakType

String. Default 'narrow'. Format of the peaks. Only relevant if no consensus peak file has been provided.

Only needed if no consensus peak set has been provided. All individual peak files must be in the same format. See the help for DiffBind *dba* for a full list of supported formats, the most common include:

- "raw": text file file; peak score is in fourth column
- "bed": .bed file; peak score is in fifth column
- "narrow": default peak.format: narrowPeaks file (from MACS2)

minOverlap

Integer ≥ 0 or Float between 0 and 1. Default 2. Minimum overlap for peak files for a peak to be considered into the consensus peak set. Corresponds to the *minOverlap* argument in the *dba* function of *DiffBind*. Only needed when *consensusPeaks* is not provided.

Only include peaks in at least this many peak sets in the main binding matrix. Only needed when *consensusPeaks* is not provided. If *minOverlap* is between zero and one, peak will be included from at least this proportion of peaksets. For more information, see the *minOverlap* argument in the *dba* function of *DiffBind*.

Section "additionalInputFiles"

refGenome fasta

String. Default 'hg19.fasta'. Path to the reference genome FASTA file.

The path to the reference genome file in FASTA format. We recommend specifying an absolute path. Note that this file has to be in concordance with the input data; that is, the exact same genome assembly version must be used.

dir PWMScan

The path to the directory where the TF-specific PWM results are stored, in BED format. We recommend specifying an absolute path. Each TF has to have one bed file, in the format $\{TF\}.bed$. For user convenience, we provide these files as described in the publication as a separate download:

- For a pre-compiled list of 620 human TF based on the HOCOMOCO 10 database, download this file: <https://www.embl.de/download/zaugg/diffTF/PWMScan/PWMscan.human.tar.gz>
- For a pre-compiled list of 423 mouse TF based on the HOCOMOCO 10 database, download this file: <https://www.embl.de/download/zaugg/diffTF/PWMScan/PWMscan.mouse.tar.gz>

However, you may also manually create these files for additional TF of your choice. For this, see the HOCOMOCO manual.

RNASeqCounts

String. Default “”. Path to the file with RNA-Seq counts.

If no RNA-Seq data is included, set to the empty string “”. Otherwise, if the parameter *RNASeqIntegration* is set to true, we recommend specifying an absolute path for a tab-separated file with normalized RNA-Seq counts. The first line must be used for labeling the samples, with column names being identical to the sample names as specific in the sample summary table (parameter *summaryFile*). The first column must be named ENSEMBL and it must contain ENSEMBL IDs (e.g., ENSG00000028277) without dots. The IDs are then matched to the IDs from the file as specified in the parameter *HOCOMOCO_mapping*.

HOCOMOCO_mapping

String. Default “HOCOMOCO/translationTable_human.csv”. Absolute path to the file with the HOCOMOCO mapping.

If RNA-Seq integration shall be used, a translation table to associate TFs and ENSEMBL genes is needed. For convenience, we provide such a translation table for human and mouse. If you want to use your own version, check the example translation tables and construct one with an identical structure.

4.2. Input metadata

This file summarizes the data and corresponding available metadata that should be used for the analysis. The format is flexible and may contain additional columns that are ignored by the pipeline, so it can be used to capture all available information in a single place. Importantly, the file must be saved as tab-separated, the exact name does not matter as long as it is correctly specified in the

configuration file. It must contain at least contain the following columns (the exact names do matter):

- “sampleID”: The ID of the sample
- “bamReads”: path to the BAM file corresponding to the sample. Note that the BAM files must be valid BAM files with chromosome names with a “chr” as prefix. The pipeline may crash if the “chr” part is missing.
- “peaks”: absolute path to the sample-specific peak file, in the format as given by the parameter “peakType”. Only needed if no consensus peak file is provided
- conditionSummary: String with an arbitrary condition name that defines which condition the sample belongs to. There must be only exactly two different conditions across all samples (e.g., mutated and unmutated, day0 and day10, ...)
- If applicable, all additional variables from the design formula except *conditionSummary* must also be present as a separate column.

4.3. Output Folders and Files

The pipeline produces a large number of output files. In the following, the directory structure and the files are briefly outlined. As some directory or file names depend on specific parameters in the configuration file, curly brackets will be used to denote that the filename depends on a particular parameter or name. For example, {comparisonType} and {regionExtension} refer to the parameters *comparisonType* and *regionExtension* as specified in the configuration file.

Most files have one of the following file formats:

- .bed.gz (gzipped bed file)
- .tsv (tab-separated value, text file with tab as column separators)
- .rds (binary R format, read into with the function readRDS)
- .pdf (PDF format)
- .log (text format)

FOLDER FINAL_OUTPUT

- extension{regionExtension}
 - {comparisonType}.allMotifs.tsv.gz
 - {comparisonType}.TF_vs_peak_distribution.tsv
 - {comparisonType}.summary.tsv
 - {comparisonType}.summary.volcano.pdf
 - {comparisonType}.summary.circular.pdf
 - {comparisonType}.diagnosticPlots.pdf

FOLDER PEAKS

Stores peak-associated files.

- if no consensus peak file was provided (parameter “*consensusPeaks*”):
 {comparisonType}.consensusPeaks.bed and consensusPeaks_lengthDistribution.pdf:
 generated consensus peaks, before filtering (see below).
- {comparisonType}.consensusPeaks.filtered.sorted.bed: Filtered consensus peaks (removal of peaks from one of the following chromosomes: chrX, chrY, chrM, chrUn*, *random*, *hap|_g|*
 hap|_g|
 hap|_g|
- for each input BAM file {basenameBAM}:
 {basenameBAM}.overlapPeaks.bed.gz
- {comparisonType}.sampleMetadata.rds
- {comparisonType}.peaks.rds
- {comparisonType}.peaks.tsv
- {comparisonType}.normFacs.rds
- {comparisonType}.diagnosticPlots.peaks.pdf

- for each permutation {perm}:
{comparisonType}.diagnosticPlots.peaks_permutation{perm}.pdf
- {comparisonType}.DESeq.object.rds

FOLDER Logs_and_Benchmarks

Stores various log and error files.

- *.log files from R scripts: Each logfile is produced by the corresponding R script
 - 1.produceConsensusPeaks.R.log
 - 2.DESeqPeaks.R.log
 - 3.analyzeTF.{TF}.R.log for each TF {TF}
 - 4.summary1.R.log
 - 5.prepareBinning.log
 - 6.binningTF.{TF}.log for each TF {TF}
 - 7.summaryFinal.R.log
- *.log summary files: Summary logs for user convenience, produced at very end of the pipeline only. They should contain all errors and warnings from the pipeline run.
 - all.errors.log
 - all.warnings.log

FOLDER TEMP

Stores temporary and intermediate files.

- extension{regionExtension}
 - for each input BAM file {basenameBAM} and TF {TF}:
{TF}__ {basenameBAM}.resorted.gz

- conditionComparison.rds: An R rds file that stores the condition comparison as a string. Subsequent steps will read this file.
- for each permutation {perm}:
{comparisonType}.motifs.coord.permutation{perm}.bed.gz
- for each permutation {perm}:
{comparisonType}.motifs.coord.nucContent.permutation{perm}.bed.gz
- {comparisonType}.allTFData_processedForPermutations.rds and
{comparisonType}.allTFUniqueData_processedForPermutations.rds. Produced in 5.prepareBinning.R, and needed subsequently for each 6.binningTF.R step.
- consensusPeaks.filtered.bed: filtered consensus peaks file before sorting.
- for each TF {TF}:
{TF}_pwmScan.sorted.bed.gz: Coordinate-sorted version of the input PWMs.
- for each input BAM file {basenameBAM}:
{basenameBAM}.idxstats, {basenameBAM}.chrOrder1, {basenameBAM}.chrOrder2,
{basenameBAM}.resorted.gz: Various temporary input BAM-associated files for the chromosome names and order as well as BAM-congruent resorted peak files.

FOLDER TF-SPECIFIC

Stores TF-specific files.

for each TF {TF}:

- {TF}.peaks.bed.gz
- extension{regionExtension}
 - {TF}.peaks.corr.bed.gz
 - {TF}.{comparisonType}.overlapBAMs.bed.gz
 - {TF}.{comparisonType}.output.tsv

- {TF}.{comparisonType}.summary.rds
- {TF}.{comparisonType}.diagnosticPlots.pdf
- {TF}.{comparisonType}.summaryPlots.pdf
- for each permutation {perm}:
 - {TF}.{comparisonType}.diagnosticPlots_permutation{perm}.pdf
- for each permutation {perm}:
 - {TF}.{comparisonType}.summaryPlots_permutation{perm}.pdf
- {TF}.{comparisonType}.DESeq.object.rds
- {TF}.{comparisonType}.permutationResults.rds
- {TF}.{comparisonType}.permutationSummary.tsv
- {TF}.{comparisonType}.covarianceResults.rds

4.4. Working with the pipeline

4.5. Handling errors

Errors occur during the Snakemake run can principally be divided into:

- Temporary errors (often when running in a cluster setting)
 - might occur due to temporary problems such as bad nodes, file system issues or latencies
 - rerunning usually fixes the problem already. Consider using the option `–restart-times`.
- Permanent errors
 - indicates a real error related to the specific command that is executed
 - rerunning does not fix the problem as they are systematic (such as a missing tool)

To troubleshoot errors, you have to first locate the exact error. Depending on how you run Snakemake (i.e., in a cluster setting or not), check the following places:

- in locale mode: the Snakemake output on the console

- in cluster mode: either error, output or log file of the corresponding rule that threw the error

After locating the error, fix it accordingly. For example, if you receive a memory-related error, try to increase the available memory. In a cluster setting, adjust the cluster configuration file accordingly by either increasing the default memory or (preferably) or by overriding the default values for the specific rule.