# DBUtilities.py

#### Python script named "DBUtilities.py" contains several functions for working with a PostgreSQL database. Here is a pseudocode for each function in the script

start\_psql\_server(db\_user\_name, db\_host\_name, db="../database", logfile="../logfile")

* Try to connect to the PostgreSQL server
* If the server is not running, check the platform and try to start the server using the appropriate command
* If the server still cannot be started, raise an exception
* Return True if the server is started successfully

create\_db(db\_name, db\_user\_name, db\_host\_name)

* Try to connect to the database with the given name
* If the database already exists, print a message and return True
* If the database does not exist, create it and print a message, then return True
* If there is an error, print an error message and return False

open\_connection(db\_name, db\_user\_name='user', db\_host\_name='localhost')

* Connect to the database with the given name, user, and host
* Return the connection object
* close\_connection(conn)
* Close the given database connection

get\_col\_names\_from\_table(table\_name, conn)

* Get the column names from the given table using the given database connection
* Return a list of column names

create\_index(cursor, cell\_table, index\_name='indexposrange', index\_method='gist', index\_cols='posrange')

* Drop the index with the given name if it exists
* Create a new index on the given table using the given method and columns

table\_contains\_data(cursor, table\_name)

* Try to select a row from the given table using the given database cursor
* If a row is returned, print a message and return True
* If there is an error or no row is returned, print a message and return False

create\_table\_stmt\_parallel(cursor, tissue, tissuecols, tissuemotifsimputed)

* Create a new table with the given name using the given columns from the given source table using the given database cursor

get\_number\_of\_motifs(table: str, cursor)

* Get the number of motifs in the given table using the given database cursor
* Return the number of motifs

add\_column\_to\_tissue\_table(table: str, cursor, col\_name: str, col\_type: str)

* Add a new column with the given name and type to the given table using the given database cursor

update\_db\_value(value, mid, column, table, cursor)

* Update the value of the given column for the row with the given mid in the given table to the given value using the given database cursor

# GetFunMotifperTissue.py

#### "GetFunMotifperTissue.py" contains a Python script with functions for determining functional motifs per tissue based on annotated motifs and regression weights. The script also includes functions for computing functionality scores, determining significance cutoffs, and interacting with a PostgreSQL database.

#### The script includes the following key functions:

* Determination of Functional Motifs per Tissue: The main function
  + get\_functional\_motifs\_per\_tissue iterates over tissues, retrieves motif data, and determines functional motifs based on various conditions such as the presence of DHSs, TF expression, and TF binding evidence.
* Functionality Score Computation: The function compute\_functionality\_score computes the functionality score of a motif based on regression weights and specific variables.
* Significance Cutoff Determination: The function get\_significance\_cutoff determines the cutoff value for a significant functionality score based on the distribution of scores from functional and non-functional motifs.

## File "GetFunMotifperTissue.py" contains a Python script with functions for determining functional motifs per tissue based on annotated motifs and regression weights. The script also includes functions for computing functionality scores, determining significance cutoffs, and interacting with a PostgreSQL database. Here is a pseudocode for each function in the script:

#### DHS\_present(motif\_info) -> bool:

* Check if DHSs is present for a motif
* Return a boolean value indicating whether DHSs is present

#### TFs\_expressed(motif\_info) -> bool:

* Check if the TF belonging to motifs is expressed
* Return a boolean value indicating whether the TF is expressed

#### TF\_ChIP\_seq\_data\_available(motif\_info) -> bool:

* Check if TF ChIP-seq data is available for a motif
* Return a boolean value indicating whether TF ChIP-seq data is available

#### TF\_binding\_evidence(motif\_info) -> bool:

* Check if there is evidence for binding of a matching TF to a motif
* Return a boolean value indicating whether there is evidence for binding

#### compute\_functionality\_score(motif: object, params, weighted\_variable: list, tissue: str, cursor) -> float:

* Compute the functionality score of a motif based on regression weights and specific variables
* Update the database using the cursor
* Return the functionality score

#### get\_significance\_cutoff(funMotifs: list, nonFunMotifs: list, params, weighted\_variable: list, tissue, cursor) -> float:

* Determine the cutoff value for a significant functionality score based on the distribution of scores from functional and non-functional motifs
* Get functional motifs and non-functional motifs in data frames
* Compute the functionality scores for the functional motifs and non-functional motifs
* Compute the mean and standard deviation of the scores
* Compute the significance cutoff (intersection of distributions of scores of functional vs. non-functional motifs)
* Return the significance cutoff

#### make\_string\_from\_list(lst)

* Create a string out of a list of names, to use it in a SQL statement
* Return the string

#### get\_motif\_data\_for\_tissue(tissue, columns, cursor, mid=None, df: bool = False) -> object:

* Given a tissue, return the data of a database for this tissue as a data frame or a list
* Create an SQL command
* Get data from the database
* Return the data

#### get\_functional\_motifs(params, tissue, weighted\_variables: list, cursor) -> list:

* Determine functional motifs per tissue based on annotated motifs and regression weights
* Add two columns to tissue tables to save functionality score and whether the motif is functional
* Get the number of motifs
* Loop over motifs
* Check if DHSs is present and TF is expressed
* Check if TF ChIP-seq data is available and there is evidence for binding of a matching TF to a motif
* Compute the functionality score of a motif
* Append the motif to the list of functional motifs or non-functional motifs
* Determine the significance cutoff
* Return the list of functional motifs

#### get\_functional\_motifs\_per\_tissue(params, cursor, tissues: list, weighted\_variables=None) -> dict:

* Return functional motifs per tissue based on annotated motifs and regression weights
* Add columns to the tissue tables to save functionality score and whether the motif is functional
* Loop over the tissues
* Get functional motifs for each tissue
* Return a dictionary of functional motifs per tissue

# funMotifsMain.py

## This Python script is part of a pipeline for analyzing functional motifs in the human genome. It consists of six sections, each performing a specific task, such as collecting resources, scoring motifs, and overlapping motifs with non-coding mutations. Below is a pseudocode representation of the main sections of the script:

## Section 1

#### parse\_args():

# Parse command line arguments

# Return the parsed arguments

## Section 2

#### save\_parameters\_to\_file(params, file\_path):

* Convert parameters to a dictionary if they are not already in one
* Write the dictionary to a file in JSON format

#### extract\_variables\_from\_file(file\_path):

* Extracts variables and their values from a file
* returns them as a dictionary

## Section 3:

### Establish database connection

* Use the parsed arguments to get parameters
* Set the temp directory for bedtools operations
* If force\_overwrite, delete results directory to compute section 1 and 2 again

## Section 4:

### Section 4.1:

## Collect Resources

* Combine all data tracks into a bed4 files one per chr, also record assay types
* Retrieves the TF family name for each TF name
* Given a GTEX file, retrieve gene expression from each tissue for each TF name
* Get tissues with gene expression
* Get assay cell info
* Generate a default dict based on the information obtained from the tracks in data\_dir
* Generate mapping from cell types to tissue

### Section 4.2:

### Overlap Between the Generated Resources and Motifs

#### if args.annotateMotifs is True:

* Summarize overlapping motifs and their annotations

### Section 4.3

### DB Generation

#### if Utilities.get\_value(params['create\_database']) is True:

* Load parameters from the parameter file
* Start the psql server
* Generate cell tables
* Generate tissue tables
* Split motif table per chr
* Get PFM for motifs

### Section 4.4:

### Score Motifs

#### if args.regression:

* Get the logistic regression parameters
* Save the parameters to a file

## Section 5:

## Get the Functional Motifs per Tissue

#### if args.findFunMotifs:

* Get functional motifs per tissue
* Save output to database

## Section 6:

## Overlap Motifs with Non-coding Mutations and Compute Entropy

#### if args.findFunMotifVariants:

* Create output files containing functional motifs containing variant for each tissue
* Compute entropy for each tissue
* Save output to database