Documentation de ViCoD

Installation

Les fichiers fournis par l'utilisateur dépassent 2Mo qui est la limite imposée par php. Pour modifier le maximum de chargement accepté, il faut changer deux paramètres dans le fichier *php.ini* se trouvant dans le répertoire *etc/php/7.2/apache2/php.ini* :
upload_max_filesize = 200M
post_max_size = 200M
Pour le regroupement clonal fait par GTM et l'analyse de l'hétérogénéité intraclonale, il est nécessaire de créer un environnement Anaconda dans lequelle on installe les packages ci-dessous :
- python-levenshtein : conda install -c conda-forge python-levenshtein
- scikit-bio : conda install conda-forge scikit-bio
- python-newick : conda install -c bioconda python-newick
Le chemin pour accéder à Anaconda doit être modifié à la ligne 3 (source) dans le fichier run_GTM.sh (contenu dans le répertoire pipeline/GTM).

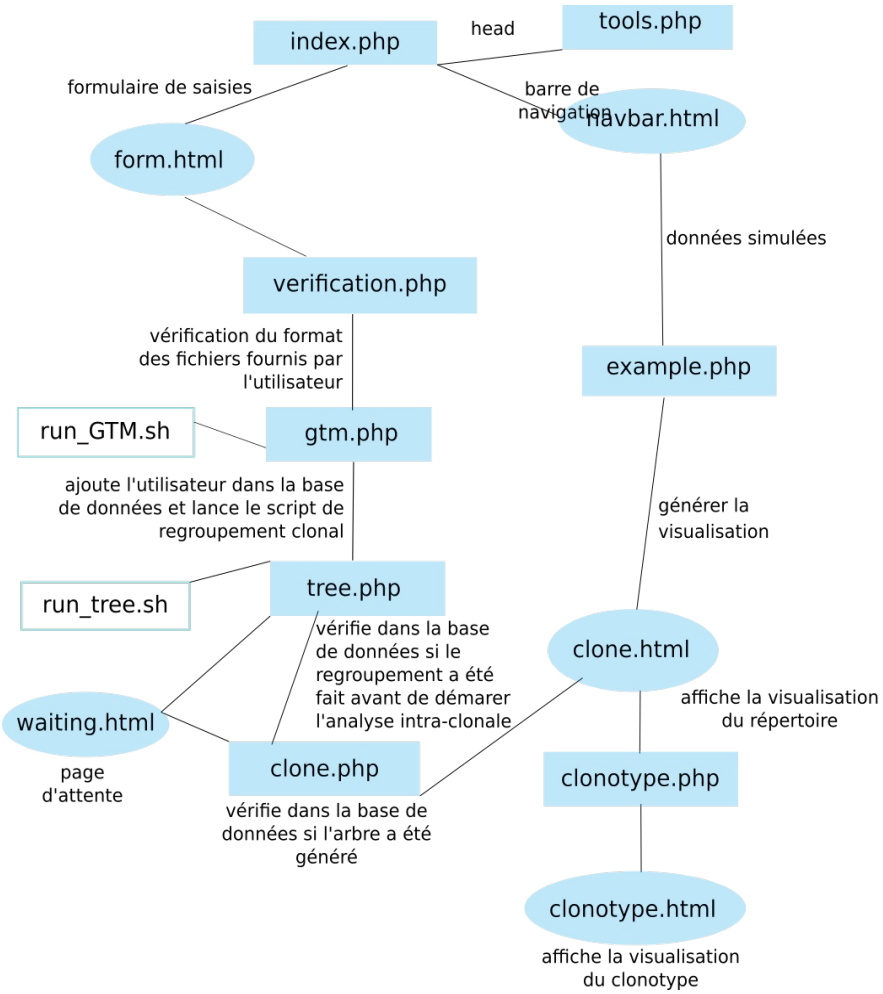

Illustration 1: Schéma de l'appel des fonctions

Description des fichiers :

- index.php : il fait appel à deux fichiers html (navbar.html et form.html) et au fichier tools.php qui contient la partie head. La page affichée correspond au formulaire permettant à aux utilisateurs de charger leurs fichiers.

- verification.php : le script vérifie le nombre de fichiers fournis par l'utilisateur ainsi que leurs formats. Si c'est le bon format il crée un répertoire avec un numéro aléatoire pour l'utilisateur.
- gtm.php : ajoute l'utilisateur à la base de données contenant l'ensemble des utilisateurs. Il gère la liste des utilisateurs en les mettant en file d'attente.
- tree.php : page chargé régulièrement pour vérifier si l'analyse GTM a été faite si c'est le cas le script run_tree.sh est exécuté.
- clone.php : vérifie dans la base de données si l'analyse d'hétérogénéité intra-clonale a été faite si c'est le cas, la visualisation est affichée.
- run_tree.sh : script exécutant un algorithme permettant la reconstruction d'un arbre phylogénique et mettant cet arbre au format JSON
- run_GTM.sh : script appelant plusieurs scripts python permettant le regroupant les données de l'utilisateur en clones et clonotypes
- clone.html : script permettant la visualisation des données générées par run_GTM.sh
- exemple.php : contient des données simulées

Base de données

La base de données a été nommée BCRVisualization elle contient une table appelée users. Cette table contient 4 colonnes : userID, creationDate, firstStep, secondStep. FirstStep et secondStep stock l'état du traitement des données (waiting, in progress, done) pour les étapes de regroupement des clones et de l'analyse de l'hétérogénéité clonale.
Requêtes utilisées pour la création de la base et la table :
CREATE DATABASE BCRVisualization ;
CREATE TABLE users (userID INT, creationDate DATE, firstStep varchar2(20), secondStep varchar2(20)) ;

Informations nécessaires pour l'ajout de l'analyse de l'hétérogénéité intraclonale

Le script permettant d'analyser l'hétérogénéité intraclonale devra être lancer à partir du script *run_tree.sh* (ligne 12) localisé dans le répertoire *pipeline/tree*. Il faudra également supprimer la ligne 15 de ce script qui permet pour le moment d'ajouter des fichier nk au dossier de l'utilisateur pour simuler la génération d'un arbre. Le répertoire *false_values_tree* devra également être supprimé.

Le nombre de clone à analyser est indiqué dans le fichier *tree.php* ligne 39 par la variable **$numClone**. Il faut mettre la valeur de cette variable à 10 si l'on souhaite observer les arbres des 10 premiers clones.

Format des données prises par le script python *clonotype_informations_json* (pipeline/tree) :
- fichier nk avec comme nomenclature par exemple : *308974_C1.nk*
- fichier json généré après l'analyse par GTM *308974_repertoire_two_levels_info.json*
- numéro du clone (attention : les clones sont numérotés à partir de 1)
Le fichier généré a pour nomenclature par exemple : *308974_C1_clonotype.json* où 308974 est l'identifiant de l'utilisateur et C1 le clone numéro 1.
Le nom de la séquence germinale pour chacun de ces arbres doit être : « **ighv** » et la valeur d'abondance de cette séquence est automatiquement fixée à 1. Cette valeur d'abondance peut être modifier dans le script python *clonotype_informations_json* (pipeline/tree) ligne 59 : **tree["value"]=1**.

Modifier le rayon des nœud dans l'arbre il faut modifier les parties en rouge dans les deux fichiers clone.html et clonotype.html :

- .on("mouseover", function(d) { var g = d3.select(this); //g object of the node
  var info = g.append('text').classed('info', true).attr('x', -25).attr('y', function(d) { return -(5+(**5+(0.25*d.data.value)**))}).text(function(d) { if(d.data.name!="ighv") { return d.data.name; }}).attr('font-size', 12) ;}) (ligne 576 dans clone.html)

- .attr("r", function(d) { return d.data.value? **5+(0.25*d.data.value)** : 0}); //set the diameter of the node (ligne 587 dans clone.html)

- .attr("dy", function(d) { return 15+(**5+(0.25*d.data.value)**)}) //set the emplacement of the text (ligne 591 dans clone.html)

- nodeUpdate.select('circle').attr('r', function(d) { return d.data.value? **5+(0.25*d.data.value)** : 0}) (ligne 607 dans clone.html)

5+(0.25*d.data.value) : ici l'échelle va de 5 à 25, un clonotype avec une abondance de 100 % aura un rayon de 25. Cette équation peut être remplacée par une valeur fixe dans ce cas tous les nœuds de l'arbre auront le même rayon. d.data.value correspond à la valeur d'abondance dans le clone du clonotype sur lequel ont se trouve lorsque l'on parcours la hiérarchie des sonnées avec les fonctions ci-dessus.