## Gecko Guidelines

## 1. Parameter details :

Genetic algorithm parameters

* **pathData :** path to the count matrix of kmer
* **AlgorithmType :** Type of genetic algorithm value possible
  + IAGA : improve adaptative GA
  + AGA : adaptive genetic algorithm (historic)
  + GA : fixed translocation and mutation rate (historic)

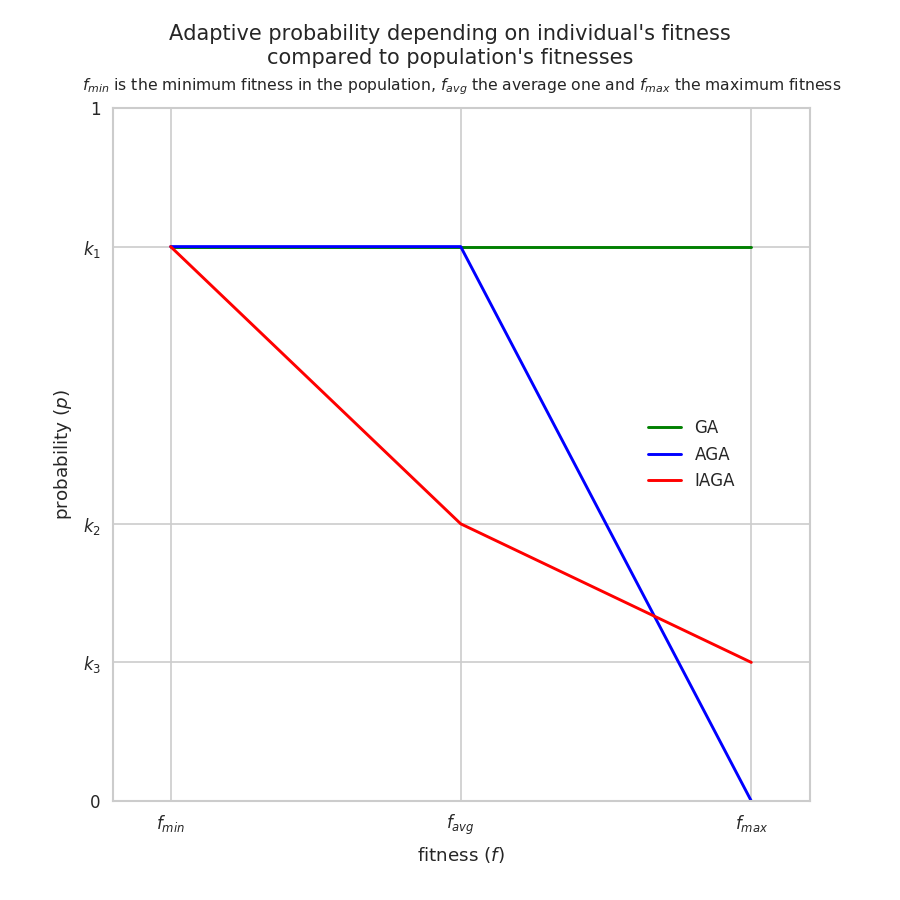
We recommend to use  IAGA for the best performance

* **Generation** : number of generations until the AGA stops, more generations will improve the search and exploration at the cost of more time. This value is highly dependent on the number of kmers to explore. Whole exome or genome experiments require more generations, small RNA experiments where the number of possible k-mers is small require less generations. *Recommended values {1000..50000}*
* **Individuals** : number of individuals per generation. A too small population size may prevent the AGA from reaching a good fitness. Above the optimal population size, the fitness will stop increasing and the computational resources to reach the same level will increase. *Recommended values {100..900}*
* **Elite** : number of individuals who will stay untouched at each generation. Larger elite sizes will allow good solutions to propagate more and thus increase convergence speed at the cost of searching broader sets of solutions. *Recommended values {0..5}*
* **killRatio** (0-1): ratio of individuals removed and randomly replaced at each generation. The individuals with the lowest fitness will be deleted. High values favor exploration but will reduce the capacity to converge to a solution. *Recommended values {0.1..0.5}*

***Mutation and crossing over rates:***

GECKO makes use of an Adaptive Genetic Algorithm that adapts the mutation and cross-over probabilities depending on the homogeneity and the performances of the population in order to converge faster and more accurately. This approach aims to maintain the population's diversity while protecting good individuals from modifications. The mutation and cross-over probabilities are decreased when the individual's fitness is high compared to the average and increased if it is low. Similarly, the probabilities are decreased when the population is heterogeneous and increased when the population is homogeneous to favor exploration of novel solutions. These probabilities are modelled by 3 linear functions depending on whether the individual is above the average fitness of the population or below it and is given by the formula below.

Here f is the individual's fitness, fmin is the fitness of the population's worst individual, favg is the population's average fitness and fmax is the fitness of the population's best individual.  
k1 is the rate applied when f=fmin, k2 when f=favg and k3 when f=fmax.

* **Mutationmode\_1kmeronly** : historic parameter. Only 1 kmer will be mutated per individual if set to 1. Recommended value: 0.
* **mutationRate1**: This is the mutation rate for individuals with the lower fitness (k1). *Recommended values {0.3..0.7}**.* Historic: for a non-adaptive GA this is the mutation rate.
* **mutationRate2**: This is the mutation rate for individuals with average fitness (k2). *Recommended values {0.05..0.2}*
* **mutationRate3**: This is the mutation rate for individuals with the best fitness (k3). *Recommended values {0..0.1}*
* **TranslocationRate1**: This is the translocation (or cross-over) rate for individuals with lower fitness (k1). *Recommended values {0.05..0.2}.* Historic: for a non-adaptive GA this is the translocation rate.
* **TranslocationRate2**: This is the translocation (or cross-over) rate for individuals with average fitness (k2)*. Recommended values {0.3..0.5}*
* **TranslocationRate3**: This is the translocation (or cross-over) rate for individuals with the best fitness (k3)*. Recommended values {0.2..0.8}*
* **Kselection** : number of individuals randomly selected to give their k-mers to another individual selected for cross-over; only the one with the best fitness will give its k-mers. This parameter has a high impact on AGA convergence . High Kselection pushes to stronger convergence but less exploration. *Recommended values {10..100} ( or put differently {5%..20%} of population size :* ***Individuals)***
* **outterRate**(0-1): ratio of samples used to test the accuracy of GECKO (this is the ratio of the category with the least samples). Larger numbers give better information on the generalisation capacity but limit the number of samples used during the AGA*. Recommended values {0.1..0.3}*
* **testRate**(0-1): ratio of inner samples (based on the smallest category number of sample). Larger numbers give more robust fitness scores but limit the number of samples used for learning. For small datasets this can to under learning *. Recommended values {0.2..0.4}*
* **scoreMaxTreshold** : stop the AGA when an individual reaches this fitness (before the number of maximum generations. Set above 1 to avoid premature stop *. Recommended values {1.1}*

Machine learning parameters :

* **Kmer** : number of kmers in each individual. This is up to the user to decide. Smaller numbers may lead to smaller accuracy scores but allow for small classifiers.
* **Method** : method of machine learning to be use. The value can be:
  + MLP : neural network
  + ADA : AdaBoost classifier
  + SDG : Linear classifiers (SVM, logistic regression, a.o.) with SGD training.
  + SVC : C-Support Vector Classification
  + linSVC : Linear Support Vector Classification.
  + KNEIG : Classifier implementing the k-nearest neighbors vote.
  + SVMfourier : Support Vector Classification with fourier approximation kernel
  + SVMnystro : Support Vector Classification with Nystroem approximation kernel

*Recommended values “linSVC” due to its speed and good generalization for small datasets*

* **hiddenNeuron**: number of hidden neurons for MLP (neural network) *Recommended values {number of Kmer/2 .. number categories}*
* **shufTrainTestDataType**:test/train method :

1 -> to add noise : mode to add noise (see **noisefactor** above)

2 -> artificially triple the matrix samples and add noise (not recommended)

3 -> no noise

* **noisefactor** : multiplication factor applied to the standard deviation of a sample to add noise to the k-mer counts. Higher values introduce higher levels of noise. This parameter is useful for datasets with a small number of samples to avoid overfitting *Recommended values {0.5..1.5}*
* **nRotativeTest**: number of rotations performed on the train/test set to determine accuracy for each individual. More homogeneous datasets require less rotations. More rotations will give a fitness score that is more representative of each individuals capacity to classify data but will increase the computational time. *Recommended values {2..6}*

Log config

* **pathLog**: path to the directory to save data results. In case of parameters with multiple values the directory will be extended with the first 2 letters of the parameter and the value corresponding to the AGA run. The pathLog is also automatically extend with ”Dir”
* **allResultHistoryBOOL** : set to 1 the fitness score of all individuals at each generation will be saved in a file *Recommended value 1*
* **generationRefreshLog**: intervals at which the AGA results will be saved. This allows users to see the evolution of the AGA solutions. At each checkpoint a numbered folder will be created with all results. Setting this interval too low will drastically increase the size of the result directory and run time *Recommended values {***Generation**/5 .. **Generation**}
* **computeAllOutters**: set to 1 the outer score of all individuals at each generation will be saved in a file, it require more computational time at each generation. Set to 0 only the outer score of the winner of each generation will be compute. *Recommended value 1*

Restart config

This section allow the user to start genetic algorithm from data genrate by previous genetic algorithm keep the section comment if you don’t use it

* **startFromPopulation:** Complete path to previous run checkpoint individuals population. The first generation will be initialise with individuals from this file ( the populations sizes must be the same ). This is usefull to restart a GA.

*Exemple :**logDir/0\_1/\_tableOfIndividuals.csv*

* **startFromOuter**: path to previous run outter information file. The algorthim will then start with same samples as outer. This is usefull to start multiple GA with same outer or restart a GA. *Exemple : logDir/\_outerSelected.csv*
* **startfromKmerSubset**: path to count file of k-mer. Only the k-mer present on this list will be selected in the pathData matrix to run the GA. It’s usually recommend to generate a new matrix for better performance on large dataset ( with ProducteurV2/sampledatamat ) *Exemple : logDir/0\_1/countkmerwin0.txt*

## 2. Running GECKO for the first time

* Perform Kmer extraction and filtering
* Select the number of Kmers per individual by doing the following:

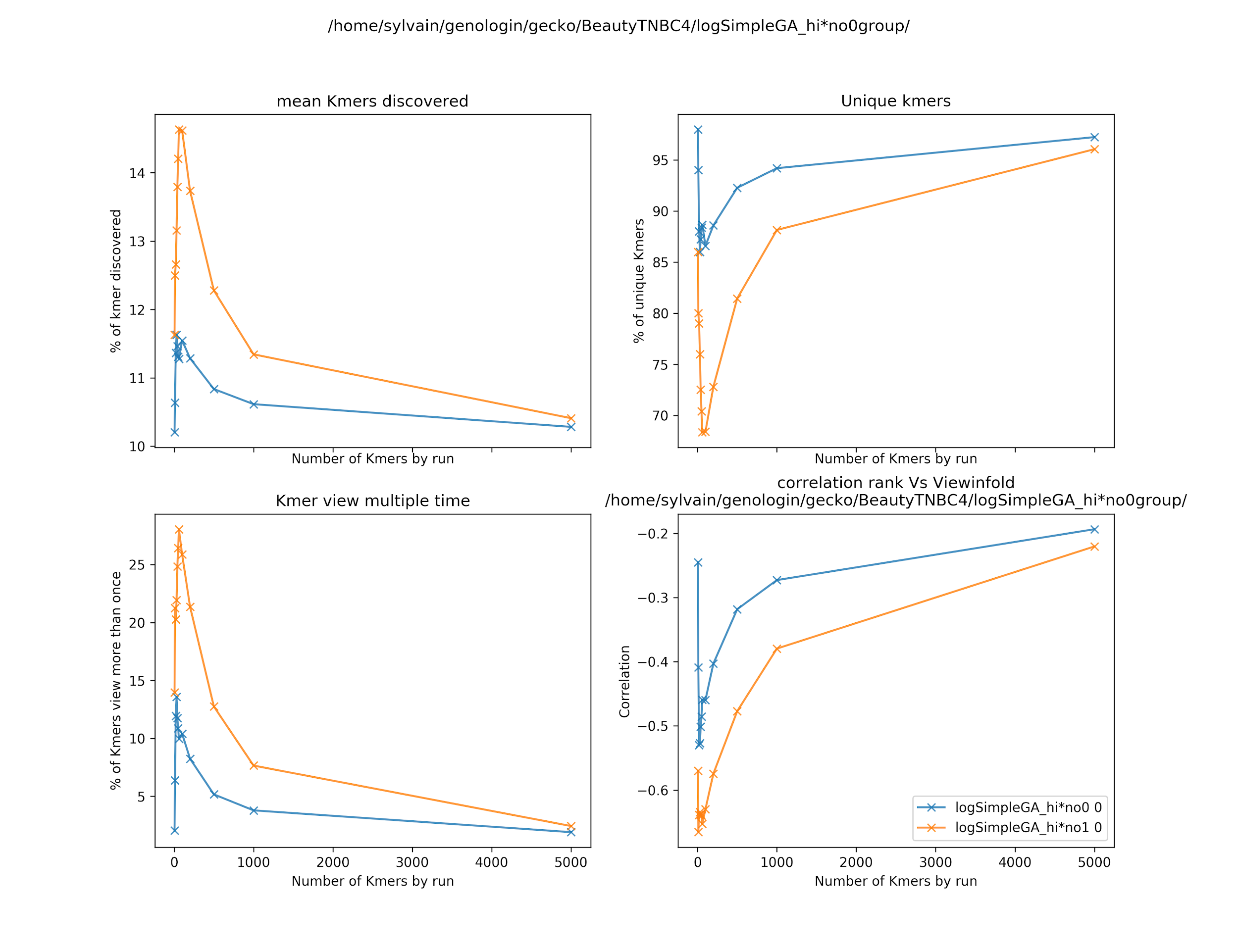
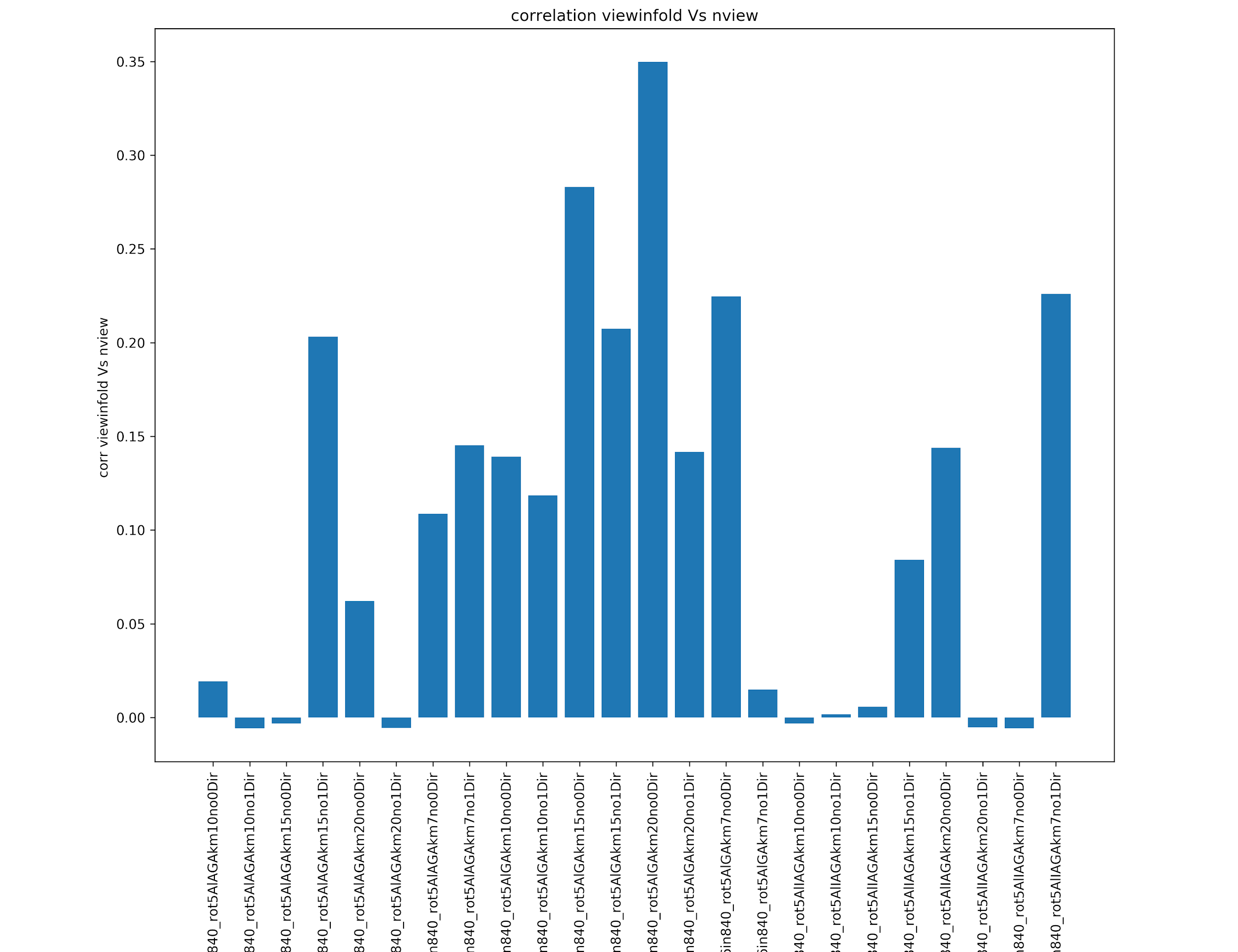
Run multiple AGAs with a broad range of kmers numbers ( 5,10,20,30,50,80,100,150), a small number of generations and high convergence parameters (kselection>80,elite>3). scoreMaxTreshold can also be set to 1 to limit the number of iterations of the AGA.

Analyze : detect for highest limit by looking inner score(fitness) = 1 and/or correlation inner/outer dropping

* Reduce kmer list and optimizing AGA parameters :

Run multiple AGA with numbers of kmer in the range of possible define in the previous step and predefined sets of convergence parameters, good amount of replicates.

Analyze : merge list of most viewed kmer , possible selection of the best AGA parameters by looking at the correlation number of view in a run and number of run where the kmer has be found *(script occurencemerge.py)*



* From merged kmer list (reduced matrix) : run AGA to found : best classifiers, clusters of solutions or minimum numbers of kmer necessary regarding your goal
* At this step, if you need to improve your classifier : It could be interesting to try differents machine learning and hyperparameter and maybe even restart to step 2 with the new selected machine learning algorithm