Scripts user guidelines for great apes project

error_rate_calc.py, functionC

Type: function

Description: This is a function that calculates the error rate for a mapped virus. The error rate is defined as the number of mapped nucleotides that do not match the reference divided by the total number of mapped nucleotides.

Input: A list where each three consecutive positions contain [Virus ID, referenceNucleotide, mappedNucleotide].
Example: [NC.006432, ‘A’, ‘A’, NC.006432, ‘A’, ‘A’, NC.006432, ‘A’, ‘G’, NC.006432, ‘A’, ‘A’, NC.006432, ‘A’, ‘A’, NC.005269, ‘C’, ‘C’, NC.005269, ‘C’, ‘C’,]

Returns: A dictionary with the virus ID as key and the error rate as values.
Example: {‘NC.006432’: 0.2, ‘NC.005269’: 0}

coverage_peak_filter_v4.py, functionB

Type: function

Description: This is a function that removes sites with unusually high coverage. It identifies high coverage peaks and removes them from further analysis if there is also a peak at low coverage. Coverage is defined as the number of nucleotides mapped to a site.

Input: (sample_file, virus_ids, mean_th=1.0, peak_locov=0.15, peak_hicov=0.15)

- sample_file: location of sample (ape.mpile.gz) files.
- virs_ids: List containing IDs of viruses to be included in the analysis. If the virus ID is present in the sample files but not in this list, they will not be included.
- mean_th: Sets multiplier for the mean for determining threshold. If mean(peak) is greater than mean(rest) * mean_th the peak is removed from the data
- peak_locov: How many % of the coverage interval a low coverage peak is expected to cover
- peak_hicov: How many % of the coverage interval a high coverage peak is expected to cover

Returns: Dictionary with virus ID's as keys. The values are one dict per virus containing the keys trim_map_loc, trim_ref_nuc, trim_map_nuc and nr_removed_sites. The values of these dicts are lists with the mapping locations, the reference nucleotide, the mapped nucleotides and the number of sites that have been removed respectively

Details: The way the script identifies a peak is by first determining the low coverage sites, high coverage sites and medium coverage sites. With default parameters high and low coverage is defined as the 15 % highest and the 15 % lowest coverage, and medium coverage is everything in between. If the average number of sites with high coverage is greater than the average number of sites with medium coverage (multiplied by an optional scaler) there is a high coverage peak. This peak is removed if there also is a low coverage peak, which is defined in the same way with a predetermined scaler of 1.2.

abundance_calc_v2.py, functionA

Type: function

Description: This is a function that removes sites with unusually high coverage.

Input: a tuple where each consecutive two positions contain virus ID and mapped nucleotides.
Example: tup = [‘NC.006432’, ‘AAAA, ‘NC.006432’, ‘CC’, ‘NC.006432’, ‘CCC’, NC.006432, ‘GG’]

Returns: Dictionary with NCBI virus id:s as keys and abundance of virus (number of virus copies present) as values.
Example: {'NC_008892.1': 6.2272727272727275, 'NC_006276.1': 2.65, 'NC_016447.1': 1.5583756345177664, ...}

Incoherence_filter_SW.py, incoherence_filter_sw

Type: function

Description: This is a function that filters out regions with incoherent mapping in viruses. The incoherence score is defined as the number of mapped nucleotides that are not the same as the most commonly mapped nucleotide for a given position, divided by the total number of mapped nucleotides for that position.
Example:
AAA = 0
AAT = 0.33
AATT = 0.5
ATCG = 0.75

Input: (Cov_result, incoherenceThreshold, smooth, binsize, jumpSize)

- Cov_result: A dictionary with virus ID’s as keys. The values are another dictionary, containing the keys ‘trim_map_loc’, ‘trim_ref_nuc’ and ‘trim_map_nuc’. The values of these dictionaries are lists with mapping locations, reference nucleotides and mapped nucleotides.
  Example:
  {‘NC.006432’: {‘trim_map_loc’: [3517, 3518, 3519, 3522, 3523], ‘trim_ref_nuc’: [‘A’, ‘A’, ‘C’, ‘T’, ‘A’], ‘trim_map_nuc’: [‘AAAA’, ‘AAT’, ‘CC’, ‘TTGTT’, ‘AAAA’]},
  ‘NC.005269’: {‘trim_map_loc’: [201, 202], ‘trim_ref_nuc’: [‘C’, ‘G’], ‘trim_map_nuc’: [‘C’, ’GG’]}}

- incoherenceThreshold: A number that defines the threshold for filtering. Regions with average values above this threshold will be removed.
- smooth: Should take values 1 or 0. 1 represents the case when the user does want to smooth the regions with a sliding window before filtering. 0 is the case where no smoothing will be done, and the function will remove each individual site with an incoherence above the threshold.
- binsize: defines the size of the sliding window for smoothing.
- jumpSize: Defines the maximum distance between two mapped positions in order for the sliding window to keep sliding. If the distance between a mapped position and the next exceeds this number, the sliding window will not slide over this gap, and instead jump to the start of the new region.

Returns: A dictionary with the virus ID as keys. The values are another dictionary, containing the keys 'positions', 'mapped_nucs', 'ref_nuc' and 'nr_removed_sites'. The values of these dictionaries are lists with mapping locations, mapped nucleotides, reference nucleotides and number of sites that were removed by the filter.
Example:
{'NC.006432': {'positions': [3517, 3518, 3519], 'mapped_nucs': ['AAAA', 'AAT', 'CC''], 'ref_nuc': ['A', 'A', 'C''], 'nr_removed_sites: '2' },
'NC.005269': {'positions': [201, 202], 'mapped_nucs': ['C', 'GG'] 'ref_nuc': ['C', 'G'], 'nr_removed_sites: '0'}}

Details
The script goes through one virus key in the Cov_result dictionary at a time. For each virus, it creates a list with all the positions, and a list with the corresponding fractions.
Example: Positions = [3517, 3518, 3519, 3522, 3523], Fractions = [0, 0.33, 0, 0.2, 0]

if smooth == 1:

> It then goes through the positions to find every jump that is longer than jumpSize (if Positions[i + 1] > Positions[i] + jumpSize)
>
> Mapped regions separated by jumps are stored as a dictionary called 'regions', where the key is simply the number of the region, and the value is a list will all positions belonging to that region.
>
> The script then handles one region at a time. A sliding window of width windowSize is applied to the region. It moves one nucleotide per slide and the mean incoherence score is calculated for each window. When the edge of the window reaches the end of the region, the window stops and jumps to the start of the next region.
>
> If the region is smaller than windowSize, the whole region is considered to be one window and the average incoherence score is calculated for that region.
>
> If there are no jumps in the mapping longer than jumpSize, the window will slide along the whole mapped regions.

All positions belonging to a window with an incoherence score above incoherenceThreshold are then removed.

If smooth == 0:

In this case, the script will simply go through all indexes in Fractions and delete the site if it's incoherence score is above the threshold. It will also delete the corresponding index in Positions.

The filtered values are then stored as lists and returned as the output specified above.

Main.py

type: main

Description: This is the main script used to filter viruses and calculate a score for each virus.

file paths:

- outfile_path = 'test_output/output.tsv'
  The location and name of the output file generated by the script
- path = 'sample_data/'
  Directory where all the input ape.mplie.gz files are stored
- virus_size_file = "virus_data/virus_genome_sizes.tsv"
  Location and name of tsv file containing virus IDs and genome sizes
- virus_name_file = "virus_data/virus_names.tsv"
  Location and name of tsv file containing virus IDs and names.

required functions:

- incoherence_filter_sw from Incoherence_filter_SW
- map_percent_filter from map_percent_filter_v4
- functionA from abundance_calc_v2
- functionB from coverage_peak_filter_v4
- functionC from error_rate_calc_v2

Specified variables:

- MappedThreshold = float: Specify a minimum fraction of the virus genome that should be mapped in order to be included in the output file. Set this to zero to not remove any viruses based on this condition.
- incoherenceThreshold = float: Specify the input parameter incoherenceThrehold to the function incoherence_filter_sw. This defines the incoherence score threshold below which regions/positions will be removed by the function.

- ErrorRateThreshold = float: Specify a maximum error rate of the viruses allowed in order to be included in the output file. Set this to one to not remove any viruses based on this condition.
- AbundanceThreshold = float: Specify a minimum number of viruses found in order to be included in the output file. Set this to zero to not remove any viruses based on this condition.
- smooth = int: Specify the input parameter smooth to to the function incoherence_filter_sw. Set this to 1 if you wish the incoherence filtering to be done using sliding windows, and to 0 if you don't want any smoothing in the incoherence filtering. We strongly recommend this to be set to 1.
- binsize = int: Specify the input parameter binsize to the function incoherence_filter_sw. It defines the size of the sliding window used for smoothing.
- jumpSize = int: Specify the input parameter jumpSize to the function incoherence_filter_sw. It defines the maximum distance between mapped sites in order for the sliding window to keep sliding across the positions.

output:
Excel and tsv files where the columns contain file name, virusID, virus name, abundance, error rate, %of virus genome present in the sample and how many percent of the sites were filtered out. The column names are set to be fileName, virusId, virusName, abundance, coverage, errorRate, percentRemoved.

analysing_output_v5.py

type: main

Description: This is a main script used visualize the output file generated by the main script.

file paths:

- inputFile = 'test_output/output.tsv
  The location and name of the output file generated by the main script
- sampleStats = 'test_output/sample_stats_complete.xlsx'
  Location and name of excel sheet containing information about the samples. Each row is one sample (ape) and the columns contain information about geographical origin, species, if it is wild or captive etc. The columns should have appropriately named headers.
- virus_size_file = “virus_data/virus_genome_sizes.tsv”
  Location and name of tsv file containing virus IDs and genome sizes
- virus_name_file = “virus_data/virus_names.tsv”
  Location and name of tsv file containing virus IDs and names.

Specified variables:

- statsColumn = str. Specify the name of the column you want to colour the PCA plot by.

- virus1 to virus2 = str. If desired, specify virus IDs to be excluded from visualization. Recommended is 'NC_001422.1', which is the known contamination PhiX.
- outlier1 to outlier3 = str. If desired, specify files (apes) to be excluded from PCA plot. For example, remove outliers to get a better look at the rest.
- targets = ['Wild born', 'Captive born', 'Human'], colors =[‘r’, ‘b’, ‘g’]
  Around line 180. Change this according to column chosen in statsColumn.

fetch_names.py

Type: main

Description: This is a short script that, for the given virus IDs, fetches the virus names from ncbi. The output file is then used by the other scripts.

Specified variables:

- virus_filename = ‘sample_data/virus_genome_sizes.tsv’
  Path and name of file containing virus genome sizes
- virus_output = ‘sample_data/virus_names.tsv
  output file containing names of viruses
- In line 20, specify which database to fetch names from. For viruses, we have used ‘nuccore’