



Version 2 ▾

Feb 28, 2021

# Recan: R-based tool for detection of recombination in viral genomes V.2

Vasily Pavelko<sup>1</sup><sup>1</sup>International Biotechnology Center Genterium

1 Works for me dx.doi.org/10.17504/protocols.io.bsjcncje

recan



Vasily Pavelko

SUBMIT TO PLOS ONE

## ABSTRACT

Recan is an R version of the Python library that allows the identification of recombination events through construction and exploration of similarity plots based on genetic distances between nucleotide sequences. After porting I validated this package using sequences of viruses with previously identified recombinations. During benchmarking both versions demonstrated the same execution time.

To benchmark the packages we selected the same four sets of viral genomes used in the Python version of recan ([Babin, 2020](#)).

## DOI

[dx.doi.org/10.17504/protocols.io.bsjcncje](https://dx.doi.org/10.17504/protocols.io.bsjcncje)

## PROTOCOL CITATION

Vasily Pavelko 2021. Recan: R-based tool for detection of recombination in viral genomes. **protocols.io**  
<https://dx.doi.org/10.17504/protocols.io.bsjcncje>  
Version created by Vasily Pavelko

## LICENSE

— This is an open access protocol distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

## CREATED

Feb 19, 2021

## LAST MODIFIED

Feb 28, 2021

## PROTOCOL INTEGER ID

47430

## DISCLAIMER:

### DISCLAIMER – FOR INFORMATIONAL PURPOSES ONLY; USE AT YOUR OWN RISK

The protocol content here is for informational purposes only and does not constitute legal, medical, clinical, or safety advice, or otherwise; content added to [protocols.io](#) is not peer reviewed and may not have undergone a formal approval of any kind. Information presented in this protocol should not substitute for independent professional judgment, advice, diagnosis, or treatment. Any action you take or refrain from taking using or relying upon the information presented here is strictly at your own risk. You agree that neither the Company nor any of the authors, contributors, administrators, or anyone else associated with [protocols.io](#), can be held responsible for your use of the

## Benchmarking of recan package in R

1 Calculating average execution time of code using Human Immunodeficiency Virus datasets.

2 Calculating average execution time of code using Hepatitis C virus datasets.

```
setwd("/path/to/sequences/")
ls <- list.files(path = ".")
t <- 3
{
  seq <- read.fasta(ls[t])
  mat <- (replicate(30, system.time({seqSim(seq, window = 400, shift = 200)})))
  print(apply(mat, MARGIN = 1, mean)*1000)
  print(apply(mat, MARGIN = 1, sd)*1000)
}
```

3 Calculating average execution time of code using Norovirus datasets.

```
setwd("/path/to/sequences/")
ls <- list.files(path = ".")
t <- 7
{
  seq <- read.fasta(ls[t])
  mat <- (replicate(30, system.time({seqSim(seq, window = 400, shift = 200)})))
  print(apply(mat, MARGIN = 1, mean)*1000)
  print(apply(mat, MARGIN = 1, sd)*1000)
}
```

4 Calculating average execution time of code using Lumpy Skin Disease Virus datasets.

```
setwd("/path/to/sequences/")
ls <- list.files(path = ".")
t <- 6
{
  seq <- read.fasta(ls[t])
  mat <- (replicate(30, system.time({seqSim(seq, window = 400, shift = 200)})))
  print(apply(mat, MARGIN = 1, mean)*1000)
  print(apply(mat, MARGIN = 1, sd)*1000)
}
```

- 5 Calculating average execution time of code using Human Immunodeficiency Virus datasets.

```
library(seqcombo)
{
  mat <- (replicate(30, system.time({simplot("hiv.fasta", query = 1,
                                         window = 400, step = 200,
                                         group = FALSE, id, sep,
                                         sd = FALSE)})))
  print(apply(mat, MARGIN = 1, mean)*1000)
  print(apply(mat, MARGIN = 1, sd)*1000)
}
```

- 6 Calculating average execution time of code using Hepatitis C virus datasets.

```
library(seqcombo)
{
  mat <- (replicate(30, system.time({simplot("hcv.fasta", query = 1,
                                         window = 400, step = 200,
                                         group = FALSE, id, sep,
                                         sd = FALSE)})))
  print(apply(mat, MARGIN = 1, mean)*1000)
  print(apply(mat, MARGIN = 1, sd)*1000)
}
```

- 7 Calculating average execution time of code using Norovirus datasets.

```
library(seqcombo)
{
  mat <- (replicate(30, system.time({simplot("norovirus.fasta", query = 1,
                                         window = 400, step = 200,
                                         group = FALSE, id, sep,
                                         sd = FALSE)})))
  print(apply(mat, MARGIN = 1, mean)*1000)
  print(apply(mat, MARGIN = 1, sd)*1000)
}
```

- 8 Calculating average execution time of code using Lumpy Skin Disease Virus datasets.

```
library(seqcombo)
{
  mat <- (replicate(30, system.time({simplot("lsdv.fasta", query = 1,
                                         window = 400, step = 200,
                                         group = FALSE, id, sep,
                                         sd = FALSE)})))
  print(apply(mat, MARGIN = 1, mean)*1000)
  print(apply(mat, MARGIN = 1, sd)*1000)
}
```

#### Benchmarking of recan library in Python

- 9 Calculating average execution time of code using Human Immunodeficiency Virus datasets. Each step I run 2 times as a preparatory steps, and then 7 runs used for calculation average time.

```
from datetime import datetime
sim_obj = Simgen("/path/to/sequences/hiv.fasta")
start_time = datetime.now()
sim_obj.simgen(window=400, shift=200, pot_rec=1)
time_elapsed = datetime.now() - start_time
print('Time elapsed (hh:mm:ss.ms) {}'.format(time_elapsed))
```

- 10 Calculating average execution time of code using Hepatitis C virus datasets.

```
from datetime import datetime
sim_obj = Simgen("/path/to/sequences/hcv.fasta")
sim_obj.get_info()
start_time = datetime.now()
sim_obj.simgen(window=400, shift=200, pot_rec=1)
time_elapsed = datetime.now() - start_time
print('Time elapsed (hh:mm:ss.ms) {}'.format(time_elapsed))
```

- 11 Calculating average execution time of code using Norovirus datasets.

```
from datetime import datetime
sim_obj = Simgen("/path/to/sequences/norovirus.fasta")
start_time = datetime.now()
sim_obj.simgen(window=400, shift=200, pot_rec=1)
time_elapsed = datetime.now() - start_time
print('Time elapsed (hh:mm:ss.ms) {}'.format(time_elapsed))
```

## 12 Calculating average execution time of code using Lumpy Skin Disease Virus datasets.

```
from datetime import datetime
sim_obj = Simgen("/path/to/sequences/lsv.fasta")
start_time = datetime.now()
sim_obj.simgen(window=400, shift=200, pot_rec=1)
time_elapsed = datetime.now() - start_time
print('Time elapsed (hh:mm:ss.ms) {}'.format(time_elapsed))
```