Graph Functions

Contents

Prerequisites

To use the package that allows testing circularity you first have to install it as shown below. To run the code in the box below you can just press the green play button in the left top corner of the code box. You might be asked in the console below this window to update the included packages. Afterwards it might take a while to install all 3 packages.

```
install.packages("Rcpp", repos = "https://packages.othr.de/cran/" )
install.packages("devtools", repos = "https://packages.othr.de/cran/")
devtools::install_github("StarmanMartin/GCATR", ref = "Testing")
```

Factoring a graph

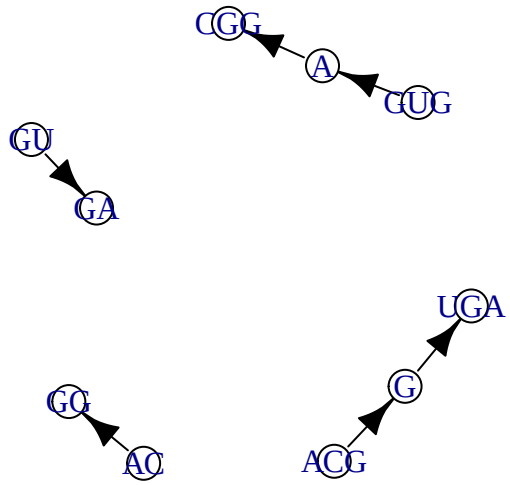
Now that you have installed all prerequisistes you can try out the GCATR package.

As with other functions there is three ways to provide input vectors/codons You can provide the input as an R vector, as a string split by spaces or stating the tuple length. The example below demonstrates the first two ways. The last way is demonstrated in the code sample below this one.

```
code_as_vector <- c("ACG", "AGU", "CGU")
code_as_split_string <- c("ACG AGU CGU")
```

To execute the following code snippet you just have to press the little green arrow on the right side of the code box. The code creates an igraph from the vectors passed to it. It contains all possible edges from given vectors.

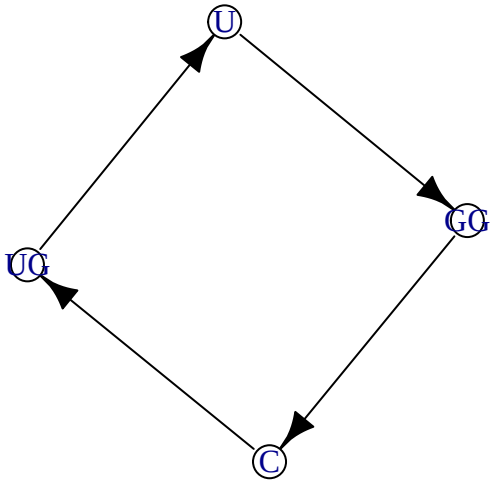
```
graph <- GCATR::code_factor_graph("ACGGGUGA",tuple_length = 4)
plot(graph)
```



Factoring a cycle

This functions returns the vectors that violate the rules of circular codes as an igraph. In this example the first codon combined with the second one allow to form the codons GGC and UGU in frameshift one (U|GG + C|UG) when read on a circle while the third and fourth codon allow to form the codons CUG and UGG in frameshift two (GG|C + UG|U) when read on a circle, thus violating the condition of circularity.

```
factored_cycle <- GCATR::code_factor_cycle("UGG CUG GGC UGU")
plot(factored_cycle)
```



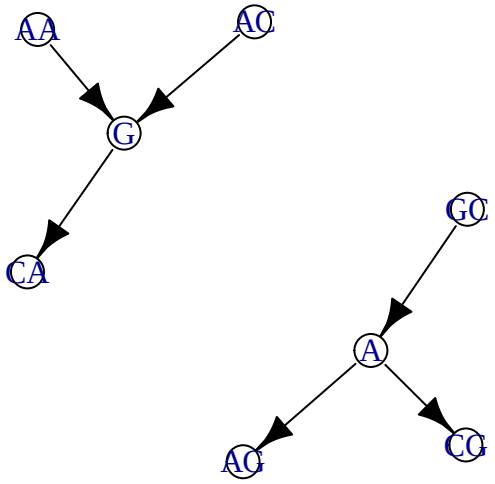
If the graph is circular NULL will be returned.

```
circular_code_example <- GCATR::code_factor_cycle("GGUGGCACUACCAGCAGUGACGAUGUCGUUAAUAUUAACAUCGCUGCC", 3)
print(circular_code_example)
## NULL
```

Factoring the longest path

This function returns the longest paths of a graph. In this example there is two paths that qualify as a longest paths, since they have equal lenghts.

```
longest_path <- GCATR::code_factor_longest_path("ACG GCA AAG")
plot(longest_path)
```



If the longest path would be infinite due to being circular, it will return NULL.

```
longest_path_is_infinite <- GCATR::code_factor_longest_path("UGGGUG", 3)
print(length(longest_path_is_infinite))
## [1] 0
```

Cycles as vector

This function returns the vectors that violate the rules of circular codes in a single combined vector.

```
cycled_vector <- GCATR::code_get_one_cycles_as_vector("UGG CUG GGC UGU")
print(cycled_vector)
## [1] "U"  "GG" "C"  "UG" "U"
```

If the code is circular the function will return an empty vector.

```
cycled_vector <- GCATR::code_get_one_cycles_as_vector("GGUGGCACUACCAGCAGUGACGAUGUCGUUAAUAUUAACAUCGCUGCC
print(length(cycled_vector))
## [1] 0
```

All cycles as vector

This functions returns the vectors from given input that violate the rules of circular codes in a list of cycle vectors.

```
cycles <- GCATR::code_get_all_cycles_as_vector(c("AG","CG","GA","TC","TT"))
print(cycles)
## [[1]]
## [1] "A" "G" "A"
##
## [[2]]
## [1] "T" "T"
```

If the code is circular, an empty list is returned

```
cycle_vectors <- GCATR::code_get_all_cycles_as_vector("GGUGGCACUACCAGCAGUGACGAUGUCGUUAAUAUUAACAUCGCUGCC
print(length(cycle_vectors))
## [1] 0
```

Get one longest path as vector

This functions returns one of the longest paths from its input as a vector.

```
longest_path <- GCATR::code_get_one_longest_path_as_vector("ACG GCA AAG")
print(longest_path)
## [1] "AC" "G"  "CA"
```

If the code is circular, an empty vector is returned

```
longest_path <- GCATR::code_get_one_longest_path_as_vector("AGCGGATCTT",2)
print(length(longest_path))
## [1] 0
```

Get all longest paths

This functions returns a list of vectors containing all longesth paths from its input.

```
longest_path <- GCATR::code_get_all_longest_path_as_vector("ACG GCA AAG")
print(longest_path)
## [[1]]
## [1] "AC" "G"  "CA"
##
## [[2]]
## [1] "GC" "A"  "AG"
##
## [[3]]
## [1] "GC" "A"  "CG"
##
## [[4]]
## [1] "AA" "G"  "CA"
```

If code has a circle, it does return an empty list.

```
empty <- GCATR::code_get_one_longest_path_as_vector("AGCGGATCTT",2)
print(length(empty))
## [1] 0
```