# Testing for circularity

## 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")
```

## Testing for self-complementarity

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

To execute the following code snippet you just have to press the little green arrow on the right side of the code box. You should be shown FALSE when executing the code snippet as the provided code is not self-complementary.

```
GCATR::code_check_if_self_complementary('UGG GUG')
## [1] FALSE
```

There is three options to provide the string that represents the code you are trying to test.

Option 1 is providing the string with a specific tuple length like this:

```
GCATR::code_check_if_self_complementary('UGGGUG', 3)
## [1] FALSE
```

Option 2 is providing the string split in tuples by spaces like so:

```
GCATR::code_check_if_self_complementary('UGG GUG')
## [1] FALSE
```

The last option is to provide each tuple separate in a vector as so: This last example should come up TRUE, as the provided code is ciruclar.

```
GCATR::code_check_if_self_complementary(c('UGG', 'GUG', 'GAG'))
## [1] FALSE
```

## Testing for maximum circularity

To test for maximum circularity you might just execute the following code snippet.

Obviously the string can be provided in the different ways mentioned in the testing for self-complementarity section.

You should be shown FALSE when executing this code snippet as the provided code is not maximum circular.

```
GCATR::code_check_if_circular('UGG GUG')
## [1] FALSE
```

## Testing for k-circularity

Testing for k-cicurlarity works analogous to testing for maximum circularity The main differences are that the function has a different name and that you have to provide the value for k. Obviously the string can be provided in the different ways mentioned in the testing for self-complementarity section.

k = 1, should come up FALSE

```
GCATR::code_check_if_k_circular(1,'UGGGUG', 3)
## [1] FALSE
```

k = 2, should come up TRUE

```
GCATR::code_check_if_k_circular(2,'ACGGUACGUCGGUAC',3)
## [1] TRUE
```

k = 3, should come up FALSE

```
GCATR::code_check_if_k_circular(3,'ACG GUA CGU CGG UAC')
## [1] FALSE
```

k = 4, should come up TRUE

```
GCATR::code_check_if_k_circular(4,c('GGU', 'GGC', 'ACU', 'ACC', 'AGC', 'AGU', 'GAC', 'GAU', 'GUC', 'GUU
## [1] TRUE
```

## Testing for cn-circularity

Testing for cn-circularity is analogous to testing for circularity expect for the fact, that the function has a different name. Obviously the string can be provided in the different ways mentioned in the testing for self-complementarity section.

This example should come up FALSE

```
GCATR::code_check_if_cn_circular('UGG GUG')
## [1] FALSE
```

While this one should come up TRUE

```
GCATR::code_check_if_cn_circular(c('GGU','GGC','ACU','ACC','AGC','AGU','GAC','GAU','GUC','GUU','AAU','A
## [1] TRUE
```

## Testing for comma-freeness

Testing for comma-freeness also works analogous to the other function calls. Just execute the following code snippet to try it out. Obviously the string can be provided in the different ways mentioned in the testing for self-complementarity section.

The example should come up TRUE.

```
GCATR::code_check_if_comma_free('GGU GGC ACU ACC AGC AGU GAC GAU GUC GUU AAU AUU AAC AUC GCU GCC')
## [1] TRUE
```