Explanation

```
library(thecentraldogma)
```

Packages description

A collection of packages containing functions to work with sequences of DNA and RNA including relevant objects and tables.

random_dna

Description: The random_dna function creates a random string of DNA of nucleic bases (A,T,G,C) with a user defined lenght.

Parameters:

lenght_of_DNA: must be an integer. Defines the lenght of the DNA string.

Examples:

```
random_dna(lenght_of_dna = 10)
#> [1] "AAAGAGGGGC"
#Outputs a string of dna bases that's 10 long.
```

replacement_of_matches

Description:

- Takes a DNA sequence and replaces all "T" with "U" resulting in the corresponding RNA sequence.

Parameters:

- DNA_sequence: must be a string

Function:

- gsub: replacement of matches

Usage:

```
replace_T_with_U(DNA_sequence)
```

Examples:

```
RNA_sequence <- replace_T_with_U(DNA_sequence = "GATTA")
print(RNA_sequence)
#> [1] "GAUUA"
```

seq_to_codons

Description

seq_to_codons is a function that takes a nucleotide sequence and splits it into the codons, starting at the position specified.

Usage

```
seq_to_codons(sequence, start = 1)
```

Arguments

sequence: A nucleotide sequence start: The first position in the nucleotide sequence that should start the codon splitting.

The codons of the input sequence is outputted as a vector of strings.

Examples

```
seq_to_codons("ACTGATCATGA", start=1)
#> [1] "ACT" "GAT" "CAT"
```

translate_codons

Description
The function translate_codons translates a vector containing nucleotide triplet(s) (codons) to a single string of amino acids.

Parameters
"codons" is a vector containing triplet(s) of nucleotides.
"standard_codon_table" is a predefined table containing the standard codons and their corresponding amino acid.

Example usage

```
codons <- c("AUG", "UUU", "UGA")
translate_codons(codons)
#> [1] "MF_"
```

plot_count

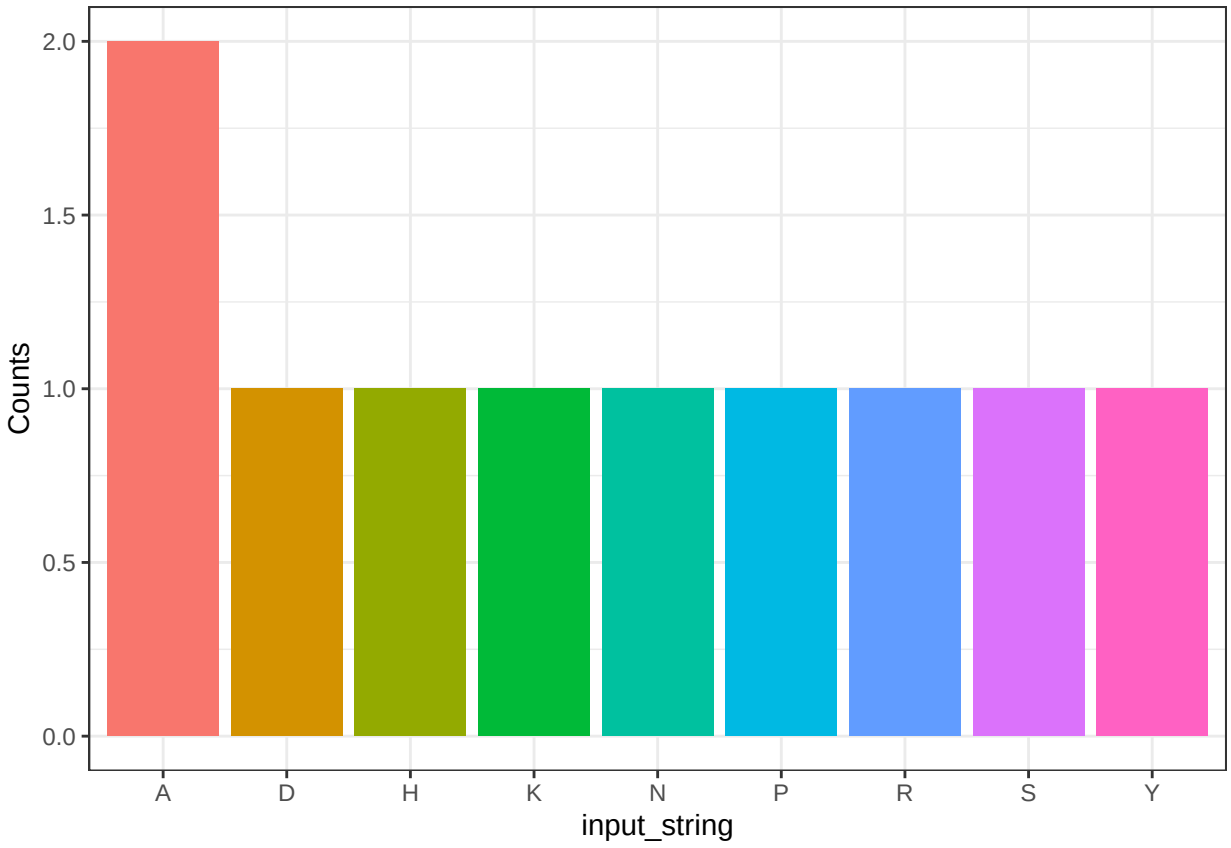
This function plots the frequency of amino acids.

The parameter *input_string* is a character string that is analyzed for unique character frequencies.

The function uses the packages stringr to split a string then counts frequency of character. Lastly it plots the frequency in columns.

Example:

```
plot <- amino_frequency_plot_local("SYRKHDAPNA")
print(plot)
```


2.0
1.5
1.0
0.5
0.0
Counts
A
D
H
K
N
P
R
S
Y
input_string