Emoji Package Manual

Tek

Emoji Data set

The complete Emoji data set is saved under the 'data' directory. This complete data set is read and named 'uni_info'. Some Emojis are a combination of two or more basic Emojis. For example, Emoji 'boy: light skin tone' is a combination of 'boy' (U+1F466) and 'light skin tone' (U+1F3FB). Data 'basic_uni_info' is a data set of the basic Emojis. There are many different ways to encode 'Unicode'. The data set includes the following encoding types: 'U+hexadecimal', 'UTF-8 hexadecimal', 'hexadecimal', and 'numeric character reference (NCR)'.

Hexadecimal UTF-8

Searching for UTF-8 hexadecimal code

Hexadecimal UTF-8 a common way to encode Unicode Emojis. Function 'find_code_utf8_hex' will target escaped hexadecimal UTF-8 character codes in a string. These character codes will start with an escape '\x' followed by two digits of alphanumeric character. For example, Emoji 'grinning face' has 'UTF-8 hex' code as '\xf0\x9f\x98\x80'. If 'grinning face' was embedded in a string, 'find_code_utf8_hex' function will generate 'f09f9880'.

```
eg_strng="StringStart\\xe2\\x9a\\xa0SomeString\\xf0\\x9f\\x91\\x91StringEnd"
eg_strng %>% find_code_utf8_hex
```

```
## [1] "e29aa0f09f9191"
```

Split the UTF-8

Hexadecimal UTF-8 codes can be categorized into three sets based on its initial alphabet: 'c', 'e', and 'f'. UTF-8 hex that starts with a letter 'c' will be 4 characters long, 'e' will be 6 characters long, and 'f' will be 8 characters long. Function 'get_code_utf8_hex' will split the unstructured strings of UTF-8 hex codes.

```
eg_strng %>% find_code_utf8_hex %>% get_code_utf8_hex
```

```
## [1] "e29aa0"   "f09f9191"
```

Match with the data base

Once the UTF-8 hex codes are successfully recovered from a given string, match them with the 'uni_info' data set to retrieve its entire information. Note that NOT ALL UTF-8 hex codes that starts with 'c', 'e', of 'f' is a code for Emojis. In the example, name of the code will be captured.

```
eg_strng %>% find_code_utf8_hex %>% get_code_utf8_hex %>% sapply(., emoji_name_hex)
```

```
##    e29aa0  f09f9191
## "warning"   "crown"
```

Another example using strings scraped from the Twitter with hash-tag #kickass.

```
txt2=readLines("../example_data_set/kickass.txt")
txt2 %>% find_code_utf8_hex() %>% get_code_utf8_hex() -> check.this
uni_info$utf_8_hex[which(check.this %in% uni_info$utf_8_hex)] %>% sapply(., emoji_name_hex) %>% unique
```

```
## [1] "kissing cat face with closed eyes" "weary cat face"
## [3] "crying cat face"                   "pouting cat face"
## [5] "see-no-evil monkey"                "hear-no-evil monkey"
```

What to do next

different types of encoding (U+hexadecimal should be the next)

more utility functions:

translater function: automatically translates the entire string.

Keyword search function: Given keyword(s) -> find unicode or vise versa.

image link function: match the Emoji image, given a unicode.

Unicode stat function?: if multiple strings are given, write a function that generates some statistics about Emoji?

Scoring function? or maybe include in the data set: positive, nutural negative Emojis?