R Package `miscset`
User Manual

Sven E. Templer
`sven.templer@gmail.com`

May 15, 2014

Contents

Part I
Preface

1 Introduction

The package **miscset** provides several R tools to read, create, modify and write different types of data. In the following examples, all available functions will be presented including explanations of their usage. Find the source code online at `http://github.com/svenetempler/miscset`.

2 Installation

To install the package first install the **devtools** package from cran via `install.packages("devtools")`. Then you can install the package from github with `install_github("svenetempler/miscset")`. After installation load the package with

```
require(miscset)

## Loading required package:  miscset
```

Part II
Functions

3 Numeric Functions

3.1 Generate triangular numbers - `ntri`

Return triangular numbers with

```
ntri(12)

##  [1]  0  1  3  6 10 15 21 28 36 45 55 66
```

3.2 Scale numeric vectors - `scale0`

Scale all values in a vector from 0 to 1 with

```
scale0(-1:3)

## [1] 0.00 0.25 0.50 0.75 1.00
```

4 Summarizing

4.1 Print a sorted table - `sortable`

Return a sorted table of vectors like `sort(table())`

```
sortable(c(1, 1, 2, 2, 2, 3))

## 2 1 3
## 3 2 1
```

```
sortable(c(1, 1, 2, 2, 2, 3), F)

## 3 1 2
## 1 2 3
```

5 Data Formatting

5.1 Transform to squared matrix - `squarematrix`

The function `squarematrix` can generate a symmetric (square) matrix from an unsymmetric matrix by using the column and row names and filling empty pairs with `NA`.

```
matA <- matrix(1:6, 2, dimnames = list(2:3, 1:3))
matA

##   1 2 3
## 2 1 3 5
## 3 2 4 6

squarematrix(matA)

##    1  2  3
## 1 NA NA NA
## 2  1  3  5
## 3  2  4  6
```

5.2 Generate a pairwise list - `enpaire`

The function `enpaire` creates a pairwise list of matrix values with upper and lower triangle values represented in a separate column. The diagonal is not returned.

```
matB <- matrix(letters[1:9], 3, 3, dimnames = list(1:3, 1:3))
matB

##   1   2   3
## 1 "a" "d" "g"
## 2 "b" "e" "h"
## 3 "c" "f" "i"

enpaire(matB)

##   row col upper lower
## 1   1   2     d     b
## 2   1   3     g     c
## 3   2   3     h     f
```

6 Text String Manipulation

6.1 Prepend zeroes to unify number lengths - `leading0`

The function `leading0` aims to create e.g. index names with a common string length. It creates character strings from numeric values while attaching `0` in front of the number up to a certain length of total digits of each string.

```
paste0("page", leading0(1:10, 3))

##  [1] "page001" "page002" "page003" "page004" "page005" "page006" "page007"
##  [8] "page008" "page009" "page010"
```

6.2 Extract substrings by pattern - `strextr`

The function `strextr` lets you extract substrings by defining a pattern of the part to extract.

```
strA <- c("A1 B1 C1", "A2 B2", "AA A1", "AA", "B1 A1", "BB AB A1")
strA

## [1] "A1 B1 C1" "A2 B2"    "AA A1"    "AA"       "B1 A1"    "BB AB A1"

strextr(strA, "^[AB][[:digit:]]$")

## [1] NA   NA   "A1" NA   NA   "A1"

strextr(strA, "^[AB][[:digit:]]$", mult = T)

## [[1]]
## [1] "A1" "B1"
##
## [[2]]
## [1] "A2" "B2"
##
## [[3]]
## [1] "A1"
##
## [[4]]
## [1] NA
##
## [[5]]
## [1] "B1" "A1"
##
## [[6]]
## [1] "A1"

strextr(strA, "^[AB][[:digit:]]$", mult = T, unlist = T)

## [1] "A1" "B1" "A2" "B2" "A1" NA   "B1" "A1" "A1"

strextr(strA, "^[C][[:digit:]]$")

## [1] "C1" NA   NA   NA   NA   NA
```

6.3 Extract substrings by splitting - `strpart`

Similar to `strextr` the function `strpart` supplies a method to extract a substring, but by defining the `n`th part of the string split by a separator.

```
strC <- c("abc", "abcb", "abc")
strpart(strC, "", 4)

## [1] NA  "b" NA
```

6.4 Reverse strings - `strrev`

With `strrev` you can create the reversed version of strings.

```
strrev(strC)

## [1] "cba"  "bcba" "cba"
```

6.5 Multiple pattern replacement - `msub`, `mgsub`

`msub` and `mgsub` behave like `sub` and `gsub` but they replace multiple patterns. Replacement is done in order of the pattern input.

```
patA <- c("a", "b")
txtA <- c("aba", "aca", "bc")
msub(patA, "", txtA)

## [1] "a"  "ca" "c"

mgsub(patA, "", txtA)

## [1] ""  "c" "c"
```

7 Pattern Matching

7.1 Get index of expression - `gregexprind`

```
patB <- c("a")
txtB <- c("abab", "ab", "xyz", NA)
gregexprind(patB, txtB, 1)

## [1]  1  1 NA NA

gregexprind(patB, txtB, 2)

## [1]  3 NA NA NA

gregexprind(patB, txtB, "last")

## [1]  3  1 NA NA
```

7.2 Multiple pattern search - `mgrepl`

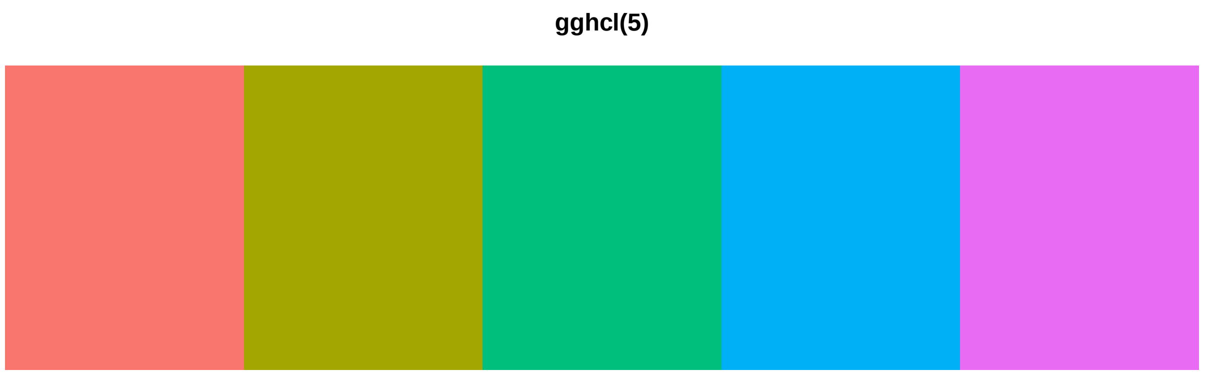
With `mgrepl()` you can search for not only one character expression, and use any logical function to combine the results for each single expression.

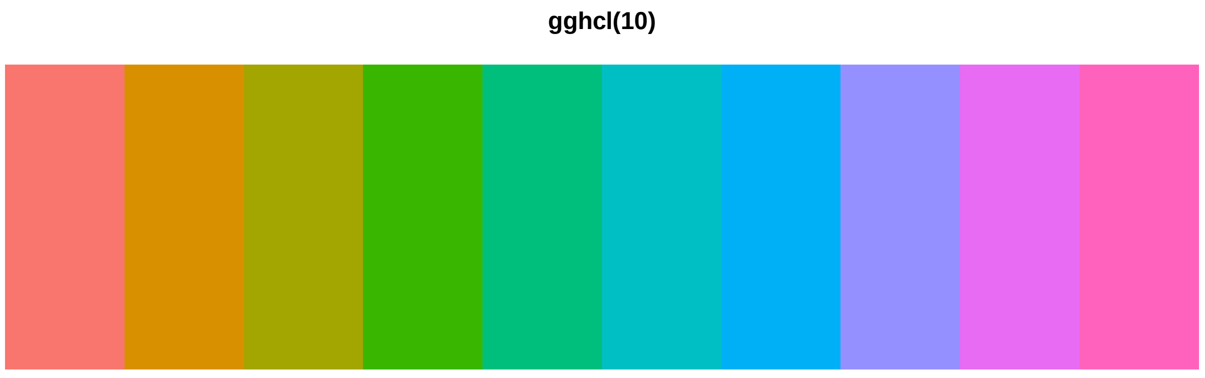
```
mgrepl(patA, txtA, any)

## [1] TRUE TRUE TRUE

mgrepl(patA, txtA, all)

## [1]  TRUE FALSE FALSE
```

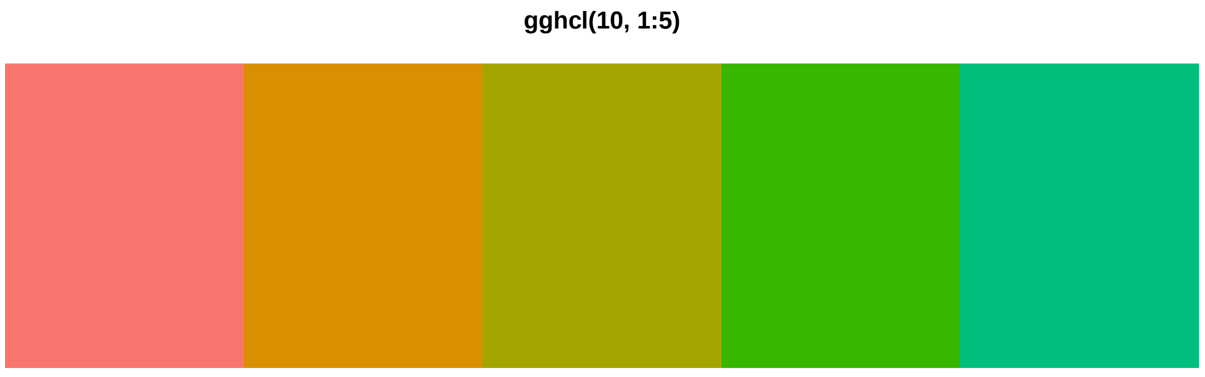
8 Graphical Tools

8.1 Create a color palette - `gghcl`

`gghcl()` creates color palettes. It enhances the `hcl` function. See some examples:







9 System Tools

9.1 List details from and remove all objects - `lsall`, `rmall`

With `lsall()` all object names, their length, class, mode and size is returned in a data.frame from a specified environment. `rmall()` removes the complete list of objects at the global environment.

```
lsall()

## Environment: R_GlobalEnv
## Objects:
##    Name Length     Class      Mode  Size Unit
## 1 matA      6    matrix   numeric 768.0 byte
## 2 matB      9    matrix character   1.3   Kb
## 3 patA      2 character character 152.0 byte
## 4 patB      1 character character  96.0 byte
## 5  pts     10    matrix   numeric 248.0 byte
## 6 strA      6 character character 392.0 byte
## 7 strC      3 character character 168.0 byte
## 8 txtA      3 character character 216.0 byte
## 9 txtB      4 character character 216.0 byte
```

```
rmall()
lsall()
```

```
## Environment: R_GlobalEnv
## Objects:
## NULL
```