Simulating and manipulating strings

Random name sampling with `randomNames()` (1/2)

- **Install & load the relevant package**

```
install.packages("randomNames")

library(randomNames)
```

1 The `randomNames` package provides a solution to sample names based on gender and ethnicity

- **`randomNames()` samples names based on gender or ethnicity**

```
randnames <- randomNames(2, gender = c("Male", "Female"))

randnames
```

2 Specify the number of names, that should be generated

3 Specify the gender. Take a look a the help file to get an overview of all options!

```
"Nicholas, Samuel" "Anderson, Jesse"
```

4 If you want to have the same output, execute `set.seed(123)` before generating the names

Random name sampling with `randomNames()` (2/2)

- **Sample first and/or last names**

```
set.seed(123)
randomNames(2, gender = c("Male", "Female"),
            return.complete.data = T)$first_name
"Samuel" "Jesse"
```

`return.complete.data=T` allows to return all the data as data.frame / data.table

```
set.seed(123)
randomNames(2, gender = c("Male", "Female"),
            return.complete.data = T)$last_name
"Nichols"  "Anderson"
```

Manipulate strings in R (1/4)

The `stringr` package provides key functionalities to manipulate strings:

- **Trim leading and trailing whitespaces**

```
str_trim("   this is a    test   ")
```

OUTPUT

```
[1] "this is a    test"
```

Install & load the relevant package:
`install.packages("stringr")`
`library(stringr)`

Non-leading or trailing white spaces will be retained

- **Pad strings with zeros to create strings of equal width**

```
str_pad(c(1,12,123,1234), width=4, side="left", pad="0")
```

OUTPUT

```
[1] "0001" "0012" "0123" "1234"
```

Has to be a single padding character

Numbers are converted to character vectors automatically

Manipulate strings in R (1/4)

The `stringr` package provides key functionalities to manipulate strings:

- **Trim leading and trailing whitespaces**

```
str_trim("   this is a    test   ")
```

OUTPUT

```
[1] "this is a    test"
```

Install & load the relevant package:
`install.packages("stringr")`
`library(stringr)`

1

Non-leading or trailing white spaces will be retained

2

- **Pad strings with zeros to create strings of equal width**

```
str_pad(c(1,12,123,1234), width=4, side="left", pad="0")
```

OUTPUT

```
[1] "0001" "0012" "0123" "1234"
```

Has to be a single padding character

4

Numbers are converted to character vectors automatically

3

Manipulate strings in R (1/4)

The `stringr` package provides key functionalities to manipulate strings:

- **Trim leading and trailing whitespaces**

```
str_trim("  this is a  test  ")
```

OUTPUT

```
[1] "this is a  test"
```

Install & load the relevant package:
`install.packages("stringr")`
`library(stringr)`

Non-leading or trailing white spaces will be retained

- **Pad strings with zeros to create strings of equal width**

```
str_pad(c(1,12,123,1234), width=4, side="left", pad="0")
```

OUTPUT

```
[1] "0001" "0012" "0123" "1234"
```

Has to be a single padding character

Numbers are converted to character vectors automatically

Manipulate strings in R (2/4)

- **Search for strings in a vector**

```
set.seeds(40)
Friends <-randomNames(3,
          return.complete.data=T)$first_name
print(Friends)
"Sarah"     "Caitlin"   "Savannah"

str_detect(friends, pattern="ah")
```

OUTPUT

```
[1] TRUE FALSE  TRUE
```

Returns a logical vector indicating whether the vector elements contains `ah`

Manipulate strings in R (3/4)

- **Replace string in a vector**

To replace **all** a's with @'s, use `str_replace_all()` (1)

```
str_replace(Friends, pattern="a", replacement="@")
```

OUTPUT

```
[1] "S@rah"    "C@itlin"  "S@vannah"
```

- **Caution: The `str_replace()`-function is case sensitive**

```
str_detect(Friends, pattern="sa")
```

There is no string with `sa` (2)

```
FALSE FALSE FALSE
```

```
str_detect(Friends, pattern="Sa")
```

There are two strings with `Sa` (3)

```
TRUE FALSE  TRUE
```

Manipulate strings in R (4/4)

Avoid mistakes by converting strings to lower/upper case:

- **Transform strings to lower case**

```
tolower(Friends)
```

OUTPUT

```
[1] "sarah"    "caitlin"  "savannah"
```

1 All characters are translated to lower case

- **Transform strings to upper case**

```
toupper(Friends)
```

OUTPUT

```
[1] "SARAH"    "CAITLIN"  "SAVANNAH"
```

2 All characters are translated to upper case

Advanced string manipulation: Use the `split()`-function

```
Example <- c("Craig, Sarah", "van den Ochtend, Jeroen")
```

- **Split strings by pattern**

```
unlist(str_split(Example,","))
```

1 The strings are split at the pattern ","

2 `str_split()` returns a list with the splitted strings in a single sublist. Use `unlist()` to get all strings in a single vector

OUTPUT

```
"Craig"   " Sarah"   "van den Ochtend"    " Jeroen"
```

3 Remember that `str_trim()` can be used to remove the space before and after words

- **Caution: Be careful with pattern**

```
unlist(str_split(Example," "))
```

OUTPUT

```
"Craig,"   "Sarah"   "van"  "den"   "Ochtend,"  "Jeroen"
```

Advanced string manipulation: Use the `split()`-function

```
Example <- c("Craig, Sarah", "van den Ochtend, Jeroen")
```

- **Split strings by pattern**

```
unlist(str_split(Example,","))
```

1 The strings are split at the pattern ","

2 `str_split()` returns a list with the splitted strings in a single sublist. Use `unlist()` to get all strings in a single vector

OUTPUT

```
"Craig"   " Sarah"   "van den Ochtend"    " Jeroen"
```

3 Remember that `str_trim()` can be used to remove the space before and after words

- **Caution: Be careful with pattern**

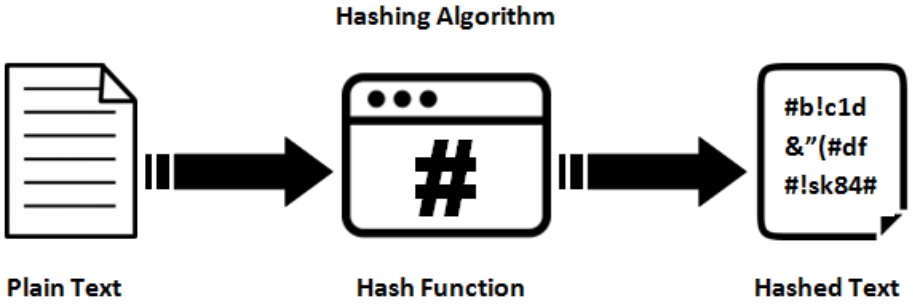
```
unlist(str_split(Example," "))
```

OUTPUT

```
"Craig,"   "Sarah"   "van"   "den"   "Ochtend,"   "Jeroen"
```

Advanced string manipulation: Anonymize customer ids

Hashing is a one-way function that changes a plain text to a unique digest that is irreversible.



A hash is a number generated from a string of text 1

- **Hash `Friends` with the `digest` library:**

There are various hashing algorithms available, which can be specified 2

```
sapply(Example,digest,algo="sha1",USE.NAMES = FALSE)
```

```
OUTPUT [1] "ba19461723212d9446aa8153a238585d27bfa7b5"
"f4c2e44c00e4a64bf610dc9c7e564c07689d3184"
```

https://www.ssl2buy.com/wiki/difference-between-hashing-and-encryption