

talk07 练习与作业

目录

0.1 练习和作业说明	1
0.2 talk07 内容回顾	1
0.3 练习与作业：用户验证	2
0.4 练习与作业 1：字符串操作	2
0.5 练习与作业 2：regular expression 正则表达式练习	8
0.6 练习与作业 3：探索题	14

0.1 练习和作业说明

将相关代码填写入以 “{r}” 标志的代码框中，运行并看到正确的结果；

完成后，用工具栏里的”Knit” 按键生成 PDF 文档；

将 PDF 文档改为：姓名-学号-talk07 作业.pdf，并提交到老师指定的平台/钉群。

0.2 talk07 内容回顾

1. string basics

- length
- uppercase, lowercase
- unite, separate

- string comparisons, sub string

2. regular expression

- detect patterns
- locate patterns
- extract patterns
- replace patterns

0.3 练习与作业：用户验证

请运行以下命令，验证你的用户名。

如你当前用户名不能体现你的真实姓名，请改为拼音后再运行本作业！

```
Sys.info()[["user"]]
```

```
## [1] "lucas"
```

```
Sys.getenv("HOME")
```

```
## [1] "/Users/lucas"
```

0.4 练习与作业 1：字符串操作

0.4.1 用 `stringr` 包实现以下操作

使用变量: `x <- c('weihua', 'chen');`

1. 每个 element/成员的长度
2. 每个成员首字母大写
3. 取每个成员的前两个字符

4. 合并为一个字符串，用 ‘,’ 间隔
5. 数一下每个成员中元音字母（vowel letter）的数量

```
## 代码写这里，并运行；
library(stringi)
library(stringr)

# Load the data
x =
  c('weihua', 'chen')

x_length =
  length(x)

x_upper =
  str_to_title(x)

x_sub =
  str_sub(x, 1, 2)

x_collapse =
  str_c(x, collapse = ', ')

x_count =
  str_count(x, '[aeiou]')

print(x_length)
```

```
## [1] 2
```

```
print(x_upper)
```

```
## [1] "Weihua" "Chen"
```

```
print(x_sub)
```

```
## [1] "we" "ch"
```

```
print(x_collapse)
```

```
## [1] "weihua, chen"
```

```
print(x_count)
```

```
## [1] 4 1
```

0.4.2 用 `mtcars` 变量作练习

1. 筛选出所有的奔驰车 (Mercedes-Benz);
2. 筛选出所有非奔驰车;
3. 处理行名, 将其中的品牌与车型分开。比如: Mazda RX4 Wag => 'Mazda', 'RX4 Wag'

```
## 代码写这里, 并运行;
```

```
# Load the package
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
```

```
## v dplyr      1.1.3      v purrr      1.0.2
```

```
## v forcats    1.0.0      v readr      2.1.4
```

```
## v ggplot2    3.4.3      v tibble     3.2.1
```

```
## v lubridate  1.9.2      v tidyr      1.3.0
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()    masks stats::lag()
```

```
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
```

```

# Sift through all the Mercedes-Benz cars
mtcars_Merc =
  mtcars %>%
    rownames_to_column('car') %>%
    filter(grepl('Merc', car)) %>%
    select(car, everything())

# Sift through all the non-Mercedes-Benz cars
mtcars_none_Merc =
  mtcars %>%
    rownames_to_column('car') %>%
    filter(!grepl('Merc', car)) %>%
    select(car, everything())

# Split the car names into brand and model
mtcars_split =
  mtcars %>%
    rownames_to_column('car') %>%
    separate(car, c('brand', 'model'), sep = ' ')

```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 3 rows [2, 4, 29].
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 1 rows [6].
```

```

# Print the results
print(mtcars_Merc)

```

```
##           car  mpg  cyl  disp  hp drat   wt  qsec vs am gear carb
## 1  Merc 240D 24.4    4 146.7  62 3.69 3.19 20.0  1  0    4    2
## 2  Merc 230 22.8    4 140.8  95 3.92 3.15 22.9  1  0    4    2
## 3  Merc 280 19.2    6 167.6 123 3.92 3.44 18.3  1  0    4    4
## 4  Merc 280C 17.8    6 167.6 123 3.92 3.44 18.9  1  0    4    4
## 5  Merc 450SE 16.4    8 275.8 180 3.07 4.07 17.4  0  0    3    3
```

```
## 6 Merc 450SL 17.3 8 275.8 180 3.07 3.73 17.6 0 0 3 3
## 7 Merc 450SLC 15.2 8 275.8 180 3.07 3.78 18.0 0 0 3 3
```

```
print(mtcars_none_Merc)
```

```
##          car  mpg  cyl  disp  hp  drat   wt   qsec vs  am gear carb
## 1      Mazda RX4 21.0   6 160.0 110 3.90 2.620 16.46 0  1    4    4
## 2      Mazda RX4 Wag 21.0   6 160.0 110 3.90 2.875 17.02 0  1    4    4
## 3      Datsun 710 22.8   4 108.0  93 3.85 2.320 18.61 1  1    4    1
## 4      Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44 1  0    3    1
## 5      Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02 0  0    3    2
## 6          Valiant 18.1   6 225.0 105 2.76 3.460 20.22 1  0    3    1
## 7          Duster 360 14.3   8 360.0 245 3.21 3.570 15.84 0  0    3    4
## 8      Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98 0  0    3    4
## 9      Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82 0  0    3    4
## 10     Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42 0  0    3    4
## 11          Fiat 128 32.4   4  78.7  66 4.08 2.200 19.47 1  1    4    1
## 12          Honda Civic 30.4   4  75.7  52 4.93 1.615 18.52 1  1    4    2
## 13          Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90 1  1    4    1
## 14          Toyota Corona 21.5   4 120.1  97 3.70 2.465 20.01 1  0    3    1
## 15      Dodge Challenger 15.5   8 318.0 150 2.76 3.520 16.87 0  0    3    2
## 16          AMC Javelin 15.2   8 304.0 150 3.15 3.435 17.30 0  0    3    2
## 17          Camaro Z28 13.3   8 350.0 245 3.73 3.840 15.41 0  0    3    4
## 18      Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05 0  0    3    2
## 19          Fiat X1-9 27.3   4  79.0  66 4.08 1.935 18.90 1  1    4    1
## 20          Porsche 914-2 26.0   4 120.3  91 4.43 2.140 16.70 0  1    5    2
## 21          Lotus Europa 30.4   4  95.1 113 3.77 1.513 16.90 1  1    5    2
## 22      Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.50 0  1    5    4
## 23          Ferrari Dino 19.7   6 145.0 175 3.62 2.770 15.50 0  1    5    6
## 24          Maserati Bora 15.0   8 301.0 335 3.54 3.570 14.60 0  1    5    8
## 25          Volvo 142E 21.4   4 121.0 109 4.11 2.780 18.60 1  1    4    2
```

```
print(mtcars_split)
```

##	brand	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
## 1	Mazda	RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
## 2	Mazda	RX4	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
## 3	Datsun	710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
## 4	Hornet	4	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
## 5	Hornet	Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
## 6	Valiant	<NA>	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
## 7	Duster	360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
## 8	Merc	240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
## 9	Merc	230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
## 10	Merc	280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
## 11	Merc	280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
## 12	Merc	450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
## 13	Merc	450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
## 14	Merc	450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
## 15	Cadillac	Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
## 16	Lincoln	Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
## 17	Chrysler	Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
## 18	Fiat	128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
## 19	Honda	Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
## 20	Toyota	Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
## 21	Toyota	Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
## 22	Dodge	Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
## 23	AMC	Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
## 24	Camaro	Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
## 25	Pontiac	Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
## 26	Fiat	X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
## 27	Porsche	914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
## 28	Lotus	Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
## 29	Ford	Pantera	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
## 30	Ferrari	Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6

```
## 31 Maserati      Bora 15.0    8 301.0 335 3.54 3.570 14.60 0 1    5    8
## 32      Volvo    142E 21.4    4 121.0 109 4.11 2.780 18.60 1 1    4    2
```

用 `str_c` 操作

为下面字符增加前缀和后缀，

```
x <- c("abc", NA)
```

使其最终结果为：

```
"|-abc-|" "-NA-|"
```

```
## 代码写这里，并运行；
# Load the package
library(tidyverse)

# Add prefix and suffix
x =
  c("abc", NA)

if(is.na(x[2])) {
  x[2] = 'NA'
}

x_add =
  str_c('|-', x, '-|')

# Print the results
print(x_add)
```

```
## [1] "|-abc-|" "-NA-|"
```

0.5 练习与作业 2: regular expression 正则表达式练习

0.5.1 用 starwars 变量作练习

注：需要先导入 tidyverse 包；

1. 选出所有 skin_color 包含为 white 的人，显示其 name, homeworld, species 和 skin_color；注意：有些人的 skin color 可为多个；
2. 打印出所有含有 ar 的名字；不区分大小写；

```
## 代码写这里，并运行；
# Load the package
library(tidyverse)

# Filtering data
white_characters =
  starwars %>%
    filter(str_detect(tolower(skin_color), "white")) %>%
    select(name, homeworld, species, skin_color)

ar_characters =
  starwars %>%
    filter(str_detect(tolower(name), "ar")) %>%
    select(name)

# Print the results
print(white_characters)
```

```
## # A tibble: 7 x 4
##   name          homeworld species  skin_color
##   <chr>         <chr>     <chr>   <chr>
## 1 R2-D2        Naboo     Droid   white, blue
## 2 Darth Vader  Tatooine  Human   white
## 3 R5-D4        Tatooine  Droid   white, red
## 4 Gasgano      Troiken   Xexto   white, blue
```

```
## 5 Yarael Poof Quermia   Quermian white
## 6 Shaak Ti      Shili   Togruta  red, blue, white
## 7 Grievous      Kalee    Kaleesh  brown, white
```

```
print(ar_characters)
```

```
## # A tibble: 19 x 1
##   name
##   <chr>
## 1 Darth Vader
## 2 Owen Lars
## 3 Beru Whitesun lars
## 4 Biggs Darklighter
## 5 Wilhuff Tarkin
## 6 Ackbar
## 7 Arvel Crynyd
## 8 Wicket Systri Warrick
## 9 Jar Jar Binks
## 10 Roos Tarpals
## 11 Quarsh Panaka
## 12 Darth Maul
## 13 Ben Quadinaros
## 14 Yarael Poof
## 15 Gregar Typho
## 16 Cliegg Lars
## 17 Luminara Unduli
## 18 Barriss Offee
## 19 Tarfful
```

0.5.2 用下面的 `vec` 变量作练习

```
vec <- c( "123", "abc", "wei555hua666" );
```

1. 找出含有数字的字符串；
2. 找出数字的位置；如果字符串含有多组数数字，只显示第一组；
3. 找出所有数字的位置；
4. 提取出找到的数字；如果字符串含有多组数数字，只提取第一组；
5. 提取所有的数字；
6. 将数字替换为 666；

```
## 代码写这里，并运行；
# Load the package
library(tidyverse)

# Find the first digit
vec =
  c( "123", "abc", "wei555hua666" )

vec <- c("123", "abc", "wei555hua666")

# 1. Finding strings containing numbers
strings_with_digits =
  grep("[0-9]",
    vec,
    value = TRUE)
print(strings_with_digits)

## [1] "123"          "wei555hua666"

# 2. Finding the position of the first digit
first_digit_positions =
  regexpr(
    "[0-9]",
    vec)
print(first_digit_positions)

## [1] 1 -1 4
```

```
## attr(,"match.length")
## [1] 1 -1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

```
# 3. Finding the position of all digits
```

```
all_digit_positions =
  regexpr(
    "[0-9]+",
    vec)
print(all_digit_positions)
```

```
## [1] 1 -1 4
## attr(,"match.length")
## [1] 3 -1 3
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

```
# 4. Extracting the first digit
```

```
first_digits =
  regmatches(
    vec,
    regexec(
      "[0-9]",
      vec))
print(first_digits)
```

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

```
## [[2]]  
## character(0)  
##  
## [[3]]  
## [1] "5"
```

```
# 5. Extracting all digits
```

```
all_digits =  
  regmatches(  
    vec,  
    regexec(  
      "[0-9]+",  
      vec))  
print(all_digits)
```

```
## [[1]]  
## [1] "123"  
##  
## [[2]]  
## character(0)  
##  
## [[3]]  
## [1] "555"
```

```
# 6. Replacing all digits with 666
```

```
vec_with_666 =  
  gsub(  
    "[0-9]+",  
    "666",  
    vec)  
print(vec_with_666)
```

```
## [1] "666"          "abc"          "wei666hua666"
```

0.6 练习与作业 3：探索题

0.6.1 序列分析

用序列: `seq <- "ATCTCGGCGCGCATCGCGTACGCTACTAGC"` 实现以下分析; 注: 可使用任何包:

1. 得到它的反向互补序列;
2. 计算它的 GC 含量, 用百分数表示;
3. 把它拆分成一个个 codon (即三个 nucleotide 形成一个 codon; 最后一个长度可以不为 3;

```
## 代码写这里, 并运行;
# Load the package
library(tidyverse)
library(stringr)

# Get the reverse complement
seq =
  "ATCTCGGCGCGCATCGCGTACGCTACTAGC"

# 1. Getting the reverse complement
complement =
  chartr("ATCG", "TAGC", seq)
reverse_complement =
  rev(complement)
print(reverse_complement)

## [1] "TAGAGCCGCGCGTAGCGCATGCGATGATCG"
```

```
# 2. Calculating the GC content
gc_content =
    round(
        (sum
         (str_count
          (seq, "G") +
          str_count(
              seq,
              "C"))) /
         nchar(seq)) * 100,
        2)
print(
    paste0(
        "GC content is: ",
        gc_content,
        "%."))
```

```
## [1] "GC content is: 63.33%."
```

```
# 3. 拆分成一个个 codon
codons =
    strsplit(
        seq,
        "(?<=\\G.{3})",
        perl = TRUE)[[1]]
print(codons)
```

```
## [1] "ATC" "TCG" "GCG" "CGC" "ATC" "GCG" "TAC" "GCT" "ACT" "AGC"
```

0.6.2 问答

问: `stringr::str_pad` 的作用是什么? 请举例回答

答:

`stringr::str_pad` 函数的作用是用指定的字符填充字符串，以使字符串达到指定的宽度。这可以在文本对齐等情况下非常有用。

以下是 `stringr::str_pad` 函数的一些示例用法：

```
library(stringr)

# Eg 1: Pad with zeros on the left to reach a width of 5
str_pad_1 =
  str_pad(
    "42",
    width = 5,
    pad = "0")
print(str_pad_1)
```

```
## [1] "00042"
```

```
# Result: "00042"

# Eg 2: Pad with spaces on the right to reach a width of 10
str_pad_2 =
  str_pad(
    "Hello",
    width = 10,
    pad = " ")
print(paste0(str_pad_2, "|"))
```

```
## [1] "      Hello|"
```

```
# Result: "Hello      |"

# Eg 3: Pad with dashes on both sides to reach a width of 10
str_pad_3 =
  str_pad(
```



```
    "abc",
    width = 6,
    pad = "-")
print(str_pad_3)
```

```
## [1] "---abc"
```

```
# Result: "--abc--"
```

```
# Eg 4: Pad with underscores on the left to reach a width of 8
```

```
str_pad_4 =
    str_pad(
        "string",
        width = 8,
        pad = "_")
print(str_pad_4)
```

```
## [1] "__string"
```

```
# Result: "string__"
```

```
# Eg 5: Pad with zeros on both sides to reach a width of 5
```

```
str_pad_5 =
    str_pad(
        "123",
        width = 5,
        pad = "0",
        side = "both")
print(str_pad_5)
```

```
## [1] "01230"
```

```
# 结果: "01230"
```

0.6.3 提取字符串中的 N 次重复字段

问：如何用正则表达式从字符串中提取任意长度为 2 字符的两次以上重复，比如：1212, abab, tata, 是 12 等的两次重复，898989 则是 89 的 3 次重复，下面的变量为输入：

```
c( "banana", "coconut", "1232323", "database" )
```

```
## 代码写这里，并运行；
# Load the package
library(tidyverse)

# Extracting repeated patterns
vec =
  c( "banana", "coconut", "1232323", "database" )

repeated_patterns =
  str_extract(
    vec,
    "(\\w{2})\\1+" )
print(repeated_patterns)
```

```
## [1] "anan"    "coco"    "232323" NA
```

0.6.4 正则表达式

设计一个正则表达式，可以完整识别所有以下格式的数字

```
123
123.45
0.124
-1.5
```

-0.2
+1.3
-11
-199.62

```
## 代码写这里，并运行；  
# Load the package  
library(tidyverse)  
  
# Designing a regular expression  
vec =  
  c(  
    "123",  
    "123.45",  
    "0.124",  
    "-1.5",  
    "-0.2",  
    "+1.3",  
    "-11",  
    "-199.62")  
  
regex =  
  "^[+-]?[0-9]+(\\.[0-9]+)?$"   
  
is_number =  
  str_detect(  
    vec,  
    regex)  
  
# Print the results  
print(is_number)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```