talk03 练习与作业

目录

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；

将生成的 PDF 改为：`姓名-学号-talk03 作业.pdf`，并提交到老师指定的平台/钉群。

0.2 talk03 内容回顾

- 二维表：`data.frame`, `tibble`
  - 声明
  - 操作
    - 增减行、列

    - 合并
  - 常用相关函数
    - `nrow`, `ncol`, `dim` , `str` , `head`, `tail`
  - `data.frame` 和 `tibble` 的不同
  - 高级技巧：
    - `with`, `within`
- IO
  - 系统自带函数
  - `readr` 带的函数
  - 不同格式的读取
  - 从网络、压缩文件读取

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] "mingyuwang"
```

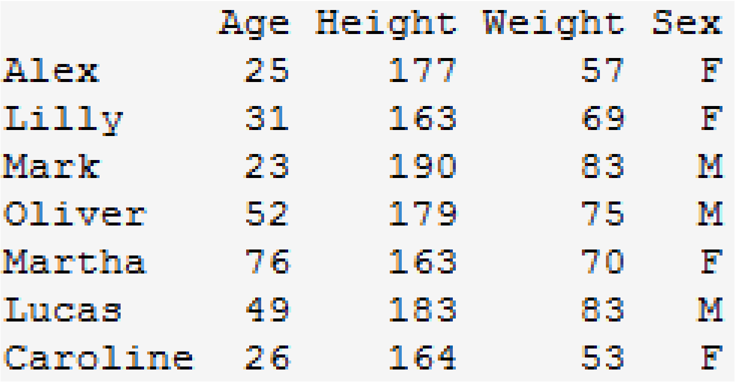
```
Sys.getenv("HOME")
```

```
## [1] "C:/Users/rhong/Documents"
```

0.4 练习与作业 1，`data.frame`

注：以下内容来自 https://www.r-exercises.com/。

- 生成下面的 `data.frame` 的前三列，之后再增加 `Sex` 这列

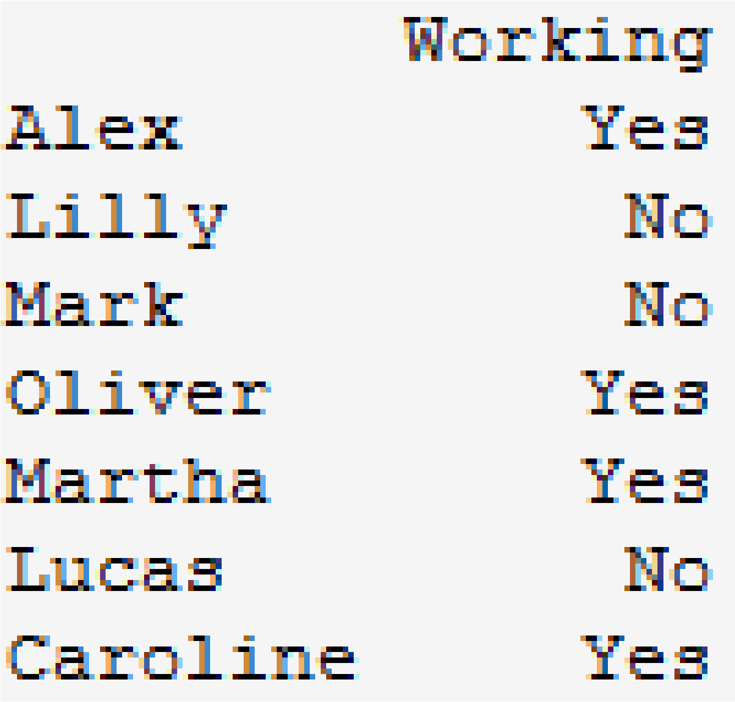


```
         Age Height Weight Sex
Alex      25    177     57   F
Lilly     31    163     69   F
Mark      23    190     83   M
Oliver    52    179     75   M
Martha    76    163     70   F
Lucas     49    183     83   M
Caroline  26    164     53   F
```

```
names <- c("Alex", "Lilly", "Mark", "Oliver", "Martha", "Lucas", "Caroline")
age <- c(25, 31, 23, 52, 76, 49, 26)
height <- c(177, 165, 180, 175, 160, 185, 170)
weight <- c(70, 50, 80, 60, 45, 90, 65)
sex <- c("F", "F", "M", "M", "F", "M", "F")
## 先生成前三列；
df1 <- data.frame(Age = age, Height = height, Weight = weight,
  row.names = names)
## 再插入第四列
df1 <- cbind(df1, sex)
## 显示最终结果
df1
```

```
##          Age Height Weight sex
## Alex      25    177     70   F
## Lilly     31    165     50   F
## Mark      23    180     80   M
## Oliver    52    175     60   M
## Martha    76    160     45   F
## Lucas     49    185     90   M
## Caroline  26    170     65   F
```

- 生成以下 `data.frame`，确保 `Working` 这列的类型是 `character`，而不是 `factor`



	Working
Alex	Yes
Lilly	No
Mark	No
Oliver	Yes
Martha	Yes
Lucas	No
Caroline	Yes

```r
working <- c("Yes", "No", "No", "Yes",  "Yes", "No", "Yes")
## 生成 data.frame
df1 <- data.frame(Working = working, row.names = names)
## 显示结果
df1
```

```
##          Working
## Alex         Yes
## Lilly         No
## Mark          No
## Oliver       Yes
## Martha       Yes
## Lucas         No
## Caroline     Yes
## 显示 Working 列的性质
with(df1, class(working))
```

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

- 检查系统自带变量 `state.center` 的内容，将其转化为 `data.frame`

```
## 代码写这里，并运行；
state.center
```

```
## $x
##  [1]  -86.7509 -127.2500 -111.6250  -92.2992 -119.7730 -105.5130  -72.3573
##  [8]  -74.9841  -81.6850  -83.3736 -126.2500 -113.9300  -89.3776  -86.0808
## [15]  -93.3714  -98.1156  -84.7674  -92.2724  -68.9801  -76.6459  -71.5800
## [22]  -84.6870  -94.6043  -89.8065  -92.5137 -109.3200  -99.5898 -116.8510
## [29]  -71.3924  -74.2336 -105.9420  -75.1449  -78.4686 -100.0990  -82.5963
## [36]  -97.1239 -120.0680  -77.4500  -71.1244  -80.5056  -99.7238  -86.4560
## [43]  -98.7857 -111.3300  -72.5450  -78.2005 -119.7460  -80.6665  -89.9941
## [50] -107.2560
##
## $y
##  [1] 32.5901 49.2500 34.2192 34.7336 36.5341 38.6777 41.5928 38.6777 27.8744
## [10] 32.3329 31.7500 43.5648 40.0495 40.0495 41.9358 38.4204 37.3915 30.6181
## [19] 45.6226 39.2778 42.3645 43.1361 46.3943 32.6758 38.3347 46.8230 41.3356
## [28] 39.1063 43.3934 39.9637 34.4764 43.1361 35.4195 47.2517 40.2210 35.5053
## [37] 43.9078 40.9069 41.5928 33.6190 44.3365 35.6767 31.3897 39.1063 44.2508
## [46] 37.5630 47.4231 38.4204 44.5937 43.0504
```

```
class(state.center)
```

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

```r
data.frame(state.center)
```

```
##            x       y
## 1   -86.7509 32.5901
## 2  -127.2500 49.2500
## 3  -111.6250 34.2192
## 4   -92.2992 34.7336
## 5  -119.7730 36.5341
## 6  -105.5130 38.6777
## 7   -72.3573 41.5928
## 8   -74.9841 38.6777
## 9   -81.6850 27.8744
## 10  -83.3736 32.3329
## 11 -126.2500 31.7500
## 12 -113.9300 43.5648
## 13  -89.3776 40.0495
## 14  -86.0808 40.0495
## 15  -93.3714 41.9358
## 16  -98.1156 38.4204
## 17  -84.7674 37.3915
## 18  -92.2724 30.6181
## 19  -68.9801 45.6226
## 20  -76.6459 39.2778
## 21  -71.5800 42.3645
## 22  -84.6870 43.1361
## 23  -94.6043 46.3943
## 24  -89.8065 32.6758
## 25  -92.5137 38.3347
## 26 -109.3200 46.8230
## 27  -99.5898 41.3356
## 28 -116.8510 39.1063
## 29  -71.3924 43.3934
## 30  -74.2336 39.9637
## 31 -105.9420 34.4764
## 32  -75.1449 43.1361
## 33  -78.4686 35.4195
## 34 -100.0990 47.2517
## 35  -82.5963 40.2210
## 36  -97.1239 35.5053
## 37 -120.0680 43.9078
## 38  -77.4500 40.9069
## 39  -71.1244 41.5928
## 40  -80.5056 33.6190
## 41  -99.7238 44.3365
## 42  -86.4560 35.6767
## 43  -98.7857 31.3897
## 44 -111.3300 39.1063
## 45  -72.5450 44.2508
## 46  -78.2005 37.5630
## 47 -119.7460 47.4231
## 48  -80.6665 38.4204
## 49  -89.9941 44.5937
## 50 -107.2560 43.0504
```

- **生成一个 50 行 * 5 列的 `matrix`，将其行名改为：row__i 格式，其中 i 为当前的行号，比如 row__1, row__2 等**

```
## 代码写这里，并运行；
m <- matrix(sample(50 * 5), nrow = 50, ncol = 5)
rownames(m) <- paste0("row_", 1:50)
m
```

```
##       [,1] [,2] [,3] [,4] [,5]
## row_1   12  129  235    2   61
## row_2   52   28  211  137   51
## row_3   248  224    5   80  206
## row_4     1  109   66  186  217
## row_5   153   29  111    6  160
## row_6    90  195  218  200  232
## row_7   159   65  181  117  190
## row_8   126  179   40  239  156
## row_9   106   47   81  135    9
## row_10  157  185  241  246   59
## row_11   32   11  175  167  220
## row_12    8   38   89   99  113
## row_13   37  243  141   31  140
## row_14   30   77   82  114   74
## row_15  233  203   48  119   78
## row_16  132  209  101  204  249
## row_17  221  138   50  112   76
## row_18  130   57   44   54  116
## row_19   27   84   67   20   86
## row_20  166  110   34  189   93
## row_21  198  171  182   95   18
## row_22  149  191  127   15  213
## row_23  147   53    4  226   70
## row_24  158   56  164  180   49
## row_25  161   73  210  145  193
## row_26  216   39   46  244  219
## row_27  229   98  227   22   72
## row_28  199  139   36  163  118
## row_29  247  173  234  177  231
## row_30    7  103  197  170  183
## row_31   60  212   21  151  102
## row_32  201  208  214  162  225
## row_33   64  152  230  120   85
## row_34  107   41   25  143  250
## row_35  207   92  242  146  238
## row_36   97  105  236   91  194
## row_37   79   43   55   96  121
## row_38   10   24   62  228  223
## row_39  100  222   35  237  122
## row_40   17  178  128   16  205
## row_41  165  150   94   19  196
## row_42  148  115  192   23   26
## row_43  174    3  188   42  187
## row_44  133  202  136   33  125
## row_45   75  176  155   45   88
## row_46  104  142   68  184  123
## row_47  215   58  240  169  134
## row_48   87  108   63   71  131
## row_49  245   69   83  168   14
## row_50   13  144  154  172  124
```

- **使用系统自带变量 VADeaths，做如下练习：**
- 检查 VADeaths 的类型，如果不是 data.frame，则转换之；
- 添加新的一列，取名 Total，其值为每行的总合
- 调整列的顺序，将 Total 变为第一列。

```
## 代码写这里，并运行；
class(VADeaths)
```

```
## [1] "matrix" "array"
```

```
df_deaths <- data.frame(VADeaths)
df_deaths$Total <- rowSums(df_deaths)
df_deaths <- df_deaths[, c(ncol(df_deaths), 1:(ncol(df_deaths) - 1))]
df_deaths
```

```
##       Total Rural.Male Rural.Female Urban.Male Urban.Female
## 50-54  44.2       11.7          8.7       15.4          8.4
## 55-59  67.7       18.1         11.7       24.3         13.6
## 60-64 103.5       26.9         20.3       37.0         19.3
## 65-69 161.6       41.0         30.9       54.6         35.1
## 70-74 241.4       66.0         54.3       71.1         50.0
```

- **用系统自带的 `swiss` 数据做练习：**
- 取子集，选取第 1，2，3，10，11，12 and 13 行，第 `Examination`, `Education` 和 `Infant.Mortality` 列；
- 将 `Sarine` 行 `Infant.Mortality` 列的值改为 NA；
- 增加一列，命名为 `Mean`，其值为当前行的平均值；

```
## 代码写这里，并运行；
(df_swiss <- swiss[c(1, 2, 3, 10, 11, 12, 13),
  c("Examination", "Education", "Infant.Mortality")])
```

```
##              Examination Education Infant.Mortality
## Courtelary            15        12             22.2
## Delemont               6         9             22.2
## Franches-Mnt           5         5             20.2
## Sarine                16        13             24.4
## Veveyse               14         6             24.5
## Aigle                 21        12             16.5
## Aubonne               14         7             19.1
```

```
df_swiss["Sarine", "Infant.Mortality"] <- NA
df_swiss
```

```
##              Examination Education Infant.Mortality
## Courtelary             15          12               22.2
## Delemont                6           9               22.2
## Franches-Mnt            5           5               20.2
## Sarine                 16          13                 NA
## Veveyse                14           6               24.5
## Aigle                  21          12               16.5
## Aubonne                14           7               19.1
```

```
df_swiss$Mean <- rowMeans(df_swiss)
head(df_swiss)
```

```
##              Examination Education Infant.Mortality     Mean
## Courtelary            15        12             22.2 16.40000
## Delemont               6         9             22.2 12.40000
## Franches-Mnt           5         5             20.2 10.06667
## Sarine                16        13               NA       NA
## Veveyse               14         6             24.5 14.83333
## Aigle                 21        12             16.5 16.50000
```

- 将下面三个变量合并生成一个 **data.frame**

Id <- LETTERS

x <- seq(1,43,along.with=Id)

y <- seq(-20,0,along.with=Id)

```
## 代码写这里，并运行；
Id <- LETTERS
x <- seq(1, 43, along.with = Id)
y <- seq(-20, 0, along.with = Id)
data.frame(Id, x, y)
```

```
##    Id     x     y
## 1   A  1.00 -20.0
## 2   B  2.68 -19.2
## 3   C  4.36 -18.4
## 4   D  6.04 -17.6
## 5   E  7.72 -16.8
## 6   F  9.40 -16.0
## 7   G 11.08 -15.2
## 8   H 12.76 -14.4
## 9   I 14.44 -13.6
## 10  J 16.12 -12.8
## 11  K 17.80 -12.0
## 12  L 19.48 -11.2
## 13  M 21.16 -10.4
## 14  N 22.84  -9.6
## 15  O 24.52  -8.8
## 16  P 26.20  -8.0
## 17  Q 27.88  -7.2
## 18  R 29.56  -6.4
## 19  S 31.24  -5.6
## 20  T 32.92  -4.8
## 21  U 34.60  -4.0
## 22  V 36.28  -3.2
## 23  W 37.96  -2.4
## 24  X 39.64  -1.6
## 25  Y 41.32  -0.8
## 26  Z 43.00   0.0
```

问： `seq` 函数中的 `along.with` 参数的意义是什么？请举例说明。

答：seq 函数中的 along.with 参数的意义是指定序列的长度，即 along.with 参数指定的变量的长度。

```
## 代码写这里，并运行；
seq(1, 19, along.with = 1:10)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

```
seq(1, 19, along.with = 1:19)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
```

- **提供代码，合并以下两个 data.frame**

```
> df1 的内容
Id Age
1 14
2 12
3 15
4 10

>df2 的内容
Id Sex Code
1 F a
2 M b
3 M c
4 F d
```

合并之后的结果：

```
> M
Id Age Sex Code
1 14 F a
2 12 M b
3 15 M c
4 10 F d
```

```
## 代码写这里，并运行；
library(dplyr)
```

```
## Warning: 程辑包'dplyr'是用R版本4.1.3 来建造的
```

```
##
## 载入程辑包：'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
Id <- 1:4
Age <- c(14, 12, 15, 10)
Sex <- c("F", "M", "M", "F")
Code <- c("a", "b", "c", "d")
df1 <- data.frame(Id, Age)
df2 <- data.frame(Id, Sex, Code)
M <- left_join(df1, df2, by = "Id")
M
```

```
##   Id Age Sex Code
## 1  1  14   F    a
## 2  2  12   M    b
## 3  3  15   M    c
## 4  4  10   F    d
```

- 从上面的 `data.frame` 中删除 `code` 列

```
## 代码写这里，并运行；
(M <- M[, -4])
```

```
##   Id Age Sex
## 1  1  14   F
## 2  2  12   M
## 3  3  15   M
## 4  4  10   F
```

- **练习，回答代码中的问题**

```
## 1. 生成一个10 行2 列的data.frame
df3 <- data.frame( data = 1:10, group = c("A","B") )
## 2. 增加一列，其长度是1，可以吗？
cbind(df3, newcol = 1);
## 3. 增加一列，其长度是10，可以吗？
cbind(df3, newcol = 1:10);
## 4. 增加一列，其长度是2，可以吗？
cbind(df3, newcol = 1:2);
## 5. 增加一列，其长度是3，可以吗？
cbind(df3, newcol = 1:3);
```

答：2. 可以，3. 可以，4. 可以，5. 不可以。增加的列的长度必须与原 data.frame 的行数是整数倍。

0.5 练习与作业 2，`tibble`

- **运行以下代码，生成一个新的 `tibble`：**

```r
## 如果系统中没有 lubridate 包，则安装：
if (!require("lubridate")){
  chooseCRANmirror();
  install.packages("lubridate");
}
```

```
## 载入需要的程辑包：lubridate
```

```
## Warning: 程辑包'lubridate'是用R版本4.1.3 来建造的
```

```
##
## 载入程辑包：'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(lubridate);

if (!require("tibble")){
  chooseCRANmirror();
  install.packages("tibble");
}
```

```
## 载入需要的程辑包：tibble
```

```
## Warning: 程辑包'tibble'是用R版本4.1.3 来建造的
```

```r
library(tibble);

tibble(
  a = lubridate::now() + runif(1e3) * 86400,
  b = lubridate::today() + runif(1e3) * 30,
  c = 1:1e3,
  d = runif(1e3),
  e = sample(letters, 1e3, replace = TRUE)
)
```

```
## # A tibble: 1,000 x 5
##    a                   b              c      d e
##    <dttm>              <date>     <int>  <dbl> <chr>
##  1 2022-09-10 05:54:58 2022-09-11     1 0.424  p
##  2 2022-09-10 13:50:03 2022-09-17     2 0.989  b
##  3 2022-09-09 22:29:07 2022-09-11     3 0.0877 d
##  4 2022-09-09 20:08:58 2022-10-01     4 0.154  a
##  5 2022-09-10 02:48:02 2022-09-22     5 0.797  r
##  6 2022-09-10 10:02:32 2022-09-23     6 0.220  a
##  7 2022-09-10 03:20:27 2022-10-01     7 0.467  n
##  8 2022-09-10 13:51:06 2022-09-13     8 0.0739 i
##  9 2022-09-10 05:49:20 2022-10-04     9 0.765  g
## 10 2022-09-10 01:21:40 2022-09-20    10 0.276  n
## # ... with 990 more rows
```

从中可以看出，`tibble` 支持一些细分数据类型，包括：

- `<dttm>`
- `<date>`

等；

- 生成一个如下的 `tibble`，完成以下任务：

```
df <- tibble(
  x = runif(5),
  y = rnorm(5)
)
```

任务：

- 取一列，比如 x 这一列，得到一个 tibble；
- 取一列，比如 y 这一列，得到一个 vector；

```
## 代码写这里，并运行；
df <- tibble(
  x = runif(5),
  y = rnorm(5)
)
df["x"]
```

```
## # A tibble: 5 x 1
##       x
##   <dbl>
## 1 0.522
## 2 0.426
## 3 0.926
## 4 0.563
## 5 0.443
```

```
df$y
```

```
## [1]  1.15456672 -0.60203375 -0.06326746 -0.21816910 -0.88190168
```

- 用 tibble 函数创建一个新的空表，并逐行增加一些随机的数据，共增加三行：

```
## 代码写这里，并运行；
## 新 tibble, with defined columns ... 创建表头
tb <- tibble( name = character(), age = integer(), salary = double() );
```

```
## 增加三行随机数据;
tb <- add_row(tb, name = sample(letters, 3),
  age = sample(50, 3), salary = sample(1000, 3))
tb
```

```
## # A tibble: 3 x 3
##   name    age salary
##   <chr> <int>  <dbl>
## 1 k        31    510
## 2 h        18    277
## 3 z        21    627
```

- ** 请解释为什么下面第一行代码能够运行成功，但第二个不行？**

这个可以：

`data.frame(a = 1:6, b = LETTERS[1:2]);`

但下面这个不行：

`tibble(a = 1:6, b = LETTERS[1:2]);`

问：为什么？tibble 循环的规则是什么？

答：Only values of size one are recycled.

- **`attach` 和 `detach`:**

问：这个两个函数的用途是什么？请用 `iris` 这个系统自带变量举例说明。

答：attach() 函数将一个数据框绑定到当前的搜索路径中，这样就可以直接使用数据框中的变量名而不用加上数据框的名字。detach() 函数将一个数据框从当前的搜索路径中分离出来。

```
try(head(Septal.Length))
```

```
## Error in head(Septal.Length) : 找不到对象'Septal.Length'
```

```
# Sepal.Length 是 iris 数据框中的变量名，但是在当前的搜索路径中没有，所以不能直接使用
attach(iris)
# attach() 函数将 iris 数据框绑定到当前的搜索路径中，之后就可以直接访问 Sepal.Length 这个变
head(Sepal.Length)
```

```
## [1] 5.1 4.9 4.7 4.6 5.0 5.4
```

```
detach(iris)
```

- **使用内置变量 airquality：**
- 检查它是否是 tibble；
- 如果不是，转化为 tibble；

```
## 代码写这里，并运行；
is_tibble(airquality)
```

```
## [1] FALSE
```

```
airquality <- as_tibble(airquality)
is_tibble(airquality)
```

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

- **问：tibble::enframe 函数的用途是什么？请举例说明：**

答：enframe() 函数将一个向量转换为一个数据框，其中第一列是向量的名字，第二列是向量的值。

```
head(iris$Sepal.Length, n = 10)
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

```
enframe(iris$Sepal.Length[1:10])
```

```
## # A tibble: 10 x 2
##     name value
##    <int> <dbl>
##  1     1   5.1
##  2     2   4.9
##  3     3   4.7
##  4     4   4.6
##  5     5   5
##  6     6   5.4
##  7     7   4.6
##  8     8   5
##  9     9   4.4
## 10    10   4.9
```

- **简述 `tibble` 相比 `data.frame` 的优势？并用实例展示**

答：tibble 显示每一列的数据类型，而 data.frame 不显示。tibble 按顺序计算列，而 data.frame 同时计算所有列。tibble 取子集将得到一个 tibble，而 data.frame 取子集可能得到 vector。data.frame 存在部分匹配的问题，而 tibble 不存在。

```
## 代码写这里，并运行；
df1 <- data.frame(aaa = 1:6, b = LETTERS[1:6])
tib1 <- tibble(aaa = 1:6, b = LETTERS[1:6])
# dataframe 子集变成了向量类型而不是 data.frame
df1[, 1]
```

```
## [1] 1 2 3 4 5 6
```

```
# tibble 子集仍然是 tibble 类型
tib1[, 1]
```

```
## # A tibble: 6 x 1
##     aaa
##   <int>
## 1     1
## 2     2
## 3     3
## 4     4
## 5     5
## 6     6
```

```
# tibble 按顺序计算列; dataframe 同时计算所有列.
tibble(a = 1:6, b = a * 2)
```

```
## # A tibble: 6 x 2
##         a     b
##   <int> <dbl>
## 1     1     2
## 2     2     4
## 3     3     6
## 4     4     8
## 5     5    10
## 6     6    12
```

```
try(data.frame(a = 1:6, b = a * 2))
```

```
## Error in data.frame(a = 1:6, b = a * 2) : 找不到对象'a'
```

0.6 练习与作业 3：IO

- **提供代码，正确读取以下文件：**

注：数据在当前目录下的 data/ 子目录里

- Table0.txt
- Table1.txt
- Table2.txt
- Table3.txt
- Table4.txt
- Table5.txt
- Table6.txt
- states1.csv
- states2.csv

注 2：每个文件读取需要提供两种方法，一种是利用系统自带函数，另一种是 readr 包的函数；

```
## 用系统自带函数，并显示读取的内容；
read.table("data/Table0.txt", header = FALSE)
```

```
##          V1 V2  V3 V4 V5
## 1      Alex 25 177 57  F
## 2     Lilly 31 163 69  F
## 3      Mark 23 190 83  M
## 4    Oliver 52 179 75  M
## 5    Martha 76 163 70  F
## 6     Lucas 49 183 83  M
## 7 Caroline 26 164 53  F
```

```
read.table("data/Table1.txt", header = TRUE)
```

```
##        Name Age Height Weight Sex
## 1      Alex  25    177     57   F
## 2     Lilly  31    163     69   F
## 3      Mark  23    190     83   M
## 4    Oliver  52    179     75   M
## 5    Martha  76    163     70   F
## 6     Lucas  49    183     83   M
## 7 Caroline  26    164     53   F
```

```
read.table("data/Table2.txt", header = TRUE, quote = "/", skip = 1)
```

```
##        Name Age Height Weight Sex
## 1      Alex  25    177     57   F
## 2     Lilly  31    163     69   F
## 3      Mark  23    190     83   M
## 4    Oliver  52    179     75   M
## 5    Martha  76    163     70   F
## 6     Lucas  49    183     83   M
## 7 Caroline  26    164     53   F
```

```
read.table("data/Table3.txt", header = TRUE, skip = 1,
  na.strings = c("", "NA", "--", "*", "**"))
```

```
##        Name Age Height Weight Sex
## 1      Alex  25    177     57   F
## 2     Lilly  31     NA     69   F
## 3      Mark  NA    190     83   M
## 4    Oliver  52    179     75   M
## 5    Martha  76     NA     70   F
## 6     Lucas  49    183     NA   M
## 7 Caroline  26    164     53   F
```

```r
# Table4.txt 身高数据 Height 中的值有逗号需要去掉
table4 <- read.table("data/Table4.txt", header = TRUE,
  na.strings = c("", "NA", "--", "*", "**"))
within(table4, Height <- as.numeric(gsub(",", "", Height)))
```

```
##       Name Age Height Weight Sex
## 1     Alex  25    177     57   F
## 2    Lilly  31     NA     69   F
## 3     Mark  NA    190     83   M
## 4   Oliver  52    179     75   M
## 5   Martha  76     NA     70   F
## 6    Lucas  49    183     NA   M
## 7 Caroline  26    164     53   F
```

```r
table5 <- read.table("data/Table5.txt", header = TRUE, sep = ";",
  na.strings = c("", "NA", "--", "*", "**"))
table5$Height <- as.numeric(gsub(",", "", table5$Height))
table5
```

```
##       Name Age Height Weight Sex
## 1     Alex  25    177     57   F
## 2    Lilly  31     NA     69   F
## 3     Mark  NA    190     83   M
## 4   Oliver  52    179     75   M
## 5   Martha  76     NA     70   F
## 6    Lucas  49    183     NA   M
## 7 Caroline  26    164     53   F
```

```r
# Table6.txt 中 @ 为注释符号，需要去掉
read.table("data/Table6.txt", header = TRUE,
  comment.char = "@", skip = 1)
```

```
##         Name Age Height Weight Sex
## 1       Alex 25     177     57   F
## 2      Lilly 31     163     69   F
## 3       Mark 23     190     83   M
## 4     Oliver 52     179     75   M
## 5     Martha 76     163     70   F
## 6      Lucas 49     183     83   M
## 7   Caroline 26     164     53   F
## 8       Alex 25     177     57   F
## 9      Lilly 31     163     69   F
## 10      Mark 23     190     83   M
## 11    Oliver 52     179     75   M
## 12    Martha 76     163     70   F
## 13     Lucas 49     183     83   M
## 14  Caroline 26     164     53   F
## 15      Alex 25     177     57   F
## 16     Lilly 31     163     69   F
## 17      Mark 23     190     83   M
## 18    Oliver 52     179     75   M
## 19    Martha 76     163     70   F
## 20     Lucas 49     183     83   M
## 21  Caroline 26     164     53   F
## 22      Alex 25     177     57   F
## 23     Lilly 31     163     69   F
## 24      Mark 23     190     83   M
## 25    Oliver 52     179     75   M
## 26    Martha 76     163     70   F
## 27     Lucas 49     183     83   M
## 28  Caroline 26     164     53   F
## 29      Alex 25     177     57   F
## 30     Lilly 31     163     69   F
## 31      Mark 23     190     83   M
## 32    Oliver 52     179     75   M
## 33    Martha 76     163     70   F
## 34     Lucas  49    183     83   M
## 35  Caroline  26    164     53   F
## 36      Alex  25    177     57   F
## 37     Lilly  31    163     69   F
## 38      Mark  23    190     83   M
## 39    Oliver  52    179     75   M
## 40    Martha  76    163     70   F
## 41     Lucas  49    183     83   M
## 42  Caroline  26    164     53   F
## 43      Alex  25    177     57   F
## 44     Lilly  31    163     69   F
## 45      Mark  23    190     83   M
## 46    Oliver  52    179     75   M
## 47    Martha  76    163     70   F
## 48     Lucas  49    183     83   M
## 49  Caroline  26    164     53   F
## 50      Alex  25    177     57   F
## 51     Lilly  31    163     69   F
## 52      Mark  23    190     83   M
## 53    Oliver  52    179     75   M
## 54    Martha  76    163     70   F
## 55     Lucas  49    183     83   M
## 56  Caroline  26    164     53   F
## 57      Alex  25    177     57   F
## 58     Lilly  31    163     69   F
## 59      Mark  23    190     83   M
## 60    Oliver  52    179     75   M
## 61    Martha  76    163     70   F
## 62     Lucas  49    183     83   M
## 63  Caroline  26    164     53   F
## 64      Alex  25    177     57   F
## 65     Lilly  31    163     69   F
## 66      Mark  23    190     83   M
## 67    Oliver  52    179     75   M
## 68    Martha  76    163     70   F
## 69     Lucas  49    183     83   M
## 70  Caroline  26    164     53   F
## 71      Alex  25    177     57   F
## 72     Lilly  31    163     69   F
## 73      Mark  23    190     83   M
## 74    Oliver  52    179     75   M
## 75    Martha  76    163     70   F
## 76     Lucas  49    183     83   M
## 77  Caroline  26    164     53   F
## 78      Alex  25    177     57   F
## 79     Lilly  31    163     69   F
## 80      Mark  23    190     83   M
## 81    Oliver  52    179     75   M
## 82    Martha  76    163     70   F
## 83     Lucas  49    183     83   M
## 84  Caroline  26    164     53   F
## 85      Alex  25    177     57   F
## 86     Lilly  31    163     69   F
## 87      Mark  23    190     83   M
## 88    Oliver  52    179     75   M
## 89    Martha  76    163     70   F
## 90     Lucas  49    183     83   M
## 91  Caroline  26    164     53   F
## 92      Alex  25    177     57   F
## 93     Lilly  31    163     69   F
## 94      Mark  23    190     83   M
## 95    Oliver  52    179     75   M
## 96    Martha  76    163     70   F
## 97     Lucas  49    183     83   M
## 98  Caroline  26    164     53   F
## 99      Alex  25    177     57   F
## 100    Lilly  31    163     69   F
## 101     Mark  23    190     83   M
## 102   Oliver  52    179     75   M
## 103   Martha  76    163     70   F
## 104    Lucas  49    183     83   M
## 105 Caroline  26    164     53   F
```

```r
read.csv("data/states1.csv", header = TRUE)
```

```
##                  X Population Income Illiteracy Life.Exp Murder HS.Grad Frost
## 1         Alabama       3615   3624        2.1    69.05   15.1    41.3    20
## 2          Alaska        365   6315        1.5    69.31   11.3    66.7   152
## 3         Arizona       2212   4530        1.8    70.55    7.8    58.1    15
## 4        Arkansas       2110   3378        1.9    70.66   10.1    39.9    65
## 5      California      21198   5114        1.1    71.71   10.3    62.6    20
## 6        Colorado       2541   4884        0.7    72.06    6.8    63.9   166
## 7     Connecticut       3100   5348        1.1    72.48    3.1    56.0   139
## 8        Delaware        579   4809        0.9    70.06    6.2    54.6   103
## 9         Florida       8277   4815        1.3    70.66   10.7    52.6    11
## 10        Georgia       4931   4091        2.0    68.54   13.9    40.6    60
## 11         Hawaii        868   4963        1.9    73.60    6.2    61.9     0
## 12          Idaho        813   4119        0.6    71.87    5.3    59.5   126
## 13       Illinois      11197   5107        0.9    70.14   10.3    52.6   127
## 14        Indiana       5313   4458        0.7    70.88    7.1    52.9   122
## 15           Iowa       2861   4628        0.5    72.56    2.3    59.0   140
## 16         Kansas       2280   4669        0.6    72.58    4.5    59.9   114
## 17       Kentucky       3387   3712        1.6    70.10   10.6    38.5    95
## 18      Louisiana       3806   3545        2.8    68.76   13.2    42.2    12
## 19          Maine       1058   3694        0.7    70.39    2.7    54.7   161
## 20       Maryland       4122   5299        0.9    70.22    8.5    52.3   101
## 21  Massachusetts       5814   4755        1.1    71.83    3.3    58.5   103
## 22       Michigan       9111   4751        0.9    70.63   11.1    52.8   125
## 23      Minnesota       3921   4675        0.6    72.96    2.3    57.6   160
## 24    Mississippi       2341   3098        2.4    68.09   12.5    41.0    50
## 25       Missouri       4767   4254        0.8    70.69    9.3    48.8   108
## 26        Montana        746   4347        0.6    70.56    5.0    59.2   155
## 27       Nebraska       1544   4508        0.6    72.60    2.9    59.3   139
## 28         Nevada        590   5149        0.5    69.03   11.5    65.2   188
## 29  New Hampshire        812   4281        0.7    71.23    3.3    57.6   174
## 30     New Jersey       7333   5237        1.1    70.93    5.2    52.5   115
## 31     New Mexico       1144   3601        2.2    70.32    9.7    55.2   120
## 32       New York      18076   4903        1.4    70.55   10.9    52.7    82
## 33 North Carolina       5441   3875        1.8    69.21   11.1    38.5    80
## 34   North Dakota        637   5087        0.8    72.78    1.4    50.3   186
## 35           Ohio      10735   4561        0.8    70.82    7.4    53.2   124
## 36       Oklahoma       2715   3983        1.1    71.42    6.4    51.6    82
## 37         Oregon       2284   4660        0.6    72.13    4.2    60.0    44
## 38   Pennsylvania      11860   4449        1.0    70.43    6.1    50.2   126
## 39   Rhode Island        931   4558        1.3    71.90    2.4    46.4   127
## 40 South Carolina       2816   3635        2.3    67.96   11.6    37.8    65
## 41   South Dakota        681   4167        0.5    72.08    1.7    53.3   172
## 42      Tennessee       4173   3821        1.7    70.11   11.0    41.8    70
## 43          Texas      12237   4188        2.2    70.90   12.2    47.4    35
## 44           Utah       1203   4022        0.6    72.90    4.5    67.3   137
## 45        Vermont        472   3907        0.6    71.64    5.5    57.1   168
## 46       Virginia       4981   4701        1.4    70.08    9.5    47.8    85
## 47     Washington       3559   4864        0.6    71.72    4.3    63.5    32
## 48  West Virginia       1799   3617        1.4    69.48    6.7    41.6   100
## 49      Wisconsin       4589   4468        0.7    72.48    3.0    54.5   149
## 50        Wyoming        376   4566        0.6    70.29    6.9    62.9   173
##      Area
## 1   50708
## 2  566432
## 3  113417
## 4   51945
## 5  156361
## 6  103766
## 7    4862
## 8    1982
## 9   54090
## 10  58073
## 11   6425
## 12  82677
## 13  55748
## 14  36097
## 15  55941
## 16  81787
## 17  39650
## 18  44930
## 19  30920
## 20   9891
## 21   7826
## 22  56817
## 23  79289
## 24  47296
## 25  68995
## 26 145587
## 27  76483
## 28 109889
## 29   9027
## 30   7521
## 31 121412
## 32  47831
## 33  48798
## 34  69273
## 35  40975
## 36  68782
## 37  96184
## 38  44966
## 39   1049
## 40  30225
## 41  75955
## 42  41328
## 43 262134
## 44  82096
## 45   9267
## 46  39780
## 47  66570
## 48  24070
## 49  54464
## 50  97203
```

```
# states2.csv 中 分隔符是 ";"
stats2 <- read.csv("data/states2.csv", header = TRUE, sep = ";")
stats2[, 4:7] <- lapply(stats2[, 4:7], gsub,
    pattern = ",", replacement = "\\.") %>%
  lapply(as.numeric)
stats2
```

```
##                  X Population Income Illiteracy Life.Exp Murder HS.Grad Frost
## 1          Alabama       3615   3624        2.1    69.05   15.1    41.3    20
## 2           Alaska        365   6315        1.5    69.31   11.3    66.7   152
## 3          Arizona       2212   4530        1.8    70.55    7.8    58.1    15
## 4         Arkansas       2110   3378        1.9    70.66   10.1    39.9    65
## 5       California      21198   5114        1.1    71.71   10.3    62.6    20
## 6         Colorado       2541   4884        0.7    72.06    6.8    63.9   166
## 7      Connecticut       3100   5348        1.1    72.48    3.1    56.0   139
## 8         Delaware        579   4809        0.9    70.06    6.2    54.6   103
## 9          Florida       8277   4815        1.3    70.66   10.7    52.6    11
## 10         Georgia       4931   4091        2.0    68.54   13.9    40.6    60
## 11          Hawaii        868   4963        1.9    73.60    6.2    61.9     0
## 12           Idaho        813   4119        0.6    71.87    5.3    59.5   126
## 13        Illinois      11197   5107        0.9    70.14   10.3    52.6   127
## 14         Indiana       5313   4458        0.7    70.88    7.1    52.9   122
## 15           Iowa       2861   4628        0.5    72.56    2.3    59.0   140
## 16         Kansas       2280   4669        0.6    72.58    4.5    59.9   114
## 17       Kentucky       3387   3712        1.6    70.10   10.6    38.5    95
## 18      Louisiana       3806   3545        2.8    68.76   13.2    42.2    12
## 19          Maine       1058   3694        0.7    70.39    2.7    54.7   161
## 20       Maryland       4122   5299        0.9    70.22    8.5    52.3   101
## 21  Massachusetts       5814   4755        1.1    71.83    3.3    58.5   103
## 22       Michigan       9111   4751        0.9    70.63   11.1    52.8   125
## 23      Minnesota       3921   4675        0.6    72.96    2.3    57.6   160
## 24    Mississippi       2341   3098        2.4    68.09   12.5    41.0    50
## 25       Missouri       4767   4254        0.8    70.69    9.3    48.8   108
## 26        Montana        746   4347        0.6    70.56    5.0    59.2   155
## 27       Nebraska       1544   4508        0.6    72.60    2.9    59.3   139
## 28         Nevada        590   5149        0.5    69.03   11.5    65.2   188
## 29  New Hampshire        812   4281        0.7    71.23    3.3    57.6   174
## 30     New Jersey       7333   5237        1.1    70.93    5.2    52.5   115
## 31     New Mexico       1144   3601        2.2    70.32    9.7    55.2   120
## 32       New York      18076   4903        1.4    70.55   10.9    52.7    82
## 33 North Carolina       5441   3875        1.8    69.21   11.1    38.5    80
## 34   North Dakota        637   5087        0.8    72.78    1.4    50.3   186
## 35           Ohio      10735   4561        0.8    70.82    7.4    53.2   124
## 36       Oklahoma       2715   3983        1.1    71.42    6.4    51.6    82
## 37         Oregon       2284   4660        0.6    72.13    4.2    60.0    44
## 38   Pennsylvania      11860   4449        1.0    70.43    6.1    50.2   126
## 39   Rhode Island        931   4558        1.3    71.90    2.4    46.4   127
## 40 South Carolina       2816   3635        2.3    67.96   11.6    37.8    65
## 41   South Dakota        681   4167        0.5    72.08    1.7    53.3   172
## 42      Tennessee       4173   3821        1.7    70.11   11.0    41.8    70
## 43          Texas      12237   4188        2.2    70.90   12.2    47.4    35
## 44           Utah       1203   4022        0.6    72.90    4.5    67.3   137
## 45        Vermont        472   3907        0.6    71.64    5.5    57.1   168
## 46       Virginia       4981   4701        1.4    70.08    9.5    47.8    85
## 47     Washington       3559   4864        0.6    71.72    4.3    63.5    32
## 48  West Virginia       1799   3617        1.4    69.48   6.7    41.6   100
## 49      Wisconsin       4589   4468        0.7    72.48   3.0    54.5   149
## 50        Wyoming        376   4566        0.6    70.29   6.9    62.9   173
##       Area
## 1    50708
## 2   566432
## 3   113417
## 4    51945
## 5   156361
## 6   103766
## 7     4862
## 8     1982
## 9    54090
## 10   58073
## 11    6425
## 12   82677
## 13   55748
## 14   36097
## 15   55941
## 16   81787
## 17   39650
## 18   44930
## 19   30920
## 20    9891
## 21    7826
## 22   56817
## 23   79289
## 24   47296
## 25   68995
## 26  145587
## 27   76483
## 28  109889
## 29    9027
## 30   7521
## 31 121412
## 32  47831
## 33  48798
## 34  69273
## 35  40975
## 36  68782
## 37  96184
## 38  44966
## 39   1049
## 40  30225
## 41  75955
## 42  41328
## 43 262134
## 44  82096
## 45   9267
## 46  39780
## 47  66570
## 48  24070
## 49  54464
## 50  97203
```

```
## 用 readr 包的函数读取，并显示读取的内容;
if (!require("readr")){
  chooseCRANmirror()
  install.packages("readr")
}
```

```
## 载入需要的程辑包：readr
```

```
## Warning: 程辑包'readr'是用R版本4.1.3 来建造的
```

```
library(readr)
# 读取 Table0.txt 分隔符有空格和制表符
read_table("data/Table0.txt", col_names = FALSE)
```

```
## 
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_character(),
##   X2 = col_double(),
##   X3 = col_double(),
##   X4 = col_double(),
##   X5 = col_character()
## )
```

```
## # A tibble: 7 x 5
##   X1           X2    X3    X4 X5   
##   <chr>     <dbl> <dbl> <dbl> <chr>
## 1 Alex         25   177    57 F    
## 2 Lilly        31   163    69 F    
## 3 Mark         23   190    83 M    
## 4 Oliver       52   179    75 M    
## 5 Martha       76   163    70 F    
## 6 Lucas        49   183    83 M    
## 7 Caroline     26   164    53 F    
```

```
# 读取 Table1.txt, 注意这里的 col_names = TRUE
read_table("data/Table1.txt", col_names = TRUE)
```

```
## 
## -- Column specification --------------------------------------------------------
## cols(
##   Name = col_character(),
##   Age = col_double(),
##   Height = col_double(),
##   Weight = col_double(),
##   Sex = col_character()
## )
```

```
## # A tibble: 7 x 5
##   Name       Age Height Weight Sex
##   <chr>    <dbl>  <dbl>  <dbl> <chr>
## 1 Alex        25    177     57 F
## 2 Lilly       31    163     69 F
## 3 Mark        23    190     83 M
## 4 Oliver      52    179     75 M
## 5 Martha      76    163     70 F
## 6 Lucas       49    183     83 M
## 7 Caroline    26    164     53 F
```

```
# 读取 Table2.txt
table2 <- read_table("data/Table2.txt", col_names = TRUE, skip = 1)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Name = col_character(),
##   Age = col_double(),
##   Height = col_double(),
##   Weight = col_double(),
##   Sex = col_character()
## )
```

```
table2[, c(1,5)] <- lapply(table2[, c(1, 5)],
  gsub, pattern = "/", replacement = "")
table2
```

```
## # A tibble: 7 x 5
##   Name        Age Height Weight Sex
##   <chr>    <dbl>  <dbl>  <dbl> <chr>
## 1 Alex        25    177     57 F
## 2 Lilly       31    163     69 F
## 3 Mark        23    190     83 M
## 4 Oliver      52    179     75 M
## 5 Martha      76    163     70 F
## 6 Lucas       49    183     83 M
## 7 Caroline    26    164     53 F
```

```
# 读取 Table3.txt
read_table("data/Table3.txt", col_names = TRUE, skip = 1,
  na = c("", "NA", "--", "*", "**"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Name = col_character(),
##   Age = col_double(),
##   Height = col_double(),
##   Weight = col_double(),
##   Sex = col_character()
## )
```

```
## # A tibble: 7 x 5
##   Name        Age Height Weight Sex
##   <chr>    <dbl>  <dbl>  <dbl> <chr>
## 1 Alex        25    177     57 F
## 2 Lilly       31     NA     69 F
## 3 Mark        NA    190     83 M
## 4 Oliver      52    179     75 M
## 5 Martha      76     NA     70 F
## 6 Lucas       49    183     NA M
## 7 Caroline    26    164     53 F
```

```
# 读取 Table4.txt
read_table("data/Table4.txt", col_names = TRUE,
  na = c("", "NA", "--", "*", "**"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Name = col_character(),
##   Age = col_double(),
##   Height = col_number(),
##   Weight = col_double(),
##   Sex = col_character()
## )

## # A tibble: 7 x 5
##   Name       Age Height Weight Sex
##   <chr>    <dbl>  <dbl>  <dbl> <chr>
## 1 Alex        25    177     57 F
## 2 Lilly       31     NA     69 F
## 3 Mark        NA    190     83 M
## 4 Oliver      52    179     75 M
## 5 Martha      76     NA     70 F
## 6 Lucas       49    183     NA M
## 7 Caroline    26    164     53 F
```

```
# 读取 Table5.txt
read_delim("data/Table5.txt", col_names = TRUE, delim = ";",
  na = c("", "NA", "--", "*", "**"))
```

```
## Rows: 7 Columns: 5

## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (2): Name, Sex
## dbl (2): Age, Weight
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 7 x 5
##   Name        Age Height Weight Sex
##   <chr>    <dbl>  <dbl>  <dbl> <chr>
## 1 Alex         25    177     57 F
## 2 Lilly        31     NA     69 F
## 3 Mark         NA    190     83 M
## 4 Oliver       52    179     75 M
## 5 Martha       76     NA     70 F
## 6 Lucas        49    183     NA M
## 7 Caroline     26    164     53 F
```

```
# 读取 Table6.txt
read_table("data/Table6.txt", col_names = TRUE, skip = 1,
  comment = "@")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Name = col_character(),
##   Age = col_double(),
##   Height = col_double(),
##   Weight = col_double(),
##   Sex = col_character()
## )
```

```
## # A tibble: 105 x 5
##    Name        Age Height Weight Sex
##    <chr>    <dbl>  <dbl>  <dbl> <chr>
##  1 Alex        25    177     57 F
##  2 Lilly       31    163     69 F
##  3 Mark        23    190     83 M
##  4 Oliver      52    179     75 M
##  5 Martha      76    163     70 F
##  6 Lucas       49    183     83 M
##  7 Caroline    26    164     53 F
##  8 Alex        25    177     57 F
##  9 Lilly       31    163     69 F
## 10 Mark        23    190     83 M
## # ... with 95 more rows
```

```
# 读取 states1.csv
read_csv("data/states1.csv", col_names = TRUE)
```

```
## New names:
## Rows: 50 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): ...1 dbl (8): Population, Income, Illiteracy, Life Exp, Murder, HS Grad,
## Frost, Area
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
## # A tibble: 50 x 9
##    ...1        Population Income Illiteracy Life E~1 Murder HS Gr~2 Frost   Area
##    <chr>            <dbl>  <dbl>      <dbl>    <dbl>  <dbl>   <dbl> <dbl>  <dbl>
##  1 Alabama           3615   3624        2.1     69.0   15.1    41.3    20  50708
##  2 Alaska             365   6315        1.5     69.3   11.3    66.7   152 566432
##  3 Arizona           2212   4530        1.8     70.6    7.8    58.1    15 113417
##  4 Arkansas          2110   3378        1.9     70.7   10.1    39.9    65  51945
##  5 California       21198   5114        1.1     71.7   10.3    62.6    20 156361
##  6 Colorado          2541   4884        0.7     72.1    6.8    63.9   166 103766
##  7 Connecticut       3100   5348        1.1     72.5    3.1    56      139   4862
##  8 Delaware           579   4809        0.9     70.1    6.2    54.6    103   1982
##  9 Florida           8277   4815        1.3     70.7   10.7    52.6     11  54090
## 10 Georgia           4931   4091        2       68.5   13.9    40.6     60  58073
## # ... with 40 more rows, and abbreviated variable names 1: `Life Exp`,
## #   2: `HS Grad`
```

```
# 读取 states2.csv . colon 转换成点，使用分号作为分隔符
read_delim("data/states2.csv", col_names = TRUE,
  locale = locale(decimal_mark = ","), delim = ";")
```

```
## New names:
## Rows: 50 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: ";" chr
## (1): ...1 dbl (8): Population, Income, Illiteracy, Life Exp, Murder, HS Grad,
## Frost, Area
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
## # A tibble: 50 x 9
##    ...1        Population Income Illiteracy Life E~1 Murder HS Gr~2 Frost   Area
##    <chr>            <dbl>  <dbl>      <dbl>    <dbl>  <dbl>   <dbl> <dbl>  <dbl>
##  1 Alabama           3615   3624        2.1     69.0   15.1    41.3     20  50708
##  2 Alaska             365   6315        1.5     69.3   11.3    66.7    152 566432
##  3 Arizona           2212   4530        1.8     70.6    7.8    58.1     15 113417
##  4 Arkansas          2110   3378        1.9     70.7   10.1    39.9     65  51945
##  5 California       21198   5114        1.1     71.7   10.3    62.6     20 156361
##  6 Colorado          2541   4884        0.7     72.1    6.8    63.9    166 103766
##  7 Connecticut       3100   5348        1.1     72.5    3.1    56      139   4862
##  8 Delaware           579   4809        0.9     70.1    6.2    54.6    103   1982
##  9 Florida           8277   4815        1.3     70.7   10.7    52.6     11  54090
## 10 Georgia           4931   4091        2       68.5   13.9    40.6     60  58073
## # ... with 40 more rows, and abbreviated variable names 1: `Life Exp`,
## #   2: `HS Grad`
```