talk05 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 PDF 文档改为：`姓名-学号-talk05 作业.pdf`，并提交到老师指定的平台/钉群。

0.2 Talk05 内容回顾

- dplyr 、tidyr (超级强大的数据处理) part 1
  - 长宽数据转换
  - dplyr 几个重要函数

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！

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

```
## [1] "sicheng.wu"
```

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

```
## [1] "/home/vkorpela"
```

0.4 练习与作业 1：dplyr 练习

0.4.1 使用 mouse.tibble 变量做统计

- 每个染色体（或 scaffold）上每种基因类型的数量、平均长度、最大和最小长度，挑出最长和最短的基因
- 去掉含有 500 以下基因的染色体（或 scaffold），按染色体（或 scaffold）、数量高 -> 低进行排序

```r
## 代码写这里，并运行；
library(tidyverse)
```

```
## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1

## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
mouse.tibble <- read_tsv("../data/talk04/mouse_genes_biomart_sep2018.txt")
```

```
## Rows: 138532 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (5): Gene stable ID, Transcript stable ID, Protein stable ID, Transcript...
## dbl (1): Transcript length (including UTRs and CDS)
##
## 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.
```

```
# 预处理出基因数超过 500 的染色体或 scaffold
CHR_list <- mouse.tibble %>%
  select(
    CHR = `Chromosome/scaffold name`,
    GENE_ID = `Gene stable ID`
  ) %>%
  group_by(CHR) %>%
  summarise(count = n_distinct(GENE_ID)) %>%
  filter(count > 500)
CHR_list <- unique(CHR_list$CHR)

# 筛选
mouse.selected <- mouse.tibble %>%
  select(
    CHR = `Chromosome/scaffold name`,
    TYPE = `Transcript type`,
    GENE_ID = `Gene stable ID`,
    GENE_LEN = `Transcript length (including UTRs and CDS)`
  ) %>%
  group_by(CHR, TYPE) %>%
  summarize(
    gene_count = n_distinct(GENE_ID),
    mean_len = mean(GENE_LEN),
    max_len = max(GENE_LEN),
    min_len = min(GENE_LEN)
  ) %>%
  filter(CHR %in% CHR_list) %>%
  arrange(CHR, desc(gene_count))
```

```
## `summarise()` has grouped output by 'CHR'. You can override using the `.groups`
## argument.
```

```
mouse.selected
```

```
## # A tibble: 521 x 6
## # Groups:   CHR [21]
##    CHR   TYPE                    gene_count mean_len max_len min_len
##    <chr> <chr>                        <int>    <dbl>   <dbl>   <dbl>
##  1 1     protein_coding                1200    2700.   40378      75
##  2 1     retained_intron                645    1748.    8483     230
##  3 1     processed_pseudogene           627     728.    4530      30
##  4 1     TEC                            479    2241.    8163     133
##  5 1     processed_transcript           462     951.    7640      65
##  6 1     lincRNA                        347    1207.    9720     154
##  7 1     nonsense_mediated_decay        314    1844.   10770     284
##  8 1     antisense                      224    1236.    7928      78
##  9 1     miRNA                          128      98.0    442      53
## 10 1     snRNA                          105     113.     191      55
## # ... with 511 more rows
```

0.4.2 使用 grades 变量做练习

1. 装入 grades 变量;

library(dplyr); grades <- read_tsv(file = "data/talk05/grades.txt");

```r
grades <- read_tsv("../data/talk05/grades.txt")
```

```
## Rows: 9 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (2): name, course
## dbl (1): grade
##
## 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.
```

2. 尝试使用 spread 和 gather 函数将其变宽后再变长；

```r
## 代码写这里，并运行；
(grades.spreaded <- grades %>%
  spread(course, grade, fill = NA))
```

```
## # A tibble: 3 x 6
##   name         Bioinformatics Chemistry Chinese English Microbiology
##   <chr>                 <dbl>     <dbl>   <dbl>   <dbl>        <dbl>
## 1 Kang Ning               100        76      20      NA           NA
## 2 Weihua Chen              99        NA      NA      99           89
## 3 Zhi Liu                  NA        NA      69      50          100
```

```r
(grades.gathered <- grades.spreaded %>%
  gather("course", "grade", -name))
```

```
## # A tibble: 15 x 3
##    name        course         grade
##    <chr>       <chr>          <dbl>
##  1 Kang Ning   Bioinformatics   100
##  2 Weihua Chen Bioinformatics    99
##  3 Zhi Liu     Bioinformatics    NA
##  4 Kang Ning   Chemistry         76
##  5 Weihua Chen Chemistry         NA
##  6 Zhi Liu     Chemistry         NA
##  7 Kang Ning   Chinese           20
##  8 Weihua Chen Chinese           NA
##  9 Zhi Liu     Chinese           69
## 10 Kang Ning   English           NA
## 11 Weihua Chen English           99
## 12 Zhi Liu     English           50
## 13 Kang Ning   Microbiology      NA
## 14 Weihua Chen Microbiology      89
## 15 Zhi Liu     Microbiology     100
```

3. 研究并使用 `tidyr` 包里的 `pivot_longer` 和 `pivot_wider` 函数对 `grades` 变量进行宽长转换；

```
## 代码写这里，并运行；
(grades.spreaded2 <- grades %>%
  pivot_wider(
    names_from = course,
    values_from = grade,
    values_fill = NA
  ))
```

```
## # A tibble: 3 x 6
##   name        Microbiology English Chinese Bioinformatics Chemistry
##   <chr>              <dbl>   <dbl>   <dbl>          <dbl>     <dbl>
## 1 Zhi Liu              100      50      69             NA        NA
## 2 Weihua Chen           89      99      NA             99        NA
## 3 Kang Ning             NA      NA      20            100        76
```

```
(grades.gathered2 <- grades.spreaded2 %>%
  pivot_longer(
    cols = !name,
    names_to = "course",
    values_to = "grade"
  ))
```

```
## # A tibble: 15 x 3
##    name         course         grade
##    <chr>        <chr>          <dbl>
##  1 Zhi Liu      Microbiology     100
##  2 Zhi Liu      English           50
##  3 Zhi Liu      Chinese           69
##  4 Zhi Liu      Bioinformatics    NA
##  5 Zhi Liu      Chemistry         NA
##  6 Weihua Chen Microbiology       89
##  7 Weihua Chen English            99
##  8 Weihua Chen Chinese            NA
##  9 Weihua Chen Bioinformatics     99
## 10 Weihua Chen Chemistry          NA
## 11 Kang Ning   Microbiology       NA
## 12 Kang Ning   English            NA
## 13 Kang Ning   Chinese            20
## 14 Kang Ning   Bioinformatics    100
## 15 Kang Ning   Chemistry          76
```

4. 使用 pivot_longer 时，有时会产生 na 值，如何使用此函数的参数去除带 na 的行？

```
## 代码写这里，并运行；
(grades.gathered3 <- grades.spreaded2 %>%
  pivot_longer(
    cols = !name,
    names_to = "course",
    values_to = "grade",
    values_drop_na = TRUE
  ))
```

```
## # A tibble: 9 x 3
##   name         course         grade
##   <chr>        <chr>          <dbl>
## 1 Zhi Liu      Microbiology     100
## 2 Zhi Liu      English           50
## 3 Zhi Liu      Chinese           69
## 4 Weihua Chen Microbiology       89
## 5 Weihua Chen English            99
## 6 Weihua Chen Bioinformatics     99
## 7 Kang Ning    Chinese           20
## 8 Kang Ning    Bioinformatics   100
## 9 Kang Ning    Chemistry         76
```

5. 以下代码有什么作用？

```
grades %>% complete( name, course )
```

答：补全 grades 中所有可能的 `name` 列和 `course` 列的组合，对于表中没有取值的组合，用 `NA` 填充。

0.4.3 使用 grades2 变量做练习

首先，用下面命令生成 grades2 变量：

```
grades2 <- tibble( "Name" = c("Weihua Chen", "Mm Hu", "John Doe", "Jane Doe",
                             "Warren Buffet", "Elon Musk", "Jack Ma"),
                  "Occupation" = c("Teacher", "Student", "Teacher", "Student",
                                   rep( "Entrepreneur", 3 ) ),
                  "English" = sample( 60:100, 7 ),
                  "ComputerScience" = sample(80:90, 7),
                  "Biology" = sample( 50:100, 7),
                  "Bioinformatics" = sample( 40:90, 7)
                  );
```

然后统计：1. 每个人最差的学科和成绩分别是什么？2. 哪个职业的平均成绩最好？3. 每个职业的最佳学科分别是什么（按平均分排序)???

```
## 代码写这里，并运行；
(grades2 <- tibble(
  "Name" = c("Weihua Chen", "Mm Hu", "John Doe", "Jane Doe", "Warren Buffet", "Elon Mus
  "Occupation" = c("Teacher", "Student", "Teacher", "Student", rep( "Entrepreneur", 3))
  "English" = sample(60:100, 7),
  "ComputerScience" = sample(80:90, 7),
  "Biology" = sample(50:100, 7),
  "Bioinformatics" = sample(40:90, 7)
))
```

```
## # A tibble: 7 x 6
##   Name          Occupation   English ComputerScience Biology Bioinformatics
##   <chr>         <chr>          <int>           <int>   <int>          <int>
## 1 Weihua Chen   Teacher           75              82      71             40
## 2 Mm Hu         Student           74              88      59             82
## 3 John Doe      Teacher           88              85      90             63
## 4 Jane Doe      Student           63              89      93             45
## 5 Warren Buffet Entrepreneur      90              84      53             74
## 6 Elon Musk     Entrepreneur      98              90      91             73
## 7 Jack Ma       Entrepreneur      69              87      68             87
```

```
# 每个人最差的学科和成绩是什么？
grades2 %>%
  pivot_longer(
    cols = !Name:Occupation,
    names_to = "Course",
    values_to = "Grade",
    values_drop_na = TRUE
  ) %>%
  group_by(Name) %>%
  filter(Grade == min(Grade)) %>%
  summarize(
    worst_course = Course,
    worst_grade = Grade
  )
```

```
## # A tibble: 7 x 3
##   Name           worst_course   worst_grade
##   <chr>          <chr>                <int>
## 1 Elon Musk      Bioinformatics          73
## 2 Jack Ma        Biology                 68
## 3 Jane Doe       Bioinformatics          45
## 4 John Doe       Bioinformatics          63
## 5 Mm Hu          Biology                 59
## 6 Warren Buffet Biology                  53
## 7 Weihua Chen    Bioinformatics          40
```

```
# 哪个职业的平均成绩最好？
grades2 %>%
  pivot_longer(
    cols = !Name:Occupation,
    names_to = "Course",
    values_to = "Grade",
    values_drop_na = TRUE
  ) %>%
  group_by(Occupation) %>%
  summarise(ave_grade = mean(Grade)) %>%
  arrange(desc(ave_grade))
```

```
## # A tibble: 3 x 2
##   Occupation   ave_grade
##   <chr>            <dbl>
## 1 Entrepreneur      80.3
## 2 Teacher           74.2
## 3 Student           74.1
```

```
# 每个职业的最佳学科分别是什么？
grades2 %>%
  pivot_longer(
    cols = !Name:Occupation,
    names_to = "Course",
    values_to = "Grade",
    values_drop_na = TRUE
  ) %>%
  group_by(Occupation, Course) %>%
  summarise(ave_grade = mean(Grade)) %>%
  arrange(Occupation, desc(ave_grade)) %>%
  group_by(Occupation) %>%
  filter(ave_grade == max(ave_grade))
```

```
## `summarise()` has grouped output by 'Occupation'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 3 x 3
## # Groups:   Occupation [3]
##   Occupation   Course          ave_grade
##   <chr>        <chr>               <dbl>
## 1 Entrepreneur ComputerScience      87
## 2 Student      ComputerScience      88.5
## 3 Teacher      ComputerScience      83.5
```

0.4.4 使用 starwars 变量做计算

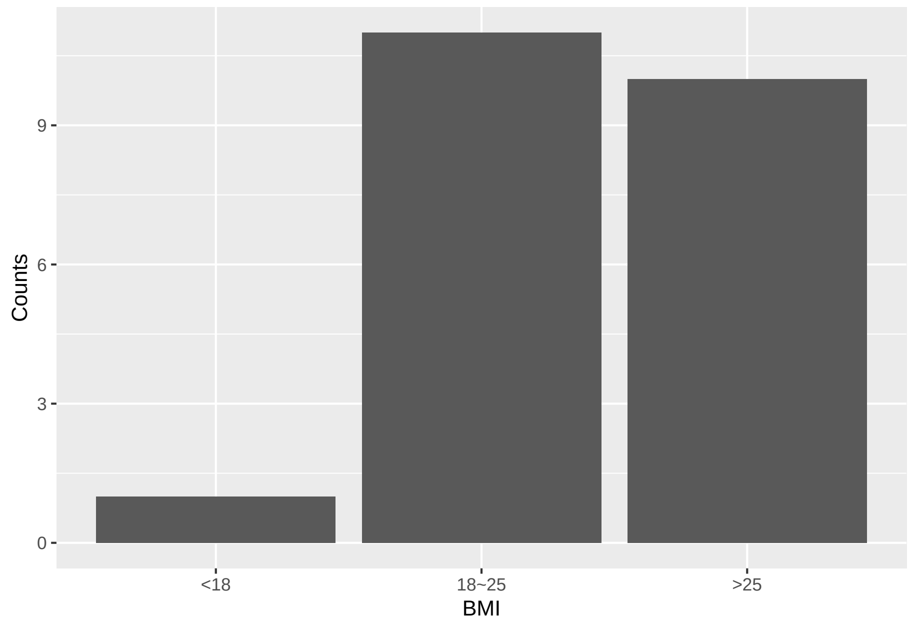
1. 计算每个人的 BMI；
2. 挑选出肥胖（BMI >= 30）的人类，并且只显示其 name，sex 和 homeworld；

```
## 代码写这里，并运行；
sw.bmi <- starwars %>%
  mutate(bmi = mass / ((height / 100) ^ 2))

sw.bmi %>%
  filter(bmi >= 30 & species == "Human") %>%
  summarise(name, sex, homeworld)
```

```
## # A tibble: 3 x 3
##   name             sex   homeworld
##   <chr>            <chr> <chr>
## 1 Darth Vader      male  Tatooine
## 2 Owen Lars        male  Tatooine
## 3 Jek Tono Porkins male  Bestine IV
```

3. 挑选出所有人类；
4. 按 BMI 将他们分为三组，<18, 18~25, >25，统计每组的人数，并用 barplot 进行展示；注意：展示时三组的按 BMI 从小到大排序；
5. 改变排序方式，按每组人数从小到大排序；

```
## 代码写这里，并运行；
sw.human <- sw.bmi %>%
  filter(species == "Human" & !is.na(bmi)) %>%
  mutate(
    bmi_group = cut(
      bmi,
      c(0, 18, 25, Inf),
      labels = c("<18", "18~25", ">25")
    )
  ) %>%
  group_by(bmi_group) %>%
  summarise(counts = n())

plot1 <-
  ggplot(
    data = sw.human,
    aes(
      x = bmi_group,
      y = counts
    )
  ) +
  geom_col() +
  xlab("BMI") +
  ylab("Counts")
plot1
```



6. 查看 starwars 的 films 列，它有什么特点？data.frame 可以实现类似的功能吗？

答：films 列的每一项都是一个字符串列表，data.frame 实现不了类似的功能。

7. 为 starwars 增加一列，用于统计每个角色在多少部电影中出现。

```
## 代码写这里，并运行；
(starwars.expanded <- starwars %>%
  mutate(film_count = lengths(films)))
```

```
## # A tibble: 87 x 15
##    name         height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##    <chr>         <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
##  1 Luke Skywa~     172    77 blond   fair    blue         19 male  mascu~ Tatooi~
##  2 C-3PO           167    75 <NA>    gold    yellow      112 none  mascu~ Tatooi~
##  3 R2-D2            96    32 <NA>    white,~ red          33 none  mascu~ Naboo
##  4 Darth Vader    202   136 none    white   yellow     41.9 male  mascu~ Tatooi~
##  5 Leia Organa    150    49 brown   light   brown      19   fema~ femin~ Aldera~
##  6 Owen Lars      178   120 brown,~ light   blue       52   male  mascu~ Tatooi~
##  7 Beru White~    165    75 brown   light   blue       47   fema~ femin~ Tatooi~
##  8 R5-D4           97    32 <NA>    white,~ red        NA   none  mascu~ Tatooi~
##  9 Biggs Dark~    183    84 black   light   brown      24   male  mascu~ Tatooi~
## 10 Obi-Wan Ke~    182    77 auburn~ fair    blue-g~    57   male  mascu~ Stewjon
## # ... with 77 more rows, 5 more variables: species <chr>, films <list>,
## #   vehicles <list>, starships <list>, film_count <int>, and abbreviated
## #   variable names 1: hair_color, 2: skin_color, 3: eye_color, 4: birth_year,
## #   5: homeworld
```

```
starwars.expanded$film_count
```

```
##  [1] 5 6 7 4 5 3 3 1 1 6 3 2 5 4 1 3 3 1 5 5 3 1 1 2 1 2 1 1 1 1 1 3 1 2 1 1 1 2
## [39] 1 1 2 1 1 3 1 1 1 3 3 3 2 2 2 1 3 2 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1 2 1 1 2
## [77] 1 1 2 2 1 1 1 1 1 1 3
```