talk05 练习与作业

目录

0.1 练习和作业说明 1
0.2 Talk05 内容回顾 1
0.3 练习与作业：用户验证 1
0.4 练习与作业 1：dplyr 练习 2

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
  - pipe
  - 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！

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

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

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

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

```
getwd(); ## 显示当前工作目录
```

```
## [1] "/Users/wchen/workspace/华中科技大学/06 - 学生培养/060 - 教学/062-本科教学/R for
```

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

挑战题（可选做）：

实现上述目标（即：去掉少于 500 基因的染色体、排序、并统计）时不使用中间变量；

```
## 代码写这里，并运行；
library(dplyr);library(readr)
```

```
##
## Attaching package: 'dplyr'

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

```
##     filter, lag

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

```
mouse.tibble<- read_delim("../data/talk04/mouse_genes_biomart_sep2018.txt" ,delim = "\t
##chr_order <- c(1:19,"X", "Y")
str(mouse.tibble);
```

```
## spc_tbl_ [138,532 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Gene stable ID                             : chr [1:138532] "ENSMUSG00000064372" "
##  $ Transcript stable ID                       : chr [1:138532] "ENSMUST00000082423" "
##  $ Protein stable ID                          : chr [1:138532] NA NA "ENSMUSP00000081
##  $ Transcript length (including UTRs and CDS): num [1:138532] 67 67 1144 69 519 ...
##  $ Transcript type                            : chr [1:138532] "Mt_tRNA" "Mt_tRNA" "p
##  $ Chromosome/scaffold name                   : chr [1:138532] "MT" "MT" "MT" "MT" ..
##  - attr(*, "spec")=
##   .. cols(
##   ..   `Gene stable ID` = col_character(),
##   ..   `Transcript stable ID` = col_character(),
##   ..   `Protein stable ID` = col_character(),
##   ..   `Transcript length (including UTRs and CDS)` = col_double(),
##   ..   `Transcript type` = col_character(),
##   ..   `Chromosome/scaffold name` = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
##mouse.tibble <-m_g %>% filter( `Chromosome/scaffold name` );
##( chr.stats <- mouse.tibble %>% count( `Chromosome/scaffold name` ) %>% arrange(-n) )
##( chr.stats <- mouse.tibble %>% count( `Chromosome/scaffold name` ) %>% arrange(n) );



mouse.tibble %>%group_by(`Chromosome/scaffold name`, `Transcript type`) %>%summarise( a
```

```
## `summarise()` has grouped output by 'Chromosome/scaffold name'. You can
## override using the `.groups` argument.

## # A tibble: 919 x 4
## # Groups:   Chromosome/scaffold name [117]
##    `Chromosome/scaffold name` `Transcript type`              avg count
##    <chr>                      <chr>                         <dbl> <int>
##  1 7                          transcribed_unitary_pseudogene 7198.     3
##  2 CHR_MG3656_PATCH           protein_coding                5753.     4
##  3 18                         polymorphic_pseudogene        4615       1
##  4 CHR_MG3833_MG4220_PATCH    TEC                           4490       1
##  5 8                          sense_overlapping             4316.     3
##  6 CHR_MG153_PATCH            lincRNA                       3877       1
##  7 CHR_MG74_PATCH             antisense                     3821       1
##  8 CHR_MG3251_PATCH           lincRNA                       3816       1
##  9 CHR_MG4261_PATCH           antisense                     3815       1
## 10 CHR_MG3490_PATCH           protein_coding                3814.    13
## # i 909 more rows
```

```
plot1 <-
mouse.tibble %>%
arrange( `Chromosome/scaffold name`, -`Transcript length (including UTRs and CDS)` );

plot1 %>%
  group_by(`Chromosome/scaffold name`) %>%
  summarise(
    longest = first(`Transcript length (including UTRs and CDS)`),
    shortest=last(`Transcript length (including UTRs and CDS)`),
    avg = mean(`Transcript length (including UTRs and CDS)`)
  ) %>%
  arrange(-avg)
```

```
## # A tibble: 117 x 4
##    `Chromosome/scaffold name` longest shortest   avg
##    <chr>                        <dbl>    <dbl> <dbl>
##  1 CHR_MG3656_PATCH              7155     4475 5753.
##  2 CHR_PWK_PHJ_MMCHR11_CTG2      3588     3516 3545.
##  3 JH584304.1                    4060     2373 3216.
##  4 CHR_MG3496_PATCH              6446      366 2998.
##  5 CHR_MG4266_PATCH              5389       73 2631.
##  6 CHR_CAST_EI_MMCHR11_CTG4      3588      134 2589.
##  7 JH584292.1                    2953     1304 2572.
##  8 CHR_WSB_EIJ_MMCHR11_CTG2      3546      246 2331.
##  9 CHR_MG3530_PATCH              8609      101 2284.
## 10 CHR_MG3490_PATCH              6641       70 2283
## # i 107 more rows
```

```
##unique_genes <- mouse.tibble %>%distinct(`Chromosome/scaffold name`, `Gene stable ID`

##unique_genes %>%group_by(`Chromosome/scaffold name`) %>%summarise(  count = n() ) %>%

chr.stats <- mouse.tibble %>%
  count(`Chromosome/scaffold name`) %>%
  arrange(-n);

chr.stats <- chr.stats %>%
  filter(n >= 500)

chr.stats
```

```
## # A tibble: 21 x 2
##    `Chromosome/scaffold name`     n
##    <chr>                      <int>
##  1 7                          12344
##  2 2                          10877
##  3 5                           8955
##  4 11                          8673
##  5 1                           8553
##  6 9                           8030
##  7 6                           7845
##   8 4                          7573
##   9 3                          6938
##  10 10                         6568
## # i 11 more rows
```

0.4.2 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；
library(tidyr);
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)
                  );
```

```r
grades.melted <- grades2 %>%
gather( course, grade, -Name, -Occupation, na.rm = T );

grades.melted %>%
group_by(Name, Occupation) %>%
summarise( avg_grades = mean( grade ), courses_count = n() ) %>%
arrange( -avg_grades );
```

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

```
## # A tibble: 7 x 4
## # Groups:   Name [7]
##   Name          Occupation   avg_grades courses_count
##   <chr>         <chr>             <dbl>         <int>
## 1 John Doe      Teacher            86.2             4
## 2 Mm Hu         Student            80               4
## 3 Warren Buffet Entrepreneur       79.2             4
## 4 Jack Ma       Entrepreneur       78               4
## 5 Jane Doe      Student            74               4
## 6 Elon Musk     Entrepreneur       70.8             4
## 7 Weihua Chen   Teacher            70.2             4
```

```r
##1
grades.melted2 <-
grades.melted %>%
arrange( Name, -grade );

grades.melted2 %>%
group_by(Name) %>%
summarise( worst_course = last( course ),
worst_grade=last( grade ),
avg_grades = mean( grade ) ) %>%
arrange( avg_grades );
```

```
## # A tibble: 7 x 4
##   Name           worst_course   worst_grade avg_grades
##   <chr>          <chr>                <int>      <dbl>
## 1 Weihua Chen    Biology                 64       70.2
## 2 Elon Musk      Biology                 52       70.8
## 3 Jane Doe       English                 62       74
## 4 Jack Ma        Bioinformatics          66       78
## 5 Warren Buffet Bioinformatics          62       79.2
## 6 Mm Hu          Bioinformatics          46       80
## 7 John Doe       Bioinformatics          73       86.2
```

```
##2
grades.melted %>%
group_by(Occupation) %>%
summarise( avg_grades = mean( grade ), courses_count = n() ) %>%
arrange( -avg_grades );
```

```
## # A tibble: 3 x 3
##   Occupation   avg_grades courses_count
##   <chr>             <dbl>         <int>
## 1 Teacher            78.2             8
## 2 Student            77               8
## 3 Entrepreneur       76              12
```

```
##3
grades.melted3 <-
grades.melted %>%
arrange( Occupation, -grade );

grades.melted3 %>%
group_by(Occupation,course) %>%
summarise( grade_avg = mean( grade )) %>%
  group_by(Occupation)%>%  arrange(Occupation,-grade_avg) %>%
summarise( best_grade=first( course ),best_course_avg = first( grade_avg )) %>%
arrange( - best_course_avg );
```

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

## # A tibble: 3 x 3
##   Occupation   best_grade      best_course_avg
##   <chr>        <chr>                     <dbl>
## 1 Student      ComputerScience            87.5
## 2 Teacher      ComputerScience            87.5
## 3 Entrepreneur ComputerScience            85
```

0.4.3 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`；

```r
## 代码写这里，并运行；
library(dplyr)
starwars$BMI <- starwars$mass / (starwars$height / 100)^2
starwars
```

```
## # A tibble: 87 x 15
##    name     height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Luke Sk~    172    77 blond      fair       blue            19   male  mascu~
##  2 C-3PO       167    75 <NA>       gold       yellow         112   none  mascu~
##  3 R2-D2        96    32 <NA>       white, bl~ red             33   none  mascu~
##  4 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
##  5 Leia Or~    150    49 brown      light      brown           19   fema~ femin~
##  6 Owen La~    178   120 brown, gr~ light      blue            52   male  mascu~
##  7 Beru Wh~    165    75 brown      light      blue            47   fema~ femin~
##  8 R5-D4        97    32 <NA>       white, red red             NA   none  mascu~
##  9 Biggs D~    183    84 black      light      brown           24   male  mascu~
## 10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57    male  mascu~
## # i 77 more rows
## # i 6 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>, BMI <dbl>
```

```
obese_humans <- starwars %>%
  filter(species == "Human" & BMI >= 30) %>%
  select(name, sex, homeworld)

obese_humans
```

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

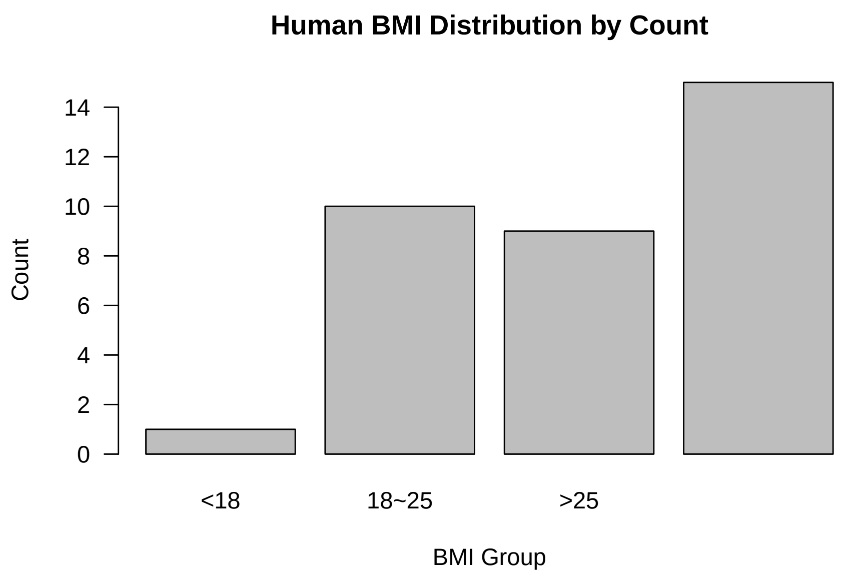
3. 挑选出所有人类；
4. 按 BMI 将他们分为三组，<18, 18~25, >25，统计每组的人数，并用 barplot 进行展示；注意：展示时三组的按 BMI 从小到大排序；
5. 改变排序方式，按每组人数从小到大排序；

```
## 代码写这里，并运行；
library(dplyr)
library(ggplot2)
humans<-starwars %>%
  filter(species == "Human" ) %>%
arrange( -BMI );
humans;
```

```
## # A tibble: 35 x 15
##    name      height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Owen La~     178   120 brown, gr~ light      blue            52    male  mascu~
##  2 Darth V~     202   136 none       white      yellow          41.9 male  mascu~
##  3 Beru Wh~     165    75 brown      light      blue            47    fema~ femin~
##   4 Wedge A~    170    77 brown      fair      hazel          21    male  mascu~
##   5 Luke Sk~    172    77 blond      fair      blue           19    male  mascu~
##   6 Palpati~    170    75 grey       pale      yellow         82    male  mascu~
##   7 Lobot       175    79 none       light     blue           37    male  mascu~
##   8 Lando C~    177    79 black      dark      brown          31    male  mascu~
##   9 Biggs D~    183    84 black      light     brown          24    male  mascu~
##  10 Han Solo    180    80 brown      fair      brown          29    male  mascu~
## # i 25 more rows
## # i 6 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>, BMI <dbl>
```

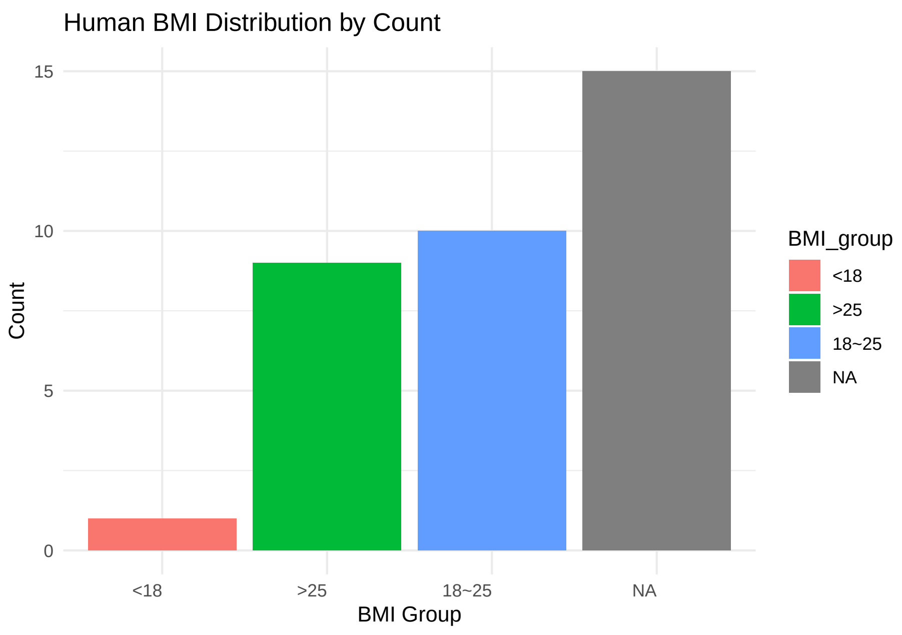
```
humans <- humans %>%
  mutate(BMI_group = cut(BMI,
                         breaks = c(-Inf, 18, 25, Inf),
                         labels = c("<18", "18~25", ">25")))
       bmi_counts <- humans %>%
  count(BMI_group)

barplot(height = bmi_counts$n, names.arg = bmi_counts$BMI_group, main = "Human BMI Dist
```



```r
bmi_counts <- bmi_counts %>%
  mutate(BMI_group = reorder(BMI_group, n))

ggplot(bmi_counts, aes(x = BMI_group, y = n, fill = BMI_group)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(title = "Human BMI Distribution by Count", x = "BMI Group", y = "Count") +
  theme(axis.text.x = element_text( hjust = 1))
```



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：其特点是它是一个列表（list）型数据，其中每个元素对应一个角色，并存储了该角色参与的所有电影名称，通常表现为字符串数组。这个列允许存在多样性，即不是所有的角色都有相同的电影数目，有的角色可能只参与一部电影，而有的则参与多部，非常适合反映现实生活中的多元联系。

data.frame 确实可以实现类似的功能，它能够容纳混合类型的列，包括字符列（如电影名称）、列表列（如 films）。通过设置相应的列结构，data.frame

可以有效地存储和管理一对多关系的数据，方便进行统计分析和数据挖掘操作。

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;
library(dplyr);library(purrr)
starwars <- starwars %>%
  mutate(films_count = map_dbl(films, ~ length(.x)))
starwars
```

```
## # A tibble: 87 x 16
##    name      height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Luke Sk~     172    77 blond      fair       blue            19   male  mascu~
##  2 C-3PO        167    75 <NA>       gold       yellow         112   none  mascu~
##  3 R2-D2         96    32 <NA>       white, bl~ red             33   none  mascu~
##  4 Darth V~     202   136 none       white      yellow          41.9 male  mascu~
##  5 Leia Or~     150    49 brown      light      brown           19   fema~ femin~
##  6 Owen La~     178   120 brown, gr~ light      blue            52   male  mascu~
##  7 Beru Wh~     165    75 brown      light      blue            47   fema~ femin~
##  8 R5-D4         97    32 <NA>       white, red red             NA   none  mascu~
##  9 Biggs D~     183    84 black      light      brown           24   male  mascu~
## 10 Obi-Wan~     182    77 auburn, w~ fair       blue-gray       57   male  mascu~
## # i 77 more rows
## # i 7 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>, BMI <dbl>, films_count <dbl>
```

0.4.4 使用 Theoph 变量做练习

注：以下练习请只显示结果的前 6 行；

1. 选取从 Subject 到 Dose 的列；总共有几列？

```
## 代码写这里，并运行；
stats <-
Theoph %>%
select( Subject:Dose )
head(stats,6)
```

```
##   Subject   Wt Dose
## 1       1 79.6 4.02
## 2       1 79.6 4.02
## 3       1 79.6 4.02
## 4       1 79.6 4.02
## 5       1 79.6 4.02
## 6       1 79.6 4.02
```

2. 用 filter 选取 Dose 大于 5，且 Time 高于 Time 列平均值的行；

```
## 代码写这里，并运行；
(time_avg <- mean(Theoph$Time, na.rm = TRUE))
```

```
## [1] 5.894621
```

```
stats1<-Theoph %>% filter(Dose>5 & Time>time_avg)
head(stats1)
```

```
##   Subject   Wt Dose  Time conc
## 1       5 54.6 5.86  7.02 7.09
## 2       5 54.6 5.86  9.10 5.90
## 3       5 54.6 5.86 12.00 4.37
## 4       5 54.6 5.86 24.35 1.57
## 5      10 58.2 5.50  7.08 8.02
## 6      10 58.2 5.50  9.38 7.14
```

3. 用 mutate 函数产生新列 trend，其值为 Time 与 Time 列平均值的差；注意：请去除可能产生的 na 值；

```
## 代码写这里，并运行；
stats2 <- Theoph %>%
  mutate(trend = Time-time_avg)%>%filter(!is.na(trend))
head(stats2)
```

```
##   Subject   Wt Dose Time  conc     trend
## 1       1 79.6 4.02 0.00  0.74 -5.894621
## 2       1 79.6 4.02 0.25  2.84 -5.644621
## 3       1 79.6 4.02 0.57  6.57 -5.324621
## 4       1 79.6 4.02 1.12 10.50 -4.774621
## 5       1 79.6 4.02 2.02  9.66 -3.874621
## 6       1 79.6 4.02 3.82  8.58 -2.074621
```

4. 用 `mutate` 函数产生新列 `weight_cat` ，其值根据 `Wt` 的取值范围而不同：

- 如果 `Wt` > 76.2，为 ‘Super-middleweight’，否则
- 如果 `Wt` > 72.57，为 ‘Middleweight’，否则
- 如果 `Wt` > 66.68，为 ‘Light-middleweight’
- 其它值，为 ‘Welterweight’

```
stats3 <- Theoph %>%
  mutate(weight_cat = case_when(
    Wt > 76.2 ~ "Super-middleweight",
    Wt > 72.57 ~ "Middleweight",
    Wt > 66.68 ~ "Light-middleweight",
    TRUE ~ "Welterweight"
  ))
head(stats3)
```

```
##   Subject   Wt Dose Time  conc         weight_cat
## 1       1 79.6 4.02 0.00  0.74 Super-middleweight
## 2       1 79.6 4.02 0.25  2.84 Super-middleweight
## 3       1 79.6 4.02 0.57  6.57 Super-middleweight
## 4       1 79.6 4.02 1.12 10.50 Super-middleweight
## 5        1 79.6 4.02 2.02  9.66 Super-middleweight
## 6        1 79.6 4.02 3.82  8.58 Super-middleweight
```

0.4.5 使用 iris 变量做练习

运行以下代码:

```
iris %>%
  subset(Sepal.Length > 5) %>%
  aggregate(. ~ Species, ., mean)
```

```
##      Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     setosa     5.313636    3.713636     1.509091   0.2772727
## 2 versicolor     5.997872    2.804255     4.317021   1.3468085
## 3  virginica     6.622449    2.983673     5.573469   2.0326531
```

回答以下问题:

1. 输出结果中的数值是什么?

答：从 iris 数据集中选择 Sepal.Length 大于 5 的行, 然后按照 Species 物种进行分组, 对每个分组计算所有数值型列的平均值。

2. 请解释 `aggregate` 函数的 4 个参数分析是什么作用?

答：第一个参数 x，是一个公式或数据框中的一个向量，指定了要进行聚合计算的变量。. ~ Species 表示使用 Species 作为分组变量，而对数据框中的所有其他变量进行聚合计算。第二个参数 by，第二个参数. 表示当前数据框（. 在公式中通常指代当前数据框的所有行和列）。第三个参数，是一个函数，用于对每个分组的数据执行计算。它可以是任何接受向量作为输入并返回单个值的函数。mean 表示对每个分组的变量计算平均值。第四个参数，…：是一组可选参数，它们会被传递给 FUN 函数。这些参数可以用来自定义聚合函数的行为。例如，如果你在计算平均值时想要排除 NA 值，可以在 mean 函数中使用 na.rm = TRUE 参数