

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
 - 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！

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

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

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

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

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：

```
library("tidyverse")
library("reshape2")
mouse_tibble <- read_delim("data/talk04/mouse_genes_biomart_sep2018.txt",
  delim = "\t", quote = "", show_col_types = FALSE)
```

```
## 代码写这里，并运行；
message(" 不使用中间变量实现上述目标。")
```

```
## 不使用中间变量实现上述目标。
```

```

mouse_tibble %>%
  group_by(`Chromosome/scaffold name`, `Transcript type`) %>%
  mutate(
    # 使用 mutate 函数统计每个染色体上每种基因类型的数量、平均长度、最大和最小长度
    count = n(),
    # mean_length 保留整数
    mean_length = round(mean(`Transcript length (including UTRs and CDS)`)),
    max_length = max(`Transcript length (including UTRs and CDS)`),
    min_length = min(`Transcript length (including UTRs and CDS)`),
  ) %>%
  ungroup() %>%
  rename(gene_length = `Transcript length (including UTRs and CDS)`) %>%
  # 标记出最长和最短的基因
  mutate(variable = ifelse(gene_length == max_length, "is_max",
                           ifelse(gene_length == min_length, "is_min", NA))) %>%
  arrange(`Chromosome/scaffold name`, `Transcript type`) %>%
  filter(!is.na(variable)) %>%
  select(`Gene stable ID`, `Chromosome/scaffold name`, `Transcript type`,
        count, mean_length, max_length, min_length, variable) %>%
  data.frame() %>%
  # 每个染色体上每种基因类型最长和最短的基因。可能有多个基因，用逗号分隔。
  dcast(`Chromosome.scaffold.name` + `Transcript.type` + count +
        mean_length + max_length + min_length ~ variable,
        value.var = "Gene.stable.ID",
        fun.aggregate = function(x) paste(x, collapse = ", ")) %>%
  tibble() %>%
  # 如果 is_min 为空，说明最长的基因和最短的基因相同
  mutate(is_min = ifelse(is_min == "", is_max, is_min)) %>%
  distinct() %>%
  # 去掉少于 500 基因的染色体
  filter(count > 500) %>%
  # 按染色体（或 scaffold）、数量 高 -\> 低 进行排序
  rename(gene_count = count, longest_gene = is_max,

```

```

shortest_gene = is_min) %>%
  arrange(desc(gene_count)) %>%
  # 显示前 10 行输出结果
  data.frame() %>%
  head(10)

```

```

##      Chromosome.scaffold.name Transcript.type gene_count mean_length max_length
## 1                7 protein_coding      5575         2148      19703
## 2                2 protein_coding      4863         2654     106824
## 3               11 protein_coding      4143         2432      24175
## 4                5 protein_coding      3620         2405      20084
## 5                9 protein_coding      3424         2373      25437
## 6                1 protein_coding      3369         2700      40378
## 7                4 protein_coding      3346         2446      23495
## 8                6 protein_coding      3188         2370      24060
## 9               10 protein_coding      2747         2408     123179
## 10               3 protein_coding      2722         2466      16109
##      min_length      longest_gene      shortest_gene
## 1           72 ENSMUSG000000059146 ENSMUSG000000091441
## 2           57 ENSMUSG000000051747 ENSMUSG000000092123
## 3           66 ENSMUSG000000061462 ENSMUSG000000092157
## 4           54 ENSMUSG000000061601 ENSMUSG000000079439, ENSMUSG000000079438
## 5           57 ENSMUSG000000109564 ENSMUSG000000079433
## 6           75 ENSMUSG000000044768 ENSMUSG000000090553
## 7           24 ENSMUSG000000028649 ENSMUSG000000095386
## 8           48 ENSMUSG000000030209 ENSMUSG000000092004
## 9           51 ENSMUSG000000020255 ENSMUSG000000078481
## 10          36 ENSMUSG000000046743 ENSMUSG000000090441

```

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"
);
```

2. 尝试使用 `spread` 和 `gather` 函数将其变宽后再变长;

```
## 代码写这里，并运行;
```

```
grades <- read_tsv(file = "data/talk05/grades.txt", show_col_types = FALSE)
grades_spread <- spread(grades, key = `course`, value = `grade`)
grades_gather <- gather(grades_spread, key = `course`,
  value = `grade`, -name) %>%
  filter(!is.na(`grade`)) %>%
  arrange(desc(name))
grades_spread
```

```
## # 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

```
grades_gather
```

```
## # A tibble: 9 x 3
```

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

9 Kang Ning Chinese 20

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.pivot_wider <- pivot_wider(grades, names_from = "course",
  values_from = "grade")
grades.pivot_longer <- pivot_longer(grades.pivot_wider, cols = 2:6,
  names_to = "course", values_to = "grade",
  # 去掉带 na 的行
  values_drop_na = TRUE)
grades.pivot_wider
```

```
## # 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.pivot_longer
```

```
## # 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

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;

```
message(" 使用 values_drop_na = TRUE 参数去除带 na 的行")
```

使用 `values_drop_na = TRUE` 参数去除带 `na` 的行

```
pivot_longer(grades.pivot_wider, cols = 2:6,
             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: `complete` 函数用于填充缺失值, 这里是填充 `name` 和 `course` 的缺失值, 使得每个 `name` 和 `course` 都有一个 `grade` 值。

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；  
set.seed(123)  
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)  
                )  
# 1. 每个人最差的学科和成绩分别是什么？  
grades2 %>%  
  gather(key = "course", value = "grade", -Name, -Occupation) %>%  
  group_by(Name) %>%  
  summarise(min_grade = min(grade)) %>%  
  # 每个人成绩最差的学科
```



```

left_join(grades2, by = c("Name")) %>%
melt(id.vars = c("Name", "min_grade", "Occupation")) %>%
tibble() %>%
rename(course = variable, grade = value) %>%
filter(grade == min_grade) %>%
select(Name, poor_course = course, grade) %>%
arrange(Name)

```

```

## # A tibble: 7 x 3
##   Name          poor_course  grade
##   <chr>         <fct>         <int>
## 1 Elon Musk     Biology         56
## 2 Jack Ma       Bioinformatics  78
## 3 Jane Doe      Bioinformatics  53
## 4 John Doe      English         73
## 5 Mm Hu         Bioinformatics  58
## 6 Warren Buffet Bioinformatics  56
## 7 Weihua Chen  Bioinformatics  48

```

2. 哪个职业的平均成绩最好？

```

grades2 %>%
  gather(key = "course", value = "grade", -Name, -Occupation) %>%
  group_by(Occupation) %>%
  summarise(avg_grade = mean(grade)) %>%
  arrange(desc(avg_grade))

```

```

## # A tibble: 3 x 2
##   Occupation  avg_grade
##   <chr>         <dbl>
## 1 Entrepreneur  80.4
## 2 Teacher       74.9
## 3 Student       69.2

```

3. 每个职业的最佳学科分别是什么（按平均分排序）???

```
grades2 %>%
  gather(key = "course", value = "grade", -Name, -Occupation) %>%
  group_by(Occupation, course) %>%
  summarise(avg_grade = mean(grade)) %>%
  arrange(Occupation, desc(avg_grade)) %>%
  group_by(Occupation) %>%
  rename(best_course = course) %>%
  slice(1)
```

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

```
## # A tibble: 3 x 3
## # Groups:   Occupation [3]
##   Occupation best_course avg_grade
##   <chr>      <chr>      <dbl>
## 1 Entrepreneur English      92.7
## 2 Student    ComputerScience 86
## 3 Teacher    ComputerScience 85.5
```

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;
mutate(starwars, BMI = mass / (height / 100)^2) %>%
  filter(BMI >= 30) %>%
  select(name, sex, homeworld)
```

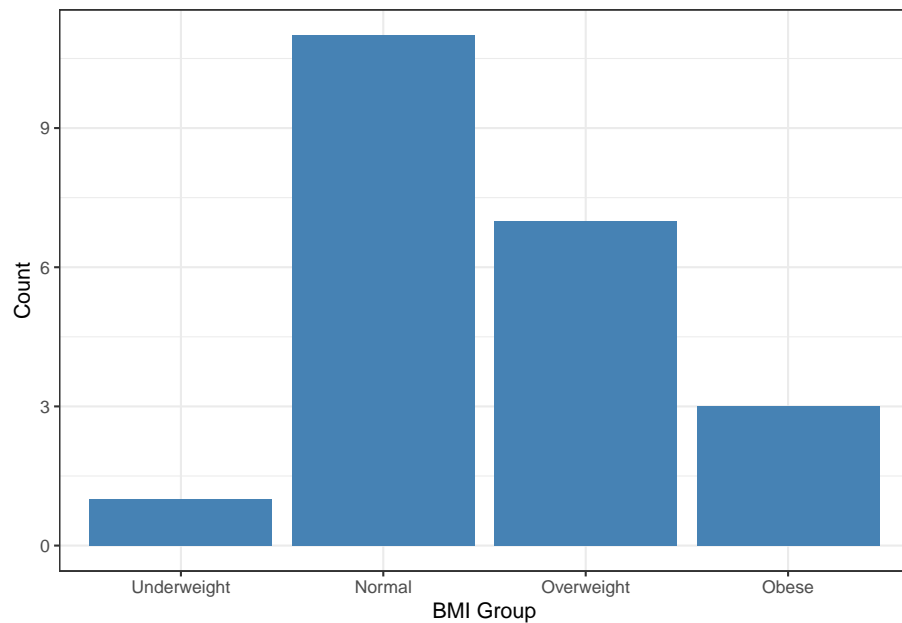
```
## # A tibble: 12 x 3
##   name                sex          homeworld
##   <chr>              <chr>      <chr>
## 1 R2-D2              none        Naboo
## 2 Darth Vader        male        Tatooine
## 3 Owen Lars          male        Tatooine
## 4 R5-D4              none        Tatooine
## 5 Jabba Desilijic Tiure hermaphroditic Nal Hutta
## 6 Jek Tono Porkins   male        Bestine IV
## 7 Yoda               male        <NA>
## 8 IG-88              none        <NA>
## 9 Bossk              male        Trandosha
## 10 Sebulba           male        Malastare
## 11 Dud Bolt          male        Vulpter
## 12 Grievous          male        Kalee
```

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;

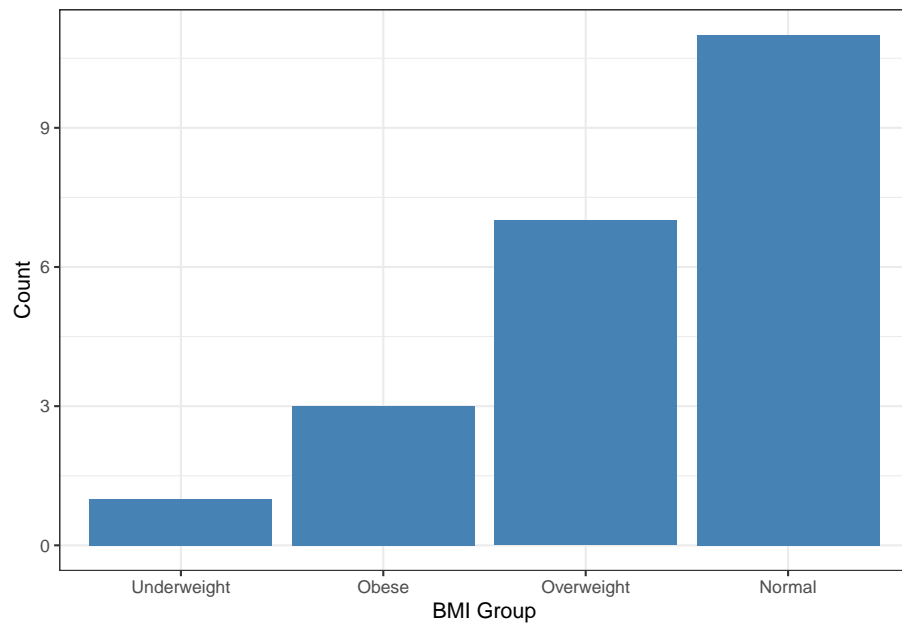
```
## 代码写这里, 并运行;
human_bmi <- filter(starwars, species == "Human") %>%
  mutate(BMI = mass/(height/100)^2) %>%
  mutate(BMI_group = ifelse(BMI < 18, "Underweight",
                            ifelse(BMI < 25, "Normal",
                                    ifelse(BMI < 30, "Overweight", "Obese")))) %>%
  group_by(BMI_group) %>%
  summarise(count = n())

human_bmi %>%
  filter(!is.na(BMI_group)) %>%
  ggplot(aes(x = factor(BMI_group,
                      levels = c("Underweight", "Normal", "Overweight", "Obese"))),
```

```
    y = count)) +  
  # 调整颜色  
  geom_bar(stat = "identity", fill = "steelblue") +  
  labs(x = "BMI Group", y = "Count") +  
  theme_bw()
```



```
# human_bmi 中的 BMI_group 按照相应的 count 重新设定 levels  
human_bmi %>%  
  filter(!is.na(BMI_group)) %>%  
  ggplot(aes(x = factor(BMI_group,  
    levels = human_bmi$BMI_group[order(human_bmi$count)]),  
    y = count)) +  
  geom_bar(stat = "identity", fill = "steelblue") +  
  labs(x = "BMI Group", y = "Count") +  
  theme_bw()
```



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 列的每个元素是一个包含多个元素的列表，`data.frame` 不能实现类似的功能。

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 %>%
  # 保持所有列不变，增加一列 nr_films，
  # 使用 map_int() 函数计算每个角色出现的电影数
  mutate(nr_films = map_int(films, length)) %>%
  select(name, nr_films, 2:length(starwars), -films) %>%
  # 展示前 5 行
  data.frame() %>%
  head(5)
```

```
##           name nr_films height mass hair_color skin_color eye_color
## 1 Luke Skywalker      5   172   77      blond      fair      blue
## 2           C-3PO      6   167   75      <NA>      gold      yellow
## 3           R2-D2      7    96   32      <NA> white, blue      red
## 4   Darth Vader      4   202  136      none      white      yellow
## 5   Leia Organa      5   150   49      brown      light      brown
##  birth_year  sex    gender homeworld species
## 1      19.0  male masculine Tatooine  Human
## 2     112.0  none masculine Tatooine  Droid
## 3      33.0  none masculine  Naboo   Droid
## 4      41.9  male masculine Tatooine  Human
## 5      19.0 female  feminine Alderaan Human
##
##           vehicles           starships
## 1 Snowspeeder, Imperial Speeder Bike X-wing, Imperial shuttle
## 2
## 3
## 4
## 5           TIE Advanced x1
## 5           Imperial Speeder Bike
```

0.4.5 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；
theoph <- tibble(Theoph)
# 选取从 Subject 到 Dose 的列
select(theoph, Subject:Dose)
```

```
## # A tibble: 132 x 3
##   Subject    Wt Dose
##   <ord>    <dbl> <dbl>
## 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
## 7 1      79.6  4.02
## 8 1      79.6  4.02
## 9 1      79.6  4.02
## 10 1     79.6  4.02
## # ... with 122 more rows
```

```
message(" 总共有", ncol(select(theoph, Subject:Dose)), " 列")
```

```
## 总共有3列
```

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;
```

```
theoph %>%
  filter(Time > mean(Time)) %>%
  filter(Dose > 5)
```

```
## # A tibble: 12 x 5
##   Subject    Wt Dose  Time  conc
##   <ord>    <dbl> <dbl> <dbl> <dbl>
## 1 5      54.6  5.86  7.02  7.09
## 2 5      54.6  5.86  9.1   5.9
## 3 5      54.6  5.86  12    4.37
## 4 5      54.6  5.86  24.4  1.57
## 5 10     58.2  5.5   7.08  8.02
## 6 10     58.2  5.5   9.38  7.14
## 7 10     58.2  5.5  12.1  5.68
## 8 10     58.2  5.5  23.7  2.42
## 9 12     60.5  5.3   7.07  6.59
```

```
## 10 12      60.5  5.3   9.03  6.11
## 11 12      60.5  5.3  12.0   4.57
## 12 12      60.5  5.3  24.2   1.17
```

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；
theoph %>%
  mutate(trend = Time - mean(Time, na.rm = TRUE)) %>%
  filter(!is.na(trend))
```

```
## # A tibble: 132 x 6
##   Subject    Wt Dose  Time  conc  trend
##   <ord>    <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1      79.6  4.02  0     0.74 -5.89
## 2 1      79.6  4.02  0.25  2.84 -5.64
## 3 1      79.6  4.02  0.57  6.57 -5.32
## 4 1      79.6  4.02  1.12 10.5  -4.77
## 5 1      79.6  4.02  2.02  9.66 -3.87
## 6 1      79.6  4.02  3.82  8.58 -2.07
## 7 1      79.6  4.02  5.1   8.36 -0.795
## 8 1      79.6  4.02  7.03  7.47  1.14
## 9 1      79.6  4.02  9.05  6.89  3.16
## 10 1     79.6  4.02 12.1   5.94  6.23
## # ... with 122 more rows
```

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’

代码写这里，并运行；

```
theoph %>%
  mutate(weight_cat =
    ifelse(Wt > 76.2, "Super-middleweight",
    ifelse(Wt > 72.57, "Middleweight",
    ifelse(Wt > 66.68, "Light-middleweight", "Welterweight")))) %>%
  print(n = 20)
```

A tibble: 132 x 6

```
##   Subject    Wt Dose  Time  conc weight_cat
##   <ord>    <dbl> <dbl> <dbl> <dbl> <chr>
## 1 1      79.6  4.02  0     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 10.5  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
## 7 1      79.6  4.02  5.1   8.36 Super-middleweight
## 8 1      79.6  4.02  7.03  7.47 Super-middleweight
## 9 1      79.6  4.02  9.05  6.89 Super-middleweight
## 10 1     79.6  4.02 12.1   5.94 Super-middleweight
## 11 1     79.6  4.02 24.4   3.28 Super-middleweight
## 12 2     72.4  4.4   0     0    Light-middleweight
## 13 2     72.4  4.4   0.27  1.72 Light-middleweight
## 14 2     72.4  4.4   0.52  7.91 Light-middleweight
## 15 2     72.4  4.4   1     8.31 Light-middleweight
## 16 2     72.4  4.4   1.92  8.33 Light-middleweight
## 17 2     72.4  4.4   3.5   6.85 Light-middleweight
## 18 2     72.4  4.4   5.02  6.08 Light-middleweight
## 19 2     72.4  4.4   7.03  5.4  Light-middleweight
## 20 2     72.4  4.4   9     4.55 Light-middleweight
## # ... with 112 more rows
```