# R for bioinformatics, data wrangler, part 1

## HUST Bioinformatics course series

Wei-Hua Chen (CC BY-NC 4.0)

10 三月, 2020

# section 1: TOC

# 前情提要

1. IO, project management, working enviroment management
2. factors: R 中最重要的概念之一

- factors 基本概念
- factors 操作
- factors 在做图中的使用
- ggplot2 和 dplyr 初步

# 今次提要

- dplyr 、tidyr (超级强大的数据处理) part 1

# section 2: data wrangler - dplyr

# dplyr

## what is `dplyr` ?

- the next iteration of plyr,
- focussing on only data frames (tibble),
- row-based manipulation,
- dplyr is faster and has a more consistent API.


**Figure 1:** dplyr logo

# dplyr, overview

dplyr provides a consistent set of verbs that help you **solve the most common data manipulation challenges**:

- select() 选择列，根据列名规则
- filter() 按规则过滤行
- mutate() 增加新列，从其它列计算而得（不改变行数）
- summarise() 将多个值转换为单个值（通过 mean, median, sd 等操作），生成新列（总行数减少，通常与 `group_by` 配合使用）
- arrange() 对行进行排序

# dplyr 安装

```
# The easiest way to get dplyr is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just dplyr:
install.packages("dplyr")
```

Development version

```
# install.packages("devtools")
devtools::install_github("tidyverse/dplyr")
```

Get the cheatsheet at here

# an example of dplyr

## get the data ready

```
mouse.tibble <- read_delim( file = "data/talk04/mouse_genes_biomart_sep2018.txt",
                            delim = "\t", quote = "" );
```

```
## Parsed with column specification:
## 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()
## )
```

# 查看 mouse.tibble 的內容

```
table( mouse.tibble$`Transcript type` );
```

```
##
##           3prime_overlapping_ncRNA                        antisense
##                                  3                             4289
##      bidirectional_promoter_lncRNA                        IG_C_gene
##                                267                               21
##                    IG_C_pseudogene                        IG_D_gene
##                                  1                               19
##                    IG_D_pseudogene                        IG_J_gene
##                                  3                               14
##                         IG_LV_gene                     IG_pseudogene
##                                  4                                2
##                          IG_V_gene                  IG_V_pseudogene
##                                301                              155
##                            lincRNA                     macro_lncRNA
##                               8557                                2
##                             miRNA                        misc_RNA
##                               2265                              572
##                           Mt_rRNA                         Mt_tRNA
##                                  2                               22
##                     non_stop_decay       nonsense_mediated_decay
##                                 26                             6755
##              polymorphic_pseudogene         processed_pseudogene
##                                 94                             9425
##               processed_transcript               protein_coding
##                              15572                            58384
```

# 查看 mouse.tibble 的內容, cont.

```
table( mouse.tibble$`Chromosome/scaffold name`);
```

```
##
##                          1                         10
##                       8553                       6568
##                         11                         12
##                       8673                       5308
##                         13                         14
##                       5618                       5843
##                         15                         16
##                       5142                       4425
##                         17                         18
##                       5050                       2096
##                         19                          2
##                       2982                      10877
##                          3                          4
##                       6938                       7573
##                          5                          6
##                       8955                       7845
##                          7                          8
##                      12344                       6385
##                          9     CHR_CAST_EI_MMCHR11_CTG4
##                       8030                         14
##    CHR_CAST_EI_MMCHR11_CTG5            CHR_MG104_PATCH
##                         40                         17
##             CHR_MG117_PATCH            CHR_MG132_PATCH
##                         52                         29
```

# 分析任务

1. 将染色体限制在常染色体和 XY 上（去掉未组装的小片段）；处理行
2. 将基因类型限制在 protein_coding, miRNA 和 lincRNA 这三种；处理行
3. 统计每条染色体上不同类型基因（protein_coding, miRNA, lincRNA）的数量
4. 按染色体（正）、基因数量（倒）进行排序

# 用 dplyr 实现

```
dat <- mouse.tibble %>%
  ## 1.

  filter( `Chromosome/scaffold name` %in% c( 1:19, "X", "Y" )   ) %>%

  ## 2.
  filter( `Transcript type` %in% c( "protein_coding", "miRNA", "lincRNA" ) ) %>%

  ## change column name ...
  select( CHR = `Chromosome/scaffold name`, TYPE = `Transcript type`,
          GENE_ID = `Gene stable ID`,
          GENE_LEN =  `Transcript length (including UTRs and CDS)`  ) %>%

  ## 3.
  group_by( CHR, TYPE ) %>%
  summarise( count = n_distinct( GENE_ID ), mean_len = mean( GENE_LEN ) ) %>%

  ## 4.
  arrange(  CHR  , desc( count ) );
```

# 检查运行结果

| CHR | TYPE | count | mean_len |
|---|---|---|---|
| 1 | protein_coding | 1200 | 2699.59009 |
| 1 | lincRNA | 347 | 1206.76149 |
| 1 | miRNA | 128 | 97.97656 |
| 10 | protein_coding | 1020 | 2408.16454 |
| 10 | lincRNA | 398 | 1220.35543 |
| 10 | miRNA | 91 | 89.87912 |
| 11 | protein_coding | 1640 | 2431.87666 |
| 11 | lincRNA | 189 | 1134.49174 |
| 11 | miRNA | 137 | 87.48905 |
| 12 | protein_coding | 644 | 2523.94822 |
| 12 | lincRNA | 327 | 1277.14979 |
| 12 | miRNA | 146 | 86.24658 |
| 13 | protein_coding | 831 | 2380.41499 |
| 13 | lincRNA | 428 | 1251.04552 |
| 13 | miRNA | 97 | 105.52577 |

这种显示格式通常被称为：**长数据格式**!! 又称为**数据扁平化**

# 数据扁平化的优点？

1. 便于用 dplyr 或 tapply 等进行计算；
2. 更灵活，用于保存稀疏数据

# 适合扁平化的数据举例

## 成绩单

```
library(RMySQL);
library(dplyr);

mysql.dbname = "r4ds_test";
dbCon <- dbConnect(MySQL(), user="r4ds", password="r4ds",
                   dbname=mysql.dbname );

grades <- dbGetQuery(dbCon, "SELECT * FROM grades_stats");

knitr::kable( head(grades, n=20) )
```

| name | course | grade |
|---|---|---|
| Zhi Liu | Microbiology | 100 |
| Zhi Liu | English | 50 |
| Zhi Liu | Chinese | 69 |
| Weihua Chen | Microbiology | 89 |
| Weihua Chen | English | 99 |
| Weihua Chen | Bioinformatics | 99 |
| Kang Ning | Bioinformatics | 100 |
| Kang Ning | Chinese | 20 |
| Kang Ning | Chemistry | 76 |

# 长数据与宽数据之间的转换

## 什么是宽数据？

```
dat2 <- dat %>% select( CHR, TYPE, `count` ) %>%  spread( TYPE, count  );
knitr::kable( head(dat2, n=10) );
```

| CHR | lincRNA | miRNA | protein_coding |
|---|---|---|---|
| 1 | 347 | 128 | 1200 |
| 10 | 398 | 91 | 1020 |
| 11 | 189 | 137 | 1640 |
| 12 | 327 | 146 | 644 |
| 13 | 428 | 97 | 831 |
| 14 | 281 | 71 | 901 |
| 15 | 215 | 94 | 781 |
| 16 | 176 | 76 | 661 |
| 17 | 114 | 73 | 1066 |
| 18 | 43 | 57 | 524 |

# 宽数据举例 2

```
grades2 <- grades %>% spread( course, grade );
knitr::kable( grades2 );
```

| name | Bioinformatics | Chemistry | Chinese | English | Microbiology |
|---|---|---|---|---|---|
| Kang Ning | 100 | 76 | 20 | NA | NA |
| Weihua Chen | 99 | NA | NA | 99 | 89 |
| Zhi Liu | NA | NA | 69 | 50 | 100 |

可以想像，如果以此为输入，用 R 计算每个人的平均成绩、不及格门数、总学分，将会是很繁琐的一件事（但对其它工具（如 Excel）可能会比较简单）

# spread explained!

```
grades2 <- grades %>% spread( course, grade );
```

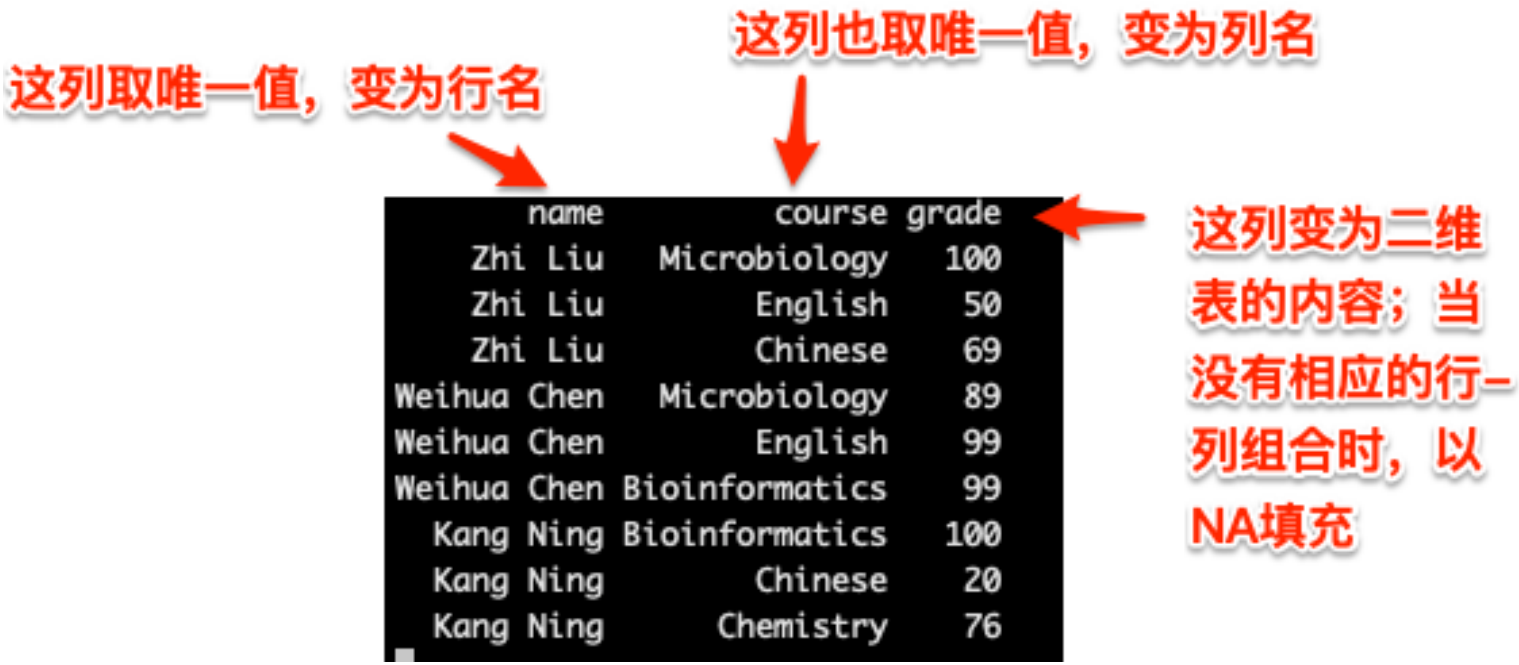


**Figure 2:** spread function explained

# 宽数据转为长数据

use gather() function in tidyr

```
grades_melted <- grades2 %>% gather( course, grade, -name ); ## 注意参数的使用 ~~
knitr::kable( head( grades_melted ) );
```

| name | course | grade |
|---|---|---|
| Kang Ning | Bioinformatics | 100 |
| Weihua Chen | Bioinformatics | 99 |
| Zhi Liu | Bioinformatics | NA |
| Kang Ning | Chemistry | 76 |
| Weihua Chen | Chemistry | NA |
| Zhi Liu | Chemistry | NA |

# gather explained!

```
grades_melted <- grades2 %>% gather( course, grade, -name ); ## 注意参数的使用 ~~
```

-name: 此列保留

列名变为第一列，取名为 course

| name | Bioinformatics | Chemistry | Chinese | English | Microbiology |
|---|---|---|---|---|---|
| Kang Ning | 100 | 76 | 20 | NA | NA |
| Weihua Chen | 99 | NA | NA | 99 | 89 |
| Zhi Liu | NA | NA | 69 | 50 | 100 |

值变为第二列，取名为 grade

**Figure 3:** annotated gather function

# 有 NA 值怎么办？

```
grades_melted1 <- grades_melted[ !is.na(grades_melted$grade),  ];
grades_melted2 <- grades_melted[ complete.cases( grades_melted )  ,  ];

## -- 更好的方法 ~~
grades_melted <- grades2 %>% gather( course, grade, -name , na.rm = T );
```

# 宽长数据转换练习

用 spread 和 gather 对下面的数据 mini_iris 进行宽长转换:

```
( mini_iris <- iris[ c(1, 51, 101),  ] );
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 51           7.0         3.2          4.7         1.4 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
```

iris 是鸢尾属一些物种花瓣的量表

# 宽长数据转换练习，cont.

```
## -- 注意: 第一、二个参数可以自行命名，分别对应原始数据中的 column names 及 values ...
mini_iris.melted <- mini_iris %>% gather( type, dat, -Species );
knitr::kable( mini_iris.melted  );
```

| Species | type | dat |
|---|---|---|
| setosa | Sepal.Length | 5.1 |
| versicolor | Sepal.Length | 7.0 |
| virginica | Sepal.Length | 6.3 |
| setosa | Sepal.Width | 3.5 |
| versicolor | Sepal.Width | 3.2 |
| virginica | Sepal.Width | 3.3 |
| setosa | Petal.Length | 1.4 |
| versicolor | Petal.Length | 4.7 |
| virginica | Petal.Length | 6.0 |
| setosa | Petal.Width | 0.2 |
| versicolor | Petal.Width | 1.4 |
| virginica | Petal.Width | 2.5 |

# 比较复杂的例子

```
grades2 <- read_delim( file = "data/talk05/grades2.txt", delim = "\t",
                       quote = "", col_names = T);
knitr::kable( grades2 );
```

| name | class | course | grade |
|---|---|---|---|
| CHEN | 1 | bioinformatics | 90 |
| CHEN | 1 | chemistry | 92 |
| CHEN | 2 | chinese | 35 |
| CHEN | 3 | german | 62 |
| LI | 1 | bioinformatics | 44 |
| LI | 2 | chinese | 68 |
| LI | 3 | microbiology | 95 |
| LI | 3 | japanese | 90 |
| WANG | 1 | bioinformatics | 35 |
| WANG | 1 | chemistry | 76 |
| WANG | 1 | mathmatics | 82 |
| WANG | 3 | german | 100 |
| WANG | 3 | spanish | 78 |

# 怎么用 spread 把它变为以下的格式？

```
## # A tibble: 8 x 10
##   name  class bioinformatics chemistry chinese german japanese mathmatics
##   <chr> <dbl>          <dbl>     <dbl>   <dbl>  <dbl>    <dbl>       <dbl>
## 1 CHEN      1             90        92      NA     NA       NA          NA
## 2 CHEN      2             NA        NA      35     NA       NA          NA
## 3 CHEN      3             NA        NA      NA     62       NA          NA
## 4 LI        1             44        NA      NA     NA       NA          NA
## 5 LI        2             NA        NA      68     NA       NA          NA
## 6 LI        3             NA        NA      NA     NA       90          NA
## 7 WANG      1             35        76      NA     NA       NA          82
## 8 WANG      3             NA        NA      NA    100       NA          NA
## # ... with 2 more variables: microbiology <dbl>, spanish <dbl>
```

又怎么把它变回来 ???

# dplyr 常用函数示例

## 先创建一个新 tibble

```
grades <- 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)
                  );
grades;
```

```
## # A tibble: 7 x 6
##   Name          Occupation   English ComputerScience Biology Bioinformatics
##   <chr>         <chr>          <int>           <int>   <int>          <int>
## 1 Weihua Chen   Teacher           74              86      69             43
## 2 Mm Hu         Student           96              90      76             60
## 3 John Doe      Teacher           86              87      87             45
## 4 Jane Doe      Student           98              80      80             56
## 5 Warren Buffet Entrepreneur      70              89     100             46
## 6 Elon Musk     Entrepreneur      75              88      90             80
## 7 Jack Ma       Entrepreneur      80              83      99             51
```

# use gather & dplyr functions

Question: 1. 每个人平均成绩是多少？2. 哪个人的平均成绩最高？

```
grades.melted <- grades %>%
  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 );
```

```
## # A tibble: 7 x 4
## # Groups:   Name [7]
##   Name          Occupation   avg_grades courses_count
##   <chr>         <chr>             <dbl>          <int>
## 1 Elon Musk     Entrepreneur       83.2              4
## 2 Mm Hu         Student            80.5              4
## 3 Jane Doe      Student            78.5              4
## 4 Jack Ma       Entrepreneur       78.2              4
## 5 John Doe      Teacher            76.2              4
## 6 Warren Buffet Entrepreneur       76.2              4
## 7 Weihua Chen   Teacher            68                4
```

# use gather & dplyr functions

问题：每个人的最强科目是什么 ??

```
grades.melted %>%
  arrange( Name, -grade ) %>%
  group_by(Name) %>%
  summarise( avg_grades = mean( grade ), best_course = first( course ),
             best_grade = first( grade ) ) %>%
  arrange( -avg_grades );
```

```
## # A tibble: 7 x 4
##   Name          avg_grades best_course     best_grade
##   <chr>              <dbl> <chr>                <int>
## 1 Elon Musk           83.2 Biology                 90
## 2 Mm Hu               80.5 English                 96
## 3 Jane Doe            78.5 English                 98
## 4 Jack Ma             78.2 Biology                 99
## 5 John Doe            76.2 ComputerScience         87
## 6 Warren Buffet       76.2 Biology                100
## 7 Weihua Chen         68   ComputerScience         86
```

# dplyr::summarise 的其它操作

dplyr::**first**
First value of a vector.

dplyr::**last**
Last value of a vector.

dplyr::**nth**
Nth value of a vector.

dplyr::**n**
# of values in a vector.

dplyr::**n_distinct**
# of distinct values in a vector.

**IQR**
IQR of a vector.

**min**
Minimum value in a vector.

**max**
Maximum value in a vector.

**mean**
Mean value of a vector.

**median**
Median value of a vector.

**var**
Variance of a vector.

**sd**
Standard deviation of a vector.

**Figure 4:** dplyr::summarise 可用的操作

# 练习 & 作业

问题：

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）???

上交：

1. 产生的数据（导出为 tsv 格式）
2. 分析结果（copy & paste 到单独的文本文件里）
3. 完整的可独立运行的代码

# 更多练习，使用 starwars tibble

```
head(starwars);
```

```
## # A tibble: 6 x 13
##   name  height  mass hair_color skin_color eye_color birth_year gender
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr>
## 1 Luke~    172    77 blond      fair       blue             19   male
## 2 C-3PO    167    75 <NA>       gold       yellow          112   <NA>
## 3 R2-D2     96    32 <NA>       white, bl~ red              33   <NA>
## 4 Dart~    202   136 none       white      yellow           41.9 male
## 5 Leia~    150    49 brown      light      brown            19   female
## 6 Owen~    178   120 brown, gr~ light      blue             52   male
## # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

**note** 包含 87 行 13 列，星战部分人物的信息，包括身高、体重、肤色等

用 `?starwars` 获取更多帮助

# dplyr::mutate - 产生新列，不改变行数

而 dplyr::summarise 则会使列数减少（通常与 group_by 联合使用）

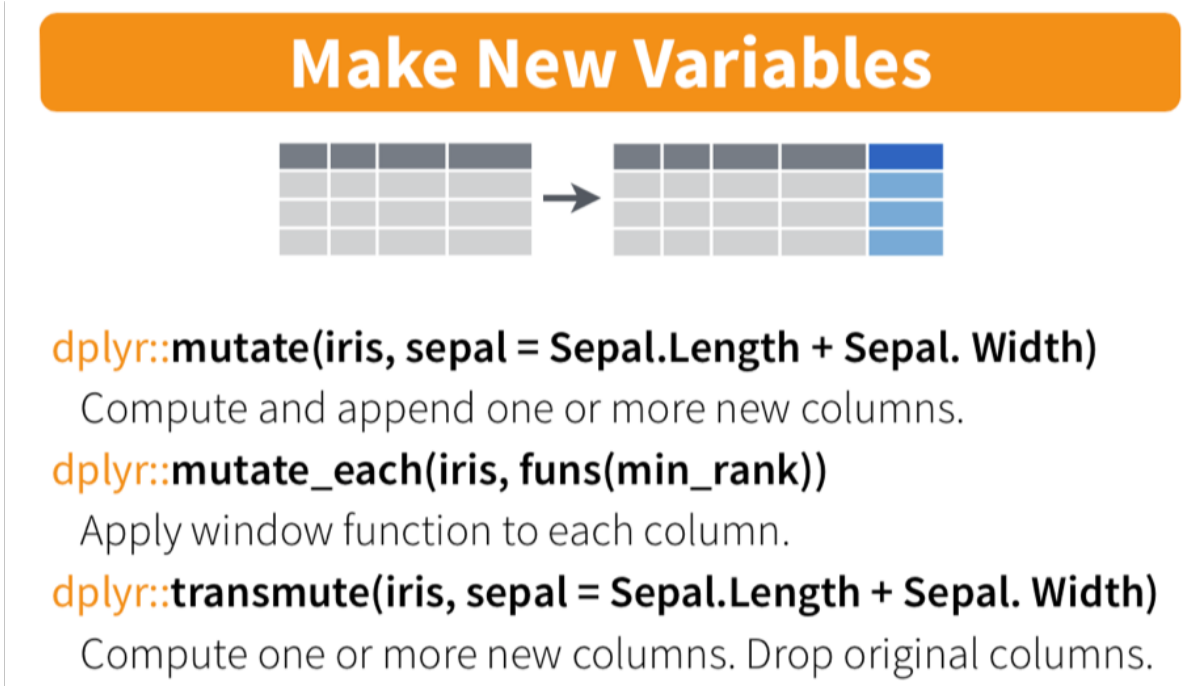


**Figure 5:** dplyr::mutate

另见下页的例子

# dplyr::select - 取列

目标：

- 取出相关列，用于计算人物的 BMI

```
stats <-
  starwars %>%
  select( name, height, mass ) %>%
  mutate( bmi = mass / ( (height / 100 ) ^ 2 ) ) ;

head(stats);
```

```
## # A tibble: 6 x 4
##   name           height  mass   bmi
##   <chr>           <int> <dbl> <dbl>
## 1 Luke Skywalker    172    77  26.0
## 2 C-3PO             167    75  26.9
## 3 R2-D2              96    32  34.7
## 4 Darth Vader       202   136  33.3
## 5 Leia Organa       150    49  21.8
## 6 Owen Lars         178   120  37.9
```

# dplyr::select - 取列, cont.

由于 name, height 和 mass 正好是相邻列，可以用 name:mass 获取：

```
stats <-
  starwars %>%
  select( name:mass ) %>%
  mutate( bmi = mass / ( (height / 100 ) ^ 2 ) ) ;

head(stats);
```

```
## # A tibble: 6 x 4
##   name           height  mass   bmi
##   <chr>           <int> <dbl> <dbl>
## 1 Luke Skywalker    172    77  26.0
## 2 C-3PO             167    75  26.9
## 3 R2-D2              96    32  34.7
## 4 Darth Vader       202   136  33.3
## 5 Leia Organa       150    49  21.8
## 6 Owen Lars         178   120  37.9
```

# dplyr::select - 取列, cont.

获取与颜色相关的列: hair_color, skin_color, eye_color

```
stats2 <- starwars %>%
  select( name, ends_with("color") );

head(stats2);
```

```
## # A tibble: 6 x 4
##   name           hair_color  skin_color  eye_color
##   <chr>          <chr>       <chr>       <chr>
## 1 Luke Skywalker blond       fair        blue
## 2 C-3PO          <NA>        gold        yellow
## 3 R2-D2          <NA>        white, blue red
## 4 Darth Vader    none        white       yellow
## 5 Leia Organa    brown       light       brown
## 6 Owen Lars      brown, grey light       blue
```

# dplyr::select - 去除列, cont.

请自行检查以下操作的结果

```
head( starwars %>% select( -hair_color, -eye_color )  );
```

# dplyr::select - 其它操作, cont.

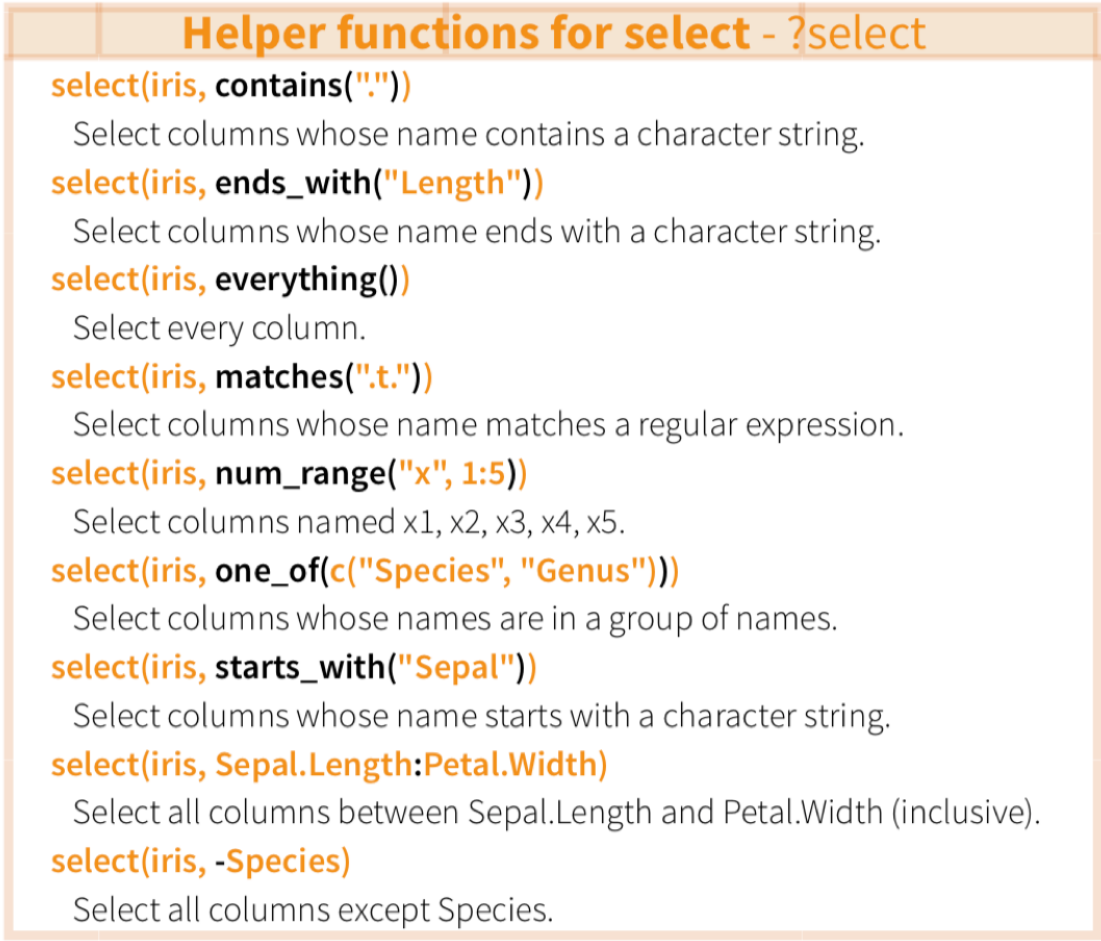


**Figure 6:** dplyr::select 支持的操作

# dplyr::filter - 行操作

任务：从星战中挑选金发碧眼的人物

```
starwars %>% select( name, ends_with("color"), gender, species ) %>%
  filter( hair_color == "blond" & eye_color == "blue" );
```

```
## # A tibble: 3 x 6
##   name             hair_color skin_color eye_color gender species
##   <chr>            <chr>      <chr>      <chr>     <chr>  <chr>
## 1 Luke Skywalker   blond      fair       blue      male   Human
## 2 Anakin Skywalker blond      fair       blue      male   Human
## 3 Finis Valorum    blond      fair       blue      male   Human
```

# dplyr 中其它取行的操作

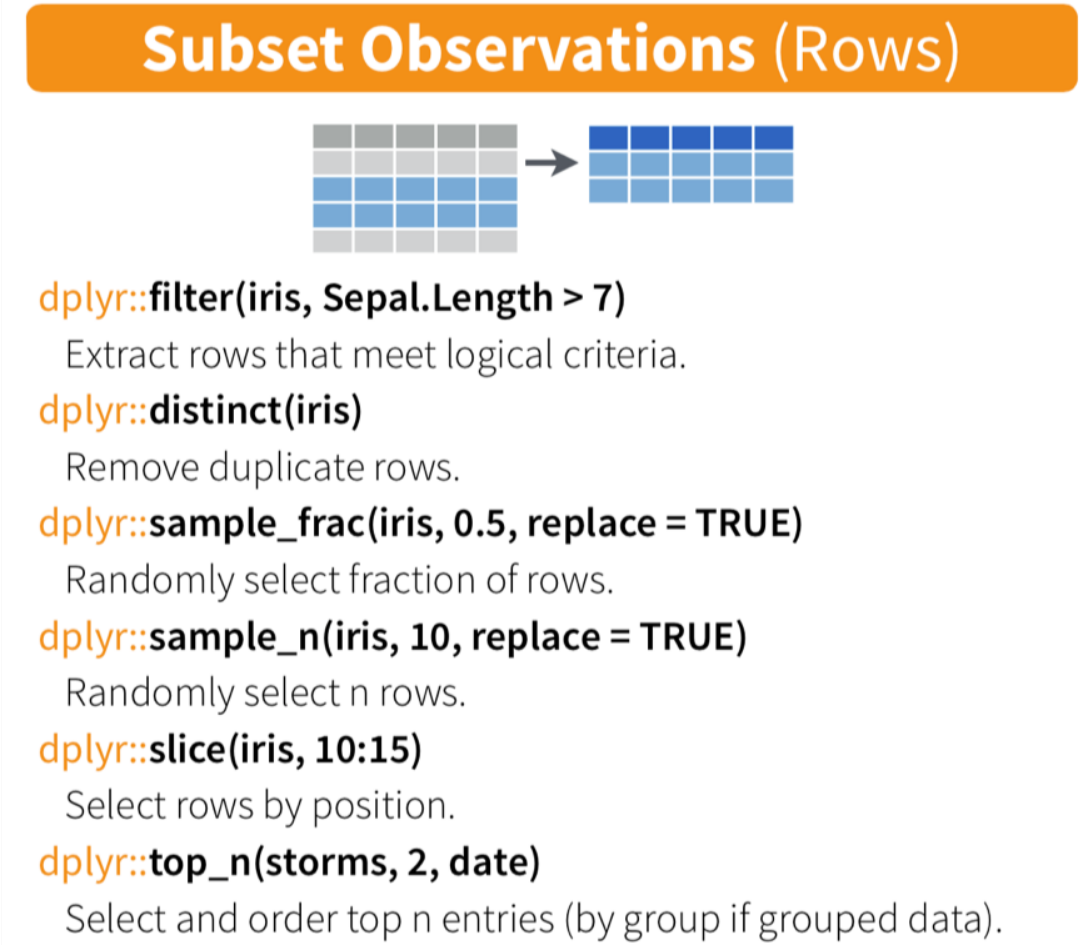


**Figure 7:** dplyr 与行相关的操作

# tidyr::separate

https://r4ds.had.co.nz/tidy-data.html

# tidyr::unite

https://r4ds.had.co.nz/tidy-data.html

# section 3：练习与作业

# more to read

https://www.dataschool.io/dplyr-tutorial-for-faster-data-manipulation-in-r/
https://pages.rstudio.net/Webinar-Series-Recording-Essential-Tools-for-R.html
https://github.com/tidyverse/dplyr http://genomicsclass.github.io/book/pages/dplyr_tutorial.html
http://stat545.com/block009_dplyr-intro.html https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

# 练习

1. let's get started with dplyr
2. dplyr: more smooth data exploration
3. some more exercise
4. dplyr: 50 examples; 包含了许多本章并未讲到的内容

# 作业

1. 前半部分提到的作业
2. 使用 mouse.tibble 数据，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，挑出最长和最短的基因
   - 去掉含有 500 以下基因的染色体，按染色体、数量高 -> 低进行排序

要求上交：

1. 完整能运行的代码，
2. 保存至文本文件的输出结果

# 下次提要

dplyr, tidyr 和 forcats 的更多功能与生信操作实例