R for bioinformatics, data wrangler, part 1

HUST Bioinformatics course series

Wei-Hua Chen (CC BY-NC 4.0)

27 September, 2021

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

问题点评

1. ggplot2 问题
2. 长宽数据转换 & pipe ...

```
N %>% gather( ind, values );
```

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,
- focusing on only data frames (also tibble),
- row-based manipulation,
- dplyr is faster and has a more consistent API.

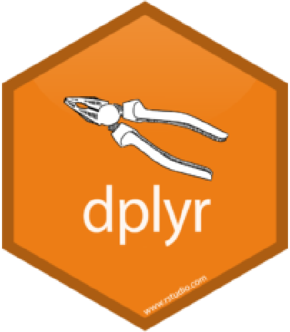


Figure 1: dplyr logo

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 = "" );
```

```
## 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.
```

查看 mouse.tibble 的内容

```
( ttype.stats <- mouse.tibble %>% count( `Transcript type` ) %>% arrange(-n) );
```

```
## # A tibble: 48 x 2
##    `Transcript type`           n
##    <chr>                   <int>
##  1 protein_coding          58384
##  2 retained_intron         21021
##  3 processed_transcript    15572
##  4 processed_pseudogene     9425
##  5 lincRNA                  8557
##  6 nonsense_mediated_decay  6755
##  7 antisense                4289
##  8 TEC                      3265
##  9 unprocessed_pseudogene   2650
## 10 miRNA                    2265
## # ... with 38 more rows
```

查看 mouse.tibble 的内容，cont.

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

```
## # A tibble: 117 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
## # ... with 107 more rows
```

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 ) );
```

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

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

灵活性：

- 应对不同学生选择不同课程的情况
- 可随时增加新的课程

长数据变宽

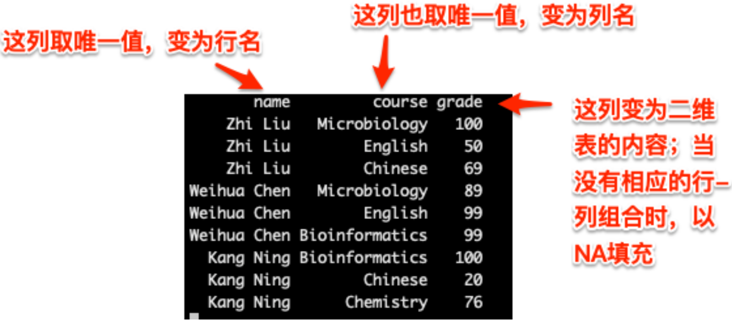
```
grades2 <- grades %>% spread( course, grade );
knitr::kable( grades2 );
```

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 );
```



这列取唯一值，变为行名

这列也取唯一值，变为列名

```
       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
```

这列变为二维表的内容；当没有相应的行–列组合时，以NA填充

Figure 2: spread function explained

宽数据转为长数据

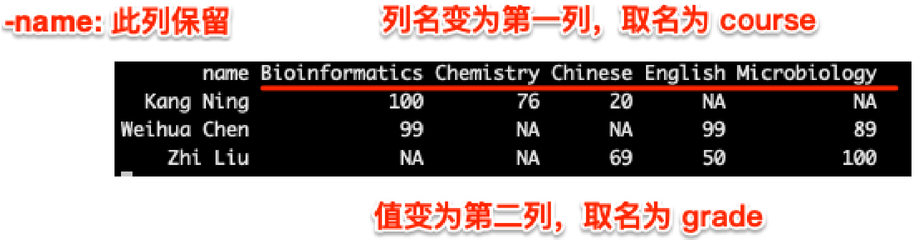
use gather() function in tidyr

```
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

长宽转换之 mouse.tibble

```
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

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

这是哪种数据类型？长还是宽？?

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           69              82      55             60
## 2 Mm Hu         Student           94              84      71             44
## 3 John Doe      Teacher           80              90      61             41
## 4 Jane Doe      Student           92              81      92             83
## 5 Warren Buffet Entrepreneur      63              89      60             73
## 6 Elon Musk     Entrepreneur      74              88      94             72
## 7 Jack Ma       Entrepreneur      91              80      69             47
```

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 );

## 检查数据 ...
knitr::kable( head(grades.melted) );
```

Name	Occupation	course	grade
Weihua Chen	Teacher	English	69
Mm Hu	Student	English	94
John Doe	Teacher	English	80
Jane Doe	Student	English	92
Warren Buffet	Entrepreneur	English	63
Elon Musk	Entrepreneur	English	74

成绩分析，cont

```
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 Jane Doe      Student            87               4
## 2 Elon Musk     Entrepreneur       82               4
## 3 Mm Hu         Student            73.2             4
## 4 Jack Ma       Entrepreneur       71.8             4
## 5 Warren Buffet Entrepreneur       71.2             4
## 6 John Doe      Teacher            68               4
## 7 Weihua Chen   Teacher            66.5             4
```

```
## 显示最终结果
knitr::kable( head( grades.melted ) );
```

Name	Occupation	course	grade
Weihua Chen	Teacher	English	69
Mm Hu	Student	English	94

use gather & dplyr functions

问题： 每个人的最强科目是什么 ??

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

knitr::kable( head(grades.melted2) );
```

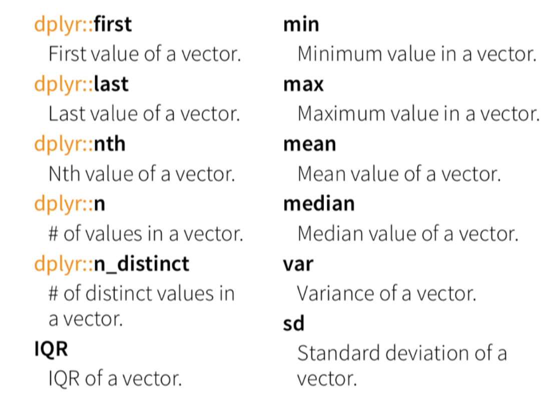
Name	Occupation	course	grade
Elon Musk	Entrepreneur	Biology	94
Elon Musk	Entrepreneur	ComputerScience	88
Elon Musk	Entrepreneur	English	74
Elon Musk	Entrepreneur	Bioinformatics	72
Jack Ma	Entrepreneur	English	91
Jack Ma	Entrepreneur	ComputerScience	80

最强科目问题，cont.

```
grades.melted2 %>%
  group_by(Name) %>%
  summarise( best_course = first( course ),
             best_grade = first( grade ),
             avg_grades = mean( grade ) ) %>%
  arrange( -avg_grades );
```

```
## # A tibble: 7 x 4
##   Name          best_course     best_grade avg_grades
##   <chr>         <chr>                <int>      <dbl>
## 1 Jane Doe      English                 92       87
## 2 Elon Musk     Biology                 94       82
## 3 Mm Hu         English                 94       73.2
## 4 Jack Ma       English                 91       71.8
## 5 Warren Buffet ComputerScience         89       71.2
## 6 John Doe      ComputerScience         90       68
## 7 Weihua Chen   ComputerScience         82       66.5
```

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

更多练习，使用 starwars tibble

```
head(starwars);
```

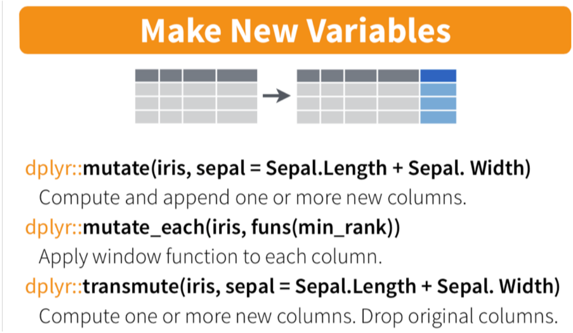
```
## # A tibble: 6 x 14
##   name     height  mass hair_color  skin_color eye_color birth_year sex   gender
##   <chr>     <int> <dbl> <chr>       <chr>      <chr>          <dbl> <chr> <chr>
## 1 Luke Sk~    172    77 blond       fair       blue            19   male  mascu~
## 2 C-3PO       167    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, grey light      blue            52   male  mascu~
## # ... 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）



Make New Variables

dplyr::**mutate(iris, sepal = Sepal.Length + Sepal. Width)**

Compute and append one or more new columns.

dplyr::**mutate_each(iris, funs(min_rank))**

Apply window function to each column.

dplyr::**transmute(iris, sepal = Sepal.Length + Sepal. Width)**

Compute one or more new columns. Drop original columns.

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);
```

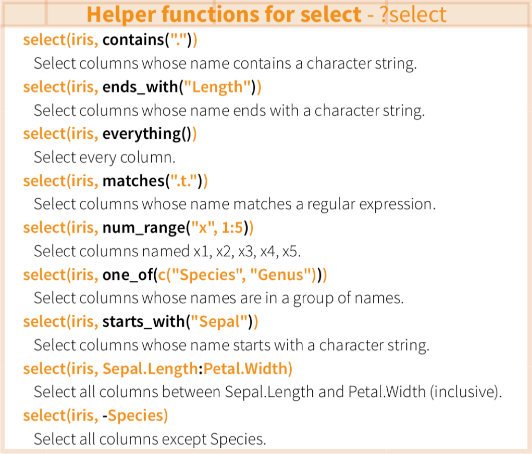
```
## # A tibble: 6 x 4
##   name           hair_color  skin_color  eye_color
##   <chr>          <chr>       <chr>       <chr>
## 1 Luke Skywalker blond       fair        blue
## 2 C-3PO          <NA>        gold        yellow
## 3 R2-D2          <NA>        white, blue red
## 4 Darth Vader    none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



Helper functions for select - ?select

`select(iris, contains("."))`

Select columns whose name contains a character string.

`select(iris, ends_with("Length"))`

Select columns whose name ends with a character string.

`select(iris, everything())`

Select every column.

`select(iris, matches(".t."))`

Select columns whose name matches a regular expression.

`select(iris, num_range("x", 1:5))`

Select columns named x1, x2, x3, x4, x5.

`select(iris, one_of(c("Species", "Genus")))`

Select columns whose names are in a group of names.

`select(iris, starts_with("Sepal"))`

Select columns whose name starts with a character string.

`select(iris, Sepal.Length:Petal.Width)`

Select all columns between Sepal.Length and Petal.Width (inclusive).

`select(iris, -Species)`

Select all columns except Species.

Figure 6: dplyr::select 支持的操作

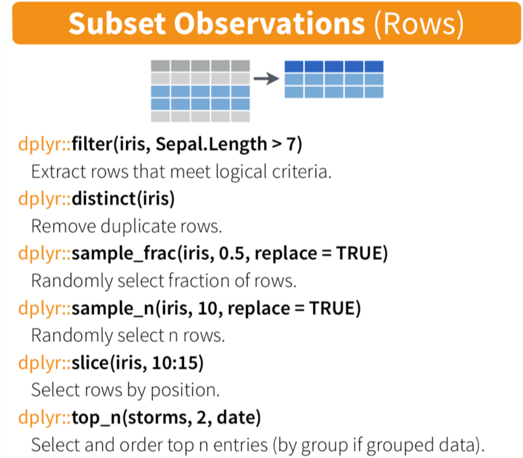
dplyr::filter - 行操作

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      masculine Human
## 2 Anakin Skywalker blond      fair       blue      masculine Human
## 3 Finis Valorum    blond      fair       blue      masculine Human
```

dplyr 中其它取行的操作



Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

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

练习 & 作业

- Exercises and homework 目录下 talk05-homework.Rmd 文件;
- 完成时间: 见钉群的要求

小结

今次提要

- dplyr 、tidyr (超级强大的数据处理) part 1

下次预告

- dplyr, tidyr 和 forcats 的更多功能与生信操作实例

important

- all codes are available at Github: https://github.com/evolgeniusteam/R-for-bioinformatics