

# R for bioinformatics, data wrangler, part 1

## HUST Bioinformatics course series

Wei-Hua Chen (CC BY-NC 4.0)

25 September, 2023

# 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

- pipe
- dplyr 、 tidyr (超级强大的数据处理) part 1

## section 2: pipe

# 什么是 pipe ?

- pipe 就是 `%>%`
- it comes from the `magrittr` package by **Stefan Milton Bache**
- Packages in the tidyverse load `%>%` for you automatically, so you don't usually load `magrittr` explicitly.
- 实质是中间值的传递

示例:

## 比如: 这段代码可以合并为:

```
library(tidyverse); ## 装入包
```

```
library(magrittr);
```

```
a <- subset( swiss, Fertility > 20 );
```

```
cor.test(a$Fertility, a$Education);
```

```
##
```

```
## Pearson's product-moment correlation
```

```
##
```

```
## data: a$Fertility and a$Education
```

```
## t = -5.9536, df = 45, p-value = 3.659e-07
```

```
## alternative hypothesis: true correlation is not equal to 0
```

```
## 95 percent confidence interval:
```

```
## -0.7987075 -0.4653206
```

```
## sample estimates:
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

# pipe 版本

```
## -- 新代码 ...
swiss %>%
  subset(., Fertility > 20) %$%
  cor.test( Education , Fertility );

##
## Pearson's product-moment correlation
##
## data: Education and Fertility
## t = -5.9536, df = 45, p-value = 3.659e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.7987075 -0.4653206
## sample estimates:
## cor
## -0.6637889
```



# 是否所有函数都支持 pipe ?

是的。

通常需要用 `.` 指代传递来的数据，并以参数的形式赋予下游函数：

```
swiss %>% do( head(., n = 4 ) );
```

```
##           Fertility Agriculture Examination Education Catholic
## Courtelary      80.2          17.0           15          12      9.96
## Delemont        83.1          45.1            6           9     84.84
## Franches-Mnt    92.5          39.7            5           5     93.40
## Moutier         85.8          36.5           12           7     33.77
##
##           Infant.Mortality
## Courtelary              22.2
## Delemont                22.2
## Franches-Mnt            20.2
## Moutier                 20.3
```

## 也可以写为

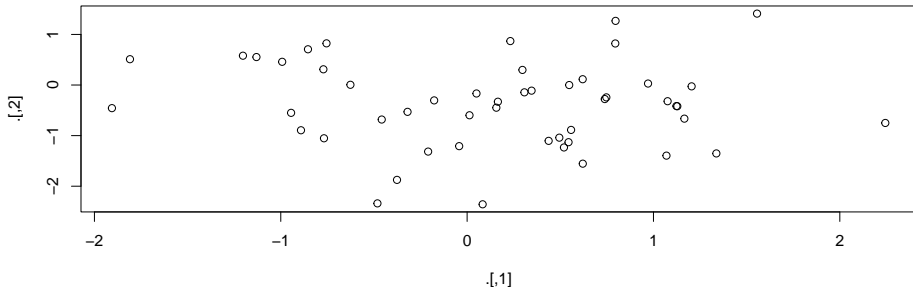
```
swiss %>% head(., n = 4 );
```

```
##           Fertility Agriculture Examination Education Catholic
## Courtelary      80.2          17.0           15          12      9.96
## Delemont        83.1          45.1            6           9     84.84
## Franches-Mnt    92.5          39.7            5           5     93.40
## Moutier         85.8          36.5           12           7     33.77
##
##           Infant.Mortality
```

# 其它形式的 pipe

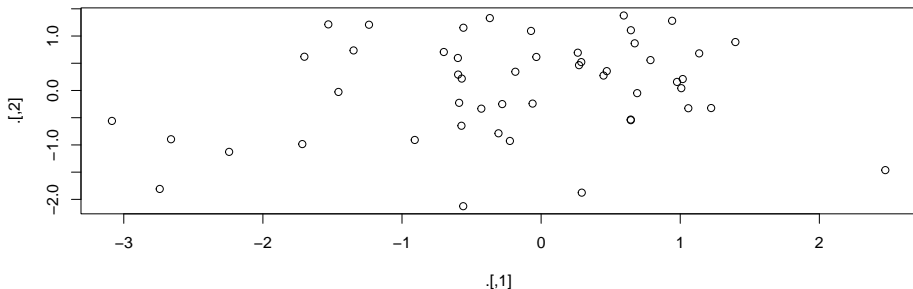
%T>% : 返回上游的值 (???)

```
## 示例: res1 是空值 ...
res1 <-
  rnorm(100) %>%
  matrix(ncol = 2) %>%
  plot();
```



## %T>%: 返回上游值 (left-side values)

```
## 示例: res2 是 matrix() 内容 ...  
res2 <-  
  rnorm(100) %>%  
  matrix(ncol = 2) %T>%  
  plot();
```



# %T>%: 返回上游值 (left-side values), cont.

```
head(res2);
```

```
##           [,1]      [,2]
## [1,] -0.06135298 -0.2408832
## [2,] -0.57274726 -0.6476997
## [3,] -1.34824906  0.7354253
## [4,]  0.44864168  0.2739711
## [5,]  1.05821923 -0.3261680
## [6,] -1.52959082  1.2139992
```

# %% : attach ???

```
attach( mtcars ); ## note the warning message ...
```

```
## The following object is masked from package:ggplot2:
##
##      mpg
```

```
cor.test( cyl, mpg ); ## 汽缸数与燃油效率
```

```
##
## Pearson's product-moment correlation
##
## data:  cyl and mpg
## t = -8.9197, df = 30, p-value = 6.113e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9257694 -0.7163171
## sample estimates:
##      cor
## -0.852162
```

# %% : attach ??? , cont.

```
detach( mtcars );
with( mtcars, cor.test( cyl, mpg ) );
```

```
##
## Pearson's product-moment correlation
##
## data:  cyl and mpg
## t = -8.9197, df = 30, p-value = 6.113e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9257694 -0.7163171
## sample estimates:
##      cor
## -0.852162
```

# %% : attach ??? , cont.

```
mtcars %$%
  cor.test( cyl, mpg );
```

```
##
## Pearson's product-moment correlation
##
## data:  cyl and mpg
## t = -8.9197, df = 30, p-value = 6.113e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9257694 -0.7163171
## sample estimates:
##      cor
## -0.852162
```

# 其它 pipe 及注意事项

```
## 双向 pipe  
mtcars %<>% transform(cyl = cyl * 2);
```

## 注

- pipe 的使用可以使思路更清晰
- 因此，尽量使用 %>%（方向明确），而不使用其它方向不明确的 pipe



## section 3: 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
## # i 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
## # i 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

# 课堂练习：使用常用函数解决问题

## 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      84          83          77          85
## 2 Mm Hu      Student      72          81          82          46
## 3 John Doe   Teacher      89          84          55          79
## 4 Jane Doe   Student      69          80          90          43
## 5 Warren Buffet Entrepreneur 95          89          54          89
## 6 Elon Musk  Entrepreneur 67          88          59          83
## 7 Jack Ma    Entrepreneur 71          82          88          59
```

# 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	84
Mm Hu	Student	English	72
John Doe	Teacher	English	89
Jane Doe	Student	English	69
Warren Buffet	Entrepreneur	English	95
Elon Musk	Entrepreneur	English	67

# 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 Weihua Chen Teacher      82.2           4
## 2 Warren Buffet Entrepreneur 81.8           4
## 3 John Doe    Teacher      76.8           4
## 4 Jack Ma     Entrepreneur 75             4
## 5 Elon Musk   Entrepreneur 74.2           4
## 6 Jane Doe    Student      70.5           4
## 7 Mm Hu       Student      70.2           4
```

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

Name	Occupation	course	grade
Weihua Chen	Teacher	English	84

# 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	ComputerScience	88
Elon Musk	Entrepreneur	Bioinformatics	83
Elon Musk	Entrepreneur	English	67
Elon Musk	Entrepreneur	Biology	59
Jack Ma	Entrepreneur	Biology	88
Jack Ma	Entrepreneur	ComputerScience	82

# 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 Weihua Chen  Bioinformatics    85       82.2
## 2 Warren Buffet English          95       81.8
## 3 John Doe     English          89       76.8
## 4 Jack Ma      Biology          88        75
## 5 Elon Musk    ComputerScience  88       74.2
## 6 Jane Doe     Biology          90       70.5
## 7 Mm Hu       Biology          82       70.2
```

# 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 2:** dplyr::summarise 可用的操作



# 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 3: dplyr 与行相关的操作**

# 练习考察

问题 1: 每个人的最差科目是什么 ??

## \*\* 列的使用! \*\*, 以 starwars tibble 为例

```
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 Sky~    172    77 blond      fair        blue         19  male  mascu~
## 2 C-3P0        167    75 <NA>      gold        yellow       112  none  mascu~
## 3 R2-D2         96    32 <NA>      white, bl~  red          33  none  mascu~
## 4 Darth Va~    202   136 none      white       yellow       41.9  male  mascu~
## 5 Leia Org~    150    49 brown     light       brown        19  fema~  femin~
## 6 Owen Lars    178   120 brown, gr~ light       blue         52  male  mascu~
## # i 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

## 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 4:** 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-3P0             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 5: dplyr::select 支持的操作**

# 同时对行列进行操作

## 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
```

# tidyr::separate

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

# tidyr::unite

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

## 练习考察

问题 2: 从 `starwars` 中选出  $18 < \text{bmi} < 25$  的人物, 统计他们的 `homeworld` 分布情况

提示: 1. 需计算 `bmi`; 2. 用 `count` 函数计算 `homeworld` 的分布

## section 4 : 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

- pipe
- dplyr 、 tidyr (超级强大的数据处理) part 1

## 下次预告

- 长宽数据转换
- 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>