

Data transformation with dplyr :: CHEAT SHEET

%>% 的作用在于把前面的内容放到后面函数中，作为第一个参数。使用这个符号的好处有：

1. 使代码更加易读
2. 减少中间变量

dplyr functions work with pipes and expect tidy data. In tidy data:



Each **variable** is in its own **column**



Each **observation**, or **case**, is in its own **row**



pipes

`x %>% f(y)` becomes `f(x, y)`

Summarise Cases

Apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function



`summarise(.data, ...)`
Compute table of summaries.
`summarise(mtcars, avg = mean(mpg))`

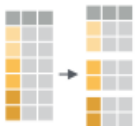


`count(.data, ..., wt = NULL, sort = FALSE, name = NULL)` Count number of rows in each group defined by the variables in ... Also `tally()`.
`count(mtcars, cyl)`

计算mtcars的mpg的均值

Group Cases

Use `group_by(.data, ..., .add = FALSE, .drop = TRUE)` to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.



`mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))`

把mtcars按照cyl分类，然后计算每一类中mpg的均值

Use `rowwise(.data, ...)` to group data into individual rows. dplyr functions will compute results for each row. Also apply functions to list-columns. See tidy cheat sheet for list-column workflow.



`starwars %>%
rowwise() %>%
mutate(film_count = length(films))`

Starwars是一个list，这里需要特别注意。把starwars这个list转变成一个table然后加了另外一列count原本starwars list里面每一个film的个数。

`ungroup(x, ...)` Returns ungrouped copy of table.
`ungroup(g_mtcars)`

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.



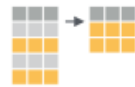
`filter(.data, ..., .preserve = FALSE)` Extract rows that meet logical criteria.
`filter(mtcars, mpg > 20)`

选择mtcars里面mpg>20的行。



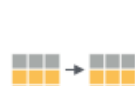
`distinct(.data, ..., .keep_all = FALSE)` Remove rows with duplicate values.
`distinct(mtcars, gear)`

剔除重复的行

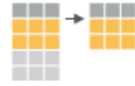


`slice(.data, ..., .preserve = FALSE)` Select rows by position.
`slice(mtcars, 10:15)`

Slice是一个很有用的语句，在这里slice用来按照你想要的数据的行位置进行选择。这里选择10：15行。



`slice_sample(.data, ..., n, prop, weight_by = NULL, replace = FALSE)` Randomly select rows. Use `n` to select a number of rows and `prop` to select a fraction of rows.
`slice_sample(mtcars, n = 5, replace = TRUE)`



`slice_min(.data, order_by, ..., n, prop, with_ties = TRUE)` and `slice_max()` Select rows with the lowest and highest values.
`slice_min(mtcars, mpg, prop = 0.25)`

`slice_head(.data, ..., n, prop)` and `slice_tail()` Select the first or last rows.
`slice_head(mtcars, n = 5)`

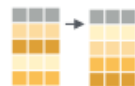
选择数据最前面或者最后的几行查看。

Logical and boolean operators to use with filter()

<code>==</code>	<code><</code>	<code><=</code>	<code>is.na()</code>	<code>%in%</code>	<code> </code>	<code>xor()</code>
<code>!=</code>	<code>></code>	<code>>=</code>	<code>!is.na()</code>	<code>!</code>	<code>&</code>	

See `?base::Logic` and `?Comparison` for help.

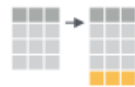
ARRANGE CASES



`arrange(.data, ..., .by_group = FALSE)` Order rows by values of a column or columns (low to high). Use `with_desc()` to order from high to low.
`arrange(mtcars, mpg)`
`arrange(mtcars, desc(mpg))`

arrange排列数据。这里是重新排列mtcars这个数据按照mpg值从小到大，desc()是从大到小。

ADD CASES



`add_row(.data, ..., .before = NULL, .after = NULL)` Add one or more rows to a table.
`add_row(cars, speed = 1, dist = 1)`

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.



`pull(.data, var = -1, name = NULL, ...)` Extract column values as a vector, by name or index.
`pull(mtcars, wt)`

Pull可以把原本行的内容提取出来变成一个vector



`select(.data, ...)` Extract columns as a table.
`select(mtcars, mpg, wt)`

Select把table里的列提取出来转变成另一个table



`relocate(.data, ..., .before = NULL, .after = NULL)` Move columns to new position.
`relocate(mtcars, mpg, cyl, .after = last_col())`

relocate把table的列移到其他位置

Use these helpers with `select()` and `across()`

e.g. `select(mtcars, mpg:cyl)`

<code>contains(match)</code>	<code>num_range(prefix, range)</code>	: e.g. <code>mpg:cyl</code>
<code>ends_with(match)</code>	<code>all_of(x)/any_of(x, ..., vars)</code>	- e.g. <code>-gear</code>
<code>starts_with(match)</code>	<code>matches(match)</code>	<code>everything()</code>

MANIPULATE MULTIPLE VARIABLES AT ONCE



`across(.cols, .funs, ..., .names = NULL)` Summarise or mutate multiple columns in the same way.
`summarise(mtcars, across(everything(), mean))`

across是一个很有用的语句，across可以帮助我们一次计算table里所有列我们所需要的statistics。这句summarise(mtcars, across(everythin(), mean))的意思是计算mtcars这个table所有列的均值。这个语句最终生成的是一个table。



`c_across(.cols)` Compute across columns in row-wise data.
`transmute(rowwise(UKgas), total = sum(c_across(1:2)))`

按照行计算所需要的statistics

MAKE NEW VARIABLES

Apply **vectorized functions** to columns. Vectorized functions take vectors as input and return vectors of the same length as output (see back).

vectorized function



`mutate(.data, ..., .keep = "all", .before = NULL, .after = NULL)` Compute new column(s). Also **`add_column()`**, **`add_count()`**, and **`add_tally()`**.
`mutate(mtcars, gpm = 1 / mpg)`

为mtcars数据增加另一列 (1/mpg)



`transmute(.data, ...)` Compute new column(s), drop others.
`transmute(mtcars, gpm = 1 / mpg)`

建立另外一个table，这个table只有一列是(1/mpg)



`rename(.data, ...)` Rename columns. Use **`rename_with()`** to rename with a function.
`rename(cars, distance = dist)`

为数据里面的列的名字重新命名。

Vectorized Functions

TO USE WITH MUTATE ()

mutate() and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function

OFFSET

dplyr::**lag()** - offset elements by 1
dplyr::**lead()** - offset elements by -1

CUMULATIVE AGGREGATE

dplyr::**cumall()** - cumulative all()
dplyr::**cumany()** - cumulative any()
dplyr::**cummax()** - cumulative max()
dplyr::**cummean()** - cumulative mean()
dplyr::**cummin()** - cumulative min()
dplyr::**cumprod()** - cumulative prod()
dplyr::**cumsum()** - cumulative sum()

RANKING

dplyr::**cume_dist()** - proportion of all values <=
dplyr::**dense_rank()** - rank w ties = min, no gaps
dplyr::**min_rank()** - rank with ties = min
dplyr::**ntile()** - bins into n bins
dplyr::**percent_rank()** - min_rank scaled to [0,1]
dplyr::**row_number()** - rank with ties = "first"

MATH

+, -, *, /, ^, %/%, %% - arithmetic ops
log(), **log2()**, **log10()** - logs
<, <=, >, >=, !=, == - logical comparisons
dplyr::**between()** - x >= left & x <= right
dplyr::**near()** - safe == for floating point numbers

MISCELLANEOUS

dplyr::**case_when()** - multi-case if_else()

```
starwars %>%  
  mutate(type = case_when(  
    height > 200 | mass > 200 ~ "large",  
    species == "Droid" ~ "robot",  
    TRUE ~ "other")  
  )
```

dplyr::**coalesce()** - first non-NA values by element across a set of vectors
dplyr::**if_else()** - element-wise if() + else()
dplyr::**na_if()** - replace specific values with NA
dplyr::**pmax()** - element-wise max()
dplyr::**pmin()** - element-wise min()

Summary Functions

TO USE WITH SUMMARISE ()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function

COUNT

dplyr::**n()** - number of values/rows
dplyr::**n_distinct()** - # of uniques
dplyr::**sum(!is.na())** - # of non-NA's

POSITION

mean() - mean, also **mean(!is.na())**
median() - median

LOGICAL

mean() - proportion of TRUE's
sum() - # of TRUE's

ORDER

dplyr::**first()** - first value
dplyr::**last()** - last value
dplyr::**nth()** - value in nth location of vector

RANK


quantile() - nth quantile
min() - minimum value
max() - maximum value


SPREAD

IQR() - Inter-Quartile Range
mad() - median absolute deviation
sd() - standard deviation
var() - variance

Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.


tibble::**rownames_to_column()**
Move row names into col.
a <- rownames_to_column(mtcars,
var = "C")


tibble::**column_to_rownames()**
Move col into row names.
column_to_rownames(a, var = "C")

Also tibble::**has_rownames()** and
tibble::**remove_rownames()**.

计算lag和lead, 在time series 分析会经常用到。

计算cumulative statistics

排列数据

把startwars这个list变成一个table, 并且建立另一个列, 基于height, mass, 和species这几个列, 如果height>200或者mass>200, type=large, 如果species是Droid, type=robot, 其他情况是other.

数一行或者一列有多少
数一行或者一列有多少不重复的个数
数一行或者一列有多少不是缺失的数据

计算均值和中位数

计算分位数, 最小值, 最大值。

把原本的rownames转变成另一个column。

把上述再转回来

Combine Tables

COMBINE VARIABLES

x	y	
A B C	E F G	A B C E F G
a t 1	a t 3	a t 1 a t 3
b u 2	b u 2	b u 2 b u 2
c v 3	d w 1	c v 3 d w 1

bind_cols(..., .name_repair) Returns tables placed side by side as a single table. Column lengths must be equal. Columns will NOT be matched by id (to do that look at Relational Data below), so be sure to check that both tables are ordered the way you want before binding.

RELATIONAL DATA

Use a "Mutating Join" to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

left_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ..., keep = FALSE, na_matches = "na") Join matching values from y to x.

right_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ..., keep = FALSE, na_matches = "na") Join matching values from x to y.

inner_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ..., keep = FALSE, na_matches = "na") Join data. Retain only rows with matches.

full_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ..., keep = FALSE, na_matches = "na") Join data. Retain all values, all rows.

COMBINE CASES

x	y	
A B C	A B C	A B C
a t 1	a t 1	a t 1
b u 2	b u 2	b u 2
c v 3	c v 3	c v 3
d w 4	d w 4	d w 4

bind_rows(..., .id = NULL) Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names (as pictured).

Use a "Filtering Join" to filter one table against the rows of another.

x	y	
A B C	A B D	
a t 1	a t 3	
b u 2	b u 2	
c v 3	d w 1	

semi_join(x, y, by = NULL, copy = FALSE, ..., na_matches = "na") Return rows of x that have a match in y. Use to see what will be included in a join.

anti_join(x, y, by = NULL, copy = FALSE, ..., na_matches = "na") Return rows of x that do not have a match in y. Use to see what will not be included in a join.

Use a "Nest Join" to inner join one table to another into a nested data frame.

nest_join(x, y, by = NULL, copy = FALSE, keep = FALSE, name = NULL, ...) Join data, nesting matches from y in a single new data frame column.

SET OPERATIONS

intersect(x, y, ...) Rows that appear in both x and y.

setdiff(x, y, ...) Rows that appear in x but not y.

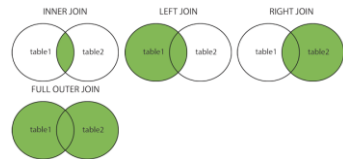
union(x, y, ...) Rows that appear in x or y. (Duplicates removed). **union_all()** retains duplicates.

Use **setequal()** to test whether two data sets contain the exact same rows (in any order).

把两个table按照列合并成一个table。这种情况下，多数需要保证x和y每一行有相同的ID。

把两个table按照行合并，一般情况下要保证两个table有相同的列。

这里的left join, right join, etc. 和SQL的概念是一样的。具体可以参考下图：



在合并的过程中加一些条件，比如值保留有相同值的行，或者只保留没有相同值的行。

基于不同的columns进行match

按照如图所示的方法合并两个table。

COLUMN MATCHING FOR JOINS

by = c("col1", "col2", ...) Use to specify one or more common columns to match on. **left_join(x, y, by = "A")**

by = c("col1" = "col2") Use a named vector, to match on columns that have different names in each table. **left_join(x, y, by = c("C" = "D"))**

suffix to specify the suffix to give to unmatched columns that have the same name in both tables. **left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))**