使用DPLYR进行数据转换

Data Transformation with dplyr : : CHEAT SHEET

dplyr要求数据符合tidy data原则。使用%>%(pipe) 符号在函数之间对数据进行传送。
tidy data的特点：

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

Each **variable** is in its own **column**
每一列是一个变量

&

Each **observation**, or **case**, is in its own **row**
每一行是一个记录

pipes

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

pipe符号的作用是把x作为一个参数传给下一个函数

记录汇总

这些是将 **summary** 方法应用 于数据表的列生成新的包含统计值的数据表. Summary 支持各种以向量作为输入值返回标量值的函数(详见后页).

summary 支持的函数

summarise(.data, …)
计算汇总统计表
summarise(mtcars, avg = mean(mpg))

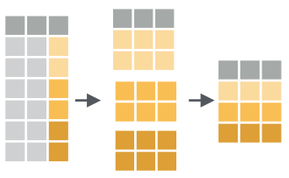
count(x, ..., wt = NULL, sort = FALSE)
根据...定义的变量的分组，计算行的数量，另参见 **tally**().
count(iris, Species)

变体

summarise_all() - 对所有的列应用汇总函数
summarise_at() - 对特定列应用汇总函数
summarise_if() - 对特定类型的列应用汇总函数

记录分组

用 **group_by()** 生成一个表格“分组后”的副本，之后 dplyr 会分别对各组单独进行操作，之后再汇总进



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

group_by(.data, ..., add = FALSE)
返回按照...分组后的副本

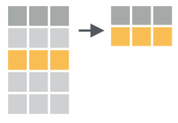
g_iris <- group_by(iris, Species)

ungroup(x, …)
group_by的逆转函数
返回未分组的数据
ungroup(g_iris)

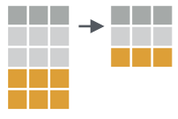
记录数据处理（行）

提取部分记录（筛选）

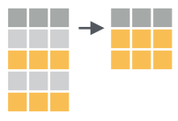
这些函数会返回数据表的部分行作为数据框的子集



filter(.data, …) 返回符合...逻辑判定的行
filter(iris, Sepal.Length > 7)

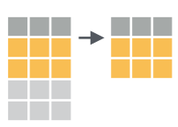


distinct(.data, ..., .keep_all = FALSE) 不仅进行筛选，同时还去处结果中的重复行
distinct(iris, Species)



sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) 从数据表抽取一定比例的行
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) 从数据表中随机抽取size行*sample_n(iris, 10, replace = TRUE)*



slice(.data, …) 使用行序号选取行
slice(iris, 10:15)

top_n(x, n, wt) 选择前n行(如果是分组表格，返回各组前n行). *top_n(iris, 5, Sepal.Width)*

filter()可以使用的逻辑和布尔运算符

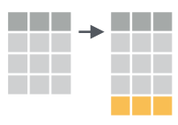
<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

参考**?base::logic** 和 **?Comparison** 阅读进一步说明

排列

arrange(.data, …) 依据某列的值从小到大排序, 如果要从大到小排序，在列名外加上 **desc()**
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

添加行



add_row(.data, ..., .before = NULL, .after = NULL) 为数据表添加更多记录
add_row(faithful, eruptions = 1, waiting = 1)

处理变量（列）

提取特定变量

按照需求返回列的各种函数

pull(.data, var = -1) 将一列的数值提取成向量. 按照列名称或序号提取
pull(iris, Sepal.Length)

select(.data, …)
将提取的内容提取成表格. 另见**select_if()**.
select(iris, Sepal.Length, Species)

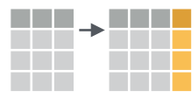
下面这些辅助函数有助于 **select ()**的使用,
e.g. select(iris, starts_with("Sepal"))

contains(match)
ends_with(match)
matches(match)
num_range(prefix, range)
one_of(…)
starts_with(match)
:, e.g. mpg:cyl
-, e.g, -Species

生成新的变量

下面这些将 向量函数应用于 各列. 向量函数以向量为输入返回相同长度的向量作为输出(详细见后页).

vectorized function



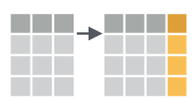
mutate(.data, …)
计算新的变量，加入当前表格.
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …)
计算新的变量，舍去其余的变量
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …) 将函数应用于所有列 通过命名列表可以提供多个函数另见**mutate_if()**
mutate_all(faithful, funs(log(.), log2(.)))
mutate_if(iris, is.numeric, funs(log(.)))

mutate_at(.tbl, .cols, .funs, …) 将多个函数应用于特定列. vars以及select辅助函数联合使用协助选择

mutate_at(iris, vars(-Species), funs(log(.)))



add_column(.data, ..., .before = NULL, .after = NULL) 添加新的变量 另见 **add_count()**, **add_tally()**. *add_column(mtcars, new = 1:32)*

rename(.data, …) 重命名变量
rename(iris, Length = Sepal.Length)

向量函数

与 **MUTATE ()**共同使用

mutate() 和 **transmute()** 利用向量函数
将现有数据列转换成新的数据列
向量函数以向量作为输入
返回同样长度的向量

vectorized function

偏移

dplyr::**lag()** - ff将元素偏移1个单位
dplyr::**lead()** - Off将所有元素向前偏移一个单位

累积计算

dplyr::**cumall()** - 直到第一个FALSE前的所有
dplyr::**cumany()** 第一个TRUE之后的所有
cummax() 累积最大值
dplyr::**cummean()**累积均数
cummin()累积最小值
cumprod() 累积阶乘
cumsum() -累积求和

排序

dplyr::**cume_dist()** - 小于等于当前值占多少比例
dplyr::**dense_rank()** - 排序，相同大小时统一采用最小，无空隙
dplyr::**min_rank()** 排序，相同大小时统一采用最小
dplyr::**ntile()** - 分成n等分后所在第几份
dplyr::**percent_rank()** - min_rank 缩小到 [0,1] 区间
dplyr::**row_number()** - 排序，相同大小时最先出现的排在前面

数学计算

+, - , *, /, ^, %/%, %% - 常规R符号
log(), log2(), log10() - logs
<, <=, >, >=, !=, == - 逻辑比较
dplyr::**between()** - x >= left & x <= right
dplyr::**near()** - 用于浮点数的 ==，不容易因为精度出错

其他

dplyr::**case_when()** - 多情况的 if_else()
dplyr::**coalesce()** 多个向量，第一个不是NA的element 就获得采用
dplyr::**if_else()** - 向量版的 if() + else()
dplyr::**na_if()** - 将特定值替换为 NA
pmax() - 多个向量逐个元素的max()
pmin() - 多个向量逐个元素的min()
dplyr::**recode()** - 向量版本的switch()
dplyr::**recode_factor()** - 向量版本的switch()用于因子类型

汇总函数

与 **SUMMARISE ()**共同使用

summarise() 将汇总函数应用于数据列
生成新的数据表。汇总函数的输入是向量
返回值是标量

.

summary function

计数

dplyr::**n()** - 值/行的数量
dplyr::**n_distinct()** - 唯一值的数量
sum(!is.na()) - 非NA值的数量

平均值和中位数

mean() - 平均值， **mean(!is.na())**
median() - 中位数

LOGICALS

mean() - 真值的比例
sum() - 真值的数量

顺序

dplyr::**first()** - 首位值
dplyr::**last()** - 末尾值
dplyr::**nth()** - 向量的第n个值

排序

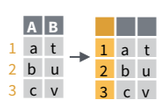
quantile() - 第n个25%
min() - 最小值
max() - 最大值

离散

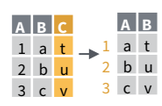
IQR() - 1/4到3/4之间的距离
mad() - 中位数绝对偏差
sd() - 标准差
var() - 方差

行名称

tidyverse数据不使用行名称（一种储存在数据表外的变量。如果要使用行名称的变量，首先把行名称变成一列



rownames_to_column()
将一个行名称转换成变量.
a <- rownames_to_column(iris, var = "C")

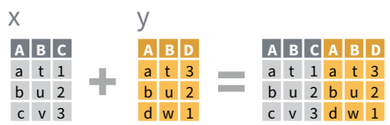


column_to_rownames()
将一个变量转换成行名称
column_to_rownames(a, var = "C")

Also **has_rownames()**, **remove_rownames()**

表组合

COMBINE VARIABLES



使用**bind_cols()** 将两个表左右贴合
类似cbind()

bind_cols(...) Returns tables placed side by side as a single table.
贴合以前确保两个表的行数一样

用 " **Mutating Join**"将两个表按照特定列进行组合。不同的join在两个表的内容保存上有所差别

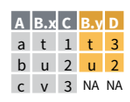
ff

left_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...)
x的全保留， y 无法match的设置为na

right_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),...)
y全保留， x无法match的设置为na

inner_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),...)
只保留两表匹配的行

full_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...)
保留两表所有的行

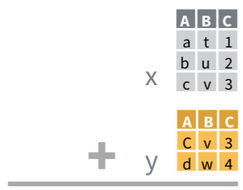


用 **by = c("col1", "col2")** 来指定基于什么列来进行匹配
left_join(x, y, by = "A")

如果两个列名称不同**by = c("col1" = "col2")**,可以通过命名向量解决这个问题
diff
left_join(x, y, by = c("C" = "D"))

suffix 可以指定给名称重复时添加的后缀
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

COMBINE CASES



用 **bind_rows()** 将两个表上下贴合

bind_rows(..., .id = NULL)
将两个表上下叠加
.id非空的时候，会额外添加一列记录表格来源

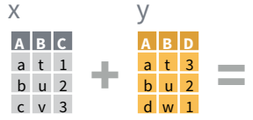
intersect(x, y, ...)
x和y表共有的行

setdiff(x, y, ...)
x有但是 y没有的表

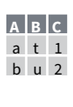
union(x, y, ...)
x和y贴合，但是移除重复值

用 **setequal()** 可以检测两个表是否含有同样的行（顺序不同也可以检测）

EXTRACT ROWS



Filtering Join" 可以将一个数据表的内容通过join与另一个表进行比对，从而实现fiter



semi_join(x, y, by = NULL, ...)
返回x中可以和y join 的行
这个可以用来预览什么行会被join

anti_join(x, y, by = NULL, ...)
返回x中不会被join的行
这可可以用来预览什么行不会被join