# DataFrames.jl 速查表

(for version 1.x)

## 整齐的数据 - 数据整理的基础

在一个整齐的数据集中:


每个 变量(**variable**)被保存在一个对应的列(column)中.

&


每条记录被保存在自己的行(row)中.

整齐的数据会使数据分析更简单更直观.
DataFrames.jl 可以帮助整理你的数据.

需要详细说明,请查看DataFrames.jl 官方文档:
http://juliadata.github.io/DataFrames.jl/latest/

## 创建数据框(DataFrame)

**DataFrame(x = [1,2,3], y = 4:6, z = 9)**
通过 向量, range 或 常量 格式的列数据创建数据框.

**DataFrame([(x=1, y=2), (x=3, y=4)])**
通过 命名元组的列表创建数据框.

**DataFrame("x" => [1,2], "y" => [3,4])**
通过 键-值 对(k-v pairs) 来创建数据框.

**DataFrame(rand(5, 3), [:x, :y, :z])**
**DataFrame(rand(5, 3), :auto)**
通过矩阵创建数据框.

**DataFrame()**
创建一个没有任何列的空数据框.

**DataFrame(x = Int[], y = Float64[])**
创建一个指定列名和列类型的数据框.

**DataFrame(mytable)**
通过一个支持Tables.jl接口的数据源来创建数据框.

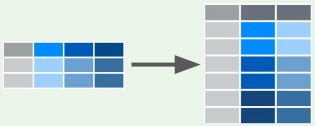
## 数据框 描述

**describe(df)**
所有列的统计汇总信息.

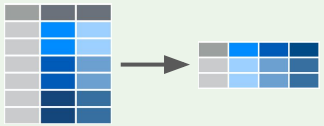
**describe(df, :mean, :std)**
所有列上的指定统计信息.

**describe(df, extrema => :extrema)**
所有列上的自定义统计信息.

## 修改数据形状 - 改变布局



**stack(df, [:sibsp, :parch])**
将列数据转换为行数据, 并指定新的列名.



**unstack(df, :variable, :value)**
将行数据转换为列数. :variable, :value是两个默认参数.

## 数据排序

***原地修改**: 用 sort!*

**sort(df, :age)**
按age列升序(默认)排序

**sort(df, :age, rev = true)**
按age列倒序排序

**sort(df, [:age, order(:sibsp, rev = true)])**
按 age升序和 *sibsp*降序排序

## 按行选择数据

### 函数方法

**first(df, 5)** or **last(df, 5)**
查看df的前5行 或 最后5行.

**unique(df)**
**unique(df, [:pclass, :survived])**
返回数据框中唯一的,独有的行数据.

**filter(:sex => ==("male"), df)**
**filter(row -> row.sex == "male", df)**
返回符合 *sex* == "male" 的所有行数据.
*注意: 第一种的语法执行性能更好.*

**subset(df, :survived)**
**subset(df, :sex => x -> x .== "male")**
返回 符合条件(判断为真)的所有行.
*注意: "survived" 列是 Bool类型的*

### 索引方法

**df[6:10, :]**
返回 6 to 10 行的数据

**df[df.sex .== "male", :]**
返回 *sex*列值等于"male"的所有行.

**df[findfirst(==(30), df.age), :]**
返回*age* 等于30的第一行.

**df[findall(==(1), df.pclass), :]**
返回*pclass* 列值等于1的所有行.

***原地修改**: 用 unique!, filter!, 或 subset!*

## 按列选择数据

### 函数方法

**select(df, :sex)**
**select(df, "sex")**
**select(df, [:sex, :age])**
查询想要的列.

**select(df, 2:5)**
通过索引查询某些列.

**select(df, r"^s")**
通过正则表达式查询某些列.

**select(df, Not(:age))**
查询除了*age*以外的所有列.

**select(df, Between(:name, :age))**
查询 *name* 和*age* 之间的所有列.

### 索引方法

**df[:, [:sex, :age]]**
所选列的数据副本.

**df[!, [:sex, :age]]**
所选列的原始列向量.

*注释: 索引方法可以同时选择行和列!*

***原地修改**: 用 select!*

## 查看元数据

**names(df)**
**propertynames(df)**
返回所有列名.

**nrow(df)**
**ncol(df)**
获取行数和列数

**columnindex(df, "sex")**
返回指定列的索引

## 处理缺失数据

**dropmissing(df)**
**dropmissing(df, [:age, :sex])**
返回没有缺失值的所有行.

**allowmissing(df)**
**allowmissing(df, :sibsp)**
允许指定列中有缺失值

**disallowmissing(df)**
**disallowmissing(df, :sibsp)**
不允许指定列中有缺失值.

**completecases(df)**
**completecases(df, [:age, :sex])**
返回Bool数组, 行中没有缺失值的都返回为true.

***原地修改**: 用 dropmissing!, allowmissing!, disallowmissing!*

# 累积统计和滑动统计

## 累积统计

**select(df, :x => cumsum)**
**select(df, :x => cumprod)**
查询 *x列的累计和 及 累计积.*

**select(df, :x => v -> accumulate(min, v))**
**select(df, :x => v -> accumulate(max, v))**
获取 *列x 中的累计 最大值 和 最小值.*

**select(df, :x => v -> cumsum(v) ./ (1:length(v)))**
获取列 *x的上地累计平均值.*

## 滑动统计 (也叫滚动统计,滑窗统计)

**select(df, :x => (v -> runmean(v, n)))**
**select(df, :x => (v -> runmedian(v, n)))**
**select(df, :x => (v -> runmin(v, n)))**
**select(df, :x => (v -> runmax(v, n)))**
计算 列x 上 滑动窗口大小 *n* 的滑动平均, 中值 和 最大值.

*这些 **run*** 滑窗函数(或更多) 可以从RollingFunctions.jl 包中获取.*

# Ranking and Lead/Lag 函数

**select(df, :x => ordinalrank)** # 1234
**select(df, :x => competerank)** # 1224
**select(df, :x => denserank)** # 1223
**select(df, :x => tiedrank)** # 1 2.5 2.5 4

*这些 ***rank** 函数 来自 StatsBase.jl 包.*

**select(df, :x => lead)** # shift up
**select(df, :x => lag)** # shift down

*这些 **lead和 lag** 函数来自 ShiftedArrays.jl 包.*

# 构建数据管道

```
@pipe df |>
  filter(:sex => ==("male"), _) |>
  groupby(_, :pclass) |>
  combine(_, :age => mean)
```

*这个 @pipe 宏 来自 Pipe.jl 包. 下划线 会被 |>前的操作的返回值自动替换 .*

# 数据聚合

## 聚合变量

**combine(df, :survived => sum)**
**combine(df, :survived => sum => :survived)**
在指定列上进行聚合; 目标列名是可选的.

**combine(df, :age => (x -> mean(skipmissing(x))))**
在指定列上用匿名函数进行聚合操作.

**combine(df, [:parch, :sibsp] .=> maximum)**
用广播语法把一个聚合函数作用到多个指定列上.

## 把聚合结果添加为列

**transform(df, :fare => mean => :average_fare)**
在原始数据框中加入一个用聚合值填充的新列.

**select(df, :name, :fare, :fare => mean => :average_fare)**
返回指定列和一个指定列的聚合值的新列.

## 把逐行处理结果添加为列

**transform(df, [:parch, :sibsp] => ByRow(+) => :relatives)**
在指定列上应用一个函数进行聚合, 把聚合值 添加为一个新列.

**transform(df, :name => ByRow(x -> split(x, ",")) => [:lname, :fname])**
在指定列上应用一个能返回多个结果的函数, 用生成的聚合值添加多个新列.

***提示:** 用 **skipmissing** 函数来移除缺失值.*

# 数据分组

**gdf = groupby(df, :pclass)**
**gdf = groupby(df, [:pclass, :sex])**
通过一个或多个列进行数据分组.

**keys(gdf)**
获取分组结果中 各子数据框的 键(keys).

**gdf[(1,)]**
用键值元组去查找特定的组.

**combine(gdf, :survived => sum)**
对每个组中的所有行应用一个聚合函数. 返回单个数据框.

**combine(gdf) do sdf**
  **DataFrame(survived = sum(sdf.survived))**
**end**
在所有组的子数据框上应用一个函数, 并组各组的结果.

**combine(gdf, AsTable(:) => t -> sum(t.parch .+ t.sibsp))**
对组中每个子数据框应用一个函数,并组合结果.

***提示:***
*可以通过下面的函数把分组概述信息添加到所有行中:*
- ***select***
- ***select!***
- ***transform***
- ***transform!***

# 数据组合/联接

**innerjoin(df1, df2, on = :id)**

| id | x | y |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

| id | z |
|---|---|
| 1 | 10 |
| 2 | 11 |
| 4 | 12 |
| 5 | 13 |

**leftjoin(df1, df2, on = :id)**

| id | x | y |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

| id | z |
|---|---|
| 1 | 10 |
| 2 | 11 |
| 4 | 12 |
| 5 | 13 |

**rightjoin(df1, df2, on = :id)**

| id | x | y |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

| id | z |
|---|---|
| 1 | 10 |
| 2 | 11 |
| 4 | 12 |
| 5 | 13 |

**outerjoin(df1, df2, on = :id)**

| id | x | y |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

| id | z |
|---|---|
| 1 | 10 |
| 2 | 11 |
| 4 | 12 |
| 5 | 13 |

**semijoin(df1, df2, on = :id)**

| id | x | y |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

| id | z |
|---|---|
| 1 | 10 |
| 2 | 11 |
| 4 | 12 |
| 5 | 13 |

**antijoin(df1, df2, on = :id) #差集**

| id | x | y |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

| id | z |
|---|---|
| 1 | 10 |
| 2 | 11 |
| 4 | 12 |
| 5 | 13 |

**vcat(df1, df2)**

| id | x | y |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |

| id | x | y |
|---|---|---|
| 3 | 10 | 12 |
| 4 | 11 | 13 |

*多个数据框可以被水平或垂直的组合 .*

**hcat(df1, df2)**

| id | x |
|---|---|
| 1 | 4 |
| 2 | 5 |

| y |
|---|
| 7 |
| 8 |