データハンドリング系

鈴木瑞人

東京大学大学院　新領域創成科学研究科

メディカル情報生命専攻

博士課程1年

今回よく出てくる人物紹介

Hadley Wickham氏

http://hadley.nz/

Who is Hadley Wickham?

- Chief Scientist at RStudio
- Adjunct Professor of Statistics at the University of Auckland, Stanford University, and Rice University.

HADLEY WICKHAM   TEACHING   CODE   PERSONAL

TEACHING

If you'd like to learn more about what I do, and how to use R effectively, I'd recommend starting with one of my books:

- R for Data Science, with Garrett Grolemund, introduces the key tools for doing data science with R.
- ggplot2: elegant graphics for data analysis shows you how to use ggplot2 to create graphics that help you understand your data.
- Advanced R helps you master R as a programming language, teaching you what makes R tick.
- R packages teaches good software engineering practices for R, using packages for bundling, documenting, and testing your code.

I also teach in person workshops from time-to-time; see the RStudio workshops page for more details.

http://hadley.nz/

HADLEY WICKHAM TEACHING CODE PERSONAL

TEACHING

If you'd like to learn more about what I do, and how to use R effectively, I'd recommend starting with one of my books:

- R for Data Science, with Garrett Grolemund, introduces the key tools for doing data science with R.
- ggplot2: elegant graphics for data analysis shows you how to use ggplot2 to create graphics that help you understand your data.
- Advanced R helps you master R as a programming language, teaching you what makes R tick.
- R packages teaches good software engineering practices for R, using packages for bundling, documenting, and testing your code.

I also teach in person workshops from time-to-time; see the RStudio workshops page for more details.

http://hadley.nz/

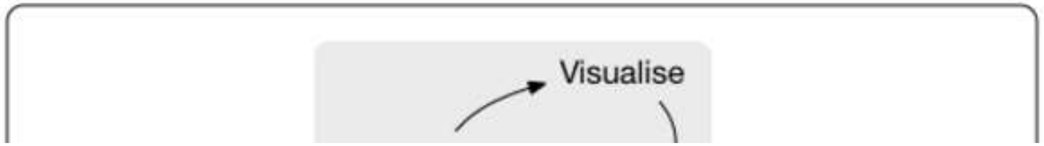
R for Data Science

1 Introduction

Data science is an exciting discipline that allows you to turn raw data into understanding, insight, and knowledge. The goal of "R for Data Science" is to help you learn the most important tools in R that will allow you to do data science. After reading this book, you'll have the tools to tackle a wide variety of data science challenges, using the best parts of R.

1.1 What you will learn

Data science is a huge field, and there's no way you can master it by reading a single book. The goal of this book is to give you a solid foundation in the most important tools. Our model of the tools needed in a typical data science project looks something like this:



http://r4ds.had.co.nz/introduction.html

HADLEY WICKHAM   TEACHING   CODE   PERSONAL

TEACHING

If you'd like to learn more about what I do, and how to use R effectively, I'd recommend starting with one of my books:

- R for Data Science, with Garrett Grolemund, introduces the key tools for doing data science with R.
- ggplot2: elegant graphics for data analysis shows you how to use ggplot2 to create graphics that help you understand your data.
- Advanced R helps you master R as a programming language, teaching you what makes R tick.
- R packages teaches good software engineering practices for R, using packages for bundling, documenting, and testing your code.

I also teach in person workshops from time-to-time; see the RStudio workshops page for more details.

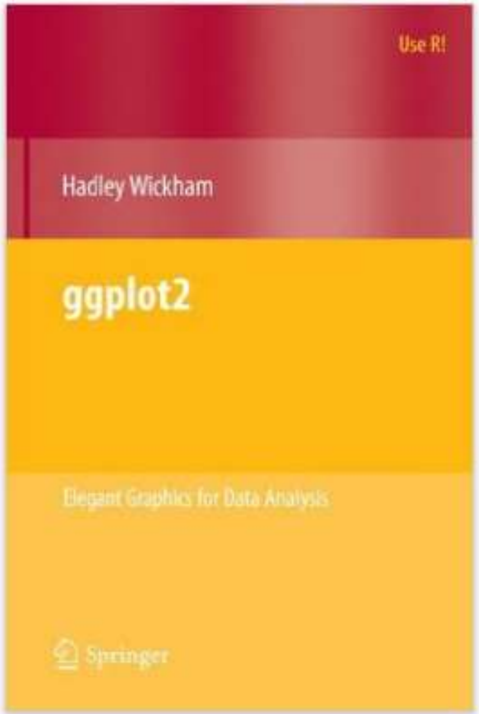
http://hadley.nz/

ggplot2: Elegant Graphics for Data Analysis (Use R!) 1st ed. 2009. Corr. 3rd printing 2010 Edition

by Hadley Wickham (Author)

57 customer reviews

Look inside



ISBN-13: 978-0387981406
ISBN-10: 0387981403

Why is ISBN important?

Paperback from $50.13 | Other Sellers from $50.13

More Buying Choices

14 New from $64.34 | 30 Used from $50.13

44 used & new from $50.13

See All Buying Options

There is a newer edition of this item:

ggplot2: Elegant Graphics for Data Analysis (Use R!)
$56.99
(57)
In Stock.

Provides both rich theory and powerful applications

Figures are accompanied by code required to produce them

Full color figures

Customer Reviews

57

4.3 out of 5 stars

5 star	58%
4 star	23%
3 star	12%
2 star	5%
1 star	2%

https://www.amazon.com/dp/0387981403/ref=cm_sw_su_dp?tag=ggplot2-20

HADLEY WICKHAM | TEACHING | CODE | PERSONAL

TEACHING

If you'd like to learn more about what I do, and how to use R effectively, I'd recommend starting with one of my books:

- R for Data Science, with Garrett Grolemund, introduces the key tools for doing data science with R.
- ggplot2: elegant graphics for data analysis shows you how to use ggplot2 to create graphics that help you understand your data.
- Advanced R helps you master R as a programming language, teaching you what makes R tick.
- R packages teaches good software engineering practices for R, using packages for bundling, documenting, and testing your code.

I also teach in person workshops from time-to-time; see the RStudio workshops page for more details.

http://hadley.nz/

Want to learn from me in person? I'm next teaching in DC, Sep 14-15.

Want a physical copy of this material? Buy a book from amazon!.

Contents

How to contribute

Edit this page

This is the companion website for "**Advanced R**", a book in Chapman & Hall's R Series. The book is designed primarily for R users who want to improve their programming skills and understanding of the language. It should also be useful for programmers coming to R from other languages, as it explains some of R's quirks and shows how some parts that seem horrible do have a positive side.

http://adv-r.had.co.nz/

HADLEY WICKHAM TEACHING CODE PERSONAL

TEACHING

If you'd like to learn more about what I do, and how to use R effectively, I'd recommend starting with one of my books:

- R for Data Science, with Garrett Grolemund, introduces the key tools for doing data science with R.
- ggplot2: elegant graphics for data analysis shows you how to use ggplot2 to create graphics that help you understand your data.
- Advanced R helps you master R as a programming language, teaching you what makes R tick.
- $\boxed{\text{R packages}}$ teaches good software engineering practices for R, using packages for bundling, documenting, and testing your code.

I also teach in person workshops from time-to-time; see the RStudio workshops page for more details.

http://hadley.nz/

R packages by Hadley Wickham

Table of contents

Want to learn from me in person? I'm next teaching in DC, Sep 14-15.

Want a physical copy of this material? Buy from amazon!.

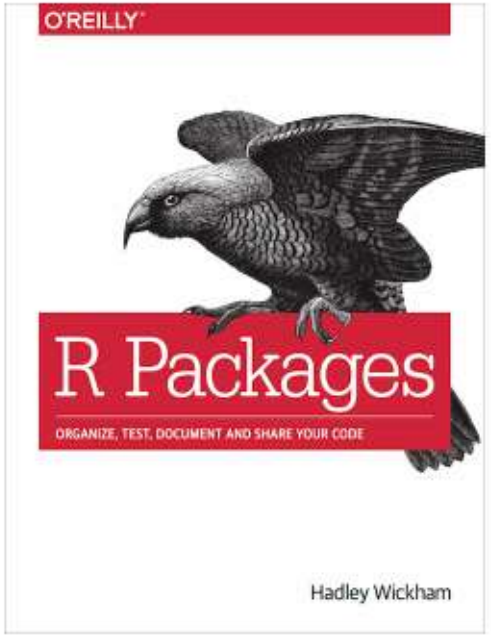
Contents

How to contribute

Edit this page

R packages

This is the book site for **"R packages"**. It was published with O'Reilly in April 2015. You can order a copy from Amazon.



Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data. In this section you'll learn how to turn your code into packages that others can easily download and use. Writing a package can seem overwhelming at first. So start with the basics and

http://r-pkgs.had.co.nz/

CODE

Most of my work is in the form of open source R code, which you can find on my github. You can roughly divide my work into three categories: tools for data science, tools for data import, and software engineering tools.

DATA SCIENCE

- ggplot2 for visualising data.
- dplyr for manipulating data.
- tidyr for tidying data.
- stringr for working with strings.
- lubridate for working with date/times.

DATA IMPORT

- readr for reading .csv and fwf files.
- readxl for reading .xls and .xlsx files.
- haven for SAS, SPSS, and Stata files.
- httr for talking to web APIs.
- rvest for scraping websites.
- xml2 for importing XML files.

SOFTWARE ENGINEERING

- devtools for general package development.
- roxygen2 for in-line documentation.
- testthat for unit testing

彼の作品には、有名なパッケージが勢ぞろい。
Reshape2など、有名でも載せられていない彼の作品も多数。

http://hadley.nz/

本講義資料では、Hadley氏の作品が頻出します。彼に感謝しましょう。

- 彼のパッケージについて、詳しく知りたければ、彼のサイトに行って調べましょう。

データの出力①(復習)

```
#iris_table_out.csvファイルを出力
write.table(iris, "iris_table_out.csv", sep = ",")
#iris_table_out.csvファイルを読み込み
result1 <- read.table("iris_table_out.csv", sep = ",", header = TRUE)
#読み込んだ結果の表示
head(result1)
```

write.table関数または、read.table関数はセパレータの指定が必要。
この場合はコンマ。

```
> #iris_table_out.csvファイルを出力
> write.table(iris, "iris_table_out.csv", sep = ",")
> #iris_table_out.csvファイルを読み込み
> result1 <- read.table("iris_table_out.csv", sep = ",", header = TRUE)
> #読み込んだ結果の表示
> head(result1)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

データの入出力②(復習)

```
#iris_csv_out.csvファイルを出力
write.csv(iris, "iris_csv_out.csv")
#iris_csv_out.csvファイルを読み込み
result2 <- read.csv("iris_csv_out.csv", header = TRUE)
#読み込んだ結果の表示
head(result2)
```

write.csv関数または、read.csv関数はセパレータの指定が不要。

```
> #iris_csv_out.csvファイルを出力
> write.csv(iris, "iris_csv_out.csv")
> #iris_csv_out.csvファイルを読み込み
> result2 <- read.csv("iris_csv_out.csv", header = TRUE)
> #読み込んだ結果の表示
> head(result2)
  X Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1 1          5.1         3.5          1.4         0.2  setosa
2 2          4.9         3.0          1.4         0.2  setosa
3 3          4.7         3.2          1.3         0.2  setosa
4 4          4.6         3.1          1.5         0.2  setosa
5 5          5.0         3.6          1.4         0.2  setosa
6 6          5.4         3.9          1.7         0.4  setosa
```

行名を消去。

```
#iris_csv2_out.csvファイルを出力
write.csv(iris, "iris_csv2_out.csv",row.names=F)
#iris_csv2_out.csvファイルを読み込み
result3 <- read.csv("iris_csv2_out.csv", header = TRUE)
#読み込んだ結果の表示
head(result3)
```

```
> #iris_csv_out.csvファイルを出力
> write.csv(iris, "iris_csv2_out.csv",row.names=F)
> #iris_csv2_out.csvファイルを読み込み
> result3 <- read.csv("iris_csv2_out.csv", header = TRUE)
> #読み込んだ結果の表示
> head(result3)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

大きなデータを読み込む場合。

- 2008年の米国のフライトデータを読み込んでみる。
- データの大きさは、658MB。
- 航空機の発着に関する情報を収録した29列の表データ。
- データはデータExpo2009のサイトからダウンロードし、read.csv関数で読み込む。
- 解凍にはR.utilsパッケージのbunzip2関数を使用する。

パッケージのダウンロード

```
install.packages("R.utils", quiet = TRUE, dependencies=T)
library(R.utils)
```

今回使用するデータは、bzip2形式で圧縮されているため、
R.utilsパッケージのbunzip2関数を使用して解凍する。

ディレクトリ作成とデータのダウンロード

```
#カレントディレクトリ確認
getwd()
# ディレクトリ作成とデータのダウンロード(getwd()で確認したカレントディレクトリ下にdataというフォルダを作成)
dir.create("C:/Users/Administrator/Documents/data")
#dataというフォルダの下にDataExpo2009というフォルダを作成
dir.create("data/DataExpo2009")
#データのダウンロード
download.file("http://stat-computing.org/dataexpo/2009/2008.csv.bz2",
 "data/DataExpo2009/2008.csv.bz2")
```

データの解凍と読み込み

```
# bunzip2関数を用いたデータの解凍
bunzip2("data/DataExpo2009/2008.csv.bz2")
# データの読み込みと速度計測
system.time(al.2008.df <- read.csv("data/DataExpo2009/2008.csv", as.is = TRUE))
```

csvファイルを読み込むときに、数値データの他に、何か文字等が混ざっていると自動的にfactor型として読み込まれるときがある。それを防ぐためには、as.is=TRUEを入れておくとよい。

```
> getwd()
[1] "C:/Users/Administrator/Documents"
> dir.create("C:/Users/Administrator/Documents/data")
> dir()
[1] "data"        "desktop.ini" "My Music"    "My Pictures" "My Videos"
> dir.create("data/DataExpo2009")
> download.file("http://stat-computing.org/dataexpo/2009/2008.csv.bz2",
+   "data/DataExpo2009/2008.csv.bz2")
trying URL 'http://stat-computing.org/dataexpo/2009/2008.csv.bz2'
Content type 'application/x-bzip2' length 113753229 bytes (108.5 MB)
downloaded 108.5 MB

> bunzip2("data/DataExpo2009/2008.csv.bz2")
> system.time(al.2008.df <- read.csv("data/DataExpo2009/2008.csv", as.is = TRUE))
   user  system elapsed
  71.77    3.08   74.94
```

データの読み込みに74.94秒かかっている。。もっと早く読み込めないか。。

read.csv関数より高速にデータを読み込む関数

- readrパッケージ(Hadley Wickham氏作)のread_csv関数

readrパッケージのダウンロード

```
install.packages("readr", quiet = TRUE, dependencies=T)
library(readr)
```

データの読み込み

```
# 2008年のフライトデータの読み込み
system.time(al.2008.readr <- read_csv("data/DataExpo2009/2008.csv"))
```

```
 user  system elapsed
14.34    1.55   18.90
```

18.9秒！！

先ほどの、74.94秒より4倍高速！

読み込んだ中身を見てみる。

#先頭3行

head(al.2008.readr, 3)

```
> #先頭3行
> head(al.2008.readr, 3)
# A tibble: 3 × 29
   Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
  <int> <int>      <int>     <int>   <int>      <int>   <int>      <int>
1  2008     1          3         4    2003       1955    2211       2225
2  2008     1          3         4     754        735    1002       1000
3  2008     1          3         4     628        620     804        750
# ... with 21 more variables: UniqueCarrier <chr>, FlightNum <int>, TailNum <chr>,
#   ActualElapsedTime <int>, CRSElapsedTime <int>, AirTime <int>, ArrDelay <int>,
#   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>, TaxiIn <int>,
#   TaxiOut <int>, Cancelled <int>, CancellationCode <chr>, Diverted <int>,
#   CarrierDelay <int>, WeatherDelay <int>, NASDelay <int>, SecurityDelay <int>,
#   LateAircraftDelay <int>
```

読み込んだ中身を見てみる

#オブジェクトのクラスの確認

```
class(al.2008.readr)
```

```
> #オブジェクトのクラスの確認
>  class(al.2008.readr)
[1] "tbl_df"     "tbl"          "data.frame"
```

data.frame以外は見慣れない。。しかし、tbl_dfやtbl型は今後使用するパッケージの関数で使用する型！

読み込んだ中身を見てみる

#各列のデータ型の確認

sapply(al.2008.readr, class)

```
> #各列のデータ型の確認
> sapply(al.2008.readr, class)
               Year               Month          DayofMonth           DayOfWeek 
          "integer"           "integer"           "integer"           "integer" 
            DepTime          CRSDepTime             ArrTime          CRSArrTime 
          "integer"           "integer"           "integer"           "integer" 
      UniqueCarrier           FlightNum             TailNum   ActualElapsedTime 
        "character"           "integer"         "character"           "integer" 
     CRSElapsedTime             AirTime            ArrDelay            DepDelay 
          "integer"           "integer"           "integer"           "integer" 
             Origin                Dest            Distance              TaxiIn 
        "character"         "character"           "integer"           "integer" 
            TaxiOut           Cancelled    CancellationCode            Diverted 
          "integer"           "integer"         "character"           "integer" 
       CarrierDelay        WeatherDelay            NASDelay       SecurityDelay 
          "integer"           "integer"           "integer"           "integer" 
  LateAircraftDelay 
          "integer" 
```

Numericや、Factor型として読み込まれていないので注意！

データを読み込むときのデータ型の指定

- col_types引数で指定する。
- 整数”i”
- 論理値”l”
- 倍精度浮動小数点”d”
- ユーロタイプの不動小数点”e”
- 日付(Y-m-d形式)”D”

もし1,2,3,4,5列目がそれぞれ、文字列、倍精度浮動小数点、整数、論理値、文字列の場合、
col_types=“cdilc” という指定になる。
すなわち先ほどの読み込みは、
al.2008.readr <- read_csv("data/DataExpo2009/2008.csv", col_types="cdilc"))

データフレームのハンドリング

データの加工・集計

- ここでは、Rのコアメンバーである、Hadley Wickham氏が作成した、dplyrパケージを用いる。
- このパッケージは、特定の条件を満たす行や列の抽出グループごとの集計などの処理を高速に行うために開発されたもの。

dplyrパッケージ

Index of /web/packages/dplyr/vignettes

Name	Last modified	Size	Description
Parent Directory		-	
data_frames.html	29-Aug-2016 11:12	15K	
databases.html	29-Aug-2016 11:12	46K	
hybrid-evaluation.html	29-Aug-2016 11:12	24K	
index.rds	29-Aug-2016 11:12	322	
introduction.html	29-Aug-2016 11:12	137K	
new-sql-backend.html	29-Aug-2016 11:12	16K	
nse.html	29-Aug-2016 11:12	16K	
two-table.html	29-Aug-2016 11:12	39K	
window-functions.html	29-Aug-2016 11:12	75K	

Apache/2.2.22 (Ubuntu) Server at cran.rstudio.com Port 443

https://cran.rstudio.com/web/packages/dplyr/vignettes/

Data frame performance

Data frame performance

2016-06-23

One of the reasons that dplyr is fast is that it's very careful about when to make copies. This section describes how this works, and gives you some useful tools for understanding the memory usage of data frames in R.

The first tool we'll use is `dplyr::location()`. It tells us the memory location of three components of a data frame object:

- the data frame itself
- each column
- each attribute

```
location(iris)
#> <0x7fdc68e309a8>
#> Variables:
#>  * Sepal.Length: <0x7fdc68f06200>
#>  * Sepal.Width:  <0x7fdc68f25000>
#>  * Petal.Length: <0x7fdc68f25600>
#>  * Petal.Width:  <0x7fdc68f25c00>
#>  * Species:      <0x7fdc6843e9e0>
#> Attributes:
#>  * names:        <0x7fdc68e30940>
#>  * row.names:    <0x7fdc6843fa00>
#>  * class:        <0x7fdc688bbb48>
```

It's useful to know the memory address, because if the address changes, then you'll know that R has made a copy. Copies are bad because they take time to create. This isn't usually a bottleneck if you have a few thousand values, but if you have millions or tens of millions of values it starts to take significant amounts of time. Unnecessary copies are also bad because they take up memory.

https://cran.rstudio.com/web/packages/dplyr/vignettes/data_frames.html

Databasesとの連携

Databases

2016-06-23

As well as working with local in-memory data like data frames and data tables, dplyr also works with remote on-disk data stored in databases. Generally, if your data fits in memory there is no advantage to putting it in a database: it will only be slower and more hassle. The reason you'd want to use dplyr with a database is because either your data is already in a database (and you don't want to work with static csv files that someone else has dumped out for you), or you have so much data that it does not fit in memory and you have to use a database. Currently dplyr supports the three most popular open source databases (sqlite, mysql and postgresql), and google's bigquery.

Since R almost exclusively works with in-memory data, if you do have a lot of data in a database, you can't just dump it into R. Instead, you'll have to work with subsets or aggregates. dplyr aims to make this task as easy as possible. If you're working with large data, it's also likely that you'll need support to get the data into the database and to ensure you have the right indices for good performance. While dplyr provides some simple tools to help with these tasks, they are no substitute for a local expert.

The motivation for supporting databases in dplyr is that you never pull down the right subset or aggregate from the database on your first try. Usually you have to iterate between R and SQL many times before you get the perfect dataset. But because switching between languages is cognitively challenging (especially because R and SQL are so perilously similar), dplyr helps you by allowing you to write R code that is automatically translated to SQL. The goal of dplyr is not to replace every SQL function with an R function; that would be difficult and error prone. Instead, dplyr only generates `SELECT` statements, the SQL you write most often as an analyst.

To get the most out of this chapter, you'll need to be familiar with querying SQL databases using the `SELECT` statement. If you have some familiarity with SQL and you'd like to learn more, I found how indexes work in SQLite and 10 easy steps to a complete understanding of SQL to be particularly helpful.

https://cran.rstudio.com/web/packages/dplyr/vignettes/databases.html

Introduction

Introduction to dplyr

2016-06-23

When working with data you must:

- Figure out what you want to do.
- Describe those tasks in the form of a computer program.
- Execute the program.

The dplyr package makes these steps fast and easy:

- By constraining your options, it simplifies how you can think about common data manipulation tasks.
- It provides simple "verbs", functions that correspond to the most common data manipulation tasks, to help you translate those thoughts into code.
- It uses efficient data storage backends, so you spend less time waiting for the computer.

This document introduces you to dplyr's basic set of tools, and shows you how to apply them to data frames. Other vignettes provide more details on specific topics:

- databases: Besides in-memory data frames, dplyr also connects to out-of-memory, remote databases. And by translating your R code into the appropriate SQL, it allows you to work with both types of data using the same set of tools.
- benchmark-baseball: see how dplyr compares to other tools for data manipulation on a realistic use case.
- window-functions: a window function is a variation on an aggregation function. Where an aggregate function uses `n` inputs to produce 1 output, a window function uses `n` inputs to produce `n` outputs.

https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html

Window functions and grouped mutate/filter

Window functions and grouped mutate/filter

2016-06-23

A **window function** is a variation on an aggregation function. Where an aggregation function, like `sum()` and `mean()`, takes n inputs and return a single value, a window function returns n values. The output of a window function depends on all its input values, so window functions don't include functions that work element-wise, like `+` or `round()`. Window functions include variations on aggregate functions, like `cumsum()` and `cummean()`, functions for ranking and ordering, like `rank()`, and functions for taking offsets, like `lead()` and `lag()`.

Window functions are used in conjunction with `mutate` and `filter` to solve a wide range of problems, some of which are shown below:

```
library(Lahman)
batting <- select(tbl_df(Batting), playerID, yearID, teamID, G, AB:H)
batting <- arrange(batting, playerID, yearID, teamID)
players <- group_by(batting, playerID)

# For each player, find the two years with most hits
filter(players, min_rank(desc(H)) <= 2 & H > 0)
# Within each player, rank each year by the number of games played
mutate(players, G_rank = min_rank(G))

# For each player, find every year that was better than the previous year
filter(players, G > lag(G))
# For each player, compute avg change in games played per year
mutate(players, G_change = (G - lag(G)) / (yearID - lag(yearID)))
```

https://cran.rstudio.com/web/packages/dplyr/vignettes/window-functions.html

パッケージのロード

```
# dplyr, nycflights13のダウンロード・インストール
install.packages("dplyr", quiet = TRUE, dependencies=T)
#nycflight13パッケージのflightsデータセットを用いる
install.packages("nycflights13", quiet = TRUE, dependencies=T)
#パッケージのロード
library(dplyr)
library(nycflights13)
#nycflights13パッケージでは、データを呼び出すことなく使用可能
#つまりdata(nycflights13) のような宣言が必要ない
```

#クラスの確認
class(flights)

```
> library(nycflights13)
> #クラスの確認
> class(flights)
[1] "tbl_df"     "tbl"        "data.frame"
```

tbl_df型か、tbl型に属しているものは、データを表示させたときに、全体が表示されず、コンソール画面を埋め尽くさないので便利。

#データの先頭の確認
flights

```
> #データの先頭の確認
> flights
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1   2013     1     1      517            515         2      830            819
2   2013     1     1      533            529         4      850            830
3   2013     1     1      542            540         2      923            850
4   2013     1     1      544            545        -1     1004           1022
5   2013     1     1      554            600        -6      812            837
6   2013     1     1      554            558        -4      740            728
7   2013     1     1      555            600        -5      913            854
8   2013     1     1      557            600        -3      709            723
9   2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# ... with 336,766 more rows, and 11 more variables: arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

data.frame形式から、tbl形式への変更

#tbl_df関数を用いる

```
y <- tbl_df(x)
```

xという、データフレームがあった場合です。
ここではxは具体的にないので、実行しないでください。

dplyrと標準Rの比較

	行の抽出	列の抽出	列の追加	行の並べ替え	データの集約	グループ化処理
dplyr	filter関数	select関数	mutate関数	arrange関数	summerize関数	group_by関数
標準のR	subset関数	subset関数	transform関数	order関数	aggregate関数	tapply関数 by関数

特定の列で条件を指定して抽出

#12月31日のレコードの抽出(AND条件で抽出)

```
filter(flights, month == 12, day == 31)
```

```
> #12月31日のレコードの抽出(AND条件で抽出)
> filter(flights, month == 12, day == 31)
# A tibble: 776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1   2013    12    31       13           2359        14      439            437
2   2013    12    31       18           2359        19      449            444
3   2013    12    31       26           2245       101      129           2353
4   2013    12    31      459            500        -1      655            651
5   2013    12    31      514            515        -1      814            812
6   2013    12    31      549            551        -2      925            900
7   2013    12    31      550            600       -10      725            745
8   2013    12    31      552            600        -8      811            826
9   2013    12    31      553            600        -7      741            754
10  2013    12    31      554            550         4     1024           1027
# ... with 766 more rows, and 11 more variables: arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

特定の列で条件を指定して抽出

#1月または31日のレコードの抽出(OR条件で抽出)

```
filter(flights, month == 1 | day == 31)
```

```
> #1月または31日のレコードの抽出(OR条件で抽出)
> filter(flights, month == 1 | day == 31)
# A tibble: 32,266 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1   2013     1     1      517            515         2      830            819
2   2013     1     1      533            529         4      850            830
3   2013     1     1      542            540         2      923            850
4   2013     1     1      544            545        -1     1004           1022
5   2013     1     1      554            600        -6      812            837
6   2013     1     1      554            558        -4      740            728
7   2013     1     1      555            600        -5      913            854
8   2013     1     1      557            600        -3      709            723
9   2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# ... with 32,256 more rows, and 11 more variables: arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

指定した列の抽出

#年、月、日の抽出

```
select(flights, year, month, day)
```

```
> #年、月、日の抽出
> select(flights, year, month, day)
# A tibble: 336,776 × 3
    year month   day
   <int> <int> <int>
1   2013     1     1
2   2013     1     1
3   2013     1     1
4   2013     1     1
5   2013     1     1
6   2013     1     1
7   2013     1     1
8   2013     1     1
9   2013     1     1
10  2013     1     1
# ... with 336,766 more rows
```

連続する列の取り出し

```
select(flights, year:day)
```

```
> select(flights, year:day)
# A tibble: 336,776 × 3
     year month   day
    <int> <int> <int>
1    2013     1     1
2    2013     1     1
3    2013     1     1
4    2013     1     1
5    2013     1     1
6    2013     1     1
7    2013     1     1
8    2013     1     1
9    2013     1     1
10   2013     1     1
# ... with 336,766 more rows
```

ー演算子による、列の削除

```
#年、月、日以外の列の抽出
select(flights, -(year:day))
```

```
> #年、月、日以外の列の抽出
> select(flights, -(year:day))
# A tibble: 336,776 × 16
   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
      <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>
1       517            515         2      830            819        11      UA
2       533            529         4      850            830        20      UA
3       542            540         2      923            850        33      AA
4       544            545        -1     1004           1022       -18      B6
5       554            600        -6      812            837       -25      DL
6       554            558        -4      740            728        12      UA
7       555            600        -5      913            854        19      B6
8       557            600        -3      709            723       -14      EV
9       557            600        -3      838            846        -8      B6
10      558            600        -2      753            745         8      AA
# ... with 336,766 more rows, and 9 more variables: flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
Warning message:
In as.POSIXlt.POSIXct(x, tz) : unable to identify current timezone 'C':
please set environment variable 'TZ'
```

mutate関数による列の追加

```
mutate(flights, gain = arr_delay - dep_delay, gain_per_hour =
gain/(air_time/60))
```

```
> mutate(flights, gain = arr_delay - dep_delay, gain_per_hour = gain/(air_time/60))
# A tibble: 336,776 × 21
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1   2013     1     1      517            515         2      830            819
2   2013     1     1      533            529         4      850            830
3   2013     1     1      542            540         2      923            850
4   2013     1     1      544            545        -1     1004           1022
5   2013     1     1      554            600        -6      812            837
6   2013     1     1      554            558        -4      740            728
7   2013     1     1      555            600        -5      913            854
8   2013     1     1      557            600        -3      709            723
9   2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# ... with 336,766 more rows, and 13 more variables: arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>, gain <dbl>,
#   gain_per_hour <dbl>
```

arrange関数による、行順番の並び替え

arrange(flights, month, arr_delay)

```
> arrange(flights, month, arr_delay)
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1   2013     1     4     1026           1030        -4     1305           1415
2   2013     1     3      941            945        -4     1153           1258
3   2013     1    14     1840           1845        -5     2117           2221
4   2013     1     3     1153           1200        -7     1442           1545
5   2013     1     3     1228           1235        -7     1503           1606
6   2013     1    27     1845           1850        -5     2110           2212
7   2013     1     3     1605           1610        -5     1816           1917
8   2013     1     3     1857           1900        -3     2200           2301
9   2013     1     4     1219           1221        -2     1454           1555
10  2013     1     6      812            819        -7     1102           1203
# ... with 336,766 more rows, and 11 more variables: arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

arrange関数による行の降順で並べ替え

#desc関数を用いることで、降順に

arrange(flights, desc(arr_delay))

```
> arrange(flights, desc(arr_delay))
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1   2013     1     9      641            900      1301     1242           1530
2   2013     6    15     1432           1935      1137     1607           2120
3   2013     1    10     1121           1635      1126     1239           1810
4   2013     9    20     1139           1845      1014     1457           2210
5   2013     7    22      845           1600      1005     1044           1815
6   2013     4    10     1100           1900       960     1342           2211
7   2013     3    17     2321            810       911      135           1020
8   2013     7    22     2257            759       898      121           1026
9   2013    12     5      756           1700       896     1058           2020
10  2013     5     3     1133           2055       878     1250           2215
# ... with 336,766 more rows, and 11 more variables: arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

summarise関数による統計量算出

summarise(flights, DepDelay = mean(dep_delay, na.rm = TRUE), ArrDelay = mean(arr_delay, na.rm = TRUE))

```
# A tibble: 1 × 2
  DepDelay ArrDelay
     <dbl>    <dbl>
1 12.63907 6.895377
```

group_by関数による、グループごとの処理

```
#機体番号ごとの平均距離・平均出発遅延時間・平均到着遅延時間の算出
planes <- group_by(flights, tailnum)
delay <- summarise(planes, Dist = mean(distance, na.rm = TRUE),
DepDelay = mean(dep_delay, na.rm = TRUE), ArrDelay = mean(arr_delay, na.rm
= TRUE))
delay
```

```
> #機体番号ごとの平均距離・平均出発遅延時間・平均到着遅延時間の算出
> planes <- group_by(flights, tailnum)
> delay <- summarise(planes, Dist = mean(distance, na.rm = TRUE),
+ DepDelay = mean(dep_delay, na.rm = TRUE), ArrDelay = mean(arr_delay, na.rm = TRUE))
> delay
# A tibble: 4,044 × 4
   tailnum     Dist   DepDelay   ArrDelay
     <chr>    <dbl>      <dbl>      <dbl>
1   D942DN 854.5000 31.5000000 31.5000000
2   N0EGMQ 676.1887  8.4915254  9.9829545
3   N10156 757.9477 17.8150685 12.7172414
4   N102UW 535.8750  8.0000000  2.9375000
5   N103US 535.1957 -3.1956522 -6.9347826
6   N104UW 535.2553  9.9361702  1.8043478
7   N10575 519.7024 22.6507353 20.6914498
8   N105UW 524.8444  2.5777778 -0.2666667
9   N107US 528.7073 -0.4634146 -5.7317073
10  N108UW 534.5000  4.2166667 -1.2500000
# ... with 4,034 more rows
```

chain関数(%>%)によるパイプ処理

```
#年、月、日を集計軸に設定
a1 <- group_by(flights, year, month, day)
a1
#年から日まで、および到着の遅延時間、出発の遅延時間を抽出
a2 <- select(a1, year:day, arr_delay, dep_delay)
a2
#年ごと、日ごとに到着の遅延時間の平均と出発の遅延時間の平均を算出
a3 <- summarise(a2, arr = mean(arr_delay, na.rm = TRUE), dep = mean(dep_delay,
na.rm = TRUE))
a3
#到着の遅延時間が50分以上かつ出発の遅延時間が50分以上の行の抽出
a4 <- filter(a3, arr >= 50 | dep >= 50)
a4
```

#年、月、日を集計軸に設定

a1 <- group_by(flights, year, month, day)

a1

```
> #年、月、日を集計軸に設定
> a1 <- group_by(flights, year, month, day)
> a1
Source: local data frame [336,776 x 19]
Groups: year, month, day [365]

    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>  <int>
1   2013     1     1      517            515         2      830            819        11      UA   1545
2   2013     1     1      533            529         4      850            830        20      UA   1714
3   2013     1     1      542            540         2      923            850        33      AA   1141
4   2013     1     1      544            545        -1     1004           1022       -18      B6    725
5   2013     1     1      554            600        -6      812            837       -25      DL    461
6   2013     1     1      554            558        -4      740            728        12      UA   1696
7   2013     1     1      555            600        -5      913            854        19      B6    507
8   2013     1     1      557            600        -3      709            723       -14      EV   5708
9   2013     1     1      557            600        -3      838            846        -8      B6     79
10  2013     1     1      558            600        -2      753            745         8      AA    301
# ... with 336,766 more rows, and 8 more variables: tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

#年から日まで、および到着の遅延時間、出発の遅延時間を抽出

a2 <- select(a1, year:day, arr_delay, dep_delay)

a2

```
> #年から日まで、および到着の遅延時間、出発の遅延時間を抽出
> a2 <- select(a1, year:day, arr_delay, dep_delay)
> a2
Source: local data frame [336,776 x 5]
Groups: year, month, day [365]

    year month   day arr_delay dep_delay
   <int> <int> <int>     <dbl>     <dbl>
1   2013     1     1        11         2
2   2013     1     1        20         4
3   2013     1     1        33         2
4   2013     1     1       -18        -1
5   2013     1     1       -25        -6
6   2013     1     1        12        -4
7   2013     1     1        19        -5
8   2013     1     1       -14        -3
9   2013     1     1        -8        -3
10  2013     1     1         8        -2
# ... with 336,766 more rows
```

#年ごと、日ごとに到着の遅延時間の平均と出発の遅延時間の平均を算出

a3 <- summarise(a2, arr = mean(arr_delay, na.rm = TRUE), dep = mean(dep_delay, na.rm = TRUE))

a3

```
> #年ごと、日ごとに到着の遅延時間の平均と出発の遅延時間の平均を算出
> a3 <- summarise(a2, arr = mean(arr_delay, na.rm = TRUE), dep = mean(dep_delay, na.rm = TRUE))
> a3
Source: local data frame [365 x 5]
Groups: year, month [?]

    year month   day        arr       dep
   <int> <int> <int>      <dbl>     <dbl>
1   2013     1     1 12.6510229 11.548926
2   2013     1     2 12.6928879 13.858824
3   2013     1     3  5.7333333 10.987832
4   2013     1     4 -1.9328194  8.951595
5   2013     1     5 -1.5258020  5.732218
6   2013     1     6  4.2364294  7.148014
7   2013     1     7 -4.9473118  5.417204
8   2013     1     8 -3.2275785  2.553073
9   2013     1     9 -0.2642777  2.276477
10  2013     1    10 -5.8988159  2.844995
# ... with 355 more rows
```

#到着の遅延時間が50分以上かつ出発の遅延時間が50分以上の行の抽出

a4 <- filter(a3, arr >= 50 | dep >= 50)
a4

```
> #到着の遅延時間が50分以上かつ出発の遅延時間が50分以上の行の抽出
> a4 <- filter(a3, arr >= 50 | dep >= 50)
> a4
Source: local data frame [12 x 5]
Groups: year, month [7]

    year month   day      arr      dep
   <int> <int> <int>    <dbl>    <dbl>
1   2013     3     8 85.86216 83.53692
2   2013     5    23 61.97090 51.14472
3   2013     6    13 63.75369 45.79083
4   2013     6    24 51.17681 47.15742
5   2013     7     1 58.28050 56.23383
6   2013     7    10 59.62648 52.86070
7   2013     7    22 62.76340 46.66705
8   2013     8     8 55.48116 43.34995
9   2013     9     2 45.51843 53.02955
10  2013     9    12 58.91242 49.95875
11  2013    12     5 51.66625 52.32799
12  2013    12    17 55.87186 40.70560
```

関数のネストとパイプ処理の比較

- ネスト処理
- パイプ処理(magiretteパッケージ)

ネスト処理

filter(summarise(select(group_by(flights, year, month, day), year:day,

arr_delay, dep_delay), arr = mean(arr_delay, na.rm = TRUE), dep = mean(dep_delay, na.rm = TRUE)), arr >= 50 | dep >= 50)

```
Source: local data frame [12 x 5]
Groups: year, month [7]

    year month   day      arr      dep
   <int> <int> <int>    <dbl>    <dbl>
1   2013     3     8 85.86216 83.53692
2   2013     5    23 61.97090 51.14472
3   2013     6    13 63.75369 45.79083
4   2013     6    24 51.17681 47.15742
5   2013     7     1 58.28050 56.23383
6   2013     7    10 59.62648 52.86070
7   2013     7    22 62.76340 46.66705
8   2013     8     8 55.48116 43.34995
9   2013     9     2 45.51843 53.02955
10  2013     9    12 58.91242 49.95875
11  2013    12     5 51.66625 52.32799
12  2013    12    17 55.87186 40.70560
```

パイプ処理

```
flights %>% group_by(year, month, day) %>% select(year:day, arr_delay, dep_delay) %>%summarise(arr = mean(arr_delay, na.rm = TRUE), dep = mean(dep_delay, na.rm = TRUE)) %>% filter(arr >= 50 | dep >= 50)
```

```
Source: local data frame [12 x 5]
Groups: year, month [7]

    year month   day      arr      dep
   <int> <int> <int>    <dbl>    <dbl>
1   2013     3     8 85.86216 83.53692
2   2013     5    23 61.97090 51.14472
3   2013     6    13 63.75369 45.79083
4   2013     6    24 51.17681 47.15742
5   2013     7     1 58.28050 56.23383
6   2013     7    10 59.62648 52.86070
7   2013     7    22 62.76340 46.66705
8   2013     8     8 55.48116 43.34995
9   2013     9     2 45.51843 53.02955
10  2013     9    12 58.91242 49.95875
11  2013    12     5 51.66625 52.32799
12  2013    12    17 55.87186 40.70560
```

dplyrパッケージのその他の関数

library(dplyr)

data(flights)

dplyrパッケージのその他の関数

```
# 出発・到着の遅延時間の平均値・中央値・標準偏差を求める
summarise_each(select(flights, dep_delay, arr_delay), funs(mean = mean(., na.rm = TRUE),median = median(., na.rm = TRUE), sd = sd(., na.rm = TRUE)))
# summarise_each関数のmatchesを使用すれば、select関数は不要。
summarise_each(flights, funs(mean = mean(., na.rm = TRUE), median = median(., na.rm = TRUE), sd = sd(., na.rm = TRUE)), matches("_delay"))
#chain関数を用いたパイプ処理
flights %>% summarise_each(funs(mean = mean(., na.rm = TRUE), median = median(., na.rm = TRUE),sd = sd(., na.rm = TRUE)), matches("_delay"))
```

# 出発・到着の遅延時間の平均値・中央値・標準偏差を求める
summarise_each(select(flights, dep_delay, arr_delay), funs(mean = mean(., na.rm = TRUE),median = median(., na.rm = TRUE), sd = sd(., na.rm = TRUE)))

```
> # 出発・到着の遅延時間の平均値・中央値・標準偏差を求める
> summarise_each(select(flights, dep_delay, arr_delay), funs(mean = mean(., na.rm = TRUE),med
# A tibble: 1 × 6
  dep_delay_mean arr_delay_mean dep_delay_median arr_delay_median dep_delay_sd arr_delay_sd
           <dbl>          <dbl>            <dbl>            <dbl>        <dbl>        <dbl>
1       12.63907       6.895377               -2               -5     40.21006     44.63329
```

# summarise_each関数のmatchesを使用すれば、select関数は不要。
summarise_each(flights, funs(mean = mean(., na.rm = TRUE), median = median(., na.rm = TRUE), sd = sd(., na.rm = TRUE)), matches("_delay"))

```
> # summarise_each関数のmatchesを使用すれば、select関数は不要。
> summarise_each(flights, funs(mean = mean(., na.rm = TRUE), median = median(., na.rm = TRUE), sd = sd(., n$
# A tibble: 1 × 6
  dep_delay_mean arr_delay_mean dep_delay_median arr_delay_median dep_delay_sd arr_delay_sd
           <dbl>          <dbl>            <dbl>            <dbl>        <dbl>        <dbl>
1       12.63907       6.895377               -2               -5     40.21006     44.63329
```

```
#chain関数を用いたパイプ処理
flights %>% summarise_each(funs(mean = mean(., na.rm = TRUE), median = median(., na.rm = TRUE),sd = sd(., na.rm = TRUE)), matches("_delay"))
```

```
> #chain関数を用いたパイプ処理
> flights %>% summarise_each(funs(mean = mean(., na.rm = TRUE), median = median(., na.rm = TRUE)
# A tibble: 1 × 6
  dep_delay_mean arr_delay_mean dep_delay_median arr_delay_median dep_delay_sd arr_delay_sd
           <dbl>          <dbl>            <dbl>            <dbl>        <dbl>        <dbl>
1       12.63907       6.895377               -2               -5     40.21006     44.63329
```

dplyrパッケージのその他の関数

#出発・到着の遅延時間を正規化する

```
mutate_each(flights, funs(scale), matches("_delay"))
```

#chain関数を用いたパイプ処理の場合

```
flights %>% mutate_each(funs(scale), matches("_delay"))
```

```
> #出発・到着の遅延時間を正規化する
> mutate_each(flights, funs(scale), matches("_delay"))
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time  dep_delay arr_time sched_arr_time   arr_delay carrier flight
   <int> <int> <int>    <int>          <int>      <dbl>    <int>          <int>       <dbl>   <chr>  <int>
1   2013     1     1      517            515 -0.2645873      830            819  0.09196327      UA   1545
2   2013     1     1      533            529 -0.2148485      850            830  0.29360647      UA   1714
3   2013     1     1      542            540 -0.2645873      923            850  0.58486888      AA   1141
4   2013     1     1      544            545 -0.3391955     1004           1022 -0.55777595      B6    725
5   2013     1     1      554            600 -0.4635425      812            837 -0.71460956      DL    461
6   2013     1     1      554            558 -0.4138037      740            728  0.11436807      UA   1696
7   2013     1     1      555            600 -0.4386731      913            854  0.27120167      B6    507
8   2013     1     1      557            600 -0.3889343      709            723 -0.46815675      EV   5708
9   2013     1     1      557            600 -0.3889343      838            846 -0.33372795      B6     79
10  2013     1     1      558            600 -0.3640649      753            745  0.02474886      AA    301
# ... with 336,766 more rows, and 8 more variables: tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
> #chain関数を用いたパイプ処理の場合
> flights %>% mutate_each(funs(scale), matches("_delay"))
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time  dep_delay arr_time sched_arr_time   arr_delay carrier flight
   <int> <int> <int>    <int>          <int>      <dbl>    <int>          <int>       <dbl>   <chr>  <int>
1   2013     1     1      517            515 -0.2645873      830            819  0.09196327      UA   1545
2   2013     1     1      533            529 -0.2148485      850            830  0.29360647      UA   1714
3   2013     1     1      542            540 -0.2645873      923            850  0.58486888      AA   1141
4   2013     1     1      544            545 -0.3391955     1004           1022 -0.55777595      B6    725
5   2013     1     1      554            600 -0.4635425      812            837 -0.71460956      DL    461
6   2013     1     1      554            558 -0.4138037      740            728  0.11436807      UA   1696
7   2013     1     1      555            600 -0.4386731      913            854  0.27120167      B6    507
8   2013     1     1      557            600 -0.3889343      709            723 -0.46815675      EV   5708
9   2013     1     1      557            600 -0.3889343      838            846 -0.33372795      B6     79
10  2013     1     1      558            600 -0.3640649      753            745  0.02474886      AA    301
# ... with 336,766 more rows, and 8 more variables: tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

テーブルの形式の変換

・data.frame形式はR特有の行列の形をしたデータ形式。

・data.frameで表現するデータの形式は大別すると

1, wide形式(横持ち形式):データの項目が横に並んだ形式

2, long形式(縦持ち形式):項目名とその値が縦に並んだ形式

があるが、これらはreshapeパッケージを用いると互いに変換できる。

reshapeは、reshape2パッケージにグレードアップして提供されている。

wide型とlong型について

wide型

```
> births.wide
  caseid v012 b2_01 b2_02 b2_03 b4_01 b4_02 b4_03
1      1   30  2000  2005    NA     1     1    NA
2      2   29  2001  2010    NA     1     2    NA
3      3   32  1999  2002  2006     1     1     1
4      4   35  1999  2009    NA     2     1    NA
5      5   34  1998    NA    NA     2    NA    NA
6      6   23    NA    NA    NA    NA    NA    NA
7      7   25  2000    NA    NA     1    NA    NA
```

行が母親のid

long型

行が子供のid

```
> births.long1
    caseid v012 time   b2 b4
1.1      1   30    1 2000  1
2.1      2   29    1 2001  1
3.1      3   32    1 1999  1
4.1      4   35    1 1999  2
5.1      5   34    1 1998  2
6.1      6   23    1   NA NA
7.1      7   25    1 2000  1
1.2      1   30    2 2005  1
2.2      2   29    2 2010  2
3.2      3   32    2 2002  1
4.2      4   35    2 2009  1
5.2      5   34    2   NA NA
6.2      6   23    2   NA NA
7.2      7   25    2   NA NA
1.3      1   30    3   NA NA
2.3      2   29    3   NA NA
3.3      3   32    3 2006  1
4.3      4   35    3   NA NA
5.3      5   34    3   NA NA
6.3      6   23    3   NA NA
7.3      7   25    3   NA NA
```

http://ryunotaro-yomimasu.blogspot.jp/2013/02/rwidelong.html

テーブルの形式の変換

```
# reshape2(Hadley氏作)パッケージのダウンロード
install.packages("reshape2", quiet = TRUE, dependencies=T)
```

```
#パッケージのロード
library(reshape2)
#データのロード
data(smiths)
smiths
```

```
> #パッケージのロード
> library(reshape2)
> #データのロード
> data(smiths)
> smiths
     subject time age weight height
1 John Smith    1  33     90   1.87
2 Mary Smith    1  NA     NA   1.54
```

melt関数での、wideからlongへの変換

```
melt(smiths)
melt(smiths, id = c("subject", "time"), measured = c("age", "weight", "height"))
```

```
> melt(smiths)
Using subject as id variables
     subject variable value
1 John Smith     time  1.00
2 Mary Smith     time  1.00
3 John Smith      age 33.00
4 Mary Smith      age    NA
5 John Smith   weight 90.00
6 Mary Smith   weight    NA
7 John Smith   height  1.87
8 Mary Smith   height  1.54
> melt(smiths, id = c("subject", "time"), measured = c("age", "weight", "height"))
     subject time variable value
1 John Smith    1      age 33.00
2 Mary Smith    1      age    NA
3 John Smith    1   weight 90.00
4 Mary Smith    1   weight    NA
5 John Smith    1   height  1.87
6 Mary Smith    1   height  1.54
```

#na.rm=TRUEにすることで、欠損値を含む行を除去する。

melt(smiths, na.rm = TRUE)

```
> #na.rm=TRUEにすることで、欠損値を含む行を除去する。
> melt(smiths, na.rm = TRUE)
Using subject as id variables
     subject variable value
1 John Smith     time  1.00
2 Mary Smith     time  1.00
3 John Smith      age 33.00
5 John Smith   weight 90.00
7 John Smith   height  1.87
8 Mary Smith   height  1.54
```

dcast関数での、longからwideへの変換

```
smithsm <- melt(smiths)
smithsm
dcast(smithsm, ... ~ variable)
dcast(smithsm, ... ~ subject)
```

```
> smithsm <- melt(smiths)
Using subject as id variables
> smithsm
     subject variable value
1 John Smith     time  1.00
2 Mary Smith     time  1.00
3 John Smith      age 33.00
4 Mary Smith      age    NA
5 John Smith   weight 90.00
6 Mary Smith   weight    NA
7 John Smith   height  1.87
8 Mary Smith   height  1.54
> dcast(smithsm, ... ~ variable)
     subject time age weight height
1 John Smith    1  33     90   1.87
2 Mary Smith    1  NA     NA   1.54
> dcast(smithsm, ... ~ subject)
  variable John Smith Mary Smith
1     time       1.00       1.00
2      age      33.00         NA
3   weight      90.00         NA
4   height       1.87       1.54
```

tidyrパッケージ

- Hadley Wickham氏によって開発された、tidyrパッケージは、より効率的に、wide形式とlong形式のデータフレームを変換する。
- tidyrパッケージは、tidy dataというコンセプトのもとに開発されている。

Journal of Statistical Software

August 2014, Volume 59, Issue 10. http://www.jstatsoft.org/

Tidy Data

Hadley Wickham
RStudio

Abstract

A huge amount of effort is spent cleaning data to get it ready for analysis, but there has been little research on how to make data cleaning as easy and effective as possible. This paper tackles a small, but important, component of data cleaning: data tidying. Tidy datasets are easy to manipulate, model and visualize, and have a specific structure: each variable is a column, each observation is a row, and each type of observational unit is a table. This framework makes it easy to tidy messy datasets because only a small set of tools are needed to deal with a wide range of un-tidy datasets. This structure also makes it easier to develop tidy tools for data analysis, tools that both input and output tidy datasets. The advantages of a consistent data structure and matching tools are demonstrated with a case study free from mundane data manipulation chores.

H.Wickham. Tidy data. The Journal of statistical software, 59(10),2014

パッケージのダウンロード･インストール

```
install.packages("tidyr", quiet = TRUE, dependencies=T)
library(tidyr)
```

gather関数による、wide形式からlong形式への変換

・tidyrパッケージのgather関数は、reshape2パッケージのmelt関数に相当。
・数列にまたがっていた値をカテゴリ変数と値の列に変換することで、wide形式のデータフレームをlong形式のデータフレームに変換する。

iris.l <- gather(iris, variable, value, -Species)

head(iris.l, 3)

```
> iris.l <- gather(iris, variable, value, -Species)
> head(iris.l, 3)
  Species     variable value
1  setosa Sepal.Length   5.1
2  setosa Sepal.Length   4.9
3  setosa Sepal.Length   4.7
```

spread関数によるlong形式からwide形式への変換

tidyrパッケージのspread関数は、reshape2パッケージのdcast関数に相当し、long形式のデータフレームを、wide形式に変換する。

```
library(dplyr)

iris.mean <- iris.l %>% group_by(Species, variable) %>%
summarise(mean = mean(value))

iris.w <- spread(iris.mean, variable, mean)

iris.w
```

```
> library(dplyr)
> iris.mean <- iris.l %>% group_by(Species, variable) %>% summarise(mea$
> iris.w <- spread(iris.mean, variable, mean)
> iris.w
Source: local data frame [3 x 5]
Groups: Species [3]

     Species Petal.Length Petal.Width Sepal.Length Sepal.Width
*     <fctr>        <dbl>       <dbl>        <dbl>       <dbl>
1     setosa        1.462       0.246        5.006       3.428
2 versicolor        4.260       1.326        5.936       2.770
3  virginica        5.552       2.026        6.588       2.974
```

separate関数によるlongからwide形式への変換

tidyrパッケージのseparate関数は、reshape2パッケージのcolsplit関数に相当し、キーとなる列を複数の列に分割する。

```
iris.l <- gather(iris, variable, value, -Species)
iris.l.sep <- separate(iris.l, variable, c("part", "variable"))
head(iris.l.sep, 3)
```

```
> iris.l <- gather(iris, variable, value, -Species)
> iris.l.sep <- separate(iris.l, variable, c("part", "variable"))
> head(iris.l.sep, 3)
  Species  part variable value
1  setosa Sepal   Length   5.1
2  setosa Sepal   Length   4.9
3  setosa Sepal   Length   4.7
```

unite関数による複数列の結合

unite関数は、separate関数の逆で、複数列の値を、一列に結合する。

iris.l.sep %>% unite("var", c(part, variable), sep = ".") %>% head(3)

```
> iris.l.sep %>% unite("var", c(part, variable), sep = ".") %>% head(3)
  Species          var value
1  setosa Sepal.Length   5.1
2  setosa Sepal.Length   4.9
3  setosa Sepal.Length   4.7
```

data.tableのハンドリング

- data.tableパッケージを使用する。

data.tableパッケージ

- data.frameを継承したdata.tableの型とそれに関する処理方法を提供する。
- キーの設定、バイナリサーチを用いた高速な検索、グループごとの処理などを提供。
- data.frameに対する処理より、data.tableに対する処理のほうが高速。

パッケージのダウンロード・インストール

```
install.packages("data.table", quiet = TRUE, dependencies=T)
```

```
#パッケージのメモリへのロード
library(data.table)
#fread関数を用いたデータの読み込み
system.time(al.2008.dt <- fread("data/DataExpo2009/2008.csv"))
```

```
> #fread関数を用いたデータの読み込み
> system.time(al.2008.dt <- fread("data/DataExpo2009/2008.csv"))
Read 7009728 rows and 29 (of 29) columns from 0.642 GB file in 00:00:12
   user  system elapsed
  10.39    0.59   11.90
Warning message:
In fread("data/DataExpo2009/2008.csv") :
  Bumped column 23 to type character on data row 179, field contains 'A'. Coercing pre$
```

```
#データ型の確認
class(al.2008.dt)
#データサイズの確認
dim(al.2008.dt)
```

```
> #データ型の確認
> class(al.2008.dt)
[1] "data.table" "data.frame"
> #データサイズの確認
> dim(al.2008.dt)
[1] 7009728      29
```

メモリ上に生成されたデータテーブルのリストの確認

・メモリ上に生成されたデータテーブルのリストは以下のように、tables関数を用いて確認できる。
・NAMEはオブジェクト名、NROWは行数、NCOLは列数、MBはデータサイズ、COLSは列名、KEYはキー。

tables()

```
> tables()
     NAME                   NROW NCOL  MB
[1,] al.2008.dt 7,009,728     29 910
     COLS
[1,] Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarr
     KEY
[1,]
Total: 910MB
```

行の抽出

#data.frameと同じように行を指定する。

al.2008.dt[1:2,]

```
> #data.frameと同じように行を指定する。
> al.2008.dt[1:2, ]
   Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime UniqueCarrier
1: 2008     1          3         4    2003       1955    2211       2225            WN
2: 2008     1          3         4     754        735    1002       1000            WN
   FlightNum TailNum ActualElapsedTime CRSElapsedTime AirTime ArrDelay DepDelay Origin
1:       335  N712SW               128            150     116      -14        8    IAD
2:      3231  N772SW               128            145     113        2       19    IAD
   Dest Distance TaxiIn TaxiOut Cancelled CancellationCode Diverted CarrierDelay
1:  TPA      810      4       8         0                         0           NA
2:  TPA      810      5      10         0                         0           NA
   WeatherDelay NASDelay SecurityDelay LateAircraftDelay
1:           NA       NA            NA                NA
2:           NA       NA            NA                NA
```

列の抽出

抽出する列名をlistの要素にして、指定したり、with=FALSEのオプションをつけて、列番号や列名を指定。

al.2008.dt[1:2, list(Year, Month, DayofMonth)]

al.2008.dt[1:2, 1:3, with = FALSE]

```
> al.2008.dt[1:2, list(Year, Month, DayofMonth)]
   Year Month DayofMonth
1: 2008     1          3
2: 2008     1          3
> al.2008.dt[1:2, 1:3, with = FALSE]
   Year Month DayofMonth
1: 2008     1          3
2: 2008     1          3
```

キーの設定

#月(Month)と曜日(DayOfWeek)の二つをキーに設定

setkey(al.2008.dt, Month, DayOfWeek)

#キーが設定されていることを確認

tables()

```
> #月(Month)と曜日(DayOfWeek)の二つをキーに設定
> setkey(al.2008.dt, Month, DayOfWeek)
> #キーが設定されていることを確認
> tables()
     NAME                NROW NCOL  MB
[1,] al.2008.dt 7,009,728   29 910
     COLS
[1,] Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarr
     KEY
[1,] Month,DayOfWeek
Total: 910MB
```

バイナリサーチによるデータの高速抽出

キーを設定することで、バイナリサーチによりデータを高速に抽出することができる。

```
#4月月曜日のフライトデータの抽出
al.2008.dt[J(4, 1)]
```

```
> #4月月曜日のフライトデータの抽出
> al.2008.dt[J(4, 1)]
       Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
    1: 2008     4          7         1    1950       1955    2153       2150
    2: 2008     4          7         1     858        900    1423       1435
    3: 2008     4          7         1     909        910    1138       1145
    4: 2008     4          7         1     649        655     920        930
    5: 2008     4          7         1    1312       1315    1541       1550
   ---
82459: 2008     4         14         1    1830       1829    1945       1951
82460: 2008     4         14         1      NA       1930      NA       2046
82461: 2008     4         14         1    1929       1930    2038       2049
82462: 2008     4         14         1    2030       2030    2146       2147
82463: 2008     4         14         1    2042       2030    2202       2149
       UniqueCarrier FlightNum TailNum ActualElapsedTime CRSElapsedTime AirTime ArrDelay
    1:            WN       609  N623SW                63             55      43        3
    2:            WN      3257  N795SW               205            215     194      -12
    3:            WN        77  N694SW                89             95      78       -7
    4:            WN        87  N342SW                91             95      78      -10
    5:            WN       214  N770SA                89             95      78       -9
   ---
82459:            DL      1965  N914DE                75             82      46       -6
82460:            DL      1966  N908DE                NA             76      NA       NA
82461:            DL      1967  N909DE                69             79      38      -11
82462:            DL      1968  N914DE                76             77      50       -1
82463:            DL      1969  N908DE                80             79      43       13
       DepDelay Origin Dest Distance TaxiIn TaxiOut Cancelled CancellationCode Diverted
    1:       -5    ABQ  AMA      277      6      14         0                         0
    2:       -2    ABQ  BWI     1670      4       7         0                         0
    3:       -1    ABQ  DAL      580      4       7         0                         0
    4:       -6    ABQ  DAL      580      3      10         0                         0
    5:       -3    ABQ  DAL      580      3       8         0                         0
   ---
82459:        1    LGA  DCA      214      3      26         0                         0
82460:       NA    DCA  LGA      214     NA      NA         1                A        0
82461:       -1    LGA  DCA      214      4      27         0                         0
82462:        0    DCA  LGA      214      3      23         0                         0
82463:       12    LGA  DCA      214      4      33         0                         0
```

高速なテーブル結合

data.tableパッケージを用いると、テーブルの結合も高速化できる。以下の例は、データセットをデータテーブルに変換し、キー設定した後に、flightsデータセットとairportsデータセットを、内部結合および外部結合している。

```
library(data.table)
library(nycflights13)
flights
airports
```

```
> library(data.table)
> library(nycflights13)
> flights
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum origin  dest
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>  <int>   <chr>  <chr> <chr>
1   2013     1     1      517            515         2      830            819        11      UA   1545  N14228    EWR   IAH
2   2013     1     1      533            529         4      850            830        20      UA   1714  N24211    LGA   IAH
3   2013     1     1      542            540         2      923            850        33      AA   1141  N619AA    JFK   MIA
4   2013     1     1      544            545        -1     1004           1022       -18      B6    725  N804JB    JFK   BQN
5   2013     1     1      554            600        -6      812            837       -25      DL    461  N668DN    LGA   ATL
6   2013     1     1      554            558        -4      740            728        12      UA   1696  N39463    EWR   ORD
7   2013     1     1      555            600        -5      913            854        19      B6    507  N516JB    EWR   FLL
8   2013     1     1      557            600        -3      709            723       -14      EV   5708  N829AS    LGA   IAD
9   2013     1     1      557            600        -3      838            846        -8      B6     79  N593JB    JFK   MCO
10  2013     1     1      558            600        -2      753            745         8      AA    301  N3ALAA    LGA   ORD
# ... with 336,766 more rows, and 5 more variables: air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
> airports
# A tibble: 1,396 × 7
     faa                           name      lat        lon   alt    tz   dst
   <chr>                          <chr>    <dbl>      <dbl> <int> <dbl> <chr>
1    04G              Lansdowne Airport 41.13047  -80.61958  1044    -5     A
2    06A  Moton Field Municipal Airport 32.46057  -85.68003   264    -5     A
3    06C            Schaumburg Regional 41.98934  -88.10124   801    -6     A
4    06N                Randall Airport 41.43191  -74.39156   523    -5     A
5    09J          Jekyll Island Airport 31.07447  -81.42778    11    -4     A
6    0A9 Elizabethton Municipal Airport 36.37122  -82.17342  1593    -4     A
7    0G6        Williams County Airport 41.46731  -84.50678   730    -5     A
8    0G7  Finger Lakes Regional Airport 42.88356  -76.78123   492    -5     A
9    0P2   Shoestring Aviation Airfield 39.79482  -76.64719  1000    -5     U
10   0S9          Jefferson County Intl 48.05381 -122.81064   108    -8     A
# ... with 1,386 more rows
```

高速なテーブル結合

data.tableパッケージを用いると、テーブルの結合も高速化できる。以下の例は、データセットをデータテーブルに変換し、キー設定した後に、flightsデータセットとairportsデータセットを、内部結合および外部結合している。

```
# データテーブルへの変換
flights.dt <- data.table(flights)
flights.dt
airports.dt <- data.table(airports)
airports.dt
```

```
> flights.dt <- data.table(flights)
> flights.dt
        year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum origin dest
     1: 2013     1   1      517            515         2      830            819        11      UA   1545  N14228    EWR  IAH
     2: 2013     1   1      533            529         4      850            830        20      UA   1714  N24211    LGA  IAH
     3: 2013     1   1      542            540         2      923            850        33      AA   1141  N619AA    JFK  MIA
     4: 2013     1   1      544            545        -1     1004           1022       -18      B6    725  N804JB    JFK  BQN
     5: 2013     1   1      554            600        -6      812            837       -25      DL    461  N668DN    LGA  ATL
    ---
336772: 2013     9  30       NA           1455        NA       NA           1634        NA      9E   3393      NA    JFK  DCA
336773: 2013     9  30       NA           2200        NA       NA           2312        NA      9E   3525      NA    LGA  SYR
336774: 2013     9  30       NA           1210        NA       NA           1330        NA      MQ   3461  N535MQ    LGA  BNA
336775: 2013     9  30       NA           1159        NA       NA           1344        NA      MQ   3572  N511MQ    LGA  CLE
336776: 2013     9  30       NA            840        NA       NA           1020        NA      MQ   3531  N839MQ    LGA  RDU
        air_time distance hour minute           time_hour
     1:      227     1400    5     15 2013-01-01 05:00:00
     2:      227     1416    5     29 2013-01-01 05:00:00
     3:      160     1089    5     40 2013-01-01 05:00:00
     4:      183     1576    5     45 2013-01-01 05:00:00
     5:      116      762    6      0 2013-01-01 06:00:00
    ---
336772:       NA      213   14     55 2013-09-30 14:00:00
336773:       NA      198   22      0 2013-09-30 22:00:00
336774:       NA      764   12     10 2013-09-30 12:00:00
336775:       NA      419   11     59 2013-09-30 11:00:00
336776:       NA      431    8     40 2013-09-30 08:00:00
```

```
> airports.dt <- data.table(airports)
> airports.dt
      faa                          name      lat        lon  alt tz dst
   1: 04G             Lansdowne Airport 41.13047  -80.61958 1044 -5   A
   2: 06A Moton Field Municipal Airport 32.46057  -85.68003  264 -5   A
   3: 06C           Schaumburg Regional 41.98934  -88.10124  801 -6   A
   4: 06N               Randall Airport 41.43191  -74.39156  523 -5   A
   5: 09J         Jekyll Island Airport 31.07447  -81.42778   11 -4   A
  ---
1392: ZUN                    Black Rock 35.08323 -108.79178 6454 -7   A
1393: ZVE         New Haven Rail Station 41.29867  -72.92599    7 -5   A
1394: ZWI      Wilmington Amtrak Station 39.73667  -75.55167    0 -5   A
1395: ZWU       Washington Union Station 38.89746  -77.00643   76 -5   A
1396: ZYP                 Penn Station 40.75050  -73.99350   35 -5   A
```

高速なテーブル結合

data.tableパッケージを用いると、テーブルの結合も高速化できる。以下の例は、データセットをデータテーブルに変換し、キー設定した後に、flightsデータセットとairportsデータセットを、内部結合および外部結合している。

```
# キーの設定
setkey(flights.dt, origin)
setkey(airports.dt, faa)
# 内部結合
flights.dt[airports.dt, nomatch = 0]
# 外部結合(airports.dtをflights.dtに右結合)
flights.dt[airports.dt, nomatch = NA, allow.cartesian = TRUE]
```

```
> # キーの設定
> setkey(flights.dt, origin)
> setkey(airports.dt, faa)
> flights.dt
        year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum origin dest
     1: 2013     1   1      517            515         2      830            819        11      UA   1545  N14228    EWR  IAH
     2: 2013     1   1      554            558        -4      740            728        12      UA   1696  N39463    EWR  ORD
     3: 2013     1   1      555            600        -5      913            854        19      B6    507  N516JB    EWR  FLL
     4: 2013     1   1      558            600        -2      923            937       -14      UA   1124  N53441    EWR  SFO
     5: 2013     1   1      559            600        -1      854            902        -8      UA   1187  N76515    EWR  LAS
    ---
336772: 2013     9  30       NA           1842        NA       NA           2019        NA      EV   5274  N740EV    LGA  BNA
336773: 2013     9  30       NA           2200        NA       NA           2312        NA      9E   3525      NA    LGA  SYR
336774: 2013     9  30       NA           1210        NA       NA           1330        NA      MQ   3461  N535MQ    LGA  BNA
336775: 2013     9  30       NA           1159        NA       NA           1344        NA      MQ   3572  N511MQ    LGA  CLE
336776: 2013     9  30       NA            840        NA       NA           1020        NA      MQ   3531  N839MQ    LGA  RDU
        air_time distance hour minute           time_hour
     1:      227     1400    5     15 2013-01-01 05:00:00
     2:      150      719    5     58 2013-01-01 05:00:00
     3:      158     1065    6      0 2013-01-01 06:00:00
     4:      361     2565    6      0 2013-01-01 06:00:00
     5:      337     2227    6      0 2013-01-01 06:00:00
    ---
336772:       NA      764   18     42 2013-09-30 18:00:00
336773:       NA      198   22      0 2013-09-30 22:00:00
336774:       NA      764   12     10 2013-09-30 12:00:00
336775:       NA      419   11     59 2013-09-30 11:00:00
336776:       NA      431    8     40 2013-09-30 08:00:00
```

```
> airports.dt
       faa                           name      lat        lon  alt tz dst
   1: 04G             Lansdowne Airport 41.13047  -80.61958 1044 -5   A
   2: 06A Moton Field Municipal Airport 32.46057  -85.68003  264 -5   A
   3: 06C           Schaumburg Regional 41.98934  -88.10124  801 -6   A
   4: 06N               Randall Airport 41.43191  -74.39156  523 -5   A
   5: 09J         Jekyll Island Airport 31.07447  -81.42778   11 -4   A
  ---
1392: ZUN                    Black Rock 35.08323 -108.79178 6454 -7   A
1393: ZVE         New Haven Rail Station 41.29867  -72.92599    7 -5   A
1394: ZWI      Wilmington Amtrak Station 39.73667  -75.55167    0 -5   A
1395: ZWU       Washington Union Station 38.89746  -77.00643   76 -5   A
1396: ZYP                  Penn Station 40.75050  -73.99350   35 -5   A
```

```
> # 内部結合
> flights.dt[airports.dt, nomatch = 0]
        year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum origin dest
     1: 2013     1   1      517            515         2      830            819        11      UA   1545  N14228    EWR  IAH
     2: 2013     1   1      554            558        -4      740            728        12      UA   1696  N39463    EWR  ORD
     3: 2013     1   1      555            600        -5      913            854        19      B6    507  N516JB    EWR  FLL
     4: 2013     1   1      558            600        -2      923            937       -14      UA   1124  N53441    EWR  SFO
     5: 2013     1   1      559            600        -1      854            902        -8      UA   1187  N76515    EWR  LAS
    ---
336772: 2013     9  30       NA           1842        NA       NA           2019        NA      EV   5274  N740EV    LGA  BNA
336773: 2013     9  30       NA           2200        NA       NA           2312        NA      9E   3525      NA    LGA  SYR
336774: 2013     9  30       NA           1210        NA       NA           1330        NA      MQ   3461  N535MQ    LGA  BNA
336775: 2013     9  30       NA           1159        NA       NA           1344        NA      MQ   3572  N511MQ    LGA  CLE
336776: 2013     9  30       NA            840        NA       NA           1020        NA      MQ   3531  N839MQ    LGA  RDU
        air_time distance hour minute           time_hour                name      lat       lon alt tz dst
     1:      227     1400    5     15 2013-01-01 05:00:00 Newark Liberty Intl 40.69250 -74.16867  18 -5   A
     2:      150      719    5     58 2013-01-01 05:00:00 Newark Liberty Intl 40.69250 -74.16867  18 -5   A
     3:      158     1065    6      0 2013-01-01 06:00:00 Newark Liberty Intl 40.69250 -74.16867  18 -5   A
     4:      361     2565    6      0 2013-01-01 06:00:00 Newark Liberty Intl 40.69250 -74.16867  18 -5   A
     5:      337     2227    6      0 2013-01-01 06:00:00 Newark Liberty Intl 40.69250 -74.16867  18 -5   A
    ---
336772:       NA      764   18     42 2013-09-30 18:00:00          La Guardia 40.77725 -73.87261  22 -5   A
336773:       NA      198   22      0 2013-09-30 22:00:00          La Guardia 40.77725 -73.87261  22 -5   A
336774:       NA      764   12     10 2013-09-30 12:00:00          La Guardia 40.77725 -73.87261  22 -5   A
336775:       NA      419   11     59 2013-09-30 11:00:00          La Guardia 40.77725 -73.87261  22 -5   A
336776:       NA      431    8     40 2013-09-30 08:00:00          La Guardia 40.77725 -73.87261  22 -5   A
```

```
> # 外部結合(airports.dtをflights.dtに右結合)
> flights.dt[airports.dt, nomatch = NA, allow.cartesian = TRUE]
        year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum origin dest
     1:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    04G   NA
     2:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    06A   NA
     3:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    06C   NA
     4:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    06N   NA
     5:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    09J   NA
    ---
338165:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    ZUN   NA
338166:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    ZVE   NA
338167:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    ZWI   NA
338168:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    ZWU   NA
338169:   NA    NA  NA       NA             NA        NA       NA             NA        NA      NA     NA      NA    ZYP   NA
        air_time distance hour minute time_hour                          name      lat        lon  alt tz dst
     1:       NA       NA   NA     NA      <NA>             Lansdowne Airport 41.13047  -80.61958 1044 -5   A
     2:       NA       NA   NA     NA      <NA> Moton Field Municipal Airport 32.46057  -85.68003  264 -5   A
     3:       NA       NA   NA     NA      <NA>           Schaumburg Regional 41.98934  -88.10124  801 -6   A
     4:       NA       NA   NA     NA      <NA>               Randall Airport 41.43191  -74.39156  523 -5   A
     5:       NA       NA   NA     NA      <NA>         Jekyll Island Airport 31.07447  -81.42778   11 -4   A
    ---
338165:       NA       NA   NA     NA      <NA>                    Black Rock 35.08323 -108.79178 6454 -7   A
338166:       NA       NA   NA     NA      <NA>      New Haven Rail Station 41.29867  -72.92599    7 -5   A
338167:       NA       NA   NA     NA      <NA>     Wilmington Amtrak Station 39.73667  -75.55167    0 -5   A
338168:       NA       NA   NA     NA      <NA>     Washington Union Station 38.89746  -77.00643   76 -5   A
338169:       NA       NA   NA     NA      <NA>                  Penn Station 40.75050  -73.99350   35 -5   A
```

#外部結合(airports.dtをflights.dtに右結合)

airports.dt[flights.dt, nomatch=NA]

```
> airports.dt[flights.dt, nomatch=NA]
        faa                  name      lat       lon alt tz dst year month day dep_time sched_dep_time dep_delay arr_time
     1: EWR Newark Liberty Intl 40.69250 -74.16867  18 -5   A 2013     1   1      517            515         2      830
     2: EWR Newark Liberty Intl 40.69250 -74.16867  18 -5   A 2013     1   1      554            558        -4      740
     3: EWR Newark Liberty Intl 40.69250 -74.16867  18 -5   A 2013     1   1      555            600        -5      913
     4: EWR Newark Liberty Intl 40.69250 -74.16867  18 -5   A 2013     1   1      558            600        -2      923
     5: EWR Newark Liberty Intl 40.69250 -74.16867  18 -5   A 2013     1   1      559            600        -1      854
    ---
336772: LGA           La Guardia 40.77725 -73.87261  22 -5   A 2013     9  30       NA           1842        NA       NA
336773: LGA           La Guardia 40.77725 -73.87261  22 -5   A 2013     9  30       NA           2200        NA       NA
336774: LGA           La Guardia 40.77725 -73.87261  22 -5   A 2013     9  30       NA           1210        NA       NA
336775: LGA           La Guardia 40.77725 -73.87261  22 -5   A 2013     9  30       NA           1159        NA       NA
336776: LGA           La Guardia 40.77725 -73.87261  22 -5   A 2013     9  30       NA            840        NA       NA
        sched_arr_time arr_delay carrier flight tailnum dest air_time distance hour minute           time_hour
     1:            819        11      UA   1545  N14228  IAH      227     1400    5     15 2013-01-01 05:00:00
     2:            728        12      UA   1696  N39463  ORD      150      719    5     58 2013-01-01 05:00:00
     3:            854        19      B6    507  N516JB  FLL      158     1065    6      0 2013-01-01 06:00:00
     4:            937       -14      UA   1124  N53441  SFO      361     2565    6      0 2013-01-01 06:00:00
     5:            902        -8      UA   1187  N76515  LAS      337     2227    6      0 2013-01-01 06:00:00
    ---
336772:           2019        NA      EV   5274  N740EV  BNA       NA      764   18     42 2013-09-30 18:00:00
336773:           2312        NA      9E   3525      NA  SYR       NA      198   22      0 2013-09-30 22:00:00
336774:           1330        NA      MQ   3461  N535MQ  BNA       NA      764   12     10 2013-09-30 12:00:00
336775:           1344        NA      MQ   3572  N511MQ  CLE       NA      419   11     59 2013-09-30 11:00:00
336776:           1020        NA      MQ   3531  N839MQ  RDU       NA      431    8     40 2013-09-30 08:00:00
```