

dplyrとtidyrを使った データラングリング cheatsheet



文法 - ラングリングに役立つおまじない

dplyr::tbl_df(iris)

データフレームからテーブルへ変換。テーブルの情報は画面に収まるよう表示され、データフレームよりも扱いやすい。

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1      5.1      3.5      1.4
2      4.9      3.0      1.4
3      4.7      3.2      1.3
4      4.6      3.1      1.5
5      5.0      3.6      1.4
...      ...      ...
Variables not shown: Petal.Width (dbl),
Species (fctr)
```

dplyr::glimpse(iris)

テーブルの要約を確認

utils::View(iris)

スプレッドシート形式でデータを確認 (Vは大文字)

dplyr::%>%

左辺のオブジェクトを、右辺関数の第一変数（または.で指定した引数）として渡す。

`x %>% f(y)` は `f(x, y)` と同じ
`y %>% f(x, ., z)` は `f(x, y, z)` と同じ

%>% を使うとコードが読みやすくなる。例えば:

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Tidy Data - Rでのデータ整形初歩

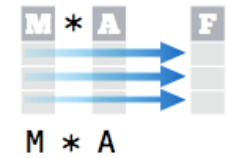
tidyの
データセット
では:



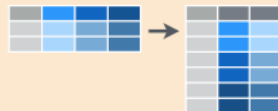
&



Tidy dataは、Rのベクトル操作を補完する。Rは変数(列)を操作しても、自動的にオブザベーション(行)を保つ。



データを変形する - データセットのレイアウト変更



`tidyr::gather(cases, "year", "n", 2:4)`
各列を行に展開
(ワイド型からロング型へ変換)



`tidyr::separate(storms, date, c("y", "m", "d"))`
一列を、複数列に分割



`tidyr::spread(pollution, size, amount)`
各行を列に展開
(ロング型からワイド型へ変換)



`tidyr::unite(data, col, ..., sep)`
複数列を、一列に集約・結合

`dplyr::data_frame(a = 1:3, b = 4:6)`
複数のベクトルを結合し、データフレーム化(最適化)

`dplyr::arrange(mtcars, mpg)`
列の値で、行を並び替え(昇順)

`dplyr::arrange(mtcars, desc(mpg))`
列の値で、行を並び替え(降順)

`dplyr::rename(tb, y = year)`
データフレームの列名を変更

行の一部を取り出す



`dplyr::filter(iris, Sepal.Length > 7)`
条件式の評価に合う行を抽出

`dplyr::distinct(iris)`
重複行を削除

`dplyr::sample_frac(iris, 0.5, replace = TRUE)`
無作為に指定割合のサンプルを抽出

`dplyr::sample_n(iris, 10, replace = TRUE)`
無作為にn行のサンプルを抽出

`dplyr::slice(iris, 10:15)`

Select rows by position. 行を位置で指定し、選択する

`dplyr::top_n(storms, 2, date)`

上位のnエンタリを並び替え抽出(グループ化済ならグループ毎に)

Rの演算子 - ?Comparison, ?base::Logic

<	より小さい	!=	等しくない
>	より大きい	%in%	グループメンバシップ
==	等しい	is.na	NA (欠損値) である
<=	等しいか、より小さい	!is.na	NA (欠損値) でない
>=	等しいか、より大きい	&, , !, xor, any, all	ブール型演算子

列の一部を取り出す



`dplyr::select(iris, Sepal.Width, Petal.Length, Species)`
列を、名前またはヘルパー関数を使って抽出

Selectのヘルパー関数 - ?select

- `select(iris, contains("."))`
列名にある文字列を含む列を選択
- `select(iris, ends_with("Length"))`
列名がある文字列で終わる列を選択
- `select(iris, everything())`
Select every column. 全ての列を選択
- `select(iris, matches(".t."))`
列名がある正規表現に一致する列を選択
- `select(iris, num_range("x", 1:5))`
列名がx1, x2, x3, x4, x5である列を選択
- `select(iris, one_of(c("Species", "Genus")))`
列名が列挙した名前に一致する列を選択
- `select(iris, starts_with("Sepal"))`
列名がある文字列で始まる列を選択
- `select(iris, Sepal.Length:Petal.Width)`
Sepal.LengthとPetal.Widthの間にある全列を選択 (二列を含む)
- `select(iris, -Species)`
Species以外の全列を選択

データを要約する



`dplyr::summarise(iris, avg = mean(Sepal.Length))`

複数行のデータを一行に要約

`dplyr::summarise_each(iris, funs(mean))`

summary関数を各列に適用

`dplyr::count(iris, Species, wt = Sepal.Length)`

変数の個数を重複を除き数え上げ(重み付け可)



summarise()は、summary関数を使う。summary関数は、複数値のベクトル入力に対し、ひとつの値を返す

例:

<code>dplyr::first</code>	ベクトル1つ目の値	<code>min</code>	ベクトルの最小値
<code>dplyr::last</code>	ベクトル最後の値	<code>max</code>	ベクトルの最大値
<code>dplyr::nth</code>	ベクトルn番目の値	<code>mean</code>	ベクトルの平均値
<code>dplyr::n</code>	ベクトルの要素数	<code>median</code>	ベクトルの中間値
<code>dplyr::n_distinct</code>	ベクトルの重複を省いた要素数	<code>var</code>	ベクトルの不偏分散
<code>IQR</code>	ベクトルのIQR	<code>sd</code>	ベクトルの不偏標準偏差

データをグルーピングする

`dplyr::group_by(iris, Species)`

Speciesが同値のものは1行にグループ化

`dplyr::ungroup(iris)`

データフレームからグループ化情報を削除

`iris %>% group_by(Species) %>% summarise(...)`

各グループ毎に要約を計算



新しい列を追加する



`dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)`

1つ以上の列を計算・生成し追加

`dplyr::mutate_each(iris, funs(min_rank))`

window関数を各列に適用

`dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)`

1つ以上の新しい列を計算・生成し元列を削除



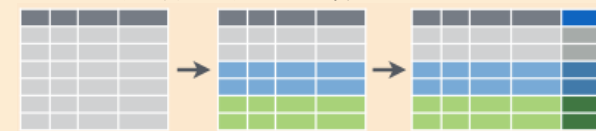
mutate()は、window関数を使う。window関数は、複数値のベクトル入力に対し、別のベクトルを返す

例:

<code>dplyr::lead</code>	前にずらす(先頭消し末尾にNA)	<code>dplyr::cumall</code>	部分条件判定(all)
<code>dplyr::lag</code>	後ろにずらす(末尾消し先頭にNA)	<code>dplyr::cumany</code>	部分条件判定(any)
<code>dplyr::dense_rank</code>	順位(同着は順位飛ばさず)	<code>dplyr::cummean</code>	部分平均
<code>dplyr::min_rank</code>	順位(同着は最小順位へ合せ次の順位飛ばす)	<code>cumsum</code>	部分和
<code>dplyr::percent_rank</code>	0~1に標準化した順位(順位%)	<code>cummax</code>	部分最大
<code>dplyr::row_number</code>	順位(同着なし、上から順。行番号)	<code>cummin</code>	部分最小
<code>dplyr::ntile</code>	n個の群に分割	<code>cumprod</code>	部分積
<code>dplyr::between</code>	値がaとbの間か否か	<code>pmax</code>	要素間の最大
<code>dplyr::cume_dist</code>	順位の%の積み上げ	<code>pmin</code>	要素間の最小

`iris %>% group_by(Species) %>% mutate(...)`

グループ毎に新しい変数を計算



データセットを結合する

a		b		
x1	x2	x1	x3	
A	1	A	T	+
B	2	B	F	
C	3	D	T	

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

`dplyr::left_join(a, b, by = "x1")`

bをaに対応付け、結合する(aが全て残る)

x1	x3	x2
A	T	1
B	F	2
D	T	NA

`dplyr::right_join(a, b, by = "x1")`

aをbに対応付け、結合する(bが全て残る)

x1	x2	x3
A	1	T
B	2	F

`dplyr::inner_join(a, b, by = "x1")`

内部結合。a, b両方にある行のみ残す。

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

`dplyr::full_join(a, b, by = "x1")`

外部結合。全ての値と行を残す。

Filtering Joins

x1	x2
A	1
B	2

`dplyr::semi_join(a, b, by = "x1")`

bに対応する行をもつaの行を抽出

x1	x2
C	3

`dplyr::anti_join(a, b, by = "x1")`

bに対応する行を持たないaの行を抽出

y		z		
x1	x2	x1	x2	
A	1	B	2	+
B	2	C	3	
C	3	D	4	

Set Operations

x1	x2
B	2
C	3

`dplyr::intersect(y, z)`

yとz両方に現れる行を抽出

x1	x2
A	1
B	2
C	3
D	4

`dplyr::union(y, z)`

yとz両方または片方に現れる行を抽出

x1	x2
A	1

`dplyr::setdiff(y, z)`

yに現れるが、zに現れない行を抽出

Binding

x1	x2
A	1
B	2
C	3

`dplyr::bind_rows(y, z)`

yに、zを新しい行として追加

x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4

`dplyr::bind_cols(y, z)`

yに、zを新しい列として追加

注意: 行は位置ベースで対応づけ