


# Data Wrangling with dplyr and tidyr

## Cheat Sheet



### Syntax - Helpful conventions for wrangling

#### `dplyr::tbl_df(iris)`

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
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)`

Information dense summary of tbl data.

#### `utils::View(iris)`

View data set in spreadsheet-like display (note capital V).

	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
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

#### `dplyr::%>%`

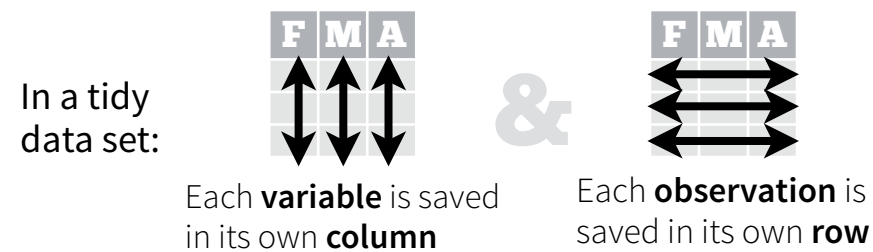
Passes object on left hand side as first argument (or . argument) of function on righthand side.

`x %>% f(y)` is the same as `f(x, y)`  
`y %>% f(x, ., z)` is the same as `f(x, y, z)`

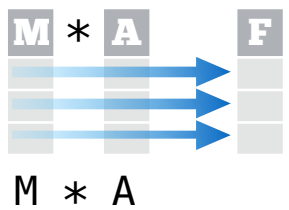
"Piping" with `%>%` makes code more readable, e.g.

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

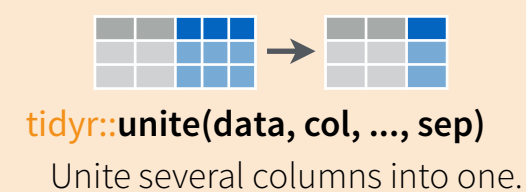
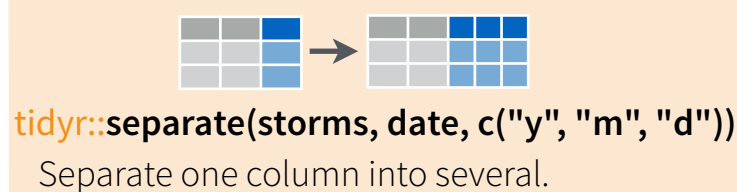
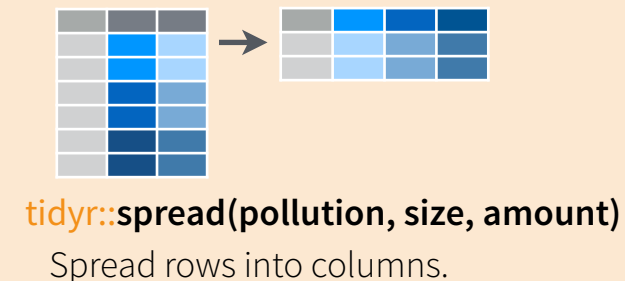
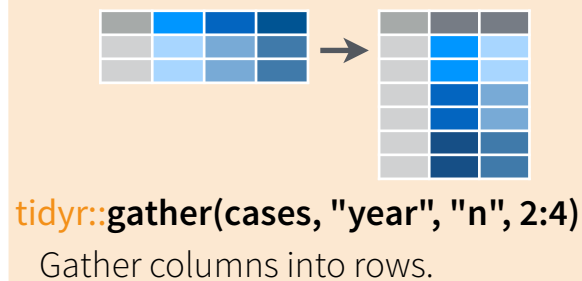
### Tidy Data - A foundation for wrangling in R



Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



### Reshaping Data - Change the layout of a data set



`dplyr::data_frame(a = 1:3, b = 4:6)`  
Combine vectors into data frame (optimized).

`dplyr::arrange(mtcars, mpg)`  
Order rows by values of a column (low to high).

`dplyr::arrange(mtcars, desc(mpg))`  
Order rows by values of a column (high to low).

`dplyr::rename(tb, y = year)`  
Rename the columns of a data frame.

### Subset Observations (Rows)



`dplyr::filter(iris, Sepal.Length > 7)`  
Extract rows that meet logical criteria.

`dplyr::distinct(iris)`  
Remove duplicate rows.

`dplyr::sample_frac(iris, 0.5, replace = TRUE)`  
Randomly select fraction of rows.

`dplyr::sample_n(iris, 10, replace = TRUE)`  
Randomly select n rows.

`dplyr::slice(iris, 10:15)`  
Select rows by position.

`dplyr::top_n(storms, 2, date)`  
Select and order top n entries (by group if grouped data).

### Subset Variables (Columns)



`dplyr::select(iris, Sepal.Width, Petal.Length, Species)`  
Select columns by name or helper function.

#### Helper functions for select - ?select

`select(iris, contains("."))`  
Select columns whose name contains a character string.

`select(iris, ends_with("Length"))`  
Select columns whose name ends with a character string.

`select(iris, everything())`  
Select every column.

`select(iris, matches(".t."))`  
Select columns whose name matches a regular expression.

`select(iris, num_range("x", 1:5))`  
Select columns named x1, x2, x3, x4, x5.

`select(iris, one_of(c("Species", "Genus")))`  
Select columns whose names are in a group of names.

`select(iris, starts_with("Sepal"))`  
Select columns whose name starts with a character string.

`select(iris, Sepal.Length:Petal.Width)`  
Select all columns between Sepal.Length and Petal.Width (inclusive).

`select(iris, -Species)`  
Select all columns except Species.

#### Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,  , !, xor, any, all	Boolean operators

## Summarise Data



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

Summarise data into single row of values.

**dplyr::summarise\_each(iris, funs(mean))**

Apply summary function to each column.

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

Count number of rows with each unique value of variable (with or without weights).



Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

**dplyr::first**

First value of a vector.

**dplyr::last**

Last value of a vector.

**dplyr::nth**

Nth value of a vector.

**dplyr::n**

# of values in a vector.

**dplyr::n\_distinct**

# of distinct values in a vector.

**IQR**

IQR of a vector.

**min**

Minimum value in a vector.

**max**

Maximum value in a vector.

**mean**

Mean value of a vector.

**median**

Median value of a vector.

**var**

Variance of a vector.

**sd**

Standard deviation of a vector.

## Group Data

**dplyr::group\_by(iris, Species)**

Group data into rows with the same value of Species.

**dplyr::ungroup(iris)**

Remove grouping information from data frame.

**iris %>% group\_by(Species) %>% summarise(...)**

Compute separate summary row for each group.



## Make New Variables



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

Compute and append one or more new columns.

**dplyr::mutate\_each(iris, funs(min\_rank))**

Apply window function to each column.

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

Compute one or more new columns. Drop original columns.



Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

**dplyr::lead**

Copy with values shifted by 1.

**dplyr::lag**

Copy with values lagged by 1.

**dplyr::dense\_rank**

Ranks with no gaps.

**dplyr::min\_rank**

Ranks. Ties get min rank.

**dplyr::percent\_rank**

Ranks rescaled to [0, 1].

**dplyr::row\_number**

Ranks. Ties got to first value.

**dplyr::ntile**

Bin vector into n buckets.

**dplyr::between**

Are values between a and b?

**dplyr::cume\_dist**

Cumulative distribution.

**dplyr::cumall**

Cumulative **all**

**dplyr::cumany**

Cumulative **any**

**dplyr::cummean**

Cumulative **mean**

**cumsum**

Cumulative **sum**

**cummax**

Cumulative **max**

**cummin**

Cumulative **min**

**cumprod**

Cumulative **prod**

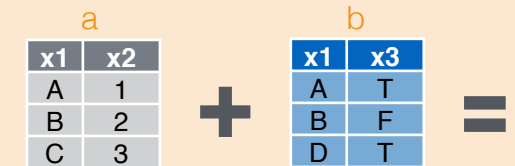
**pmax**

Element-wise **max**

**pmin**

Element-wise **min**

## Combine Data Sets



Mutating Joins

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

**dplyr::left\_join(a, b, by = "x1")**

Join matching rows from b to a.

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

**dplyr::right\_join(a, b, by = "x1")**

Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

**dplyr::inner\_join(a, b, by = "x1")**

Join data. Retain only rows in both sets.

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

**dplyr::full\_join(a, b, by = "x1")**

Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

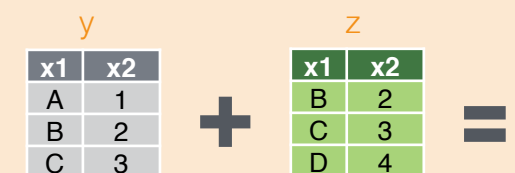
**dplyr::semi\_join(a, b, by = "x1")**

All rows in a that have a match in b.

x1	x2
C	3

**dplyr::anti\_join(a, b, by = "x1")**

All rows in a that do not have a match in b.



Set Operations

x1	x2
B	2
C	3

**dplyr::intersect(y, z)**

Rows that appear in both y and z.

x1	x2
A	1
B	2
C	3
D	4

**dplyr::union(y, z)**

Rows that appear in either or both y and z.

x1	x2
A	1

**dplyr::setdiff(y, z)**

Rows that appear in y but not z.

Binding

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

**dplyr::bind\_rows(y, z)**

Append z to y as new rows.

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

**dplyr::bind\_cols(y, z)**

Append z to y as new columns.

Caution: matches rows by position.