Data Wrangling
with dplyr and tidyr
Cheat Sheet

R Studio®

Tidy Data - A foundation for wrangling in R

In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

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

M * A

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)
```
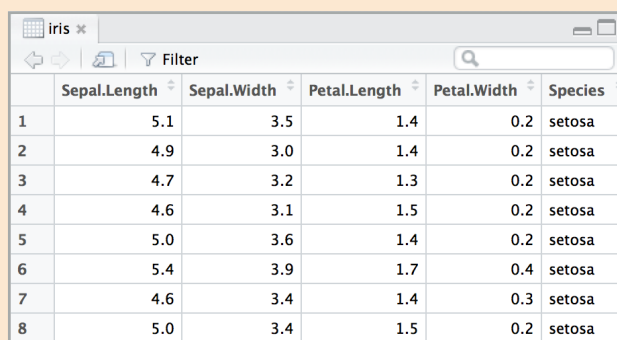
dplyr::glimpse(iris)

Information dense summary of tbl data.

utils::View(iris)

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

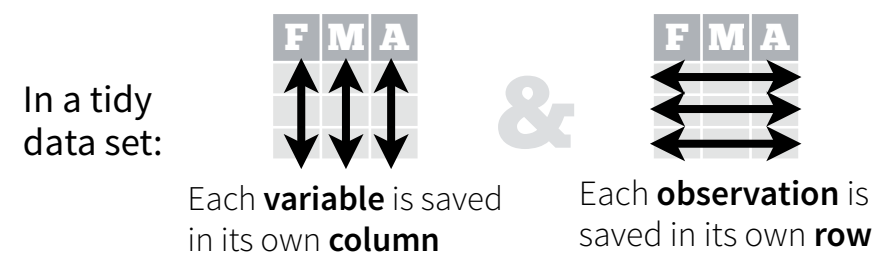|	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

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 )
```

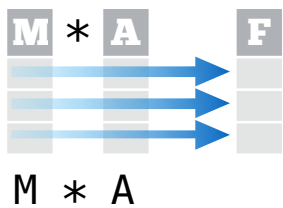"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```
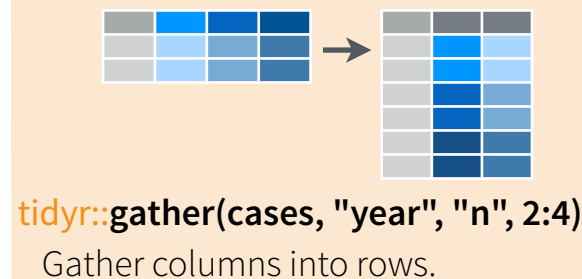
Reshaping Data - Change the layout of a data set

tidyr::gather(cases, "year", "n", 2:4)

Gather columns into rows.

tidyr::separate(storms, date, c("y", "m", "d"))

Separate one column into several.

tidyr::spread(pollution, size, amount)

Spread rows into columns.

tidyr::unite(data, col, ..., sep)

Unite several columns into one.

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).

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

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.

devtools::install_github("rstudio/EDAWR") for data sets   Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

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.	**min** Minimum value in a vector.
dplyr::**last** Last value of a vector.	**max** Maximum value in a vector.
dplyr::**nth** Nth value of a vector.	**mean** Mean value of a vector.
dplyr::**n** # of values in a vector.	**median** Median value of a vector.
dplyr::**n_distinct** # of distinct values in a vector.	**var** Variance of a vector.
IQR IQR 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::**cumall** Cumulative `all`
dplyr::**lag** Copy with values lagged by 1.	dplyr::**cumany** Cumulative `any`
dplyr::**dense_rank** Ranks with no gaps.	dplyr::**cummean** Cumulative `mean`
dplyr::**min_rank** Ranks. Ties get min rank.	**cumsum** Cumulative `sum`
dplyr::**percent_rank** Ranks rescaled to [0, 1].	**cummax** Cumulative `max`
dplyr::**row_number** Ranks. Ties got to first value.	**cummin** Cumulative `min`
dplyr::**ntile** Bin vector into n buckets.	**cumprod** Cumulative `prod`
dplyr::**between** Are values between a and b?	**pmax** Element-wise `max`
dplyr::**cume_dist** Cumulative distribution.	**pmin** Element-wise `min`

iris %>% group_by(Species) %>% mutate(...)
Compute new variables by group.

Combine Data Sets
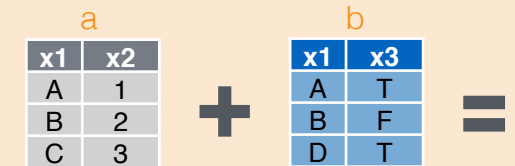
Mutating Joins

dplyr::**left_join**(a, b, by = "x1")
Join matching rows from b to a.

dplyr::**right_join**(a, b, by = "x1")
Join matching rows from a to b.

dplyr::**inner_join**(a, b, by = "x1")
Join data. Retain only rows in both sets.

dplyr::**full_join**(a, b, by = "x1")
Join data. Retain all values, all rows.

Filtering Joins
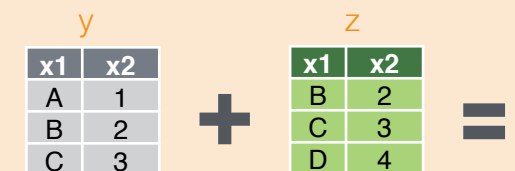

dplyr::**semi_join**(a, b, by = "x1")
All rows in a that have a match in b.

dplyr::**anti_join**(a, b, by = "x1")
All rows in a that do not have a match in b.

Set Operations

dplyr::**intersect**(y, z)
Rows that appear in both y and z.

dplyr::**union**(y, z)
Rows that appear in either or both y and z.

dplyr::**setdiff**(y, z)
Rows that appear in y but not z.

Binding

dplyr::**bind_rows**(y, z)
Append z to y as new rows.

dplyr::**bind_cols**(y, z)
Append z to y as new columns.
Caution: matches rows by position.