**Data Transformation with dplyr:**

**dplyr functions work with pipes and expect tidy data.**

- Each **variable** is in its own column
- Each **observation, or case**, is in its own row

### Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

- **summary function**
  - `summarise(.)` Compute table of summaries.
    ```r
    summarise(iris, avg = mean(mpg))
    ```
  - `count(x, ..., wt = NULL, sort = FALSE)` Count number of rows in each group defined by the variables in x. Also **tally**.
    ```r
    count(iris, Species)
    ```

### Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

- `group_by(.)` Returns copy of table grouped by
  ```r
  g_iris <- group_by(iris, Species)
  ```
- `ungroup(.)` Returns ungrouped copy of table.
  ```r
  ungroup(g_iris)
  ```

### Manipulate Cases

**EXTRACT CASES**

Row functions return a subset of rows as a new table.

- `filter(.)` Extract rows that meet logical criteria.
  ```r
  filter(iris, Sepal.Length > 7)
  ```
- `distinct(.)` Remove rows with duplicate values.
  ```r
  distinct(iris, Species)
  ```
- `sample_frac(.)` Randomly select fraction of rows.
  ```r
  sample_frac(iris, 0.5, replace = TRUE)
  ```
- `sample_n(.)` Randomly select size rows.
  ```r
  sample_n(iris, 10, replace = TRUE)
  ```
- `slice(.)` Select rows by position.
  ```r
  slice(iris, 10:15)
  ```
- `top_n(.)` Select and order top n entries (by group if grouped data).
  ```r
  top_n(iris, 5, Sepal.Width)
  ```

**Logical and boolean operators to use with filter()**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;</code></td>
<td>Less than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater than</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater than or equal</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Not equal</td>
</tr>
<tr>
<td><code>is.na()</code></td>
<td>Check if NA</td>
</tr>
<tr>
<td><code>%&gt;%</code></td>
<td>Pipe to next function</td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>Logical and operator: AND</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td><code>!</code></td>
<td>Logical not operator: NOT</td>
</tr>
<tr>
<td><code>xor()</code></td>
<td>Logical exclusive or operator: EXOR</td>
</tr>
</tbody>
</table>

**VARIATIONS**

- `summarise_all()` - Apply functions to every column.
- `summarise_at()` - Apply functions to specific columns.
- `summarise_if()` - Apply functions to all cols of one type.

### Manipulate Variables

**EXTRACT VARIABLES**

Column functions return a set of columns as a new vector or table.

- `pull(.)` Extract column values as a vector. Choose by name or index.
  ```r
  pull(iris, Sepal.Length)
  ```
- `select(.)` Extract columns as a table. Also `select_if()`.
  ```r
  select(iris, Sepal.Length, Species)
  ```

**MAKE NEW VARIABLES**

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

- `mutate(.)` Compute new column(s).
  ```r
  mutate(iris, Sepal.Length = log(Sepal.Length))
  ```
- `transmute(.)` Compute new column(s), drop others.
  ```r
  transmute(iris, Log(Sepal.Length) = log(Sepal.Length))
  ```
- `add_column(.)` Add new column(s).
  ```r
  add_column(iris, New = c(1, 2, 3))
  ```
- `rename(.)` Rename columns.
  ```r
  rename(iris, Sepal.Length = Log(Sepal.Length))
  ```

**ARRANGE CASES**

Order rows by values of a column or columns (low to high), use with `desc()` to order from high to low.

- `arrange(.)` Order rows by values of a column or columns (low to high), use with `desc()` to order from high to low.
  ```r
  arrange(iris, Sepal.Length)
  ```
- `desc(.)` To order from high to low.
  ```r
  desc(iris, Sepal.Length)
  ```

**ADD CASES**

- `add_row(.)` Add one or more rows to a table.
  ```r
  add_row(iris, Sepal.Length = 10, Petal.Length = 2)
  ```

See ?base::logic and ?Comparison for help.
Vector Functions

TO USE WITH MUTATE()

mutate() and transmute() apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function

OFFSETS
dplyr::lag() - Offset elements by 1
dplyr::lead() - Offset elements by -1

CUMULATIVE AGGREGATES
dplyr::cumsum() - Cumulative sum(dplyr::cumprod() - Cumulative product
dplyr::cumsum() - Cumulative sum

cummean() - Cumulative mean
cumany() - Cumulative any

cummin() - Cumulative min

cummax() - Cumulative max
cumsum() - Cumulative sum

RANKINGS
dplyr::n() - number of values/rows

dplyr::n_distinct() - # of uniques

dplyr::sum(is.na()) - # of non-NA's

LOCATION

mean() - mean, also mean(is.na())

median() - median

LOGICALS

mean() - Proportion of TRUE's

sum() - # of TRUE's

POSITION/ORDER

last() - last value

mean() - mean, also mean(is.na())

first() - first value

median() - median value

nth() - value in nth location of vector

rank() - rank with ties = min, no

quantile() - nth quantile

min() - minimum value

max() - maximum value

SPREAD

IQR() - Inter-Quartile Range

mad() - median absolute deviation

sd() - standard deviation

var() - variance

MISC

dplyr::case_when() - multi-case if_else()
dplyr::coalesce() - first non-NA values by element

dplyr::if_else() - element-wise if/else

dplyr::na_if() - replace missing values with NA

pmin() - element-wise min

pmax() - element-wise max

dplyr::recode() - Vectorized switch

dplyr::recode_factor() - Vectorized switch for factors

Summary Functions

TO USE WITH SUMMARISE()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function

COUNTS

dplyr::n() - number of values/rows

dplyr::n_distinct() - # of uniques

dplyr::sum(is.na()) - # of non-NA's

LOCATION

mean() - mean, also mean(is.na())

median() - median

LOGICALS

mean() - Proportion of TRUE's

sum() - # of TRUE's

POSITION/ORDER

last() - last value

mean() - mean, also mean(is.na())

first() - first value

median() - median value

nth() - value in nth location of vector

rank() - rank with ties = min, no

quantile() - nth quantile

min() - minimum value

max() - maximum value

SPREAD

IQR() - Inter-Quartile Range

mad() - median absolute deviation

sd() - standard deviation

var() - variance

Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

rownames_to_column()

Move row names into column

column_to_rownames()

Move col in row names

Also has rownames(), remove_rownames()