**Data Transformation with dplyr**

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

- Each variable is in its own column
- Each observation, or case, is in its own row
- x \%\% f(y) becomes \( f(x, y) \)

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

**VARIATIONS**

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

### 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.

- mtcars %>%
  - group_by(cyl) %>%
  - summarise(avg = mean(mpg))

- group_by(.data, ..., add = FALSE) Returns copy of table grouped by ...
  - g_iris <- group_by(iris, Species)

### Extract Cases

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

- filter(.data, ...) Extract rows that meet logical criteria.
  - filter(iris, Sepal.Length > 7)
- distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
  - distinct(iris, Species)
- sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, env = parent.frame()) Randomly select fraction of rows.
  - sample_frac(iris, 0.5, replace = TRUE)
- sample_n(tbl, size = replace = FALSE, weight = NULL, env = parent.frame()) Randomly select size rows.
  - sample_n(iris, 10, replace = TRUE)
- slice(.data, ...) Select rows by position.
  - slice(iris, 10:15)
- top_n(.data, n, wt) Select and order top n entries (by group if grouped data).
  - top_n(iris, 5, Sepal.Width)

### Manipulate Variables

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

- pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
  - pull(iris, Sepal.Length)
- select(.data, ...) Extract columns as a table. Also select_if().
  - 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(.data, ...) Compute new column(s).
  - mutate(mtcars, gpm = 1/mpg)
  - transmute(.data, ...) Compute new column(s), drop others.
    - transmute(mtcars, gpm = 1/mpg)
- mutate_all(.tbl, .funs, ...) Apply funs to every column. Use with `funs()`. Also mutate_if().
  - mutate_all(faithful, funs(log(.), log2(.)))
  - mutate_if(iris, is.numeric, funs(log(.)))
- transmute_at() Applies funs to specific columns. Use with `funs()`, `vars()` and the helper functions for select().
  - transmute_at(iris, vars(-Species), funs(log(.)))
- add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also add_count(), add_tally().
  - add_column(mtcars, new = 1:32)
- rename(.data, ...) Rename columns.
  - rename(iris, Length = Sepal.Length)
  - rename(iris, Species = Species)

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

- `<`, `<=`, `is.na()`, `%in%`, `!is.na()`, `|`, `!`, `&`
  - 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 inputs 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::cumall() - Cumulative all()
dplyr::cumany() - Cumulative any()
cummax() - Cumulative max()
dplyr::cummean() - Cumulative mean()
cummin() - Cumulative min()
cumprod() - Cumulative prod()
cumsum() - Cumulative sum()

RANKINGS
dplyr::cume_dist() - Proportion of all values <=
dplyr::dense_rank() - rank with ties = min, no gaps
dplyr::min_rank() - rank with ties = min
dplyr::ntile() - bins into n bins
dplyr::percent_rank() - min_rank scaled to [0,1]
dplyr::row_number() - rank with ties = "first"

MATH

+, -, ^, %, %%, %% - arithmetic ops
log(), log2(), log10() - logs
<, <=, >, >=, !=, == - logical comparisions
dplyr::between() - x == left & x <= right

dplyr::mean() - mean, also mean(!is.na())
dplyr::median() - median

dplyr::m() - number of values/rows
dplyr::n_distinct() - # of uniques

LOCATION

mean() - mean, also mean(!is.na())
dplyr::median() - median

LOGICALS

mean() - Proportion of TRUE's
sum() - # of TRUE's

POSITION/ORDER

first() - first value
last() - last value
nth() - value in n-th location of vector

RANK

quantile() - n-th 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 across a set of vectors
dplyr::if_else() - element-wise if () + else()
dplyr::na_if() - replace specific values with NA
pmax() - element-wise max()
pmin() - element-wise min()

RECODES

recod() - Vectorized switch()
recod_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 inputs and return single values as output.

```
summary function
```

COUNTS
dplyr::m() - number of values/rows
dplyr::n_distinct() - # of uniques

LOCATION

mean() - mean, also mean(!is.na())
dplyr::median() - median

RANKINGS

mean() - Proportion of TRUE's
sum() - # of TRUE's

POSITION/ORDER

first() - first value
last() - last value
nth() - value in n-th location of vector

RANK

quantile() - n-th 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()
```

```
column_to_rownames()
```

Also has rownames(), remove_rownames()