Data Transformation with dplyr

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges. The dplyr package allows you to perform data manipulation tasks.

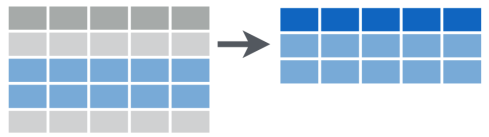
Most data manipulation tasks can be solved using a combination of five functions: filter, arrange, select, mutate, and summarize.

Main dplyr functions:

- filter(): Pick observations by their values
- select(): Pick variables by their names
- arrange(): Reorder the rows
- mutate(): Create new variable with function of existing ones
- summarise(): Collapse many columns into a summary

These all combine naturally with group_by() which allows you to perform any operation "by group", changes the scope of the above functions.

Subset Observations (Rows)



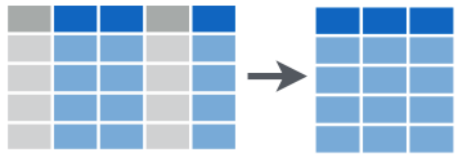
Row functions return a subset of rows as a new table.

filter(.data, ...**)**: Extract rows that meet logical criteria "..." -- Logical predicates defined in terms of the variables in .data. Multiple conditions are combined with &. Only rows where the condition evaluates to TRUE are kept.

distinct(.data, ...**)**: Remove rows with duplicate values.
slice(.data, 1:5**)**: Select rows by position 1:5.
sample_n(.data, n, replace = TRUE**)**: Randomly select rows with number n.
sample_frac(iris, 0.5, replace = TRUE**)**: Randomly select a fraction of rows with 0.5.
slice_head(.data, ...**)** and **slice_tail(**.data, ...**)** : Select the first or last rows.

Logical operators			
<	Less than	>	Greater than
<=	Less than or equal	>=	Greater than or equal
==	Equal to	!=	Not equal to
is.na	Is NA	!is.na	Is not NA

Subset Variables (Columns)



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

select(.data, ...**):** Extract columns as a table
"..." -- One or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, 'select()' will automatically start with all variables.

select(.data, contains(".")**)**: Select columns whose name contains a character string.
select(.data, ends_with("the name for character")**)**: Select columns whose name ends with a character string.
select(.data, matches(".t.")**)**: Select columns whose name matches a regular expression.
select(.data, everything()**)**: Select every column.
select(.data, num_range("x", 1:5)**)**: Select columns named x1, x2, x3, x4, x5.
relocate(.data, …**)**: Move columns to new position.

pull(.data, var = -1, name = NULL, …**)**: Extract column values as a vector, by name or index.

Reshape Data

Change the layout of a data set
arrange(.data, …, .by_group = FALSE**)**: Order rows by values of a column or columns (low to high), use with 'desc()' to order from high to low.
".by_group" -- If 'TRUE', will sort first by grouping variable. Applies to grouped data frames only.

add_row(.data, ...**)**: Add one or more rows to a table.
add_column(.data, ...**)**: Add one or more columns to a table.
rename(.data, ...**)**: Rename columns of a data frame.

Make New Variables

mutate(.data, ...**)**: Compute and append one or more new columns.
"..." -- Name-value pairs of expressions. Use 'NULL' to drop a variable.

transmute(.data, …**)**: Compute new column(s), drop others.
mutate_each(.data, funs(min_rank)**)**: Apply window function to each column. "min_rank" -- Ranks. Ties get min rank.

Summarise Data

Apply summary functions to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value.

summarise(.data, ...**)**: Compute table of summaries.
"..."-- Name-value pairs of summary functions. The name will be the name of the variable in the result. The value should be an expression that returns a single value like 'min(x)', 'n()', or 'sum(is.na(y))'.

count(.data, ...**)**: Count number of rows in each group defined by the variables in ...

Group Data

group_by(.data, ..., add=FALSE**)**: create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.
"..." -- Variables to group by. All tbls accept variable names. Some tbls will accept functions of variables. Duplicated groups will be silently dropped
"add" -- When 'add = FALSE', the default, 'group_by()' will override existing groups. To add to the existing groups, use 'add = TRUE'.

ungroup(.data, ...**)**: Remove grouping information from data frame.

Data visualization:ggplot2

Basic formula

```
 ggplot(data=name of data)+
            <GEOM_FUNCTION>
                    (mapping=aes(<MAPPINGS>),
            Position=<POSITIONS>)+
            <COORDINATE_FUNCTION>+
            <FACET_FUNCTION>+
                        <THEME_FUNCTION>
```

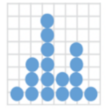
ggplot**(**data = mpg, **aes(**x = cty, y = hwy**))**

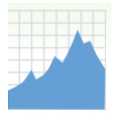
Parameters:

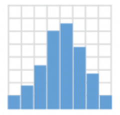
1. Add axes labels : xlab= , ylab=
2. add the title of the chart: main=
3. change color: col=
4. line type(like dashed or dotted): lty
5. line width: lwd
6. plotting figures in an array: mfrow

GEOM_FUCNTIONS(continuous):

7. geom_density(kernel = "gaussian")
8. geom_dotplot()
9. geom_boxplot()
10. geom_violin()
11. geom_bar()
12. geom_area(stat = "bin")
13. geom_freqpoly()
14. geom_histogram()







LABELS：

t + labs(x = "New x axis label", y = "New y axis label",

title ="Add a title above the plot",

subtitle = "Add a subtitle below title",

caption = "Add a caption below plot",

<AES>=”New <AES> legend title”)

THEMES:

1. theme_gray() — grey theme
2. theme_dark()-dark for constant
3. theme_bw() — white backgrounds with grid lines
4. theme_light()
5. theme_minimal() — minimal theme
6. theme_void() — empty theme
7. Theme_linedraw()
8. Theme_classic()

Facet_Function: （Facets divide a plot into subplots based on the values of one or more discrete variables. ）

1. facet_grid(cols = vars(fl))

   facet into columns based on fl

2. facet_grid(rows = vars(year))

   facet into rows based on year

3. facet_grid(rows = vars(year), cols = vars(fl))

   facet into both rows and columns

4. **facet_wrap(vars(fl))**

   wrap facets into a rectangular layout