Data Transformation with dplyr • CHEAT SHEET

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**

### Manipulate Cases
Row functions return a subset of rows as a new table.

- **filter()** Extract rows that meet logical criteria.
- **distinct()** Remove rows with duplicate values.
- **sample_frac()** Randomly select fraction of rows.
- **sample_n()** Randomly select size rows.
- **slice()** Select rows by position.
- **top_n()** Select and order top n entries (by group if grouped data).

### Manipulate 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.
- **select()** Extract columns as a table. Also **select_if()**.

### Vectorized Functions
- **mutate()** Compute new column(s), drop others.
- **transmute()** Compute new column(s), drop others.

### Pipes
f(x, y) becomes x %>% f(y).

---

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>%in%</td>
<td>in a vector</td>
</tr>
<tr>
<td>xor()</td>
<td>xor</td>
</tr>
</tbody>
</table>

**Summary functions**

- **summarise()** Compute table of summaries.
- **summarise_if()** Apply functions to every 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))**

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

- **mtcars %>% arrange(mpg, desc(mpg))**

### Add Cases
Add one or more columns to a table.

- **add_row(.data, ..., before = NULL, after = NULL)** Add one or more rows to a table.
- **rename(.data, ..., .before = NULL, .after = NULL)** Add new column(s). Also **add_column()**, **add_tally()**.

---

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browseVignettes(package = c("dplyr", "tibble")) • dplyr 0.7.0 • tibble 1.2.0 • Updated: 2019-08
**Vector Functions**

Vector functions are used to manipulate and transform data in R. They are particularly useful when working with data frames and vectors.

**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.

**OFFSETS**

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

**CUMULATIVE AGGREGATES**

- **dplyr::cumsum()** - Cumulative sum
- **dplyr::cumprod()** - Cumulative product
- **dplyr::cummax()** - Cumulative max
- **dplyr::cummin()** - Cumulative min
- **dplyr::percent_rank()** - Min rank scaled to [0,1]
- **dplyr::row_number()** - Rank with ties = "first"

**RANKINGS**

- **dplyr::cum_dist()** - Proportion of all values <=
- **dplyr::dense_rank()** - Rank w 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]

**MATH**

- `+`, `-`, `*` - arithmetic ops
- `log()`, `log2()`, `log10()` - logs
- `<`, `<=`, `>`, `>=` - logical comparisons
- `between()` - `x` <= `left` & `x` <= `right`
- `near()` - safe for floating point numbers

**SPREAD**

- **iQR()** - Inter-Quartile Range
- **mad()** - Median absolute deviation
- **sd()** - Standard deviation
- **var()** - Variance

**MISC**

- **dplyr::case_when()** - multi-case `if_else()`
- **dplyr::na_if()** - Replace specific values with NA
- **dplyr::na.omit()** - Remove rows with NA
- **dplyr::recode()** - Vectorized switch
- **dplyr::recode_factor()** - Vectorized switch

---

**Summary Functions**

Summary functions are used to perform simple calculations on columns to create a new table. Summary functions take vectors as input and return single values as output.

**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 unique values
  - **dplyr::n_unique()** - # 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**

- **dplyr::first()** - first value
- **dplyr::last()** - last value
- **dplyr::nth()** - value in nth location of vector

**RANK**

- **quantile()** - nth quantile
- **min()** - minimum value
- **max()** - maximum value

**EXTRACT ROWS**

- **row_names()**
  - **row_names()**
  - **column_to_rownames()**
  - **rownames_to_column()**

**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.

**COMBINE CASES**

- **bind_rows()** to paste tables beside each other as they are.

**COMBINE VARIABLES**

- **bind_cols()** to paste tables beside each other as they are.

---

**Extract Tables**

- **left_join()**
  - Join matching values from x to y
  - Returns tables side by side as a single table. BE SURE THAT ROWS ALIGN.

- **right_join()**
  - Join matching values from x to y

- **inner_join()**
  - Join data. Retain only rows with matches.

- **full_join()**
  - Join data. Retain all values, all rows.

- **anti_join()**
  - Return rows of x that have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.

- **semi_join()**
  - Return rows of x that have a match in y. USEFUL TO SEE WHAT WILL BE JOINED.

- **setdiff()**
  - Rows that appear in x but not y. USEFUL TO SEE WHAT WILL BE JOINED.

- **union()**
  - Rows that appear in x or y. USEFUL TO SEE WHAT WILL BE JOINED.

---

**Extract Tables**

- **by = c("col1", "col2", ...)**
  - Specify one or more common columns to match on.
  - left_join(x, y, by = "A")

- **by = c("col1" = "col2")**
  - To match on columns that have different names in each table.
  - left_join(x, y, by = c("C" = "D"))

- **suffix**
  - To give to unmatched columns that have the same name in both tables.
  - left_join(x, y, by = c("C" = "D", suffix = "1", "2"))

---

**Extract Tables**

- **has_rownames()**, **remove_rownames()**

---

**dplyr** and **tidyverse** tools for data manipulation and analysis.