# dplyr Data Transformation Cheatsheet

A comprehensive cheat sheet for data manipulation using dplyr, covering essential functions for summarizing, grouping, manipulating, extracting, and combining data.

## Summarizing and Grouping Data

### Summarizing Cases

`summarize(.data, ...)`: Computes a table of summary statistics.

**Example:**

```
mtcars |> summarize(avg_mpg = mean(mpg))
```

`.data`: The data frame to summarize.
`...`: Summary expressions that return a single value.

**Common Summary Functions:** `mean()`, `median()`, `sum()`, `min()`, `max()`, `n()`

**Example:**

```
mtcars |> summarize(avg_mpg = mean(mpg),
total_cyl = sum(cyl))
```

### Grouping Cases

`group_by(.data, ...)`: Creates a grouped copy of the table, enabling group-wise operations.

**Example:**

```
mtcars |> group_by(cyl) |> summarize(avg_mpg =
mean(mpg))
```

`.data`: The data frame to group.
`...`: Columns to group by.

`ungroup(x)`: Returns an ungrouped copy of the table.

**Example:**

```
mtcars |> group_by(cyl, gear) |>
summarize(avg_mpg = mean(mpg))
```

### Counting Cases

`count(.data, ..., wt = NULL, sort = FALSE, name = NULL)`: Counts the number of rows in each group.

**Example:**

```
mtcars |> count(cyl)
```

`.data`: The data frame to count.
`...`: Columns to group by.
`wt`: Optional weighting variable.
`sort`: Whether to sort the results.

**Example:**

```
mtcars |> count(cyl, sort = TRUE)
```

**Alternatives:** `tally()`, `add_count()`, and `add_tally()`.

## Manipulating Cases

### Extracting Cases

`filter(.data, ...)`: Extracts rows that meet specified logical criteria.

**Example:**

```
mtcars |> filter(mpg > 20)
```

`.data`: The data frame to filter.
`...`: Logical conditions to apply.

`distinct(.data, ...)`: Removes rows with duplicate values.

**Example:**

```
mtcars |> distinct(gear)
```

`slice(.data, ...)`: Selects rows by their position.

**Example:**

```
mtcars |> slice(10:15)
```

`slice_sample(.data, ..., n, prop, replace = FALSE)`: Randomly selects rows.

**Example:**

```
mtcars |> slice_sample(n = 5, replace = TRUE)
```

### Arranging Cases

`arrange(.data, ...)`: Orders rows by the values of specified columns.

**Example:**

```
mtcars |> arrange(mpg)
```

`.data`: The data frame to arrange.
`...`: Columns to order by.

`desc()`: Use with `arrange()` to order in descending order.

**Example:**

```
mtcars |> arrange(desc(mpg))
```

### Adding Cases

`add_row(.data, ...)`: Adds one or more rows to a table.

**Example:**

```
cars |> add_row(speed = 1, dist = 1)
```

`.data`: The data frame to add rows to.
`...`: Values for the new row(s).

## Manipulating Variables

### Extracting Variables

`pull(.data, var = -1)` : Extracts column values as a vector.

**Example:**

```
mtcars |> pull(wt)
```

`.data` : The data frame to extract from.
`var` : The column to extract (name or index).

`select(.data, ...)` : Extracts columns as a new table.

**Example:**

```
mtcars |> select(mpg, wt)
```

`relocate(.data, ...)` : Moves columns to a new position.

**Example:**

```
mtcars |> relocate(mpg, cyl, after =
last_col())
```

### Making New Variables

`mutate(.data, ...)` : Computes new column(s) using vectorized functions.

**Example:**

```
mtcars |> mutate(gpm = 1 / mpg)
```

`.data` : The data frame to mutate.
`...` : Expressions that compute new columns.

`rename(.data, ...)` : Renames columns.

**Example:**

```
mtcars |> rename(miles_per_gallon = mpg)
```

### Manipulating Multiple Variables at Once

`across(.cols, .fns, ...)` : Applies the same function(s) to multiple columns.

**Example:**

```
df |> summarize(across(everything(), mean))
```

`.cols` : Columns to apply the function to.
`.fns` : Function(s) to apply.

`c_across(.cols)` : Computes across columns in row-wise data.

**Example:**

```
df |> rowwise() |> mutate(x_total =
sum(c_across(1:2)))
```

## Combining Tables

### Combining Variables

`bind_cols(...)` : Returns tables placed side by side.

**Example:**

```
bind_cols(df1, df2)
```

Ensure tables have the same number of rows and are ordered correctly.

### Combining Cases

`bind_rows(...)` : Returns tables stacked on top of each other.

**Example:**

```
bind_rows(df1, df2)
```

Can add a column indicating the source table using `.id`.

**Example:**

```
bind_rows(df1, df2, .id = 'source')
```

### Relational Data (Joins)

`left_join(x, y, by = NULL)` : Join matching values from `y` to `x`.

**Example:**

```
left_join(x, y, by = 'A')
```

`right_join(x, y, by = NULL)` : Join matching values from `x` to `y`.

`inner_join(x, y, by = NULL)` : Retain only rows with matches in both tables.

`full_join(x, y, by = NULL)` : Retain all rows and values from both tables.

`semi_join(x, y, by = NULL)` : Return rows of `x` that have a match in `y`.

`anti_join(x, y, by = NULL)` : Return rows of `x` that do not have a match in `y`.