**Data Wrangling with dplyr and tidyr**

**Cheat Sheet**

**Syntax - Helpful conventions for wrangling**

- **dplyr::tbl_df(iris)**
  Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length Species
1      5.1        3.5        1.4    setosa
2      4.9        3.0        1.4    setosa
3      4.7        3.2        1.3    setosa
4      4.6        3.1        1.5    setosa
5      5.0        3.6        1.4    setosa
... Variables not shown: Petal.Width (dbl); Species (fctr)
```

- **utils::View(iris)**
  View data set in spreadsheet-like display (note capital V).

```
Sepal.Length Sepal.Width Petal.Length
1      5.1        3.5        1.4
2      4.9        3.0        1.4
3      4.7        3.2        1.3
4      4.6        3.1        1.5
5      5.0        3.6        1.4
```

- **dplyr::%>%%
  "Piping" with %>% makes code more readable, e.g.

```
iris %>>% group_by(Species) %>% summarise(avg = mean(Sepal.Length)) %>% arrange(avg)
```

**Tidy Data - A foundation for wrangling in R**

In a tidy data set:
- Each **variable** is saved in its own **column**
- Each **observation** is saved in its own **row**

**Tidy data complements R's vectorized operations. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.**

**Reshaping Data - Change the layout of a data set**

**Syntax**

- **dplyr::data_frame(a = 1:3, b = 4:6)**
  Combine vectors into data frame (optimized).
- **dplyr::arrange(mtcars, mpg)**
  Order rows by values of a column (low to high).
- **dplyr::arrange(mtcars, desc(mpg))**
  Order rows by values of a column (high to low).
- **dplyr::rename(tb, y = year)**
  Rename the columns of a data frame.

**Cheat Sheet**

**Subset Observations (Rows)**

- **dplyr::filter(iris, Sepal.Length > 7)**
  Extract rows that meet logical criteria.
- **dplyr::distinct(iris)**
  Remove duplicate rows.
- **dplyr::sample_frac(iris, 0.5, replace = TRUE)**
  Randomly select fraction of rows.
- **dplyr::sample_n(iris, 10, replace = TRUE)**
  Randomly select n rows.
- **dplyr::slice(iris, 10:15)**
  Select rows by position.
- **dplyr::top_n(storms, 2, date)**
  Select and order top n entries (by group if grouped data).

**Logic in R - Comparison, ?base::Logic**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>in</td>
<td>Is in</td>
</tr>
<tr>
<td>is.na</td>
<td>Is NA</td>
</tr>
<tr>
<td>is.not.na</td>
<td>Is not NA</td>
</tr>
<tr>
<td>,</td>
<td>And</td>
</tr>
<tr>
<td>!</td>
<td>Not</td>
</tr>
<tr>
<td>&amp;</td>
<td>Any</td>
</tr>
<tr>
<td>or</td>
<td>All</td>
</tr>
</tbody>
</table>

**Subset Variables (Columns)**

- **dplyr::select(iris, Sepal.Width, Petal.Length, Species)**
  Select columns by name or helper function.

**Helper functions for select**

- **select(iris, contains("\")**
  Select columns whose name contains a character string.
- **select(iris, ends_with("Length")**
  Select columns whose name ends with a character string.
- **select(iris, everything())**
  Select every column.
- **select(iris, matches("t")**
  Select columns whose name matches a regular expression.
- **select(iris, num_range("x", 1:5))**
  Select columns named x1, x2, x3, x4, x5.
- **select(iris, one_of(c("Species", "Genus"))**
  Select columns whose names are in a group of names.
- **select(iris, starts_with("Sepal")**
  Select columns whose name starts with a character string.
- **select(iris, Sepal.Length:Petal.Width)**
  Select all columns between Sepal.Length and Petal.Width (inclusive).
- **select(iris, -Species)**
  Select all columns except Species.
### Summarise Data

- **`dplyr::summarise(iris, avg = mean(Sepal.Length))`**
  - Summarise data into single row of values.
- **`dplyr::summarise_each(iris, funs(mean))`**
  - Apply summary function to each column.
- **`dplyr::count(iris, Species, wt = Sepal.Length)`**
  - Count number of rows with each unique value of `Species` (with or without weights).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

- **`dplyr::first`**
  - First value of a vector.
- **`dplyr::last`**
  - Last value of a vector.
- **`dplyr::nth`**
  - Nth value of a vector.
- **`dplyr::n`**
  - Number of values in a vector.
- **`dplyr::n_distinct`**
  - Number of distinct values in a vector.
- **`dplyr::IQR`**
  - IQR of a vector.
- **`dplyr::min`**
  - Minimum value in a vector.
- **`dplyr::max`**
  - Maximum value in a vector.
- **`dplyr::mean`**
  - Mean value of a vector.
- **`dplyr::median`**
  - Median value of a vector.
- **`dplyr::var`**
  - Variance of a vector.
- **`dplyr::sd`**
  - Standard deviation of a vector.

### Make New Variables

- **`dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)`**
  - Compute new variables by group.
- **`dplyr::mutate_each(iris, funs(mean))`**
  - Compute one or more new columns.
- **`dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)`**
  - Remove grouping information from data frame.

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

- **`dplyr::lead`**
  - Copy with values shifted by 1.
- **`dplyr::lag`**
  - Copy with values lagged by 1.
- **`dplyr::dense_rank`**
  - Ranks with no gaps.
- **`dplyr::min_rank`**
  - Ranks. Ties get min rank.
- **`dplyr::percent_rank`**
  - Ranks. Ties get to first value.
- **`dplyr::row_number`**
  - Ranks. Ties get to first value.
- **`dplyr::ntile`**
  - Bin vector into n buckets.
- **`dplyr::between`**
  - Are values between a and b?
- **`dplyr::cum_dist`**
  - Cumulative distribution.
- **`dplyr::cumall`**
  - Cumulative all
- **`dplyr::cumany`**
  - Cumulative any
- **`dplyr::cumsum`**
  - Cumulative sum
- **`dplyr::cummax`**
  - Cumulative max
- **`dplyr::cummin`**
  - Cumulative min
- **`dplyr::cumprod`**
  - Element-wise prod
- **`dplyr::pmax`**
  - Element-wise max
- **`dplyr::pmin`**
  - Element-wise min

### Combine Data Sets

- **`dplyr::left_join(a, b, by = "x1")`**
  - Join matching rows from b to a.
- **`dplyr::right_join(a, b, by = "x1")`**
  - Join matching rows from a to b.
- **`dplyr::inner_join(a, b, by = "x1")`**
  - Join data. Retain only rows in both sets.
- **`dplyr::full_join(a, b, by = "x1")`**
  - Join data. Retain all values, all rows.

Filtering Joins

- **`dplyr::semi_join(a, b, by = "x1")`**
  - All rows in a that have a match in b.
- **`dplyr::anti_join(a, b, by = "x1")`**
  - All rows in a that do not have a match in b.

Set Operations

- **`dplyr::intersect(y, z)`**
  - Rows that appear in both y and z.
- **`dplyr::union(y, z)`**
  - Rows that appear in either or both y and z.
- **`dplyr::setdiff(y, z)`**
  - Rows that appear in y but not z.

Binding

- **`dplyr::bind_rows(y, z)`**
  - Append z to y as new rows.
- **`dplyr::bind_cols(y, z)`**
  - Append z to y as new columns. Caution: matches rows by position.