Sorting

Sort DataFrames based on their column features.

Chapter Goals:

- Learn how to sort a DataFrame by its features
- Write code to sort an MLB player's statistics

A. Sorting by feature

When we deal with a dataset that has many features, it is often useful to sort the dataset. This makes it easier to view the data and spot trends in the values.

In pandas, the `sort_values` function allows us to sort a DataFrame by one or more of its columns. The first argument is either a column label or a list of column labels to sort by.

The `ascending` keyword argument allows us to specify whether to sort in ascending or descending order (default is ascending order, i.e. `ascending=True`).

The code below demonstrates how to use `sort_values` with a single column label. The first example sorts by `'yearID'` in ascending order, while the second sorts `'playerID'` in descending lexicographic (alphabetical) order.

```
# df is predefined
print('{}\n'.format(df))

sort1 = df.sort_values('yearID')
print('{}\n'.format(sort1))

sort2 = df.sort_values('playerID', ascending=False)
print('{}\n'.format(sort2))
```

When sorting with a list of column labels, each additional label is used to break ties. Specifically, label *i* in the list acts as a tiebreaker for label *i - 1*.

The code below demonstrates how to sort with a list of column labels.

```
# df is predefined
print('{}\n'.format(df))

sort1 = df.sort_values(['yearID', 'playerID'])
print('{}\n'.format(sort1))

sort2 = df.sort_values(['yearID', 'HR'],
                       ascending=[True, False])
print('{}\n'.format(sort2))
```

When using two column labels to sort, the list's first label represents the main sorting criterion, while the second label is used to break ties. In the example with sorting by `'yearID'` and `'playerID'`, the DataFrame is first sorted by year (in ascending order). For identical years, we sort again by player ID (in ascending order).

For multi-label inputs to `sort_values`, we are allowed to specify different sorting orders for each column label. In our second example, we specified that `'yearID'` would be sorted in ascending order, while `'HR'` would be sorted in descending order.

Time to Code!

The code exercises in this chapter involve sorting a DataFrame of yearly MLB player stats, `yearly_stats_df`.

We'll sort `yearly_stats_df` using two different methods. The first method sorts by `'yearID'` in ascending order.

Set `by_year` equal to `yearly_stats_df.sort_values` with `'yearID'` as the only argument.

```
# CODE HERE
```

The next sorting method will sort by `'HR'` in descending order.

Set `best_hr` equal to `yearly_stats_df.sort_values` with `'HR'` as the first argument and the `ascending` keyword argument set to `False`.

```
# CODE HERE
```

The final sorting method will again sort `yearly_stats_df` by `'HR'` in descending order, but this time we break ties with `'SO'` in ascending order.

Set `best_hr_so` equal to `yearly_stats_df.sort_values` with a list of the specified column labels, in the order provided. The `ascending` keyword argument should be set to `[False, True]`.

```
# CODE HERE
```