Spring 2019 Data 100/200A Midterm 1 Reference Sheet

Pandas and Matplotlib

`df` is a DataFrame; `s` is a Series.

Function	Description
`df[col]`	Returns the column labeled `col` from `df` as a Series.
`df[[col1, col2]]`	Returns a DataFrame containing the columns labeled `col1` and `col2`.
`s.loc[rows] / df.loc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their index values.
`s.iloc[rows] / df.iloc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their positions.
`s.isnull() / df.isnull()`	Returns boolean Series/DataFrame identifying missing values
`s.fillna(value) / df.fillna(value)`	Returns a Series/DataFrame where missing values are replaced by `value`
`df.drop(labels, axis)`	Returns a DataFrame without the rows or columns named `labels` along `axis` (either 0 or 1)
`df.rename(index=None, columns=None)`	Returns a DataFrame with renamed columns from a dictionary `index` and/or `columns`
`df.sort_values(by, ascending=True)`	Returns a DataFrame where rows are sorted by the values in columns `by`
`s.sort_values(ascending=True)`	Returns a sorted Series.
`s.unique()`	Returns a NumPy array of the unique values
`s.value_counts()`	Returns the number of times each unique value appears in a Series
`pd.merge(left, right, how='inner', on='a')`	Returns a DataFrame joining DataFrames `left` and `right` on the column labeled `a`; the join is of type `inner`
`left.merge(right, left_on=col1, right_on=col2)`	Returns a DataFrame joining DataFrames `left` and `right` on columns labeled `col1` and `col2`.
`df.set_index(col)`	Returns a DataFrame that uses the values in the column labeled `col` as the row index.
`df.reset_index(col)`	Returns a DataFrame that has row index 0, 1, etc., and adds the current index as a column.

Groups, Strings, & Plots

`grouped = df.groupby(by)` where `by` can be a column label or a list of labels.

Function	Description
`grouped.count()`	Return a Series containing the size of each group, excluding missing values
`grouped.size()`	Return a Series containing size of each group, including missing values
`grouped.mean()`/`grouped.min()`/`grouped.max()`	Return a Series/DataFrame containing mean/min/max of each group for each column, excluding missing values
`grouped.first()`/`grouped.last()`	Return a Series/DataFrame containing first/last element of each group for each column

`s` is a series of strings.

Function	Description
`s.str.len()`	Returns a Series containing length of each string
`s.str.lower()`/`s.str.upper()`	Returns a Series containing lowercase/uppercase version of each string
`s.str.replace(pat, repl)`	Returns a Series after replacing occurences of substrings matching regular expression `pat` with string `repl`
`s.str.contains(pat)`	Returns a boolean Series indicating whether a substring matching the regular expression `pat` is contained in each string
`s.str.extract(pat)`	Returns a Series of the first subsequence of each string that matches the regular expression `pat`. If `pat` contains one group, then only the substring matching the group is extracted

`x` and `y` are sequences of values.

Function	Description
`plt.plot(x, y)`	Creates a line plot of `x` against `y`
`plt.scatter(x, y)`	Creates a scatter plot of `x` against `y`
`plt.hist(x, bins=None)`	Creates a histogram of `x`; `bins` can be an integer or a sequence
`plt.bar(x, height)`	Creates a bar plot of categories `x` and corresponding heights `height`

.

Regular Expressions

List of all metacharacters: `. ^ $ * + ? ] [ \ | ( ) { }`

Operator	Description
.	Matches any character except **\n**
\	Escapes metacharacters
|	Matches expression on either side of expression; has lowest priority of any operator
\d, **\w**, **\s**	Predefined character group of digits (0-9), alphanumerics (a-z, A-Z, 0-9, and underscore), or whitespace, respectively
\D, **\W**, **\S**	Inverse sets of **\d**, **\w**, **\s**, respectively
*	Matches preceding character/group zero or more times
?	Matches preceding character/group zero or one times
+	Matches preceding character/group one or more times
*?, **+?**	Applies non-greedy matching to * and **+**, respectively
{m}	Matches preceding character/group exactly *m* times
{m, n}	Matches preceding character/group at least *m* times and at most *n* times; if either *m* or *n* are omitted, set lower/upper bounds to 0 and ∞, respectively
^, **$**	Matches the beginning and end of the line, respectively
[ ]	Matching group used to match any of the specified characters or range (e.g. **[abcde]**) **[a-e]**)
( )	Capturing group used to create a sub-expression
[^]	Invert matching group; e.g. **[^a-c]** matches all characters except *a*, *b*, *c*

Function	Description
`re.match(pattern, string)`	Returns a match if zero or more characters at beginning of `string` matches `pattern`, else None
`re.search(pattern, string)`	Returns a match if zero or more characters anywhere in `string` matches `pattern`, else None
`re.findall(pattern, string)`	Returns a list of all non-overlapping matches of `pattern` in `string` (if none, returns empty list)
`re.sub(pattern, repl, string)`	Returns `string` after replacing all occurrences of `pattern` with `repl`

.

SQL

For a table `x` with columns labeled `a` and `g`, here are two example `SELECT` statements:

```
SELECT a, a+1 AS b FROM x WHERE b > 2 ORDER BY -a;
```

```
SELECT g, max(a) FROM x GROUP BY g HAVING min(a) > 1;
```

Syntax	Description
`FROM s INNER JOIN t ON cond`	Inner join of tables `s` and `t` using `cond` to filter rows
`FROM s JOIN t ON cond`	Same as above.
`FROM s LEFT JOIN t ON cond`	Left outer join of tables `s` and `t` using `cond` to filter rows
`FROM s, t`	Cross join of tables `s` and `t`: all pairs of a row from `s` and one from `t`
`FROM (SELECT ...)`	Select rows from a temporary table defined by a select statement
`WHERE a IN (SELECT ...)`	Select rows for which the value in column `a` is among the values in a one-column temporary table defined by a select statement
`ORDER BY RANDOM LIMIT n`	Draw a simple random sample of `n` rows
`CASE WHEN pred THEN cons ELSE alt END`	Evaluates to `cons` if `pred` is true and `alt` otherwise; Multiple `WHEN/THEN` pairs can be included, and `ELSE` is optional