

Spring 2022 Data 100/200 Midterm 1 Reference Sheet

Pandas

Suppose `df` is a DataFrame; `s` is a Series. `pd` is the Pandas package.

Function	Description
<code>df[col]</code>	Returns the column labeled <code>col</code> from <code>df</code> as a Series.
<code>df[[col1, col2]]</code>	Returns a DataFrame containing the columns labeled <code>col1</code> and <code>col2</code> .
<code>s.loc[rows]</code> / <code>df.loc[rows, cols]</code>	Returns a Series/DataFrame with rows (and columns) selected by their index values.
<code>s.iloc[rows]</code> / <code>df.iloc[rows, cols]</code>	Returns a Series/DataFrame with rows (and columns) selected by their positions.
<code>s.isnull()</code> / <code>df.isnull()</code>	Returns boolean Series/DataFrame identifying missing values
<code>s.fillna(value)</code> / <code>df.fillna(value)</code>	Returns a Series/DataFrame where missing values are replaced by <code>value</code>
<code>df.drop(labels, axis)</code>	Returns a DataFrame without the rows or columns named <code>labels</code> along <code>axis</code> (either 0 or 1)
<code>df.rename(index=None, columns=None)</code>	Returns a DataFrame with renamed columns from a dictionary <code>index</code> and/or <code>columns</code>
<code>df.sort_values(by, ascending=True)</code>	Returns a DataFrame where rows are sorted by the values in columns <code>by</code>
<code>s.sort_values(ascending=True)</code>	Returns a sorted Series.
<code>s.unique()</code>	Returns a NumPy array of the unique values
<code>s.value_counts()</code>	Returns the number of times each unique value appears in a Series
<code>pd.merge(left, right, how='inner', on='a')</code>	Returns a DataFrame joining DataFrames <code>left</code> and <code>right</code> on the column labeled <code>a</code> ; the join is of type <code>inner</code>
<code>left.merge(right, left_on=col1, right_on=col2)</code>	Returns a DataFrame joining DataFrames <code>left</code> and <code>right</code> on columns labeled <code>col1</code> and <code>col2</code> .
<code>pd.pivot_table(df, index, columns, values=None, aggfunc='mean')</code>	Returns a DataFrame pivot table where columns are unique values from <code>columns</code> (column name or list), and rows are unique values from <code>index</code> (column name or list); cells are collected <code>values</code> using <code>aggfunc</code> . If <code>values</code> is not provided, cells are collected for each remaining column with multi-level column indexing.
<code>df.set_index(col)</code>	Returns a DataFrame that uses the values in the column labeled <code>col</code> as the row index.
<code>df.reset_index()</code>	Returns a DataFrame that has row index 0, 1, etc., and adds the current index as a column.

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

Function	Description
<code>grouped.count()</code>	Return a Series containing the size of each group, excluding missing values
<code>grouped.size()</code>	Return a Series containing size of each group, including missing values
<code>grouped.mean()/grouped.min()/grouped.max()</code>	Return a Series/DataFrame containing mean/min/max of each group for each column, excluding missing values
<code>grouped.filter(f)</code> <code>grouped.agg(f)</code>	Filters or aggregates using the given function <code>f</code>

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

Visualization

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

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

Seaborn: `x` and `y` are column names in a DataFrame `data`.

Function	Description
<code>sns.countplot(data, x)</code>	Create a barplot of value counts of variable <code>x</code> from <code>data</code>
<code>sns.histplot(data, x, kde=False)</code> <code>sns.displot(x, data, rug = True, kde = True)</code>	Creates a histogram of <code>x</code> from <code>data</code> ; optionally overlay a kernel density estimator. <code>displot</code> is similar but can optionally overlay a rug plot.
<code>sns.boxplot(data, x=None, y)</code> <code>sns.violinplot(data, x=None, y)</code>	Create a boxplot of <code>y</code> , optionally factoring by categorical <code>x</code> , from <code>data</code> . <code>violinplot</code> is similar but also draws a kernel density estimator of <code>y</code> .
<code>sns.scatterplot(data, x, y)</code>	Create a scatterplot of <code>x</code> versus <code>y</code> from <code>data</code>
<code>sns.lmplot(x, y, data, fit_reg=True)</code>	Create a scatterplot of <code>x</code> versus <code>y</code> from <code>data</code> , and by default overlay a least-squares regression line
<code>sns.jointplot(x, y, data, kind)</code>	Combine a bivariate scatterplot of <code>x</code> versus <code>y</code> from <code>data</code> , with univariate density plots of each variable overlaid on the axes; <code>kind</code> determines the visualization type for the distribution plot, can be <code>scatter</code> , <code>kde</code> or <code>hist</code>

Regular Expressions

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

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

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

Modified lecture example for a single capturing group:

```
lines = '169.237.46.168 - - [26/Jan/2014:10:47:58 -0800] "GET ... HTTP/1.1"'
re.findall(r'\\[\\d+\\/(\\w+)\\/(\\d+:\\d+:\\d+:\\d+ .+\\]', line) # returns ['Jan']
```

Modeling

Concept	Formula	Concept	Formula
L_1 loss	$L_1(y, \hat{y}) = y - \hat{y} $	Correlation r	$r = \sum_{i=1}^n \frac{x_i - \bar{x}}{\sigma_x} \frac{y_i - \bar{y}}{\sigma_y}$
L_2 loss	$L_2(y, \hat{y}) = (y - \hat{y})^2$	Linear regression estimate of y	$\hat{y} = \hat{a} + \hat{b}x$
Empirical risk with loss L	$\frac{1}{n} \sum_{i=1}^n L(y_i, \hat{y}_i)$	Slope \hat{b} of regression line	$\hat{b} = r \frac{\sigma_y}{\sigma_x}$
		Intercept \hat{a} of regression line	$\hat{a} = \bar{y} - \hat{b}\bar{x}$