

# Summer 2024 Data C100 Midterm Reference Sheet

## Pandas

Suppose `df` is a DataFrame; `s` is a Series. `import pandas as pd`

Function	Description
<code>df.shape</code>	Returns a tuple containing the number of rows and columns, in that order
<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] / df.loc[rows, cols]</code>	Returns a Series/DataFrame with rows (and columns) selected by their index values.
<code>s.iloc[rows] / df.iloc[rows, cols]</code>	Returns a Series/DataFrame with rows (and columns) selected by their positions.
<code>s.isnull() / df.isnull()</code>	Returns boolean Series/DataFrame identifying missing values
<code>s.fillna(value) / df.fillna(value)</code>	Returns a Series/DataFrame where missing values are replaced by <code>value</code>
<code>s.isin(values) / df.isin(values)</code>	Returns a Series/DataFrame of booleans indicating if each element is in <code>values</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 of <code>s</code> in the order that they appear
<code>s.value_counts()</code>	Returns the number of times each unique value appears in a Series
<code>pd.merge(left, right, how='inner', left_on=col1, right_on=col2)</code>	Returns a DataFrame joining <code>left</code> and <code>right</code> on columns labeled <code>col1</code> and <code>col2</code> ; the join is of type <code>inner</code>
<code>left.merge(right, left_on=col1, right_on=col2)</code>	Returns a DataFrame joining <code>left</code> and <code>right</code> on columns labeled <code>col1</code> and <code>col2</code> .
<code>df.pivot_table(values=None, index=None, columns=None, aggfunc='mean', fill_value=None)</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 DataFrame 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().min().max()</code>	Return a Series/DataFrame containing mean/min/max of each group for each column, excluding missing values
<code>grouped.first().last()</code>	Return a Series/DataFrame containing first/last entry 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[a:b]</code>	Returns a Series where each element is a slice of the corresponding string indexed from <code>a</code> (inclusive, optional) to <code>b</code> (non-inclusive, optional).
<code>s.str.lower()/s.str.upper()</code>	Returns a Series of lowercase/uppercase versions of each string.

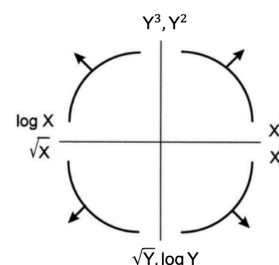
Function	Description
<code>s.str.replace(pat, repl, regex=False)</code>	Returns a Series that replaces occurrences of substrings matching <code>pat</code> with string <code>repl</code> . When <code>regex=False</code> , <code>pat</code> is treated as a literal string; when <code>regex=True</code> , <code>pat</code> is treated as a RegEx pattern.
<code>s.str.contains(pat)</code>	Returns a boolean Series indicating if a substring matching the regex <code>pat</code> is contained in each string.
<code>s.str.extract(pat)</code>	Extracts capture groups in the regex <code>pat</code> from each string. Returns a DataFrame with the extracted substrings, where the columns in the returned DataFrame correspond to the capture groups in <code>pat</code> .
<code>s.str.split(pat=" ")</code>	Splits the strings in <code>s</code> at the delimiter <code>pat</code> (defaults to a whitespace). Returns a Series of lists, where each list contains strings of the characters before and after the split.

## Visualization

Matplotlib: `x` and `y` are sequences of values. `import matplotlib.pyplot as plt`

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>

Tukey-Mosteller Bulge Diagram.



Seaborn: `x` and `y` are column names in a DataFrame `data`. `import seaborn as sns`

Function	Description
<code>sns.countplot(data=None, x=None)</code>	Create a barplot of value counts of variable <code>x</code> from <code>data</code>
<code>sns.histplot(data=None, x=None, stat='count', kde=False)</code> <code>sns.displot(data=None, x=None, kind='hist', rug=False)</code>	Creates a histogram of <code>x</code> from <code>data</code> , where bin statistics <code>stat</code> is one of 'count', 'frequency', 'probability', 'percent', and 'density'; optionally overlay a kernel density estimator. <code>displot</code> is similar but can optionally overlay a rug plot and/or a KDE plot
<code>sns.rugplot(data=None, x=None)</code>	Adds a rug plot on the x-axis of variable <code>x</code> from <code>data</code>
<code>sns.boxplot(data=None, x=None, y=None)</code> <code>sns.violinplot(data=None, x=None, y=None)</code>	Create a boxplot of a numeric feature (e.g., <code>y</code> ), optionally factoring by a category (e.g., <code>x</code> ), from <code>data</code> . <code>violinplot</code> is similar but also draws a kernel density estimator of the numeric feature
<code>sns.scatterplot(data=None, x=None, y=None)</code>	Create a scatterplot of <code>x</code> versus <code>y</code> from <code>data</code>
<code>sns.lmplot(data=None, x=None, y=None, 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(data=None, x=None, y=None, kind='scatter')</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

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 $\infty$ , respectively	<code>^ [ ]</code>	Invert character class; e.g. <code>^[a-c]</code> matches all characters except <code>a, b, c</code>

Modified lecture example for capture groups:

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

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>
<code>re.split(pattern, string)</code>	Splits <code>string</code> on occurrences of <code>pattern</code> , returning a list of substrings

## Modeling

Concept	Formula	Concept	Formula
Variance, $\sigma_x^2$	$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$	Correlation $r$	$r = \frac{1}{n} \sum_{i=1}^n \frac{x_i - \bar{x}}{\sigma_x} \frac{y_i - \bar{y}}{\sigma_y}$
$L_1$ loss	$L_1(y, \hat{y}) =  y - \hat{y} $	Linear regression estimate of $y$	$\hat{y} = \theta_0 + \theta_1 x$
$L_2$ loss	$L_2(y, \hat{y}) = (y - \hat{y})^2$	Least squares linear regression	$\hat{\theta}_0 = \bar{y} - \hat{\theta}_1 \bar{x}$ $\hat{\theta}_1 = r \frac{\sigma_y}{\sigma_x}$
Empirical risk with loss $L$	$R(\theta) = \frac{1}{n} \sum_{i=1}^n L(y_i, \hat{y}_i)$		

## Ordinary Least Squares

Multiple Linear Regression Model:  $\hat{Y} = \mathbb{X}\theta$  with design matrix  $\mathbb{X}$ , response vector  $\mathbb{Y}$ , and predicted vector  $\hat{Y}$ . If there are  $p$  features plus a bias/intercept, then the vector of parameters  $\theta = [\theta_0, \theta_1, \dots, \theta_p]^T \in \mathbb{R}^{p+1}$ . The vector of estimates  $\hat{\theta}$  is obtained from fitting the model to the sample  $(\mathbb{X}, \mathbb{Y})$ .

Concept	Formula	Concept	Formula
Mean squared error	$R(\theta) = \frac{1}{n} \ \mathbb{Y} - \mathbb{X}\theta\ _2^2$	Normal equation	$\mathbb{X}^T \mathbb{X} \hat{\theta} = \mathbb{X}^T \mathbb{Y}$
Least squares estimate, if $\mathbb{X}$ is full rank	$\hat{\theta} = (\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X}^T \mathbb{Y}$	Residual vector, $e$	$e = \mathbb{Y} - \hat{Y}$
		Multiple $R^2$ (coefficient of determination)	$R^2 = \frac{\text{variance of fitted values}}{\text{variance of } y}$
Ridge Regression L2 Regularization	$\frac{1}{n} \ \mathbb{Y} - \mathbb{X}\theta\ _2^2 + \alpha \ \theta\ _2^2$	Squared L2 Norm of $\theta \in \mathbb{R}^d$	$\ \theta\ _2^2 = \sum_{j=1}^d \theta_j^2$
Ridge regression estimate (closed form)	$\hat{\theta}_{\text{ridge}} = (\mathbb{X}^T \mathbb{X} + n\alpha I)^{-1} \mathbb{X}^T \mathbb{Y}$		
LASSO Regression L1 Regularization	$\frac{1}{n} \ \mathbb{Y} - \mathbb{X}\theta\ _2^2 + \alpha \ \theta\ _1$	L1 Norm of $\theta \in \mathbb{R}^d$	$\ \theta\ _1 = \sum_{j=1}^d  \theta_j $

## Gradient Descent

Let  $L(\theta, \mathbb{X}, \mathbb{Y})$  be an objective function to minimize over  $\theta$ , with some optimal  $\hat{\theta}$ . Suppose  $\theta^{(0)}$  is some starting estimate at  $t = 0$ , and  $\theta^{(t)}$  is the estimate at step  $t$ . Then for a learning rate  $\alpha$ , the gradient update step to compute  $\theta^{(t+1)}$  is

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_{\theta} L(\theta^{(t)}, \mathbb{X}, \mathbb{Y})$$

where  $\nabla_{\theta} L(\theta^{(t)}, \mathbb{X}, \mathbb{Y})$  is the partial derivative/gradient of  $L$  with respect to  $\theta$ , evaluated at  $\theta^{(t)}$ .