GraphLab Create™ Translator

The GraphLab Create API is easy to learn and use. See how to convert code syntax from products you already know to GraphLab Create.

Table of Contents

Constructing data objects

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Construct a one-dimensional vector	sa = gl.SArray([1, 2, 3, 4])	s = pd.Series([1, 2, 3, 4])
Construct a vector with missing values	sa = gl.SArray([1, 3, 5, None, 6])	s = pd.Series([1, 3, 5, np.nan, 6])
Construct a two-dimensional table of data	sf = gl.SFrame({'type': ['cat', 'fossa'], 'height': [15., 23.5]})	df = pd.DataFrame({'type': ['cat', 'fossa'], 'height': [15., 23.5]})
Construct an empty graph	sg = gl.SGraph()	
Convert an SFrame to a DataFrame	df = sf.to_dataframe()	
Convert a DataFrame to an SFrame	sf = gl.SFrame(df)	

Assign index name		df.index.name = 'foo'
		df.index.name
		df = df.set_index(['B'])

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)

Rename column name(part of column)
Rename of index value
		df.rename(columns={'aa': 'a', 'bb': 'b'}, inplace=True)
		df1.rename(index={1: 'a'})

Accessing data in a table

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Retrieve a single column from a table	sf['A']	df['A']
Retrieve multiple columns from a table	sf[['A', 'C']]	df[['A', 'C']]
Retrieve a single row from a table return rows if column's value equals with a specific value.	sf[3]	df.iloc[3] df.loc[df['column_name'] == some_value]
Retrieve multiple rows from a table Retrieve(slice) multiple row , column Retrieve(slice) multiple row with all column	sf[3:7]	df[3:7] df.loc[['one','two','three', 'four'],['Fresh', 'Milk', 'Frozen','Detergents']] df1.loc[['a', 'b', 'd'], :]
Retrieve the value from a single cell of a table	sf['A'][3] sf[3][‘A’] ?	df.at[3, 'A'] df[[‘A’]][3] ?

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Retrieve a subset of a table along both axes (elements num are different)	sf[3:7][['A', 'C']] :3,4,5,6(4ele) **sf[[‘A’,’C’]][3:7]**	df.loc[3:6, ['A', 'C']] : 3,4,5,6(4 ele) **df[[‘A’,’C’]][3:7] :3,4,5,6(4 ele)** df[[‘A’,’C’][3:7]] :3,4,5,6(4 ele)
Retrieve rows of a table by filtering a column	sf.filter_by(['b', 'd', 'f'], 'type')	df[df['type'].isin(['b', 'd', 'f'])]
Retrieve table rows using a boolean flag	sf[sf['A'] > 0.5]	df[df.A > 0.5] `df[df['A'] >0.5]`
Set the value of a single table entry		df.at[3, 'A'] = -1

Vector arithmetic

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Add two vectors	sf['A'] + sf['B']	df['A'] + df['B']
Subtract two vectors	sf['A'] - sf['B']	df['A'] - df['B']
Multiply two vectors, element-wise	sf['A'] * sf['B']	df['A'] * df['B']
Divide two vectors, element-wise	sf['A'] / sf['B']	df['A'] / df['B']
Raise a vector to a power, element-wise	sf['A'].apply(lambda x: x**2)	df['A']**2
Test equality of vector elements	sf['C'] == sf['D']	df['C'] == df['D']
Test inequality of vector elements	sf['C'] <= sf['D'] sf['C'] >= sf['D']	df['C'] <= df['D'] df['C'] >= df['D']

Saving and loading data tables

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Read a binary data file	sf = gl.load_sframe("my_sframe")	df = pd.read_pickle("my_dataframe")

Read data from a text file	sf = gl.SFrame.read_csv('my_sframe.csv')	df = pd.read_csv('my_dataframe.csv')
		df=pd.read_csv(‘my.csv’, name=[‘aa’, ‘bb’, ‘cc’])

Save a data table as a text file	sf.save('my_sframe', format='csv')	df.to_csv('my_dataframe.csv', index=False)
Save a data table in binary format	sf.save('my_sframe')	df.to_pickle('my_dataframe')

Data table operations

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Get the first rows of a table	sf.head(5)	df.head(5)
Get the last rows of a table	sf.tail(5)	df.tail(5)
Print a data table in the console	sf.print_rows(30)	pd.set_option('display.max_rows', 30) df
Retrieve column names	sf.column_names()	df.columns df.keys()
Retrieve column types	sf.column_types()	df.dtypes
Retrieve the row index of a table	sf = sf.add_row_number() sf['id']	df.index

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Sort based on a column		df.sort(['c1','c2'], ascending=False)
Add a column to a data table	sf['new'] = range(sf.num_rows())	df['new'] = range(len(df))
Remove a row from a data table		data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'], 'year': [2012, 2012, 2013, 2014,2014], 'reports': [4, 24, 31, 2, 3]} df = pd.DataFrame(data, index = ['Cochice', 'Pima', 'Santa Cruz', 'Maricopa', 'Yuma']) 1)df.drop(['Cochice', 'Pima']) **2) df = df[df.name != 'Tina'] :** Drop a row if it contains a certain value ("Tina")

Remove a column from a data table	sf.remove_column('new')	df = df.drop('new', axis=1)
		df.drop(df.columns[[1, 69]], axis=1, inplace=True)
		#drop column 1,69
		del df['column_name']

Concatenate columns of two tables	sf2 = sf[['A', 'B']]
	sf2.add_columns(sf[['C']])
		blocks = [df[['A', 'B']], df[['C']]]
		df2 = pd.concat(blocks, axis=1)

		Rename Column Names df.columns = ['Leader', 'Time', 'Score'] df.rename(columns={'Leader': 'Commander'}, inplace=True)

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Join two tables on common columns	sf.join(sf2)	pd.merge(df, df2)
Concatenate rows of two tables	sf.append(sf2)	df.append(df2)
Combine multiple columns into a single array or dictionary column	sf.pack_columns(['A', 'B', 'C'], dtype=dict)	
Unpack a single array or dictionary column to multiple columns	sf.unpack('value_dict')	
Stack entries in an array or dictionary column as rows	sf.stack('value_dict', new_column_name=['type', 'value'])	
Stack multiple columns as rows	sf.pack_columns(['A', 'B', 'C'], dtype=dict, new_column_name='value_dict').stack('value_dict')	df.stack()
Flatten rows into columns	sf.unstack(['type', 'value'], new_column_name='value_dict').unpack('value_dict')	df.unstack()

Manipulating data in a table

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Split data into train and test	train_data, test_data = SFrame_1.random_split(0.8, seed=0)	from sklearn.corss_validation import train_test_split X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
Apply a lambda function to a vector	sf['A'].apply(lambda x: x**2)	df['A'].apply(lambda x: x**2)
Apply a lambda function over table rows	sf.apply(lambda x: x['A'] + x['B'])	df.apply(lambda x: x['A'] + x['B'], axis=1) multi columns calculation !! df[‘new_col’]= df.apply(lambda x: x['A'] + x['B'], axis=1)
	*topic_model = gl.load_model('lda_assignment_topic_model') *x['words'] for x in topic_model.get_topics(output_type='topic_words', num_words=10)] *get_topics	
Drop missing values from a table	sf.dropna(columns=['type'])	df.dropna(subset=['type'])
Impute a value for missing table entries	sf.fillna(column='type', value='fossa')	df.fillna(value={'type': 'fossa'}, inplace=True)
Create a boolean mask for missing values in a table	mask = gl.SFrame({c: sf[c] == None for c in sf.column_names()})	mask = pd.isnull(df)

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Swap rows and columns of a table		df.T
Sort a table according to a particular column	sf.sort('A', ascending=False)	df.sort('A', ascending=False)

Convert a vector of strings into a dictionary of word counts	gl.text_analytics.count_words(sf['text'])	from collections import Counter
		import string
		document = [‘this’,’and’,….]
		word_counts = Counter(document)
		# most common 10 words
		for word, count in word_counts.most_common(10):
		    print word, count

Group and aggregate a table based on a set of columns	sf.groupby('type', [gl.aggregate.SUM('A'), gl.aggregate.SUM('B')])	df.groupby('type').sum()[['A', 'B']]
Find the unique elements in a vector	sf['type'].unique()	df['type'].unique()

Computing statistics with data tables

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)

Display statistic info	?	from IPython.display import display
		display(df.describe())

Compute the mean of a column	sf['A'].mean()	df['A'].mean()

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
Compute the mean of each column in a table	[sf[c].mean() for c in sf.column_names()]	df.mean()
Compute the minimum value of a column	sf['A'].min()	df['A'].min()
Compute the maximum value of a column	sf['A'].max()	df['A'].max()
Compute the sum of a column	sf['A'].sum()	df['A'].sum()

Read csv with column name		df = pd.read_csv('../data/example.csv',
		names=['UID', 'First Name',
		'Last Name', 'Age', 'Pre-Test Score', '
		Post-Test Score'])

Compute the sum of a column & add new row of sum		rows_list = []
		for row in input_rows:
		        dict1 = {}
		  # get input row in dictionary format
		  # key = col_name
		        dict1.update(blah..)
		        rows_list.append(dict1)
		df = pd.DataFrame(rows_list)
		-- 	
		dfi = pd.DataFrame(np.arange(6).\\
		      reshape(3,2),columns=['A','B'])
		n [2]: dfi
		Out[2]:
		   A B
		0 0 1
		1 2 3
		2 4 5
		
		In [3]: dfi.loc[:,'C'] = dfi.loc[:,'A']
		#for all row, column ‘C’
		In [4]: dfi

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)
		Out[4]: A B C 0 0 1 0 1 2 3 2 2 4 5 4 In [5]: dfi.loc[3] = 5 In [6]: dfi Out[6]: A B C 0 0 1 0 1 2 3 2 2 4 5 4 3 5 5 5
Compute the variance of a column	sf['A'].var()	df['A'].var()
Compute the standard deviation of a column	sf['A'].std()	df['A'].std()
Compute the number of nonzero elements in a column	sf['A'].nnz()	sum(abs(df['A']) > 1e-8)
Compute the number of missing values in a column	sf['A'].num_missing()	sum(pd.isnull(df['A']))
Show a statistical summary of a data table	sf.show()	df.describe()
Count the frequency of values in a column	sf.groupby('type', gl.aggregate.COUNT)	df['type'].value_counts()