Constructing data objects

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)	R (VER. 3.1.1)
Construct a one-dimensional vector	sa = gl.SArray([1, 2, 3, 4])	s = pd.Series([1, 2, 3, 4])	s = c(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])	s = c(1, 3, 5, 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]})	df = data.frame(type=c('cat', 'fossa'), height=c(15, 23.5))
Construct an empty	sg = gl.SGraph()		
Convert an SFrame to a DataFrame	df = sf.to_dataframe()		
Convert a DataFrame to an SFrame	sf = gl.SFrame(df)		

Accessing data in a table

TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)	R (VER. 3.1.1)
Retrieve a single column from a table	sf['A']	df['A']	df$A
Retrieve multiple columns from a table	sf[['A', 'C']]	df[['A', 'C']]	df[c('A', 'C')]
Retrieve a single row from a table	sf[3]	df.iloc[3]	df[4,]

Retrieve multiple rows from a table	sf[3:7]	df[3:7]	df[4:7,]
Retrieve the value from a single cell of a table	sf['A'][3]	df.at[3, 'A']	df$A[4]
Retrieve a subset of a table along both axes	sf[3:7][['A', 'C']]	df.loc[3:6, ['A', 'C']]	df[4:7, c('A', 'C')]
Retrieve rows of a table by filtering a column	sf.filter_by(['b', 'd', 'f'], 'type')	df[df['type'].isin(['b', 'd', 'f'])]	subset(df, df$type %in% c('b', 'd', 'f'))
Retrieve table rows using a boolean flag	sf[sf['A'] > 0.5]	df[df.A > 0.5]	subset(df, df$A > .5)
Set the value of a single table entry		df.at[3, 'A'] = -1	df$A[4] = -1

Vector arithmetic			
TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)	R (VER. 3.1.1)
Add two vectors	sf['A'] + sf['B']	df['A'] + df['B']	df$A + df$B
Subtract two vectors	sf['A'] - sf['B']	df['A'] - df['B']	df$A - df$B
Multiply two vectors, element-wise	sf['A'] * sf['B']	df['A'] * df['B']	df$A * df$B
Divide two vectors, element-wise	sf['A'] / sf['B']	df['A'] / df['B']	dfA / dfB
Raise a vector to a power, element-wise	sf['A'].apply(lambda x: x**2)	df['A']**2	df$A^2

Test equality of vector elements	sf['C'] == sf['D']	df['C'] == df['D']	df$C == df$D
Test inequality of vector elements	sf['C'] <= sf['D']	df['C'] <= df['D']	df$C <= df$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)	R (VER. 3.1.1)
Read a binary data file	sf = gl.load_sframe("my_sframe")	df = pd.read_pickle("my_dataframe")	load('my_dataframe.rdata')
Read data from a text file	sf = gl.SFrame.read_csv('my_sframe.csv')	df = pd.read_csv('my_dataframe.csv')	df = read.csv('my_dataframe.csv')
Save a data table as a text file	sf.save('my_sframe', format='csv')	df.to_csv('my_dataframe.csv', index=False)	write.csv(df, file='my_dataframe.csv')
Save a data table in binary format	sf.save('my_sframe')	df.to_pickle('my_dataframe')	save(df, file='my_dataframe.rdat

Data table operations			
TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)	R (VER. 3.1.1)
Get the first rows of a	sf.head(5)	df.head(5)	head(df, n=5)
Get the last rows of a	sf.tail(5)	df.tail(5)	tail(df, n=5)
Print a data table in the		pd.set_option('display.max_rows', 30)	

console	sf.print_rows(30)	df	df
Retrieve column names	sf.column_names()	df.columns	colnames(df)
Retrieve column types	sf.column_types()	df.dtypes	lapply(df, class)
Retrieve the row index of a table	sf = sf['id']	df.index	rownames(df)
Add a column to a data table	sf['new'] = range(sf.num_rows())	df['new'] = range(len(df))	df$new = 1:nrow(df)
Remove a column from a data table	sf.remove_column('new')	df = df.drop('new', axis=1)	df[, names(df) != 'new']
Concatenate columns of two tables	sf2 = sf[['A', 'B']] sf2.add_columns(sf[['C']	blocks = [df[['A', 'B']], df[['C']]] df2 = pd.concat(blocks,	df2 = cbind(df[,c('A','B')], 'C'=df$C)
Join two tables on common columns	sf.join(sf2)	pd.merge(df, df2)	merge(df, df2)
Concatenate rows of two tables	sf.append(sf2)	df.append(df2)	rbind(df, df2)
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_dic	df.stack()	
Flatten rows into columns	sf.unstack(['type', 'value'], new_column_name='value_dict').unpack('value_d	df.unstack()	

Manipulating data in a table			
TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)	R (VER. 3.1.1)
Apply a lambda function to a vector	sf['A'].apply(lambda x: x**2)	df['A'].apply(lambda x: x**2)	sapply(df$A, function(x) x^2)
Apply a lambda function over table rows	sf.apply(lambda x: x['A'] + x['B'])	df.apply(lambda x: x['A'] + x['B'], axis=1)	i = j = apply(df, 1, function(x)
Drop missing values from a table	sf.dropna(columns=['type'])	df.dropna(subset=['type'])	na.exclude(df)
Impute a value for missing table entries	sf.fillna(column='type', value='fossa')	df.fillna(value={'type': 'fossa'}, inplace=True)	ix=which(is.na(df$type)) df$type[ix] = 'fossa'

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)	data.frame(lapply(df, is.na))
Swap rows and columns of a table		df.T	t(df)
Sort a table according to a particular column	sf.sort('A', ascending=False)	df.sort('A', ascending=False)	df[order(df$A, decreasing=TRUE),]
Convert a vector of	gl.text_analytics.count_		
Group and aggregate a table based on a set of columns	sf.groupby('type', [gl.aggregate.SUM('A'), gl.aggregate.SUM('B')])	df.groupby('type').sum() [['A', 'B']]	library(plyr) ddply(df, 'type', summarize, sum(A), sum(B))
Find the unique elements in a vector	sf['type'].unique()	df['type'].unique()	unique(df$type)

Computing statistics with data tables			
TASK	GRAPHLAB CREATE (VER. 1.0)	PANDAS (VER. 0.15.0)	R (VER. 3.1.1)
Compute the mean of a column	sf['A'].mean()	df['A'].mean()	mean(df$A)
Compute the mean of each column in a table	[sf[c].mean() for c in sf.column_names()]	df.mean()	lapply(df, mean)
Compute the minimum value of a column	sf['A'].min()	df['A'].min()	min(df$A)
Compute the maximum	sf['A'].max()	df['A'].max()	max(df$A)

Compute the sum of a column	sf['A'].sum()	df['A'].sum()	sum(df$A)
Compute the variance of a column	sf['A'].var()	df['A'].var()	var(df$A)
Compute the standard deviation of a column	sf['A'].std()	df['A'].std()	sd(df$A)
Compute the number of nonzero elements in a column	sf['A'].nnz()	sum(abs(df['A']) > 1e-8)	sum(abs(df$A) > 0)
missing values in a column	sf['A'].num_missing()	sum(pd.isnull(df['A']))	sum(is.na(df$A))
Show a statistical summary of a data table	sf.show()	df.describe()	summary(df)
Count the frequency of values in a column	sf.groupby('type', gl.aggregate.COUNT)	df['type'].value_counts()	table(df$type)