a_data_processing_a

October 15, 2018

0.0.1 Imports

```
In [1]: # Imports
        import pickle
        import numpy as np
        import pandas as pd
        import urllib
        from bs4 import BeautifulSoup
        from pandas_datareader import data as web
```

1 Part One - Data Acquisition and Cleaning

1.1 Web Scraping

Web scraping the list of S&P 500 companies from website https://en.wikipedia.org/wiki/List_of_S%26P_500_companies As of January 20, 2018. NOTE: Saved a local copy 'data/201801201706 - List of S&P 500 companies - Wikipedia.html'

```
In [ ]: # https://stackoverflow.com/questions/42225204/use-pandas-to-get-multiple-tables-from-


        url = 'https://en.wikipedia.org/wiki/List_of_S%26P_500_companies'
        html_table = urllib.request.urlopen(url).read()

        # fix HTML
        soup = BeautifulSoup(html_table, "html.parser")
        # warn! id ratings-table is your page specific
        for table in soup.findChildren(attrs={'id': 'ratings-table'}):
            for c in table.children:
                if c.name in ['tbody', 'thead']:
                    c.unwrap()

        list_df = pd.read_html(str(soup), flavor="bs4")
        #len(list_df[0])
```

1.1.1 Debug

In [ ]:
```
# List of dataframes.
# The first Dataframe in the list is the conversion of the HTML table
# to a Pandas Dataframe. The first one is the one we care about. It is the
# S&P 500 Component Stocks.
type(list_df[0])
```

In [ ]:
```
# Randomly sample axis, in this case None looks like the rows.
list_df[0].sample(10,
                  random_state = None)
```

1.1.2 Pass Pandas Dataframe we care about to a more intuitive variable name.

In [ ]:
```
df_sp500_component_stocks_raw_data = list_df[0]
```

1.1.3 Save raw data to .csv

In [ ]:
```
file_name_web_scrap_raw_data = 'data/sp500_component_stocks_raw_data_201801201730.csv'
df_sp500_component_stocks_raw_data.to_csv(file_name_web_scrap_raw_data)
```

1.1.4 Work from loaded raw data instead of web scrapping each time.

The web site can change over time where as we are data locking from this point on and using the saved/cached version. If we need to web scrap again, run the first section. NOTE: The web address may change and/or the format may change.

1.1.5 Load raw data from file

In [ ]:
```
# Using same variable name.
df_sp500_component_stocks_raw_data = pd.read_csv( file_name_web_scrap_raw_data )
```

1.1.6 Debug

In [ ]:
```
df_sp500_component_stocks_raw_data.sample(10, random_state = None)
```

1.1.7 Clean the data. Make the first row the head for the columns.

In [ ]:
```
# Make the first row the header column
# NOTE: This does not get rid of the row.
df_sp500_component_stocks_cleaned = df_sp500_component_stocks_raw_data.copy()
df_sp500_component_stocks_cleaned.columns = df_sp500_component_stocks_cleaned.iloc[0]

# Re-index and drop the first row.
df_sp500_component_stocks_cleaned = df_sp500_component_stocks_cleaned.reindex(df_sp500_

# # Keep columns of interest.
# # https://stackoverflow.com/questions/14940743/selecting-excluding-sets-of-columns-i
# columns_to_keep = ['GICS Sector', 'GICS Sub Industry']
# df_sp500_component_stocks_cleaned = df_sp500_component_stocks_cleaned[columns_to_kee
```

```
# This is the main industry, select only the rows for ticker symbols for the main indu
# NOTE: The main industry is System Software (i.e. operating systems companies like Re
#       Not just broad
#
# https://stackoverflow.com/questions/17071871/select-rows-from-a-dataframe-based-on-v
#df_tickers_for_information_technology = df_sp500_component_stocks_cleaned.loc[df_sp50
#
# not plural and ignore case:
# https://stackoverflow.com/questions/32616261/filtering-pandas-dataframe-rows-by-conte
df_tickers_for_software = df_sp500_component_stocks_cleaned.loc[df_sp500_component_sto

# Related industry, semiconductors
# not plural and ignore case:
# https://stackoverflow.com/questions/32616261/filtering-pandas-dataframe-rows-by-conte
df_tickers_for_semiconductors = df_sp500_component_stocks_cleaned.loc[df_sp500_componen


# Reset index.
# NOTE: drop = True means do not make a new index and keep old.
#       inplace = True means update this variable and not return a copy
#                       leaving original intact.
df_tickers_for_software.reset_index(drop = True,
                                    inplace = True)
df_tickers_for_semiconductors.reset_index(drop = True,
                                          inplace = True)

# This is the related industry.
# NOTE: The related industry is semiconductors.
```

1.1.8 Debug

```
In [ ]: #df_sp500_component_stocks_cleaned.sample(10, random_state = None)
        #df_sp500_component_stocks_cleaned.head(10)
        #df_sp500_component_stocks_cleaned.tail(10)
        #df_tickers_for_software
        df_tickers_for_semiconductors
```

1.1.9 Save clean data to .csv

```
In [ ]: df_sp500_component_stocks_cleaned.to_csv( 'data/sp500_component_stocks_cleaned_data_20
        df_tickers_for_software.to_csv( 'data/tickers_main_industry_software_201801201826.csv')
        df_tickers_for_semiconductors.to_csv( 'data/tickers_related_industry_semiconductor_201
```

1.1.10 Load data derived from web data scraping.

```
In [ ]: df_tickers_for_software = pd.read_csv( 'data/tickers_main_industry_software_20180120182
```

```
                                             # Use the first column as the index
                                             index_col = 0)

        df_tickers_for_semiconductors = pd.read_csv( 'data/tickers_related_industry_semiconduct

                                                  # Use the first column as the index
                                                  index_col = 0)
```

1.1.11 Debug

```
In [ ]: df_tickers_for_software
```

```
In [ ]: df_tickers_for_semiconductors
```

2 Part Two - Data Processing

2.0.1 Get ticker symbols for the main industry and related industry

2.0.2 From Rubric

2.0.3 Step 1: List down all stocks in the industry

```
In [ ]: # Get the value from the column, which is a Pandas Series, and convert to a Python Lis
        list_tickers_for_software = df_tickers_for_software['Ticker symbol'].tolist()
        list_tickers_for_semiconductors = df_tickers_for_semiconductors['Ticker symbol'].tolis
```

2.0.4 Debug

```
In [345]: list_tickers_for_software
```

```
Out[345]: ['ADBE',
           'ADP',
           'ADSK',
           'AKAM',
           'ANSS',
           'ATVI',
           'CA',
           'CDNS',
           'CRM',
           'CTXS',
           'EA',
           'EBAY',
           'FB',
           'FIS',
           'FISV',
           'GOOG',
           'GOOGL',
           'INTU',
           'MA',
```

```
             'MSFT',
             'NFLX',
             'NTAP',
             'ORCL',
             'PAYX',
             'RHT',
             'SNPS',
             'SYMC',
             'TSS',
             'V',
             'VRSN',
             'WU']
```

```
In [346]: list_tickers_for_semiconductors
```

```
Out[346]: ['ADI',
             'AMAT',
             'AMD',
             'AVGO',
             'INTC',
             'KLAC',
             'LRCX',
             'MCHP',
             'MU',
             'NVDA',
             'QCOM',
             'QRVO',
             'SWKS',
             'TXN',
             'XLNX']
```

2.0.5 From Rubric

2.0.6 Step 2: Collect last one year stock price data for these stocks.

Main Industry

```
In [ ]: dict_tickers_for_software = {}

        for ticker in list_tickers_for_software:
            dict_tickers_for_software[ticker] = web.get_data_quandl(ticker,
                                                                     start = '1/19/2017',
                                                                     end = '1/19/2018')

        # df_temp_a = []
        # for ticker in list_tickers_for_software:

        #     # We get all data available
        #     df_temp_a = web.get_data_quandl(ticker)
```

```
#     # The data is dated latest first.
#     # We only want a year, so keep only 252 days.
#     dict_tickers_for_software[ticker]  = df_temp_a[:252]
```

In [ ]:
```
dict_tickers_for_semiconductors = {}

for ticker in list_tickers_for_semiconductors:
    dict_tickers_for_semiconductors[ticker] = web.get_data_quandl(ticker,
                                                                  start = '1/19/2017',
                                                                  end = '1/19/2018')

# for ticker in list_tickers_for_semiconductors:

#     # We get all data available
#     df_temp_a = web.get_data_quandl(ticker)

#     # The data is dated latest first.
#     # We only want a year, so keep only 252 days.
#     dict_tickers_for_semiconductors[ticker]  = df_temp_a[:252]
```

2.0.7 Get SPX, the ticker that represents the S&P 500 to compare market returns

In [81]:
```
spx_ticker = web.DataReader( 'SPX',
                            data_source = 'yahoo',
                            start = '1/19/2017',
                            end = '1/19/2018')
```

In [82]:
```
spx_ticker
```

Out[82]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-19	0.006	0.006	0.006	0.006	0.006	0
2017-01-20	0.045	0.045	0.045	0.045	0.045	0
2017-01-23	0.045	0.045	0.045	0.045	0.045	58000
2017-01-24	0.045	0.045	0.045	0.045	0.045	19000
2017-01-25	0.045	0.050	0.045	0.045	0.045	204250
2017-01-26	0.050	0.050	0.045	0.045	0.045	46000
2017-01-27	0.045	0.045	0.045	0.045	0.045	12000
2017-01-30	0.006	0.006	0.006	0.006	0.006	0
2017-01-31	0.040	0.040	0.040	0.040	0.040	8000
2017-02-01	0.045	0.045	0.045	0.045	0.045	28000
2017-02-02	0.045	0.045	0.045	0.045	0.045	20000
2017-02-03	0.045	0.045	0.045	0.045	0.045	0
2017-02-06	0.045	0.045	0.045	0.045	0.045	4000
2017-02-07	0.040	0.040	0.040	0.040	0.040	117735
2017-02-08	0.040	0.040	0.040	0.040	0.040	1348
2017-02-09	0.040	0.040	0.040	0.040	0.040	1000

2017-02-10	0.040	0.040	0.040	0.040	0.040	0
2017-02-13	0.040	0.040	0.040	0.040	0.040	0
2017-02-14	0.040	0.040	0.040	0.040	0.040	1000
2017-02-15	0.050	0.050	0.045	0.045	0.045	16000
2017-02-16	0.050	0.050	0.050	0.050	0.050	6320
2017-02-17	0.045	0.045	0.045	0.045	0.045	28200
2017-02-21	0.045	0.045	0.045	0.045	0.045	100500
2017-02-22	0.045	0.045	0.045	0.045	0.045	85000
2017-02-23	0.005	0.005	0.005	0.005	0.005	0
2017-02-24	0.045	0.045	0.045	0.045	0.045	46000
2017-02-27	0.005	0.005	0.005	0.005	0.005	0
2017-02-28	0.005	0.006	0.005	0.006	0.006	3234898
2017-03-01	0.045	0.045	0.040	0.040	0.040	78845
2017-03-02	0.008	0.010	0.008	0.009	0.009	3692497
...	...	...	...	...	...	...
2017-12-06	0.035	0.035	0.035	0.035	0.035	27000
2017-12-07	0.035	0.035	0.035	0.035	0.035	40590
2017-12-08	0.035	0.035	0.035	0.035	0.035	29000
2017-12-11	0.035	0.035	0.035	0.035	0.035	237500
2017-12-12	0.035	0.040	0.035	0.035	0.035	95000
2017-12-13	0.040	0.040	0.040	0.040	0.040	20000
2017-12-14	0.035	0.035	0.035	0.035	0.035	33000
2017-12-15	0.040	0.040	0.040	0.040	0.040	59000
2017-12-18	0.040	0.040	0.040	0.040	0.040	9300
2017-12-19	0.035	0.035	0.035	0.035	0.035	40560
2017-12-20	0.040	0.040	0.040	0.040	0.040	100000
2017-12-21	0.040	0.045	0.035	0.045	0.045	56500
2017-12-22	0.045	0.045	0.045	0.045	0.045	0
2017-12-26	0.045	0.045	0.045	0.045	0.045	0
2017-12-27	0.040	0.040	0.035	0.035	0.035	36820
2017-12-28	0.040	0.045	0.040	0.045	0.045	5550
2017-12-29	0.035	0.045	0.035	0.045	0.045	18320
2018-01-02	0.045	0.045	0.040	0.040	0.040	30000
2018-01-03	0.035	0.040	0.035	0.040	0.040	25350
2018-01-04	0.035	0.035	0.035	0.035	0.035	49000
2018-01-05	0.040	0.045	0.035	0.045	0.045	100000
2018-01-08	0.045	0.045	0.040	0.045	0.045	37500
2018-01-09	0.040	0.050	0.040	0.050	0.050	115290
2018-01-10	0.050	0.065	0.050	0.060	0.060	271450
2018-01-11	0.060	0.065	0.055	0.060	0.060	55515
2018-01-12	0.055	0.055	0.055	0.055	0.055	7646
2018-01-16	0.060	0.065	0.055	0.055	0.055	307800
2018-01-17	0.050	0.055	0.050	0.055	0.055	52000
2018-01-18	0.055	0.055	0.050	0.055	0.055	23070
2018-01-19	0.055	0.055	0.055	0.055	0.055	131000

[253 rows x 6 columns]

2.0.8 Debug

```
In [ ]: # #dict_tickers_for_software['MSFT']['2017-01-19']
        # df_temp_a = dict_tickers_for_software['MSFT'].reset_index()
```

```
In [ ]: # #df_temp_a.iloc[:252]
        # # df_temp_b = df_temp_a.set_index( df_temp_a.iloc[:252]['Date'],
        # #                                  drop = True )
        # df_temp_c = df_temp_a.iloc[:252]
        # #df_temp_b = dict_tickers_for_software['MSFT'].iloc['1/19/2017':'1/19/2018']
```

```
In [ ]: # df_temp_a = dict_tickers_for_software['MSFT']
```

```
In [ ]: # df_temp_a.iloc[:252]
```

```
In [ ]: dict_tickers_for_software.keys()
```

```
In [ ]: dict_tickers_for_software['MSFT']dict_tickers_for_semiconductors
```

```
In [ ]: dict_tickers_for_semiconductors.keys()
```

```
In [ ]: dict_tickers_for_semiconductors['INTC']
```

2.0.9 Save Data to Pickle

```
In [ ]: # https://stackoverflow.com/questions/11641493/how-to-cpickle-dump-and-load-separate-d

        filename = 'data/dict_tickers_for_software_201801201933.pickle'
        with open(filename,'wb') as fp:
            pickle.dump(dict_tickers_for_software,fp)

        filename = 'data/dict_tickers_for_semiconductors_201801201933.pickle'
        with open(filename,'wb') as fp:
            pickle.dump(dict_tickers_for_semiconductors,fp)
```

2.0.10 Loading Data from Pickle

```
In [26]: # https://stackoverflow.com/questions/11641493/how-to-cpickle-dump-and-load-separate-

         filename_software = 'data/dict_tickers_for_software_201801201933.pickle'
         filename_semiconductor = 'data/dict_tickers_for_semiconductors_201801201933.pickle'

         with open(filename_software,'rb') as fp:
             dict_tickers_for_software=pickle.load(fp)

         with open(filename_semiconductor,'rb') as fp:
             dict_tickers_for_semiconductors=pickle.load(fp)
```

2.0.11 Debug

```
In [ ]: dict_tickers_for_software.keys()
```

```
In [ ]: dict_tickers_for_software['MSFT']
```

```
In [ ]: dict_tickers_for_semiconductors.keys()
```

```
In [ ]: dict_tickers_for_semiconductors['INTC']
```

2.0.12 Create Pandas Panels to have multiple pages for Dataframes.

```
In [3]: # https://www.tutorialspoint.com/python_pandas/python_pandas_panel.htm

        dp_tickers_for_software = pd.Panel(dict_tickers_for_software)
        dp_tickers_for_semiconductors = pd.Panel(dict_tickers_for_semiconductors)
```

2.0.13 Debug

```
In [4]: dir(dp_tickers_for_software)
```

```
Out[4]: ['ADBE',
         'ADP',
         'ADSK',
         'AKAM',
         'ANSS',
         'ATVI',
         'CA',
         'CDNS',
         'CRM',
         'CTXS',
         'EA',
         'EBAY',
         'FB',
         'FIS',
         'FISV',
         'GOOG',
         'GOOGL',
         'INTU',
         'MA',
         'MSFT',
         'NFLX',
         'NTAP',
         'ORCL',
         'PAYX',
         'RHT',
         'SNPS',
         'SYMC',
         'TSS',
```

```
'V',
'VRSN',
'WU',
'_AXIS_ALIASES',
'_AXIS_IALIASES',
'_AXIS_LEN',
'_AXIS_NAMES',
'_AXIS_NUMBERS',
'_AXIS_ORDERS',
'_AXIS_REVERSED',
'_AXIS_SLICEMAP',
'__abs__',
'__add__',
'__and__',
'__array__',
'__array_wrap__',
'__bool__',
'__bytes__',
'__class__',
'__contains__',
'__copy__',
'__deepcopy__',
'__delattr__',
'__delitem__',
'__dict__',
'__dir__',
'__div__',
'__doc__',
'__eq__',
'__finalize__',
'__floordiv__',
'__format__',
'__ge__',
'__getattr__',
'__getattribute__',
'__getitem__',
'__getstate__',
'__gt__',
'__hash__',
'__iadd__',
'__iand__',
'__ifloordiv__',
'__imod__',
'__imul__',
'__init__',
'__init_subclass__',
'__invert__',
'__ior__',
```

```
'__ipow__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__lt__',
'__mod__',
'__module__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__nonzero__',
'__or__',
'__pow__',
'__radd__',
'__rand__',
'__rdiv__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__unicode__',
'__weakref__',
'__xor__',
'_accessors',
'_add_aggregate_operations',
'_add_numeric_operations',
'_add_series_only_operations',
'_add_series_or_dataframe_operations',
```

```
'_agg_by_level',
'_aggregate',
'_aggregate_multiple_funcs',
'_align_frame',
'_align_series',
'_apply_1d',
'_apply_2d',
'_at',
'_box_item_values',
'_builtin_table',
'_check_inplace_setting',
'_check_is_chained_assignment_possible',
'_check_percentile',
'_check_setitem_copy',
'_clear_item_cache',
'_clip_with_one_bound',
'_clip_with_scalar',
'_combine',
'_combine_const',
'_combine_frame',
'_combine_panel',
'_compare_constructor',
'_consolidate',
'_consolidate_inplace',
'_construct_axes_dict',
'_construct_axes_dict_for_slice',
'_construct_axes_dict_from',
'_construct_axes_from_arguments',
'_construct_return_type',
'_constructor',
'_constructor_expanddim',
'_constructor_sliced',
'_convert',
'_create_indexer',
'_cython_table',
'_deprecations',
'_dir_additions',
'_dir_deletions',
'_drop_axis',
'_expand_axes',
'_extract_axes',
'_extract_axes_for_slice',
'_extract_axis',
'_from_axes',
'_get_axis',
'_get_axis_name',
'_get_axis_number',
'_get_axis_resolvers',
```

```
'_get_block_manager_axis',
'_get_bool_data',
'_get_cacher',
'_get_index_resolvers',
'_get_item_cache',
'_get_join_index',
'_get_numeric_data',
'_get_plane_axes',
'_get_plane_axes_index',
'_get_value',
'_get_values',
'_getitem_multilevel',
'_gotitem',
'_homogenize_dict',
'_iat',
'_iget_item_cache',
'_iloc',
'_indexed_same',
'_info_axis',
'_info_axis_name',
'_info_axis_number',
'_init_arrays',
'_init_data',
'_init_dict',
'_init_matrix',
'_init_mgr',
'_internal_names',
'_internal_names_set',
'_is_builtin_func',
'_is_cached',
'_is_cython_func',
'_is_datelike_mixed_type',
'_is_mixed_type',
'_is_numeric_mixed_type',
'_is_view',
'_ix',
'_ixs',
'_loc',
'_maybe_cache_changed',
'_maybe_update_cacher',
'_metadata',
'_needs_reindex_multi',
'_obj_with_exclusions',
'_prep_ndarray',
'_protect_consolidate',
'_reduce',
'_reindex_axes',
'_reindex_axis',
```

```
'_reindex_multi',
'_reindex_with_indexers',
'_repr_data_resource_',
'_repr_latex_',
'_reset_cache',
'_reset_cacher',
'_selected_obj',
'_selection',
'_selection_list',
'_selection_name',
'_set_as_cached',
'_set_axis',
'_set_axis_name',
'_set_is_copy',
'_set_item',
'_set_value',
'_setup_axes',
'_shallow_copy',
'_slice',
'_stat_axis',
'_stat_axis_name',
'_stat_axis_number',
'_take',
'_to_dict_of_blocks',
'_try_aggregate_string_function',
'_typ',
'_unpickle_panel_compat',
'_update_inplace',
'_validate_dtype',
'_values',
'_where',
'_wrap_result',
'_xs',
'abs',
'add',
'add_prefix',
'add_suffix',
'agg',
'aggregate',
'align',
'all',
'any',
'apply',
'as_matrix',
'asfreq',
'asof',
'astype',
'at',
```

```
'at_time',
'axes',
'between_time',
'bfill',
'bool',
'clip',
'clip_lower',
'clip_upper',
'compound',
'conform',
'copy',
'count',
'cummax',
'cummin',
'cumprod',
'cumsum',
'describe',
'div',
'divide',
'drop',
'dropna',
'dtypes',
'empty',
'eq',
'equals',
'ffill',
'fillna',
'filter',
'first',
'floordiv',
'fromDict',
'from_dict',
'ftypes',
'ge',
'get',
'get_dtype_counts',
'get_ftype_counts',
'get_value',
'get_values',
'groupby',
'gt',
'head',
'iat',
'iloc',
'infer_objects',
'interpolate',
'is_copy',
'isna',
```

```
'isnull',
'items',
'iteritems',
'ix',
'join',
'keys',
'kurt',
'kurtosis',
'last',
'le',
'loc',
'lt',
'mad',
'major_axis',
'major_xs',
'mask',
'max',
'mean',
'median',
'min',
'minor_axis',
'minor_xs',
'mod',
'mul',
'multiply',
'ndim',
'ne',
'notna',
'notnull',
'pct_change',
'pipe',
'pop',
'pow',
'prod',
'product',
'radd',
'rank',
'rdiv',
'reindex',
'reindex_axis',
'reindex_like',
'rename',
'rename_axis',
'replace',
'resample',
'rfloordiv',
'rmod',
'rmul',
```

```
'round',
'rpow',
'rsub',
'rtruediv',
'sample',
'select',
'sem',
'set_axis',
'set_value',
'shape',
'shift',
'size',
'skew',
'slice_shift',
'sort_index',
'sort_values',
'squeeze',
'std',
'sub',
'subtract',
'sum',
'swapaxes',
'swaplevel',
'tail',
'take',
'toLong',
'to_clipboard',
'to_dense',
'to_excel',
'to_frame',
'to_hdf',
'to_json',
'to_latex',
'to_long',
'to_msgpack',
'to_pickle',
'to_sparse',
'to_sql',
'to_xarray',
'transpose',
'truediv',
'truncate',
'tshift',
'tz_convert',
'tz_localize',
'update',
'values',
'var',
```

```
 'where',
 'xs']
```

In [5]:
```
dp_tickers_for_software.items
```

Out[5]:
```
Index(['ADBE', 'ADP', 'ADSK', 'AKAM', 'ANSS', 'ATVI', 'CA', 'CDNS', 'CRM',
       'CTXS', 'EA', 'EBAY', 'FB', 'FIS', 'FISV', 'GOOG', 'GOOGL', 'INTU',
       'MA', 'MSFT', 'NFLX', 'NTAP', 'ORCL', 'PAYX', 'RHT', 'SNPS', 'SYMC',
       'TSS', 'V', 'VRSN', 'WU'],
      dtype='object')
```

2.0.14 Save to Excel

Saving to Excel allows to use each page as a ticker symbol as a stock.

In [6]:
```
dp_tickers_for_software.to_excel('data/tickers_main_industry_software_201801211359.xls
dp_tickers_for_semiconductors.to_excel('data/tickers_related_industry_semiconductor_20
```

In [83]:
```
spx_ticker.to_excel('data/ticker_spx_201801281357.xls')
```

2.0.15 Load from Excel

There does not seem to have a method to load an Excel file with multiple pages back into a Pandas Panel. The workaround is to load the Excel file as and Excel file object, find the sheet names (Excel pages,) and reconstruct a Pandas Panel.

In [2]:
```
# https://stackoverflow.com/questions/26521266/using-pandas-to-pd-read-excel-for-multi

# Reconstruct Pandas Panel for Main industry Software:
xls_tickers_for_software = pd.ExcelFile('data/tickers_main_industry_software_20180121

# .sheet_names is a property, not a method
list_tickers_for_software = xls_tickers_for_software.sheet_names

dict_tickers_for_software = {}

for ticker in list_tickers_for_software:

    dict_tickers_for_software[ticker] = pd.read_excel( xls_tickers_for_software,
                                                       ticker,
                                                       index_col = 'Date')

#-------------------------------

# Reconstruct Pandas Panel for related industry semiconductors:
xls_tickers_for_semiconductors = pd.ExcelFile('data/tickers_related_industry_semicondu

# .sheet_names is a property, not a method
list_tickers_for_semiconductors = xls_tickers_for_semiconductors.sheet_names
```

```
dict_tickers_for_semiconductors = {}

for ticker in list_tickers_for_semiconductors:

    dict_tickers_for_semiconductors[ticker] = pd.read_excel( xls_tickers_for_semicondu
                                                              ticker,
                                                              index_col = 'Date')

#------------------------------

# Create Pandas Panels to have multiple pages for Dataframes.
# https://www.tutorialspoint.com/python_pandas/python_pandas_panel.htm
dp_tickers_for_software = pd.Panel(dict_tickers_for_software)
dp_tickers_for_semiconductors = pd.Panel(dict_tickers_for_semiconductors)
```

In [86]:
```
df_spx = pd.read_excel( 'data/ticker_spx_201801281357.xls',
                        sheet_name = 0,
                        index_col = 'Date')
```

In [87]:
```
df_spx
```

Out[87]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-19	0.006	0.006	0.006	0.006	0.006	0
2017-01-20	0.045	0.045	0.045	0.045	0.045	0
2017-01-23	0.045	0.045	0.045	0.045	0.045	58000
2017-01-24	0.045	0.045	0.045	0.045	0.045	19000
2017-01-25	0.045	0.050	0.045	0.045	0.045	204250
2017-01-26	0.050	0.050	0.045	0.045	0.045	46000
2017-01-27	0.045	0.045	0.045	0.045	0.045	12000
2017-01-30	0.006	0.006	0.006	0.006	0.006	0
2017-01-31	0.040	0.040	0.040	0.040	0.040	8000
2017-02-01	0.045	0.045	0.045	0.045	0.045	28000
2017-02-02	0.045	0.045	0.045	0.045	0.045	20000
2017-02-03	0.045	0.045	0.045	0.045	0.045	0
2017-02-06	0.045	0.045	0.045	0.045	0.045	4000
2017-02-07	0.040	0.040	0.040	0.040	0.040	117735
2017-02-08	0.040	0.040	0.040	0.040	0.040	1348
2017-02-09	0.040	0.040	0.040	0.040	0.040	1000
2017-02-10	0.040	0.040	0.040	0.040	0.040	0
2017-02-13	0.040	0.040	0.040	0.040	0.040	0
2017-02-14	0.040	0.040	0.040	0.040	0.040	1000
2017-02-15	0.050	0.050	0.045	0.045	0.045	16000
2017-02-16	0.050	0.050	0.050	0.050	0.050	6320
2017-02-17	0.045	0.045	0.045	0.045	0.045	28200
2017-02-21	0.045	0.045	0.045	0.045	0.045	100500
2017-02-22	0.045	0.045	0.045	0.045	0.045	85000

```
2017-02-23  0.005  0.005  0.005  0.005      0.005        0
2017-02-24  0.045  0.045  0.045  0.045      0.045    46000
2017-02-27  0.005  0.005  0.005  0.005      0.005        0
2017-02-28  0.005  0.006  0.005  0.006      0.006  3234898
2017-03-01  0.045  0.045  0.040  0.040      0.040    78845
2017-03-02  0.008  0.010  0.008  0.009      0.009  3692497
...           ...    ...    ...    ...        ...      ...
2017-12-06  0.035  0.035  0.035  0.035      0.035    27000
2017-12-07  0.035  0.035  0.035  0.035      0.035    40590
2017-12-08  0.035  0.035  0.035  0.035      0.035    29000
2017-12-11  0.035  0.035  0.035  0.035      0.035   237500
2017-12-12  0.035  0.040  0.035  0.035      0.035    95000
2017-12-13  0.040  0.040  0.040  0.040      0.040    20000
2017-12-14  0.035  0.035  0.035  0.035      0.035    33000
2017-12-15  0.040  0.040  0.040  0.040      0.040    59000
2017-12-18  0.040  0.040  0.040  0.040      0.040     9300
2017-12-19  0.035  0.035  0.035  0.035      0.035    40560
2017-12-20  0.040  0.040  0.040  0.040      0.040   100000
2017-12-21  0.040  0.045  0.035  0.045      0.045    56500
2017-12-22  0.045  0.045  0.045  0.045      0.045        0
2017-12-26  0.045  0.045  0.045  0.045      0.045        0
2017-12-27  0.040  0.040  0.035  0.035      0.035    36820
2017-12-28  0.040  0.045  0.040  0.045      0.045     5550
2017-12-29  0.035  0.045  0.035  0.045      0.045    18320
2018-01-02  0.045  0.045  0.040  0.040      0.040    30000
2018-01-03  0.035  0.040  0.035  0.040      0.040    25350
2018-01-04  0.035  0.035  0.035  0.035      0.035    49000
2018-01-05  0.040  0.045  0.035  0.045      0.045   100000
2018-01-08  0.045  0.045  0.040  0.045      0.045    37500
2018-01-09  0.040  0.050  0.040  0.050      0.050   115290
2018-01-10  0.050  0.065  0.050  0.060      0.060   271450
2018-01-11  0.060  0.065  0.055  0.060      0.060    55515
2018-01-12  0.055  0.055  0.055  0.055      0.055     7646
2018-01-16  0.060  0.065  0.055  0.055      0.055   307800
2018-01-17  0.050  0.055  0.050  0.055      0.055    52000
2018-01-18  0.055  0.055  0.050  0.055      0.055    23070
2018-01-19  0.055  0.055  0.055  0.055      0.055   131000

[253 rows x 6 columns]
```

2.0.16 Debug

```
In [9]: dir(xls_tickers_for_software)
```

```
Out[9]: ['__class__',
         '__delattr__',
         '__dict__',
         '__dir__',
```

```
'__doc__',
'__enter__',
'__eq__',
'__exit__',
'__format__',
'__fspath__',
'__ge__',
'__getattribute__',
'__gt__',
'__hash__',
'__init__',
'__init_subclass__',
'__le__',
'__lt__',
'__module__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__',
'_io',
'_parse_excel',
'_should_parse',
'book',
'close',
'io',
'parse',
'sheet_names']
```

In [10]:
```
xls_tickers_for_software.sheet_names
```

Out[10]:
```
['ADBE',
 'ADP',
 'ADSK',
 'AKAM',
 'ANSS',
 'ATVI',
 'CA',
 'CDNS',
 'CRM',
 'CTXS',
 'EA',
 'EBAY',
```

```
         'FB',
         'FIS',
         'FISV',
         'GOOG',
         'GOOGL',
         'INTU',
         'MA',
         'MSFT',
         'NFLX',
         'NTAP',
         'ORCL',
         'PAYX',
         'RHT',
         'SNPS',
         'SYMC',
         'TSS',
         'V',
         'VRSN',
         'WU']
```

```
In [12]: dict_tickers_for_software.keys()
```

```
Out[12]: dict_keys(['ADBE', 'ADP', 'ADSK', 'AKAM', 'ANSS', 'ATVI', 'CA', 'CDNS', 'CRM', 'CTXS'
```

```
In [13]: dict_tickers_for_semiconductors.keys()
```

```
Out[13]: dict_keys(['ADBE', 'ADP', 'ADSK', 'AKAM', 'ANSS', 'ATVI', 'CA', 'CDNS', 'CRM', 'CTXS'
```

```
In [16]: dp_tickers_for_software.items
```

```
Out[16]: Index(['ADBE', 'ADP', 'ADSK', 'AKAM', 'ANSS', 'ATVI', 'CA', 'CDNS', 'CRM',
                'CTXS', 'EA', 'EBAY', 'FB', 'FIS', 'FISV', 'GOOG', 'GOOGL', 'INTU',
                'MA', 'MSFT', 'NFLX', 'NTAP', 'ORCL', 'PAYX', 'RHT', 'SNPS', 'SYMC',
                'TSS', 'V', 'VRSN', 'WU'],
               dtype='object')
```

```
In [17]: dp_tickers_for_semiconductors.items
```

```
Out[17]: Index(['ADI', 'AMAT', 'AMD', 'AVGO', 'INTC', 'KLAC', 'LRCX', 'MCHP', 'MU',
                'NVDA', 'QCOM', 'QRVO', 'SWKS', 'TXN', 'XLNX'],
               dtype='object')
```

2.0.17 From Rubric

2.0.18 Step 3: Calculate the historical distance measure between all the possible pairs of stocks.

First need to make Pandas DataFrame for both main industry (software) and related industry (semiconductors) with the ticker symbol and closing price.

NOTE: Using Adjusted Closing Price to take into account stock splits and dividends. https://www.investopedia.com/terms/a/adjusted_closing_price.asp

Simple trading strategy Illiquid stocks are removed from the investment universe. Cumulative total return index is then created for each stock (dividends included) and starting price during formation period is set to $1 (price normalization). Pairs are formed over a twelve-month period (formation period) and are then traded in next six-month period (trading period). The matching partner for each stock is found by looking for the security that minimizes the sum of squared deviations between two normalized price series. Top 20 pairs with the smallest historical distance measure are then traded and long-short position is opened when pair prices have diverged by two standard deviations and the position is closed when prices revert back.

https://quantpedia.com/Screener/Details/12

```
In [4]: # https://quantpedia.com/Screener/Details/12
        # https://stackoverflow.com/questions/18062135/combining-two-series-into-a-dataframe-i

        # Going to concat multiple Panda series into a Panda DataFrame
        list_of_series_for_software = []
        dict_of_dataframes_modified_for_software = {}
        for ticker in dp_tickers_for_software.items:

            df_temp_e = pd.DataFrame()

            #----------------------------------------

            # Getting adjusted close. It takes into account stock split and
            # dividend pay outs.
            # https://www.investopedia.com/terms/a/adjusted_closing_price.asp
            #
            # Where price includes reinvested dividends from paper Pair Trading.
            # Section 2.1 - Pairs Formation Page 11.
            # This means we do NOT include dividends as cumulative returns.
            df_temp_e['AdjClose'] = dp_tickers_for_software[ticker]['AdjClose']
            df_temp_e['ExDividend'] = dp_tickers_for_software[ticker]['ExDividend']

            # Forward fill then backward fill data.
            df_temp_e = df_temp_e.ffill().bfill()

            # Adjusted close normalized by dividing by the price of day 1 of the formation per
            df_temp_e['AdjClose_Normalized']  = df_temp_e['AdjClose'] / df_temp_e.iloc[0]['Adj

            # Get the daily returns
            # https://stackoverflow.com/questions/20000726/calculate-daily-returns-with-pandas
            df_temp_e['daily_returns'] = df_temp_e['AdjClose'] - df_temp_e['AdjClose'].shift(1

            # Fill all not a number (NaN) with 0.
            df_temp_e['daily_returns'] = df_temp_e['daily_returns'].fillna(0.0)
```

```
    # Get cumulative returns
    df_temp_e['cumulative_returns'] = df_temp_e['daily_returns'].cumsum()

    # Add dividend pay out.
    df_temp_e['cumulative_returns_with_dividends'] = df_temp_e['cumulative_returns'] +

    # Normalized cumulative returns
    df_temp_e['cumulative_returns_normalized'] = df_temp_e['cumulative_returns'] / df_t

    # Normalized cumulative returns by dividing by the price of day 1 of the formation
    df_temp_e['cumulative_returns_with_dividends_normalized'] = df_temp_e['cumulative_r

    # Forward fill then backward fill data.
    df_temp_e = df_temp_e.ffill().bfill()


    # Gather data for Debugging
    df_temp_f = df_temp_e.copy()
    dict_of_dataframes_modified_for_software[ticker] = df_temp_f

    # Copy only the Pandas Series
    #
    # Where price includes reinvested dividends from paper Pair Trading.
    # Section 2.1 - Pairs Formation Page 11.
    # This means we do NOT include dividends as cumulative returns.
    ds_temp_a = df_temp_f['AdjClose_Normalized']

    # Name the Pandas Series
    ds_temp_a.name = ticker

    # Rename the column in the Pandas Series.
    ds_temp_a.columns = [ticker]

    # Append the Pandas Series to the list.
    list_of_series_for_software.append( ds_temp_a )

# Make Pandas DataFrame from Pandas Series.
df_ticker_closing_for_software = pd.concat(list_of_series_for_software, axis = 1)

#----------------------

# Going to concat multiple Panda series into a Panda DataFrame
list_of_series_for_semiconductors = []
for ticker in dp_tickers_for_semiconductors.items:

    df_temp_e = pd.DataFrame()

    #----------------------------------------
```

```
# Getting adjusted close. It takes into account stock split and
# dividend pay outs.
# https://www.investopedia.com/terms/a/adjusted_closing_price.asp
#
# Where price includes reinvested dividends from paper Pair Trading.
# Section 2.1 - Pairs Formation Page 11.
# This means we do NOT include dividends as cumulative returns.
df_temp_e['AdjClose'] = dp_tickers_for_semiconductors[ticker]['AdjClose']
df_temp_e['ExDividend'] = dp_tickers_for_semiconductors[ticker]['ExDividend']

# Forward fill then backward fill data.
df_temp_e = df_temp_e.ffill().bfill()

# Adjusted close normalized by dividing by the price of day 1 of the formation per
df_temp_e['AdjClose_Normalized']  = df_temp_e['AdjClose'] / df_temp_e.iloc[0]['Adj

# Get the daily returns
# https://stackoverflow.com/questions/20000726/calculate-daily-returns-with-pandas
df_temp_e['daily_returns'] = df_temp_e['AdjClose'] - df_temp_e['AdjClose'].shift(1

# Fill all not a number (NaN) with 0.
df_temp_e['daily_returns'] = df_temp_e['daily_returns'].fillna(0.0)

# Get cumulative returns
df_temp_e['cumulative_returns'] = df_temp_e['daily_returns'].cumsum()

# Normalized cumulative returns
df_temp_e['cumulative_returns_normalized'] = df_temp_e['cumulative_returns'] / df_t

# Add dividend pay out.
df_temp_e['cumulative_returns_with_dividends'] = df_temp_e['cumulative_returns'] +

# Normalized cumulative returns by dividing by the price of day 1 of the formation
df_temp_e['cumulative_returns_with_dividends_normalized'] = df_temp_e['cumulative_r

# Forward fill then backward fill data.
df_temp_e = df_temp_e.ffill().bfill()


# Gather data for Debugging
df_temp_f = df_temp_e.copy()
dict_of_dataframes_modified_for_software[ticker] = df_temp_f

# Copy only the Pandas Series
#
# Where price includes reinvested dividends from paper Pair Trading.
# Section 2.1 - Pairs Formation Page 11.
```

```
        # This means we do NOT include dividends as cumulative returns.
        ds_temp_a = df_temp_f['AdjClose_Normalized']

        # Name the Pandas Series
        ds_temp_a.name = ticker

        # Rename the column in the Pandas Series.
        ds_temp_a.columns = [ticker]

        # Append the Pandas Series to the list.
        list_of_series_for_semiconductors.append( ds_temp_a )

    # Make Pandas DataFrame from Pandas Series.
    df_ticker_closing_for_semiconductors = pd.concat(list_of_series_for_semiconductors, ax
```

2.0.19 Debug

```
In [290]: df_ticker_closing_for_software
```

```
Out[290]:                 ADBE       ADP      ADSK      AKAM      ANSS      ATVI  \
          Date
          2017-01-18  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000
          2017-01-19  1.009192  1.000000  1.000000  0.991092  1.000000  0.994371
          2017-01-20  1.017649  1.002622  1.021130  0.993534  0.997968  0.997185
          2017-01-23  1.020039  0.992522  1.019505  0.976580  1.000214  0.985926
          2017-01-24  1.045317  1.002622  1.025256  0.973276  1.010589  1.000512
          2017-01-25  1.050188  0.997378  1.033383  0.987500  1.015617  1.012282
          2017-01-26  1.037595  0.990968  1.020755  0.973851  1.014547  1.009980
          2017-01-27  1.047799  0.987763  1.022256  0.976580  1.004920  1.013562
          2017-01-30  1.046236  0.990288  1.014754  0.974282  0.999893  1.013562
          2017-01-31  1.042191  0.980771  1.017004  0.985489  0.997540  1.028915
          2017-02-01  1.042008  0.925027  1.016254  0.987069  0.997326  1.035568
          2017-02-02  1.040169  0.935418  1.034759  0.997126  1.003851  1.035568
          2017-02-03  1.058645  0.940759  1.055389  0.999282  1.017970  1.023797
          2017-02-06  1.052119  0.936875  1.035509  1.001580  1.016686  1.028403
          2017-02-07  1.056715  0.935127  1.056514  1.021839  1.018933  1.024821
          2017-02-08  1.067469  0.937943  1.036884  0.913075  1.018612  1.002815
          2017-02-09  1.070319  0.948334  1.048762  0.917385  1.030271  1.016633
          2017-02-10  1.074088  0.949985  1.040635  0.918822  1.036047  1.208547
          2017-02-13  1.081441  0.957366  1.053763  0.913793  1.039362  1.169396
          2017-02-14  1.080798  0.965718  1.057014  0.913075  1.052091  1.150972
          2017-02-15  1.091369  0.967758  1.054764  0.918103  1.063857  1.163767
          2017-02-16  1.093207  0.969991  1.064516  0.922989  1.074233  1.161464
          2017-02-17  1.100009  0.968049  1.080270  0.895690  1.072521  1.159928
          2017-02-21  1.099642  0.963582  1.089647  0.900000  1.086961  1.160184
          2017-02-22  1.098171  0.975236  1.080020  0.905316  1.081185  1.157369
          2017-02-23  1.092288  0.987084  1.089022  0.900862  1.114023  1.153531
          2017-02-24  1.096700  1.000291  1.094649  0.903305  1.117767  1.165558
```

```
2017-02-27  1.091828  0.997378  1.097024  0.903161  1.147182  1.169140
2017-02-28  1.087784  0.996601  1.079020  0.899425  1.141940  1.154811
2017-03-01  1.106260  1.015150  1.111903  0.910632  1.159054  1.206244
...              ...       ...       ...       ...       ...       ...
2017-12-06  1.595000  1.144301  1.336834  0.801293  1.547973  1.563876
2017-12-07  1.605019  1.143906  1.370468  0.810920  1.564980  1.584985
2017-12-08  1.595459  1.145288  1.339835  0.811782  1.563055  1.607124
2017-12-11  1.602261  1.163057  1.335709  0.812500  1.565836  1.626173
2017-12-12  1.585991  1.155456  1.329457  0.819253  1.554498  1.657837
2017-12-13  1.625425  1.155160  1.325581  0.814080  1.548401  1.662985
2017-12-14  1.608604  1.149632  1.328457  0.815517  1.545727  1.674055
2017-12-15  1.631492  1.166710  1.355339  0.829885  1.583378  1.717818
2017-12-18  1.624690  1.171152  1.343836  0.943534  1.578672  1.699283
2017-12-19  1.608328  1.168388  1.317704  0.960057  1.579099  1.670708
2017-12-20  1.601710  1.163452  1.313203  0.957328  1.576104  1.657322
2017-12-21  1.604559  1.158911  1.305701  0.945977  1.574821  1.675342
2017-12-22  1.608604  1.153778  1.298950  0.940374  1.568724  1.660669
2017-12-26  1.603456  1.160589  1.297824  0.939943  1.565836  1.633381
2017-12-27  1.611913  1.157529  1.307577  0.936638  1.568724  1.630549
2017-12-28  1.613659  1.158023  1.313703  0.940086  1.579313  1.632094
2017-12-29  1.610810  1.156838  1.310703  0.934483  1.578672  1.630035
2018-01-02  1.633422  1.144992  1.339335  0.941954  1.588940  1.655520
2018-01-03  1.664124  1.157430  1.367592  0.947414  1.619638  1.681263
2018-01-04  1.684162  1.168486  1.401225  0.942529  1.623917  1.664530
2018-01-05  1.703649  1.167795  1.385846  0.945833  1.630656  1.708550
2018-01-08  1.700892  1.164242  1.393098  0.946552  1.644133  1.715243
2018-01-09  1.716150  1.172336  1.401725  0.969684  1.659429  1.703916
2018-01-10  1.719919  1.161379  1.393723  0.953161  1.625949  1.720392
2018-01-11  1.736557  1.156739  1.416104  0.936925  1.627554  1.782432
2018-01-12  1.792904  1.169474  1.449237  0.941379  1.649588  1.811264
2018-01-16  1.769096  1.178555  1.400100  0.926724  1.640710  1.768531
2018-01-17  1.806140  1.204813  1.412853  0.939655  1.665954  1.811264
2018-01-18  1.803475  1.193461  1.407227  0.935057  1.693657  1.796848
2018-01-19  1.799154  1.198792  1.441485  0.942960  1.727885  1.816670

                  CA      CDNS       CRM      CTXS    ...        NTAP  \
Date                                                  ...
2017-01-18  1.000000  1.000000  1.000000  1.000000    ...    1.000000
2017-01-19  1.000000  1.000000  1.000000  1.000000    ...    1.000000
2017-01-20  1.000614  0.998845  1.006087  1.003886    ...    1.015071
2017-01-23  0.996928  0.988448  1.007807  0.998057    ...    1.009489
2017-01-24  1.012289  1.010012  1.020378  1.018784    ...    1.047167
2017-01-25  0.961905  1.011167  1.038375  1.033574    ...    1.059168
2017-01-26  0.962826  0.999615  1.031494  0.966426    ...    1.049400
2017-01-27  0.956375  1.009626  1.034934  0.974414    ...    1.053865
2017-01-30  0.956375  1.005776  1.041551  0.983807    ...    1.062517
2017-01-31  0.960676  1.002310  1.046712  0.984454    ...    1.069495
2017-02-01  0.950845  0.994224  1.039831  0.961572    ...    1.061959
```

2017-02-02	0.965591	1.113978	1.060341	1.011648	...	1.067262
2017-02-03	0.978802	1.121679	1.061400	1.035269	...	1.068378
2017-02-06	0.978187	1.131305	1.055313	1.029870	...	1.070611
2017-02-07	0.967435	1.131305	1.061400	1.048092	...	1.083729
2017-02-08	0.962826	1.125915	1.067884	1.048632	...	1.087357
2017-02-09	0.970507	1.137851	1.072251	1.058485	...	1.097125
2017-02-10	0.976037	1.137081	1.067090	1.060240	...	1.101033
2017-02-13	0.978495	1.148248	1.070663	1.063344	...	1.104661
2017-02-14	0.980799	1.152099	1.070795	1.065369	...	1.102986
2017-02-15	0.981418	1.159030	1.080720	1.081971	...	1.086520
2017-02-16	0.987612	1.164420	1.069472	1.079001	...	1.132012
2017-02-17	0.993806	1.162880	1.078073	1.086560	...	1.121128
2017-02-21	0.995664	1.185599	1.087469	1.090204	...	1.130896
2017-02-22	0.997832	1.188294	1.086145	1.082106	...	1.135920
2017-02-23	0.998761	1.185214	1.086013	1.078866	...	1.142618
2017-02-24	1.010529	1.204852	1.082175	1.082241	...	1.142897
2017-02-27	1.008671	1.214478	1.079000	1.081971	...	1.151828
2017-02-28	0.999380	1.189834	1.076485	1.065639	...	1.167457
2017-03-01	1.014246	1.205622	1.109038	1.090204	...	1.200391
...	...	...	...	...	...	...
2017-12-06	1.047782	1.651906	1.361916	1.179559	...	1.619451
2017-12-07	1.043691	1.676935	1.377134	1.176589	...	1.628510
2017-12-08	1.053445	1.677320	1.368797	1.182258	...	1.649174
2017-12-11	1.048726	1.675010	1.383353	1.174565	...	1.658232
2017-12-12	1.053760	1.666923	1.371841	1.177264	...	1.636152
2017-12-13	1.049670	1.658067	1.375414	1.176319	...	1.635586
2017-12-14	1.044321	1.663073	1.379383	1.174969	...	1.623980
2017-12-15	1.060682	1.681941	1.395792	1.181448	...	1.621150
2017-12-18	1.061626	1.677320	1.390499	1.184688	...	1.634171
2017-12-19	1.058795	1.685021	1.379648	1.185903	...	1.639266
2017-12-20	1.055648	1.663843	1.368136	1.189682	...	1.614639
2017-12-21	1.054389	1.635734	1.371179	1.189682	...	1.591144
2017-12-22	1.054704	1.627262	1.358079	1.186172	...	1.593692
2017-12-26	1.058795	1.616095	1.356888	1.186847	...	1.581237
2017-12-27	1.054075	1.626877	1.358343	1.193731	...	1.589446
2017-12-28	1.054075	1.628417	1.360196	1.196296	...	1.579538
2017-12-29	1.047152	1.610320	1.352785	1.187792	...	1.565951
2018-01-02	1.056277	1.619176	1.381633	1.197106	...	1.573028
2018-01-03	1.060682	1.653061	1.393278	1.210063	...	1.615488
2018-01-04	1.062885	1.678090	1.411671	1.224236	...	1.636436
2018-01-05	1.070751	1.693878	1.430462	1.234764	...	1.665875
2018-01-08	1.065717	1.729688	1.440519	1.227880	...	1.688521
2018-01-09	1.062256	1.735464	1.444356	1.230850	...	1.689087
2018-01-10	1.053445	1.726608	1.439725	1.225856	...	1.716828
2018-01-11	1.064144	1.727763	1.443695	1.212223	...	1.749664
2018-01-12	1.072639	1.735464	1.458780	1.215327	...	1.776839
2018-01-16	1.070751	1.705044	1.440916	1.214248	...	1.762685
2018-01-17	1.084596	1.727378	1.456133	1.230850	...	1.800051

```
2018-01-18  1.075786  1.730458  1.479026  1.233414    ...     1.791842
2018-01-19  1.072010  1.755872  1.481143  1.251501    ...     1.785897

                ORCL      PAYX       RHT      SNPS      SYMC       TSS  \
Date
2017-01-18  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000
2017-01-19  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000
2017-01-20  1.016832  1.009360  1.003788  0.997702  1.003389  1.000937
2017-01-23  1.011987  1.003448  1.001894  0.998687  1.007530  0.994005
2017-01-24  1.022698  1.015928  1.020427  1.024626  1.029744  0.996441
2017-01-25  1.023973  1.011823  1.033956  1.029059  1.026355  0.968715
2017-01-26  1.023463  1.009031  1.017992  1.021179  1.024473  0.960097
2017-01-27  1.026014  1.004433  1.024080  1.026104  1.029744  0.960659
2017-01-30  1.026014  1.006076  1.021510  1.028074  1.026732  0.960472
2017-01-31  1.022953  0.997472  1.026515  1.032507  1.037274  0.949419
2017-02-01  1.016067  0.968188  1.029085  1.028074  1.025979  0.942675
2017-02-02  1.019638  0.968685  1.051272  1.053686  1.045181  0.955414
2017-02-03  1.031115  0.959089  1.056953  1.065506  1.069654  0.990446
2017-02-06  1.022698  0.953298  1.058171  1.069939  1.064006  0.989697
2017-02-07  1.021933  0.952471  1.066829  1.070432  1.070783  1.001311
2017-02-08  1.020658  0.953795  1.064123  1.063701  1.077184  1.003934
2017-02-09  1.026014  0.967692  1.072917  1.074372  1.091491  1.005807
2017-02-10  1.040296  0.967196  1.072105  1.071909  1.092620  1.003934
2017-02-13  1.047947  0.976792  1.079140  1.083238  1.094127  1.012177
2017-02-14  1.048202  0.980597  1.083469  1.080939  1.079819  1.011802
2017-02-15  1.056108  0.982417  1.096456  1.087835  1.076619  1.016486
2017-02-16  1.060699  0.984237  1.102949  1.160072  1.074731  1.010678
2017-02-17  1.072686  0.978281  1.117424  1.157610  1.087189  1.020420
2017-02-21  1.078041  0.977784  1.132982  1.165326  1.087566  1.031285
2017-02-22  1.084162  0.986057  1.145969  1.169923  1.079639  1.030910
2017-02-23  1.095639  0.999458  1.144481  1.169266  1.078884  1.032784
2017-02-24  1.100995  1.022620  1.147998  1.183057  1.088699  1.033720
2017-02-27  1.089008  1.017988  1.133387  1.186341  1.082659  1.030161
2017-02-28  1.086202  1.016168  1.120265  1.172878  1.078506  1.020607
2017-03-01  1.094619  1.039165  1.126082  1.186012  1.098136  1.035594
...              ...       ...       ...       ...       ...       ...
2017-12-06  1.249800  1.150657  1.649621  1.455426  1.040952  1.409261
2017-12-07  1.252381  1.152675  1.674784  1.483008  1.048561  1.430359
2017-12-08  1.280257  1.160749  1.694940  1.485963  1.064541  1.442604
2017-12-11  1.302713  1.160917  1.712933  1.470366  1.111718  1.443357
2017-12-12  1.300648  1.155703  1.701163  1.437859  1.100304  1.450516
2017-12-13  1.291872  1.157553  1.711174  1.448859  1.089271  1.441097
2017-12-14  1.295486  1.149311  1.706304  1.444426  1.093836  1.443546
2017-12-15  1.246702  1.173532  1.742154  1.442456  1.102587  1.465774
2017-12-18  1.231473  1.181605  1.751082  1.443441  1.090032  1.488380
2017-12-19  1.234313  1.164954  1.743236  1.450829  1.077096  1.500813
2017-12-20  1.236377  1.163944  1.650433  1.429979  1.071389  1.506841
2017-12-21  1.220374  1.154525  1.665720  1.419800  1.083564  1.479903
```

2017-12-22	1.222439	1.148302	1.662879	1.408964	1.077476	1.484424
2017-12-26	1.224246	1.158057	1.631629	1.401248	1.074052	1.490264
2017-12-27	1.222955	1.150488	1.639881	1.407158	1.082422	1.493843
2017-12-28	1.226569	1.149479	1.638663	1.406173	1.083564	1.501378
2017-12-29	1.220374	1.145106	1.624729	1.399442	1.067584	1.489887
2018-01-02	1.203597	1.132155	1.637446	1.412412	1.099924	1.478396
2018-01-03	1.231473	1.148470	1.664773	1.429322	1.099924	1.484047
2018-01-04	1.243605	1.159235	1.677219	1.442949	1.102207	1.496103
2018-01-05	1.251090	1.155703	1.679383	1.460023	1.122752	1.505899
2018-01-08	1.264254	1.148470	1.693858	1.473650	1.125035	1.528505
2018-01-09	1.266319	1.140733	1.679518	1.472500	1.103728	1.520028
2018-01-10	1.259608	1.136023	1.684389	1.466754	1.090412	1.535098
2018-01-11	1.263480	1.124586	1.706710	1.472829	1.094597	1.548096
2018-01-12	1.277934	1.138042	1.703869	1.483664	1.096500	1.544705
2018-01-16	1.279999	1.148134	1.685471	1.465605	1.060165	1.548849
2018-01-17	1.297551	1.171345	1.712662	1.493679	1.047039	1.562413
2018-01-18	1.296519	1.159067	1.697376	1.501724	1.036767	1.558457
2018-01-19	1.305553	1.168822	1.703734	1.520440	1.044756	1.568252

	V	VRSN	WU
Date			
2017-01-18	1.000000	1.000000	1.000000
2017-01-19	1.000000	1.000000	1.000000
2017-01-20	1.001346	1.000498	0.992901
2017-01-23	1.005139	0.997512	0.980596
2017-01-24	1.018353	1.002115	0.966398
2017-01-25	1.026551	1.011570	0.950308
2017-01-26	1.018475	1.014556	0.927118
2017-01-27	1.024960	1.010450	0.927118
2017-01-30	1.024104	1.005723	0.928064
2017-01-31	1.011991	0.997885	0.926645
2017-02-01	1.008687	0.998134	0.924752
2017-02-02	1.006974	1.016422	0.933743
2017-02-03	1.053224	1.021523	0.945102
2017-02-06	1.050165	1.022394	0.938476
2017-02-07	1.049553	1.023389	0.938949
2017-02-08	1.041111	1.027246	0.942262
2017-02-09	1.047106	1.026624	0.964505
2017-02-10	1.051022	1.034337	0.934217
2017-02-13	1.057629	1.031351	0.924278
2017-02-14	1.062645	1.028614	0.931377
2017-02-15	1.073107	1.030729	0.927118
2017-02-16	1.071513	1.019159	0.925225
2017-02-17	1.072126	1.023887	0.938003
2017-02-21	1.077765	1.032346	0.937530
2017-02-22	1.076294	1.022518	0.946048
2017-02-23	1.080952	1.025131	0.942735
2017-02-24	1.084017	1.036452	0.954567

```
2017-02-27  1.078745  1.026872  0.952674
2017-02-28  1.078010  1.026001  0.929484
2017-03-01  1.090881  1.046529  0.942262
...              ...       ...       ...
2017-12-06  1.349816  1.398358  0.943150
2017-12-07  1.370234  1.403583  0.948984
2017-12-08  1.384994  1.405574  0.942664
2017-12-11  1.382165  1.422244  0.949470
2017-12-12  1.395572  1.422866  0.948012
2017-12-13  1.393727  1.415526  0.941692
2017-12-14  1.388930  1.420005  0.940719
2017-12-15  1.400000  1.431077  0.947039
2017-12-18  1.396679  1.444389  0.950929
2017-12-19  1.379336  1.437547  0.952387
2017-12-20  1.379090  1.443269  0.946553
2017-12-21  1.382657  1.420627  0.936344
2017-12-22  1.386101  1.418139  0.927593
2017-12-26  1.389791  1.417144  0.918356
2017-12-27  1.402460  1.422991  0.913981
2017-12-28  1.406519  1.436925  0.927107
2017-12-29  1.402460  1.423737  0.924190
2018-01-02  1.408487  1.361657  0.927593
2018-01-03  1.422510  1.379696  0.927107
2018-01-04  1.427799  1.386788  0.945581
2018-01-05  1.461993  1.405822  1.001489
2018-01-08  1.467897  1.422120  1.044758
2018-01-09  1.465068  1.406071  1.022880
2018-01-10  1.463469  1.392884  1.032117
2018-01-11  1.474047  1.396243  1.029687
2018-01-12  1.477122  1.413536  1.026770
2018-01-16  1.480812  1.401468  1.002462
2018-01-17  1.500369  1.418263  0.983988
2018-01-18  1.514268  1.424981  0.981557
2018-01-19  1.509225  1.424235  0.983988

[253 rows x 31 columns]
```

```
In [291]: df_ticker_closing_for_semiconductors
```

```
Out[291]:             ADI      AMAT       AMD      AVGO      INTC      KLAC  \
Date
2017-01-18  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000
2017-01-19  1.000000  1.000000  1.000000  1.000000  0.994831  1.000000
2017-01-20  1.006074  1.002667  0.997953  1.029748  1.004897  1.005052
2017-01-23  1.004694  1.000000  1.014330  1.029317  1.000272  1.011707
2017-01-24  1.014771  1.008296  1.068577  1.065693  1.023395  1.020333
2017-01-25  1.028990  1.019852  1.059365  1.091183  1.028292  1.043376
2017-01-26  1.024296  1.006815  1.076766  1.100291  1.021763  1.025878
```

2017-01-27	1.061982	1.038222	1.092119	1.108590	1.033188	1.061738
2017-01-30	1.053009	1.017481	1.085977	1.095171	1.017954	1.062847
2017-01-31	1.034511	1.014815	1.061412	1.075124	1.001632	1.048799
2017-02-01	1.051491	1.037926	1.234391	1.097704	0.993471	1.060259
2017-02-02	1.038515	1.035259	1.256909	1.099644	0.997824	1.076525
2017-02-03	1.042518	1.046222	1.252815	1.111015	1.000544	1.079359
2017-02-06	1.042656	1.042370	1.395087	1.113171	0.993695	1.072089
2017-02-07	1.053424	1.053037	1.360287	1.109668	0.995887	1.080222
2017-02-08	1.069575	1.054815	1.387922	1.114949	0.996708	1.070487
2017-02-09	1.056184	1.049185	1.373593	1.106596	0.971503	1.068885
2017-02-10	1.056461	1.046519	1.389969	1.107566	0.968215	1.067652
2017-02-13	1.075649	1.050667	1.380757	1.113979	0.980818	1.081454
2017-02-14	1.074544	1.043852	1.357216	1.106542	0.984380	1.078602
2017-02-15	1.126449	1.051556	1.361310	1.110099	0.987667	1.087158
2017-02-16	1.131557	1.042370	1.327533	1.121901	0.997530	1.097697
2017-02-17	1.138597	1.059852	1.343910	1.133919	0.999448	1.096829
2017-02-21	1.143291	1.086890	1.432958	1.146529	1.000544	1.121503
2017-02-22	1.139012	1.084811	1.461617	1.154128	0.988215	1.119272
2017-02-23	1.132765	1.074411	1.465711	1.134835	0.991229	1.114684
2017-02-24	1.133737	1.078571	1.445241	1.133703	1.000818	1.119519
2017-02-27	1.142065	1.082434	1.555783	1.149386	1.000270	1.118652
2017-02-28	1.137207	1.076194	1.480041	1.136721	0.991777	1.117412
2017-03-01	1.162608	1.094913	1.531218	1.159409	0.984380	1.133779
...	...	...	...	...	...	...
2017-12-06	1.198724	1.522437	1.023541	1.440632	1.222338	1.289115
2017-12-07	1.208549	1.560938	1.027636	1.440632	1.211929	1.312202
2017-12-08	1.200268	1.534972	1.017400	1.418905	1.219525	1.304673
2017-12-11	1.204198	1.540643	1.039918	1.419123	1.228245	1.303042
2017-12-12	1.199707	1.506320	1.013306	1.412299	1.218962	1.293130
2017-12-13	1.195496	1.514975	1.034800	1.430478	1.219243	1.303293
2017-12-14	1.199145	1.531391	1.036847	1.415793	1.216993	1.310194
2017-12-15	1.216689	1.568101	1.054248	1.450950	1.253564	1.325376
2017-12-18	1.236338	1.593768	1.123849	1.443362	1.301389	1.395516
2017-12-19	1.234373	1.580636	1.120778	1.439595	1.323332	1.400535
2017-12-20	1.244197	1.590485	1.123849	1.450186	1.337960	1.410949
2017-12-21	1.244197	1.548701	1.114637	1.427749	1.315455	1.362893
2017-12-22	1.247004	1.553775	1.078813	1.432225	1.313767	1.371676
2017-12-26	1.243917	1.523631	1.070624	1.409023	1.296325	1.350722
2017-12-27	1.250513	1.542434	1.077789	1.414592	1.297169	1.346079
2017-12-28	1.254443	1.543627	1.079836	1.421689	1.300263	1.353231
2017-12-29	1.249531	1.525720	1.052201	1.402472	1.298576	1.318350
2018-01-02	1.267074	1.583322	1.123849	1.457665	1.317424	1.331650
2018-01-03	1.282794	1.611079	1.182190	1.473606	1.273257	1.347209
2018-01-04	1.281390	1.620331	1.240532	1.474097	1.249907	1.349593
2018-01-05	1.286583	1.629583	1.215967	1.482832	1.258628	1.378452
2018-01-08	1.288829	1.668980	1.256909	1.486381	1.258628	1.382216
2018-01-09	1.286162	1.637343	1.209826	1.465799	1.227120	1.370170
2018-01-10	1.264688	1.590784	1.224156	1.434846	1.195612	1.333532

2018-01-11	1.279846	1.587799	1.242620	1.438503	1.221212	1.333030
2018-01-12	1.291495	1.595261	1.230297	1.442816	1.215586	1.346205
2018-01-16	1.306653	1.626599	1.219038	1.437138	1.213617	1.350847
2018-01-17	1.341319	1.711361	1.246673	1.451114	1.248782	1.423999
2018-01-18	1.364758	1.713152	1.276356	1.472896	1.251314	1.438553
2018-01-19	1.349319	1.713152	1.288639	1.454280	1.260879	1.443196

	LRCX	MCHP	MU	NVDA	QCOM	QRVO \
Date						
2017-01-18	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
2017-01-19	1.000000	1.000000	1.000000	1.000000	0.989406	1.000000
2017-01-20	1.013623	1.018877	1.011515	0.989064	0.965454	1.046476
2017-01-23	1.017412	1.020399	1.008291	0.999334	0.842622	1.060710
2017-01-24	1.034284	1.033643	1.052510	1.020635	0.844465	1.071857
2017-01-25	1.054854	1.047648	1.085214	1.025010	0.873637	1.098268
2017-01-26	1.031487	1.029533	1.083372	1.042697	0.829879	1.086263
2017-01-27	1.066763	1.046735	1.104099	1.062857	0.832796	1.091579
2017-01-30	1.046554	1.051302	1.113772	1.046215	0.823123	1.102555
2017-01-31	1.036268	1.025270	1.110548	1.038227	0.820359	1.101183
2017-02-01	1.062252	1.041407	1.140028	1.083587	0.816060	1.107014
2017-02-02	1.057651	1.047039	1.141870	1.097280	0.808537	1.086435
2017-02-03	1.061440	1.056934	1.133118	1.087676	0.813450	1.096896
2017-02-06	1.057019	1.052976	1.121142	1.115538	0.811915	1.103756
2017-02-07	1.055576	1.059826	1.133118	1.132845	0.817903	1.110444
2017-02-08	1.045381	1.123459	1.115154	1.127900	0.812068	1.135311
2017-02-09	1.046554	1.079769	1.126209	1.106695	0.811915	1.136683
2017-02-10	1.046463	1.079312	1.107784	1.080449	0.829111	1.136512
2017-02-13	1.048899	1.075658	1.100875	1.030620	0.843390	1.140971
2017-02-14	1.043215	1.077485	1.064947	1.034424	0.851835	1.139427
2017-02-15	1.048584	1.094535	1.060341	1.036516	0.867342	1.151946
2017-02-16	1.033652	1.097138	1.058498	1.019874	0.873330	1.151603
2017-02-17	1.036629	1.096679	1.075541	1.019684	0.866882	1.151089
2017-02-21	1.062974	1.108919	1.094887	1.056200	0.871334	1.158806
2017-02-22	1.069830	1.117487	1.093966	1.054583	0.876708	1.158463
2017-02-23	1.062162	1.112438	1.081529	0.956799	0.877322	1.152461
2017-02-24	1.071725	1.114274	1.070474	0.966035	0.878551	1.135140
2017-02-27	1.076236	1.118864	1.094427	0.994123	0.879165	1.159664
2017-02-28	1.069470	1.109531	1.079687	0.966225	0.875290	1.133596
2017-03-01	1.079664	1.126666	1.130815	0.978698	0.883504	1.163608
...	...	...	...	...	...	...
2017-12-06	1.682161	1.337631	1.915246	1.805418	1.036756	1.202195
2017-12-07	1.727175	1.354369	1.989866	1.831460	1.040745	1.178529
2017-12-08	1.702522	1.354679	1.990327	1.826690	1.024950	1.164123
2017-12-11	1.694305	1.354369	1.981115	1.856930	1.039947	1.176814
2017-12-12	1.657873	1.330347	1.928144	1.820490	1.034842	1.160864
2017-12-13	1.655499	1.339026	1.936895	1.776036	1.035480	1.154004
2017-12-14	1.662712	1.324613	1.945647	1.778803	1.032289	1.118676
2017-12-15	1.702705	1.344451	1.953017	1.827358	1.033246	1.123821

2017-12-18	1.726354	1.378856	2.013358	1.887838	1.043138	1.149889
2017-12-19	1.699327	1.388465	2.025795	1.870762	1.029098	1.145430
2017-12-20	1.724162	1.398229	2.107324	1.877344	1.030694	1.150232
2017-12-21	1.695035	1.386450	2.046062	1.868664	1.027343	1.157263
2017-12-22	1.704622	1.378081	2.032243	1.862749	1.032768	1.160864
2017-12-26	1.682891	1.373587	1.946108	1.883449	1.025907	1.130166
2017-12-27	1.692570	1.370022	1.956702	1.880874	1.029736	1.153490
2017-12-28	1.694122	1.373277	1.925841	1.883068	1.027183	1.155205
2017-12-29	1.680700	1.361963	1.894058	1.845864	1.021440	1.142171
2018-01-02	1.728271	1.401173	2.011515	1.901670	1.040267	1.181273
2018-01-03	1.757672	1.422250	2.071856	2.026826	1.052073	1.172869
2018-01-04	1.765707	1.425660	2.159374	2.037510	1.053509	1.178529
2018-01-05	1.793282	1.429844	2.109627	2.054776	1.060529	1.166695
2018-01-08	1.806796	1.435424	2.098111	2.117736	1.057338	1.180587
2018-01-09	1.803965	1.433719	1.979272	2.117164	1.041383	1.183330
2018-01-10	1.750002	1.409542	1.994012	2.133762	1.041224	1.161722
2018-01-11	1.727258	1.427520	1.972363	2.136996	1.043936	1.181273
2018-01-12	1.721240	1.448132	1.971902	2.124976	1.043138	1.218316
2018-01-16	1.738406	1.446427	1.976969	2.099707	1.088929	1.188475
2018-01-17	1.872537	1.488117	2.039152	2.143683	1.085260	1.215057
2018-01-18	1.875916	1.507799	2.026255	2.141012	1.085738	1.212485
2018-01-19	1.894908	1.510279	1.969139	2.195100	1.085579	1.173041

	SWKS	TXN	XLNX
Date			
2017-01-18	1.000000	1.000000	1.000000
2017-01-19	1.000000	1.000000	1.000000
2017-01-20	1.130130	1.011776	1.000862
2017-01-23	1.149248	1.025041	0.999655
2017-01-24	1.176013	1.043313	1.016126
2017-01-25	1.170915	1.063617	1.027251
2017-01-26	1.159954	1.072821	0.989479
2017-01-27	1.172699	1.062940	1.008796
2017-01-30	1.185190	1.061169	1.019317
2017-01-31	1.172827	1.029021	1.003794
2017-02-01	1.172827	1.038965	0.989824
2017-02-02	1.164773	1.033788	1.002760
2017-02-03	1.170526	1.042098	1.004139
2017-02-06	1.168352	1.038284	1.010003
2017-02-07	1.173338	1.037739	1.007575
2017-02-08	1.172188	1.034606	1.010003
2017-02-09	1.182159	1.025751	1.004973
2017-02-10	1.178836	1.023844	1.006188
2017-02-13	1.184205	1.028748	1.032205
2017-02-14	1.173594	1.030383	1.019023
2017-02-15	1.177813	1.030792	1.021972
2017-02-16	1.181648	1.038692	1.033940
2017-02-17	1.231762	1.041281	1.041745

```
2017-02-21  1.234575  1.049999  1.045387
2017-02-22  1.236365  1.052042  1.037409
2017-02-23  1.218467  1.050816  1.022318
2017-02-24  1.210668  1.052178  1.023186
2017-02-27  1.222558  1.051497  1.023012
2017-02-28  1.212075  1.043733  1.020237
2017-03-01  1.236748  1.064575  1.035154
...              ...       ...       ...
2017-12-06  1.244139  1.346651  1.199296
2017-12-07  1.246848  1.357200  1.209993
2017-12-08  1.241689  1.360532  1.201926
2017-12-11  1.250717  1.371774  1.199120
2017-12-12  1.244139  1.366222  1.186845
2017-12-13  1.236272  1.372191  1.192983
2017-12-14  1.213445  1.391484  1.193684
2017-12-15  1.220410  1.404948  1.192106
2017-12-18  1.236014  1.433402  1.219462
2017-12-19  1.233951  1.444923  1.224021
2017-12-20  1.256520  1.454639  1.209817
2017-12-21  1.257294  1.444506  1.198945
2017-12-22  1.251877  1.445339  1.191054
2017-12-26  1.227761  1.445617  1.185442
2017-12-27  1.235757  1.450891  1.190878
2017-12-28  1.245042  1.454917  1.201225
2017-12-29  1.224536  1.449642  1.182285
2018-01-02  1.269417  1.465327  1.190352
2018-01-03  1.288762  1.505024  1.214201
2018-01-04  1.299595  1.503081  1.236121
2018-01-05  1.305527  1.514601  1.300304
2018-01-08  1.305785  1.521819  1.308896
2018-01-09  1.298434  1.532229  1.312404
2018-01-10  1.271609  1.522652  1.299427
2018-01-11  1.287988  1.536115  1.303635
2018-01-12  1.305269  1.564570  1.309247
2018-01-16  1.287859  1.567346  1.293640
2018-01-17  1.317908  1.654235  1.330290
2018-01-18  1.319198  1.615787  1.338006
2018-01-19  1.279347  1.621617  1.327660

[253 rows x 15 columns]
```

2.0.20 Minimized Sum of Squared Deviations within the Main Industry Only - Software

In [336]:
```
# https://stackoverflow.com/questions/39203662/euclidean-distance-matrix-using-panda
#pd.DataFrame( , columns = df_ticker_closing_for_semiconductors.columns, index = df_

# https://stackoverflow.com/questions/41337316/pandas-compare-all-dataframe-columns-
df = df_ticker_closing_for_software.copy()
```

```
    # This is a numpy array
    ndarray_a = df.values
    column_names = df.columns

    # Multiplying every column by every other column
    #list_for_matrix_a = []
    dict_pairs = {}
    for i in range(len(ndarray_a.T)):

        column_name_in_focus = column_names[i]
        column_values_in_focus = ndarray_a[:, i]

        # The other columns
        for j in range(len(ndarray_a.T)):

            # Skip comparison to itself
            if j == i:
                continue

            column_name_other = column_names[j]
            column_values_other = ndarray_a[:, j]

    #         value_a = column_values_in_focus.sum() - column_values_other.sum()
    #         sum_squared_deviations = value_a**2

            # These are numpy ndarrays
            ndarray_temp_a = column_values_in_focus - column_values_other
            ndarray_temp_b = np.square(ndarray_temp_a)

            sum_squared_deviations = ndarray_temp_b.sum()

            # Put in a list so we can sort it.
            # That way, we will have unique keys in the dictionary.
            # ('ADI', 'XLNX') is the same as ('XLNX', 'ADI') in this case.
            # Yes, the value for the key will be overriden, but it would be the
            # same value.
            #
            # After convert to a tuple. No real reason I can think of except convention.
            list_pair_key = sorted([column_name_in_focus, column_name_other])
            tuple_pair_key = tuple(list_pair_key)

            dict_pairs[tuple_pair_key] = sum_squared_deviations
```

In [337]:
```
dict_pairs
```

Out[337]:
```
{('ADBE', 'ADP'): 29.376129693592553,
 ('ADBE', 'ADSK'): 4.158609456332218,
```

```
('ADBE', 'AKAM'): 94.87267386981122,
('ADBE', 'ANSS'): 0.8696850922794794,
('ADBE', 'ATVI'): 6.517351640000458,
('ADBE', 'CA'): 36.511020170463276,
('ADBE', 'CDNS'): 1.2265447236409859,
('ADBE', 'CRM'): 7.085918839247967,
('ADBE', 'CTXS'): 23.967198766619457,
('ADBE', 'EA'): 6.204815344227672,
('ADBE', 'EBAY'): 16.506262595260367,
('ADBE', 'FB'): 4.966219754693422,
('ADBE', 'FIS'): 18.847753171034842,
('ADBE', 'FISV'): 19.019434180821282,
('ADBE', 'GOOG'): 12.213583359818005,
('ADBE', 'GOOGL'): 13.570674681293951,
('ADBE', 'INTU'): 9.254211425211263,
('ADBE', 'MA'): 7.418980673896577,
('ADBE', 'MSFT'): 8.58283144569894,
('ADBE', 'NFLX'): 6.259575001444054,
('ADBE', 'NTAP'): 7.682165352235385,
('ADBE', 'ORCL'): 10.948462323843168,
('ADBE', 'PAYX'): 35.64310708952115,
('ADBE', 'RHT'): 1.153945770702228,
('ADBE', 'SNPS'): 3.811917449321886,
('ADBE', 'SYMC'): 23.828023833314973,
('ADBE', 'TSS'): 8.092865305396348,
('ADBE', 'V'): 6.474267995072576,
('ADBE', 'VRSN'): 6.529076057006827,
('ADBE', 'WU'): 53.5264717931703,
('ADP', 'ADSK'): 20.28697737936668,
('ADP', 'AKAM'): 20.431752323892752,
('ADP', 'ANSS'): 23.84709906933113,
('ADP', 'ATVI'): 53.57235979823229,
('ADP', 'CA'): 1.036966431412505,
('ADP', 'CDNS'): 36.91168258814312,
('ADP', 'CRM'): 8.147196074456309,
('ADP', 'CTXS'): 1.9153141277466614,
('ADP', 'EA'): 21.35795561141269,
('ADP', 'EBAY'): 3.3230907679222694,
('ADP', 'FB'): 11.706227210933367,
('ADP', 'FIS'): 1.6134987587691578,
('ADP', 'FISV'): 1.8235753149414649,
('ADP', 'GOOG'): 4.618847052101154,
('ADP', 'GOOGL'): 3.942300296773336,
('ADP', 'INTU'): 6.380018278467871,
('ADP', 'MA'): 8.06633351721472,
('ADP', 'MSFT'): 6.495723204426145,
('ADP', 'NFLX'): 10.513119519449324,
('ADP', 'NTAP'): 14.013947246377636,
```

```
('ADP', 'ORCL'): 7.30246716338635,
('ADP', 'PAYX'): 0.8226419562575669,
('ADP', 'RHT'): 32.007280537056104,
('ADP', 'SNPS'): 13.84130036823279,
('ADP', 'SYMC'): 3.8265242380171927,
('ADP', 'TSS'): 8.179723739097994,
('ADP', 'V'): 8.77549584911996,
('ADP', 'VRSN'): 8.991016449671067,
('ADP', 'WU'): 4.0021985121418195,
('ADSK', 'AKAM'): 79.44756182547066,
('ADSK', 'ANSS'): 2.802816676496424,
('ADSK', 'ATVI'): 10.473166879619512,
('ADSK', 'CA'): 25.00253404565265,
('ADSK', 'CDNS'): 5.594513994395422,
('ADSK', 'CRM'): 4.2864926579759945,
('ADSK', 'CTXS'): 16.744195321732526,
('ADSK', 'EA'): 1.4927197476600735,
('ADSK', 'EBAY'): 9.288085300886838,
('ADSK', 'FB'): 2.345486843074066,
('ADSK', 'FIS'): 11.111437876124937,
('ADSK', 'FISV'): 11.457785548992558,
('ADSK', 'GOOG'): 7.189954599180913,
('ADSK', 'GOOGL'): 7.974748080038076,
('ADSK', 'INTU'): 5.3842964614956,
('ADSK', 'MA'): 4.727316247730728,
('ADSK', 'MSFT'): 5.841219057017932,
('ADSK', 'NFLX'): 3.651072813726262,
('ADSK', 'NTAP'): 10.686077929230263,
('ADSK', 'ORCL'): 5.08922384096354,
('ADSK', 'PAYX'): 26.287252771312566,
('ADSK', 'RHT'): 5.3463430031778,
('ADSK', 'SNPS'): 2.7797994513233686,
('ADSK', 'SYMC'): 13.912047655573508,
('ADSK', 'TSS'): 6.303920608506605,
('ADSK', 'V'): 3.866485990508485,
('ADSK', 'VRSN'): 3.838493791946847,
('ADSK', 'WU'): 39.765895809931905,
('AKAM', 'ANSS'): 84.95618664334728,
('AKAM', 'ATVI'): 138.11864153118768,
('AKAM', 'CA'): 16.636885456824604,
('AKAM', 'CDNS'): 109.05679793343567,
('AKAM', 'CRM'): 51.8854544044754,
('AKAM', 'CTXS'): 26.681146553488418,
('AKAM', 'EA'): 81.89300127017108,
('AKAM', 'EBAY'): 37.65293431813518,
('AKAM', 'FB'): 61.785387799939116,
('AKAM', 'FIS'): 32.41950857974421,
('AKAM', 'FISV'): 32.353809778653115,
```

```
('AKAM', 'GOOG'): 41.62565878305333,
('AKAM', 'GOOGL'): 39.397637759370404,
('AKAM', 'INTU'): 47.36271200879837,
('AKAM', 'MA'): 51.764804536559545,
('AKAM', 'MSFT'): 47.223042688254836,
('AKAM', 'NFLX'): 57.59766251382763,
('AKAM', 'NTAP'): 58.30983641058923,
('AKAM', 'ORCL'): 50.1097858975067,
('AKAM', 'PAYX'): 15.048243179305434,
('AKAM', 'RHT'): 99.66045731864762,
('AKAM', 'SNPS'): 65.45853719125701,
('AKAM', 'SYMC'): 33.97428230347582,
('AKAM', 'TSS'): 50.54386420484589,
('AKAM', 'V'): 54.19181180072739,
('AKAM', 'VRSN'): 54.793869519032775,
('AKAM', 'WU'): 7.571338709860404,
('ANSS', 'ATVI'): 8.154563717479729,
('ANSS', 'CA'): 29.943261681141287,
('ANSS', 'CDNS'): 2.5941690067372756,
('ANSS', 'CRM'): 4.534955933384435,
('ANSS', 'CTXS'): 18.552109268627913,
('ANSS', 'EA'): 4.295611339857258,
('ANSS', 'EBAY'): 12.272365956157463,
('ANSS', 'FB'): 3.0382590701096337,
('ANSS', 'FIS'): 14.349432684392086,
('ANSS', 'FISV'): 14.165581404285472,
('ANSS', 'GOOG'): 8.30167829382667,
('ANSS', 'GOOGL'): 9.41935164195621,
('ANSS', 'INTU'): 6.106299464298635,
('ANSS', 'MA'): 5.23414096590403,
('ANSS', 'MSFT'): 5.859227648695128,
('ANSS', 'NFLX'): 4.472687358212998,
('ANSS', 'NTAP'): 5.927731236701019,
('ANSS', 'ORCL'): 7.629628521151207,
('ANSS', 'PAYX'): 29.38047161074107,
('ANSS', 'RHT'): 2.5209045459914985,
('ANSS', 'SNPS'): 2.2015105355353706,
('ANSS', 'SYMC'): 18.897764721038637,
('ANSS', 'TSS'): 5.691196183576592,
('ANSS', 'V'): 4.188946306404346,
('ANSS', 'VRSN'): 4.3669750053096985,
('ANSS', 'WU'): 45.704650169081155,
('ATVI', 'CA'): 61.33711278896739,
('ATVI', 'CDNS'): 4.158582404160264,
('ATVI', 'CRM'): 21.406015165088377,
('ATVI', 'CTXS'): 45.44837761131093,
('ATVI', 'EA'): 9.292584427519238,
('ATVI', 'EBAY'): 32.77265383724729,
```

```
('ATVI', 'FB'): 15.839910132367695,
('ATVI', 'FIS'): 37.657095348456345,
('ATVI', 'FISV'): 37.52490631785801,
('ATVI', 'GOOG'): 28.839374630162528,
('ATVI', 'GOOGL'): 30.78521348947124,
('ATVI', 'INTU'): 24.70908645747999,
('ATVI', 'MA'): 22.54702328311286,
('ATVI', 'MSFT'): 24.91574785973942,
('ATVI', 'NFLX'): 19.668042415020217,
('ATVI', 'NTAP'): 24.090786300249558,
('ATVI', 'ORCL'): 23.358783918184045,
('ATVI', 'PAYX'): 63.12238407158666,
('ATVI', 'RHT'): 7.152630916743301,
('ATVI', 'SNPS'): 14.395739157136697,
('ATVI', 'SYMC'): 40.51242765452401,
('ATVI', 'TSS'): 24.22966939518591,
('ATVI', 'V'): 19.88524201271926,
('ATVI', 'VRSN'): 20.12178071584688,
('ATVI', 'WU'): 84.52634607386977,
('CA', 'CDNS'): 44.92324615372572,
('CA', 'CRM'): 11.863681636740417,
('CA', 'CTXS'): 2.512556877203576,
('CA', 'EA'): 25.50215903805071,
('CA', 'EBAY'): 4.951432360509962,
('CA', 'FB'): 16.12777353971046,
('CA', 'FIS'): 3.2432958740817774,
('CA', 'FISV'): 3.167146213222451,
('CA', 'GOOG'): 7.100471658897913,
('CA', 'GOOGL'): 6.139275746371604,
('CA', 'INTU'): 9.84001653266879,
('CA', 'MA'): 12.359129781010669,
('CA', 'MSFT'): 10.246127831626062,
('CA', 'NFLX'): 15.255154153362918,
('CA', 'NTAP'): 19.46742789414594,
('CA', 'ORCL'): 9.570574032140545,
('CA', 'PAYX'): 1.0556054126528271,
('CA', 'RHT'): 39.76651187557996,
('CA', 'SNPS'): 18.481677738986455,
('CA', 'SYMC'): 4.197878201911511,
('CA', 'TSS'): 12.65449529732265,
('CA', 'V'): 12.84349435028879,
('CA', 'VRSN'): 13.201898647636582,
('CA', 'WU'): 2.1979284961737853,
('CDNS', 'CRM'): 11.031947872090864,
('CDNS', 'CTXS'): 30.874006045052283,
('CDNS', 'EA'): 7.423172422335358,
('CDNS', 'EBAY'): 21.48987119937285,
('CDNS', 'FB'): 7.843372412328077,
```

('CDNS', 'FIS'): 24.65439386658227,
('CDNS', 'FISV'): 25.015929608117617,
('CDNS', 'GOOG'): 17.53980678233555,
('CDNS', 'GOOGL'): 19.139345470598055,
('CDNS', 'INTU'): 13.68628861255111,
('CDNS', 'MA'): 11.40871375099716,
('CDNS', 'MSFT'): 13.254787147719263,
('CDNS', 'NFLX'): 9.685292569671772,
('CDNS', 'NTAP'): 11.56312645877254,
('CDNS', 'ORCL'): 15.148307377550283,
('CDNS', 'PAYX'): 44.03092057783347,
('CDNS', 'RHT'): 0.9235182896273276,
('CDNS', 'SNPS'): 6.079412375484493,
('CDNS', 'SYMC'): 29.04313870451233,
('CDNS', 'TSS'): 12.547036771201697,
('CDNS', 'V'): 10.053963814709995,
('CDNS', 'VRSN'): 9.952639612236819,
('CDNS', 'WU'): 63.83878338302211,
('CRM', 'CTXS'): 5.520854365718749,
('CRM', 'EA'): 5.620478274383742,
('CRM', 'EBAY'): 2.377202652227747,
('CRM', 'FB'): 0.8317637499559798,
('CRM', 'FIS'): 3.09379877979767,
('CRM', 'FISV'): 3.1650678643526873,
('CRM', 'GOOG'): 1.037492391157313,
('CRM', 'GOOGL'): 1.3924248328373303,
('CRM', 'INTU'): 0.5105766639488711,
('CRM', 'MA'): 0.6511549699849806,
('CRM', 'MSFT'): 0.36001117965462753,
('CRM', 'NFLX'): 1.2108957423449946,
('CRM', 'NTAP'): 3.5829751108044983,
('CRM', 'ORCL'): 1.5820850174944878,
('CRM', 'PAYX'): 11.535565085868466,
('CRM', 'RHT'): 9.013410939134108,
('CRM', 'SNPS'): 1.0162211830295222,
('CRM', 'SYMC'): 6.024650812233192,
('CRM', 'TSS'): 1.1024545775554966,
('CRM', 'V'): 0.2072181498473113,
('CRM', 'VRSN'): 0.4678096232573099,
('CRM', 'WU'): 22.29733530684016,
('CTXS', 'EA'): 17.42256764275316,
('CTXS', 'EBAY'): 2.0664057033362875,
('CTXS', 'FB'): 9.18496370039216,
('CTXS', 'FIS'): 1.728746583356205,
('CTXS', 'FISV'): 0.9695653565296607,
('CTXS', 'GOOG'): 2.7079121979048635,
('CTXS', 'GOOGL'): 2.2092940469538815,
('CTXS', 'INTU'): 4.8913709356683865,

```
('CTXS', 'MA'): 7.086590128788154,
('CTXS', 'MSFT'): 4.861407447629741,
('CTXS', 'NFLX'): 9.027759466937667,
('CTXS', 'NTAP'): 10.444183675011955,
('CTXS', 'ORCL'): 4.95844833705484,
('CTXS', 'PAYX'): 2.501943680909248,
('CTXS', 'RHT'): 27.659407918969343,
('CTXS', 'SNPS'): 9.80419619358976,
('CTXS', 'SYMC'): 2.2462092216226237,
('CTXS', 'TSS'): 7.4228369480356955,
('CTXS', 'V'): 6.62745300090062,
('CTXS', 'VRSN'): 7.24828219747269,
('CTXS', 'WU'): 7.4085995127082125,
('EA', 'EBAY'): 9.214843633540891,
('EA', 'FB'): 3.0258058660745464,
('EA', 'FIS'): 11.70504140967026,
('EA', 'FISV'): 11.794745443152385,
('EA', 'GOOG'): 8.184044968271582,
('EA', 'GOOGL'): 8.965335998092671,
('EA', 'INTU'): 6.68146934576541,
('EA', 'MA'): 6.486667219218422,
('EA', 'MSFT'): 7.480453941051254,
('EA', 'NFLX'): 5.278196164196902,
('EA', 'NTAP'): 12.948636618694895,
('EA', 'ORCL'): 4.819562449968256,
('EA', 'PAYX'): 28.077198829424677,
('EA', 'RHT'): 7.8529307406098745,
('EA', 'SNPS'): 3.7633171009373787,
('EA', 'SYMC'): 13.155803735417425,
('EA', 'TSS'): 8.082116598648454,
('EA', 'V'): 4.845250902788944,
('EA', 'VRSN'): 5.211672155079992,
('EA', 'WU'): 40.958594678584845,
('EBAY', 'FB'): 3.987974429614597,
('EBAY', 'FIS'): 0.6089873412974444,
('EBAY', 'FISV'): 0.5815084586772623,
('EBAY', 'GOOG'): 1.1918500860681327,
('EBAY', 'GOOGL'): 1.0373520435504762,
('EBAY', 'INTU'): 1.9555020215727663,
('EBAY', 'MA'): 3.1530614463168445,
('EBAY', 'MSFT'): 2.2506704286098835,
('EBAY', 'NFLX'): 4.234601166058194,
('EBAY', 'NTAP'): 8.3265036552198,
('EBAY', 'ORCL'): 1.4521191149848256,
('EBAY', 'PAYX'): 5.8822104441942376,
('EBAY', 'RHT'): 18.774253278083243,
('EBAY', 'SNPS'): 5.11620294624582,
('EBAY', 'SYMC'): 1.477334513460374,
```

```
('EBAY', 'TSS'): 3.961557624755921,
('EBAY', 'V'): 2.6460063821353605,
('EBAY', 'VRSN'): 3.062778626018295,
('EBAY', 'WU'): 12.778796924220387,
('FB', 'FIS'): 5.1424069826376915,
('FB', 'FISV'): 5.50018265466027,
('FB', 'GOOG'): 2.686980027997973,
('FB', 'GOOGL'): 3.245882311693954,
('FB', 'INTU'): 1.5834553714088195,
('FB', 'MA'): 1.1242821402178662,
('FB', 'MSFT'): 1.4493476910520435,
('FB', 'NFLX'): 0.9715135404963295,
('FB', 'NTAP'): 4.9493478124464785,
('FB', 'ORCL'): 1.7804937489766783,
('FB', 'PAYX'): 16.558389369985022,
('FB', 'RHT'): 6.3083824442035965,
('FB', 'SNPS'): 0.6825352026638856,
('FB', 'SYMC'): 8.101153424870827,
('FB', 'TSS'): 1.8473552613654824,
('FB', 'V'): 0.4706843344746586,
('FB', 'VRSN'): 0.5305884543622407,
('FB', 'WU'): 28.329607998809706,
('FIS', 'FISV'): 0.2857409476522602,
('FIS', 'GOOG'): 1.3040282185579883,
('FIS', 'GOOGL'): 1.0170362853836052,
('FIS', 'INTU'): 2.1799213278728287,
('FIS', 'MA'): 3.324933816903757,
('FIS', 'MSFT'): 2.412316246121379,
('FIS', 'NFLX'): 4.858283616529405,
('FIS', 'NTAP'): 9.163823903199265,
('FIS', 'ORCL'): 2.4308061794139557,
('FIS', 'PAYX'): 3.795920433983037,
('FIS', 'RHT'): 21.064999160442934,
('FIS', 'SNPS'): 6.766284623296644,
('FIS', 'SYMC'): 2.052249740870587,
('FIS', 'TSS'): 3.7658130465777226,
('FIS', 'V'): 3.4082846707821917,
('FIS', 'VRSN'): 3.6497308698130544,
('FIS', 'WU'): 9.662710320419482,
('FISV', 'GOOG'): 1.0702903095153238,
('FISV', 'GOOGL'): 0.7679445764339363,
('FISV', 'INTU'): 2.3070740187462957,
('FISV', 'MA'): 3.9091120301367943,
('FISV', 'MSFT'): 2.626206195495933,
('FISV', 'NFLX'): 5.496221593928438,
('FISV', 'NTAP'): 8.906050140006466,
('FISV', 'ORCL'): 2.3648319164699907,
('FISV', 'PAYX'): 3.7280432472957137,
```

```
('FISV', 'RHT'): 21.913964868103836,
('FISV', 'SNPS'): 6.727171411308274,
('FISV', 'SYMC'): 1.741831468355303,
('FISV', 'TSS'): 4.428630778322396,
('FISV', 'V'): 3.683172966223803,
('FISV', 'VRSN'): 4.087982318900063,
('FISV', 'WU'): 9.619206503412636,
('GOOG', 'GOOGL'): 0.04045301814067839,
('GOOG', 'INTU'): 0.5793654361769872,
('GOOG', 'MA'): 1.57201467679406,
('GOOG', 'MSFT'): 0.6851416214529011,
('GOOG', 'NFLX'): 2.5620267781333617,
('GOOG', 'NTAP'): 5.359845564695879,
('GOOG', 'ORCL'): 1.7557676635576023,
('GOOG', 'PAYX'): 7.061983751434619,
('GOOG', 'RHT'): 14.870873023159652,
('GOOG', 'SNPS'): 3.523679515988144,
('GOOG', 'SYMC'): 4.009779642386653,
('GOOG', 'TSS'): 1.8966194684498623,
('GOOG', 'V'): 1.4285449382500413,
('GOOG', 'VRSN'): 1.7984341071492365,
('GOOG', 'WU'): 15.58635530830525,
('GOOGL', 'INTU'): 0.8224222462129634,
('GOOGL', 'MA'): 1.967264997216336,
('GOOGL', 'MSFT'): 0.9615383146739326,
('GOOGL', 'NFLX'): 3.0681795056864223,
('GOOGL', 'NTAP'): 6.032004586226634,
('GOOGL', 'ORCL'): 1.8798127945001002,
('GOOGL', 'PAYX'): 6.186157455246015,
('GOOGL', 'RHT'): 16.332732227463648,
('GOOGL', 'SNPS'): 4.190042780132229,
('GOOGL', 'SYMC'): 3.5475571926758573,
('GOOGL', 'TSS'): 2.309795545799717,
('GOOGL', 'V'): 1.8584326271076137,
('GOOGL', 'VRSN'): 2.231789356877014,
('GOOGL', 'WU'): 14.147208116409324,
('INTU', 'MA'): 0.5980149073455097,
('INTU', 'MSFT'): 0.2961582963157633,
('INTU', 'NFLX'): 1.5384904822032306,
('INTU', 'NTAP'): 4.281235179423031,
('INTU', 'ORCL'): 1.9354725729482922,
('INTU', 'PAYX'): 9.38788023137246,
('INTU', 'RHT'): 11.134611012675055,
('INTU', 'SNPS'): 2.2413013084682385,
('INTU', 'SYMC'): 5.798862864926634,
('INTU', 'TSS'): 0.8811381731971999,
('INTU', 'V'): 0.602668214336919,
('INTU', 'VRSN'): 0.8088193648550983,
```

```
('INTU', 'WU'): 19.441097299480838,
('MA', 'MSFT'): 0.39764108964701733,
('MA', 'NFLX'): 0.6369017056832544,
('MA', 'NTAP'): 3.877963044716283,
('MA', 'ORCL'): 2.6413256916262826,
('MA', 'PAYX'): 11.709249008900832,
('MA', 'RHT'): 8.450039758453155,
('MA', 'SNPS'): 1.9403223823745646,
('MA', 'SYMC'): 7.559165147367271,
('MA', 'TSS'): 0.41429499358425526,
('MA', 'V'): 0.39234837790166055,
('MA', 'VRSN'): 0.36546272129588436,
('MA', 'WU'): 22.543787681119465,
('MSFT', 'NFLX'): 1.2333220553378121,
('MSFT', 'NTAP'): 3.3963282089728937,
('MSFT', 'ORCL'): 2.1355365216745135,
('MSFT', 'PAYX'): 9.510533440478504,
('MSFT', 'RHT'): 10.450809868954009,
('MSFT', 'SNPS'): 2.058986395609634,
('MSFT', 'SYMC'): 6.190732497974761,
('MSFT', 'TSS'): 0.5873446524515048,
('MSFT', 'V'): 0.4652088816480763,
('MSFT', 'VRSN'): 0.612903633991063,
('MSFT', 'WU'): 19.7106251657355,
('NFLX', 'NTAP'): 4.456918206072289,
('NFLX', 'ORCL'): 3.1475968232629095,
('NFLX', 'PAYX'): 14.878855391721492,
('NFLX', 'RHT'): 7.3574851108681445,
('NFLX', 'SNPS'): 1.8966399696234966,
('NFLX', 'SYMC'): 8.642587930021216,
('NFLX', 'TSS'): 1.2638483920444297,
('NFLX', 'V'): 0.7914721044148963,
('NFLX', 'VRSN'): 0.8830585373131893,
('NFLX', 'WU'): 26.23052590307758,
('NTAP', 'ORCL'): 7.877315748960216,
('NTAP', 'PAYX'): 16.75099828553493,
('NTAP', 'RHT'): 9.196640717701607,
('NTAP', 'SNPS'): 3.886666292911035,
('NTAP', 'SYMC'): 14.238243898081427,
('NTAP', 'TSS'): 3.4498954116894454,
('NTAP', 'V'): 3.71029345139542,
('NTAP', 'VRSN'): 3.846009116894125,
('NTAP', 'WU'): 30.38583022136295,
('ORCL', 'PAYX'): 11.347323330799314,
('ORCL', 'RHT'): 13.280774822778579,
('ORCL', 'SNPS'): 2.710249369916319,
('ORCL', 'SYMC'): 3.565852117237573,
('ORCL', 'TSS'): 3.722227037652517,
```

```
 ('ORCL', 'V'): 1.613623668970272,
 ('ORCL', 'VRSN'): 1.8764910358029467,
 ('ORCL', 'WU'): 19.993499952513673,
 ('PAYX', 'RHT'): 38.58053388646707,
 ('PAYX', 'SNPS'): 18.258422223208996,
 ('PAYX', 'SYMC'): 5.920863136124079,
 ('PAYX', 'TSS'): 11.580753296403191,
 ('PAYX', 'V'): 12.677776781877178,
 ('PAYX', 'VRSN'): 12.967781798160182,
 ('PAYX', 'WU'): 2.403220488062168,
 ('RHT', 'SNPS'): 5.1797755243532455,
 ('RHT', 'SYMC'): 26.59712869275919,
 ('RHT', 'TSS'): 9.35679983077451,
 ('RHT', 'V'): 8.008434189190275,
 ('RHT', 'VRSN'): 7.470214569170028,
 ('RHT', 'WU'): 57.463025756540006,
 ('SNPS', 'SYMC'): 9.25913655540079,
 ('SNPS', 'TSS'): 2.7893004253939844,
 ('SNPS', 'V'): 0.8907311970171012,
 ('SNPS', 'VRSN'): 1.1277965248402264,
 ('SNPS', 'WU'): 31.367380362236766,
 ('SYMC', 'TSS'): 9.00710896810785,
 ('SYMC', 'V'): 6.625221625259926,
 ('SYMC', 'VRSN'): 7.133660727502845,
 ('SYMC', 'WU'): 10.572458910322382,
 ('TSS', 'V'): 0.898612533002861,
 ('TSS', 'VRSN'): 0.938963808222344,
 ('TSS', 'WU'): 22.493071934956873,
 ('V', 'VRSN'): 0.27237885594731637,
 ('V', 'WU'): 23.734044154960266,
 ('VRSN', 'WU'): 24.175810311065415}
```

In [338]:
```
# Sort dictionary by value
d = dict_pairs

# Note: Dictionaries prior to Python 3.6 (I believe,) are not ordered.
#       So returning in
list_minimized_sum_of_squared_deviations = [(k, d[k]) for k in sorted(d, key=d.get, r
list_minimized_sum_of_squared_deviations
```

Out[338]:
```
[(('GOOG', 'GOOGL'), 0.04045301814067839),
 (('CRM', 'V'), 0.2072181498473113),
 (('V', 'VRSN'), 0.27237885594731637),
 (('FIS', 'FISV'), 0.2857409476522602),
 (('INTU', 'MSFT'), 0.2961582963157633),
 (('CRM', 'MSFT'), 0.36001117965462753),
 (('MA', 'VRSN'), 0.36546272129588436),
 (('MA', 'V'), 0.39234837790166055),
```

```
(('MA', 'MSFT'), 0.39764108964701733),
(('MA', 'TSS'), 0.41429499358425526),
(('MSFT', 'V'), 0.4652088816480763),
(('CRM', 'VRSN'), 0.4678096232573099),
(('FB', 'V'), 0.4706843344746586),
(('CRM', 'INTU'), 0.5105766639488711),
(('FB', 'VRSN'), 0.5305884543622407),
(('GOOG', 'INTU'), 0.5793654361769872),
(('EBAY', 'FISV'), 0.5815084586772623),
(('MSFT', 'TSS'), 0.5873446524515048),
(('INTU', 'MA'), 0.5980149073455097),
(('INTU', 'V'), 0.602668214336919),
(('EBAY', 'FIS'), 0.6089873412974444),
(('MSFT', 'VRSN'), 0.612903633991063),
(('MA', 'NFLX'), 0.6369017056832544),
(('CRM', 'MA'), 0.6511549699849806),
(('FB', 'SNPS'), 0.6825352026638856),
(('GOOG', 'MSFT'), 0.6851416214529011),
(('FISV', 'GOOGL'), 0.7679445764339363),
(('NFLX', 'V'), 0.7914721044148963),
(('INTU', 'VRSN'), 0.8088193648550983),
(('GOOGL', 'INTU'), 0.8224222462129634),
(('ADP', 'PAYX'), 0.8226419562575669),
(('CRM', 'FB'), 0.8317637499559798),
(('ADBE', 'ANSS'), 0.8696850922794794),
(('INTU', 'TSS'), 0.8811381731971999),
(('NFLX', 'VRSN'), 0.8830585373131893),
(('SNPS', 'V'), 0.8907311970171012),
(('TSS', 'V'), 0.898612533002861),
(('CDNS', 'RHT'), 0.9235182896273276),
(('TSS', 'VRSN'), 0.938963808222344),
(('GOOGL', 'MSFT'), 0.9615383146739326),
(('CTXS', 'FISV'), 0.9695653565296607),
(('FB', 'NFLX'), 0.9715135404963295),
(('CRM', 'SNPS'), 1.0162211830295222),
(('FIS', 'GOOGL'), 1.0170362853836052),
(('ADP', 'CA'), 1.036966431412505),
(('EBAY', 'GOOGL'), 1.0373520435504762),
(('CRM', 'GOOG'), 1.037492391157313),
(('CA', 'PAYX'), 1.0556054126528271),
(('FISV', 'GOOG'), 1.0702903095153238),
(('CRM', 'TSS'), 1.1024545775554966),
(('FB', 'MA'), 1.1242821402178662),
(('SNPS', 'VRSN'), 1.1277965248402264),
(('ADBE', 'RHT'), 1.153945770702228),
(('EBAY', 'GOOG'), 1.1918500860681327),
(('CRM', 'NFLX'), 1.2108957423449946),
(('ADBE', 'CDNS'), 1.2265447236409859),
```

```
(('MSFT', 'NFLX'), 1.2333220553378121),
(('NFLX', 'TSS'), 1.2638483920444297),
(('FIS', 'GOOG'), 1.3040282185579883),
(('CRM', 'GOOGL'), 1.3924248328373303),
(('GOOG', 'V'), 1.4285449382500413),
(('FB', 'MSFT'), 1.4493476910520435),
(('EBAY', 'ORCL'), 1.4521191149848256),
(('EBAY', 'SYMC'), 1.477334513460374),
(('ADSK', 'EA'), 1.4927197476600735),
(('INTU', 'NFLX'), 1.5384904822032306),
(('GOOG', 'MA'), 1.57201467679406),
(('CRM', 'ORCL'), 1.5820850174944878),
(('FB', 'INTU'), 1.5834553714088195),
(('ADP', 'FIS'), 1.6134987587691578),
(('ORCL', 'V'), 1.613623668970272),
(('CTXS', 'FIS'), 1.728746583356205),
(('FISV', 'SYMC'), 1.741831468355303),
(('GOOG', 'ORCL'), 1.7557676635576023),
(('FB', 'ORCL'), 1.7804937489766783),
(('GOOG', 'VRSN'), 1.7984341071492365),
(('ADP', 'FISV'), 1.8235753149414649),
(('FB', 'TSS'), 1.8473552613654824),
(('GOOGL', 'V'), 1.8584326271076137),
(('ORCL', 'VRSN'), 1.8764910358029467),
(('GOOGL', 'ORCL'), 1.8798127945001002),
(('GOOG', 'TSS'), 1.8966194684498623),
(('NFLX', 'SNPS'), 1.8966399696234966),
(('ADP', 'CTXS'), 1.9153141277466614),
(('INTU', 'ORCL'), 1.9354725729482922),
(('MA', 'SNPS'), 1.9403223823745646),
(('EBAY', 'INTU'), 1.9555020215727663),
(('GOOGL', 'MA'), 1.967264997216336),
(('FIS', 'SYMC'), 2.052249740870587),
(('MSFT', 'SNPS'), 2.058986395609634),
(('CTXS', 'EBAY'), 2.0664057033362875),
(('MSFT', 'ORCL'), 2.1355365216745135),
(('FIS', 'INTU'), 2.1799213278728287),
(('CA', 'WU'), 2.1979284961737853),
(('ANSS', 'SNPS'), 2.2015105355353706),
(('CTXS', 'GOOGL'), 2.2092940469538815),
(('GOOGL', 'VRSN'), 2.231789356877014),
(('INTU', 'SNPS'), 2.2413013084682385),
(('CTXS', 'SYMC'), 2.2462092216226237),
(('EBAY', 'MSFT'), 2.2506704286098835),
(('FISV', 'INTU'), 2.3070740187462957),
(('GOOGL', 'TSS'), 2.309795545799717),
(('ADSK', 'FB'), 2.345486843074066),
(('FISV', 'ORCL'), 2.3648319164699907),
```

```
(('CRM', 'EBAY'), 2.377202652227747),
(('PAYX', 'WU'), 2.403220488062168),
(('FIS', 'MSFT'), 2.412316246121379),
(('FIS', 'ORCL'), 2.4308061794139557),
(('CTXS', 'PAYX'), 2.501943680909248),
(('CA', 'CTXS'), 2.512556877203576),
(('ANSS', 'RHT'), 2.5209045459914985),
(('GOOG', 'NFLX'), 2.5620267781333617),
(('ANSS', 'CDNS'), 2.5941690067372756),
(('FISV', 'MSFT'), 2.626206195495933),
(('MA', 'ORCL'), 2.6413256916262826),
(('EBAY', 'V'), 2.6460063821353605),
(('FB', 'GOOG'), 2.686980027997973),
(('CTXS', 'GOOG'), 2.7079121979048635),
(('ORCL', 'SNPS'), 2.710249369916319),
(('ADSK', 'SNPS'), 2.7797994513233686),
(('SNPS', 'TSS'), 2.7893004253939844),
(('ADSK', 'ANSS'), 2.802816676496424),
(('EA', 'FB'), 3.0258058660745464),
(('ANSS', 'FB'), 3.0382590701096337),
(('EBAY', 'VRSN'), 3.062778626018295),
(('GOOGL', 'NFLX'), 3.0681795056864223),
(('CRM', 'FIS'), 3.09379877979767),
(('NFLX', 'ORCL'), 3.1475968232629095),
(('EBAY', 'MA'), 3.1530614463168445),
(('CRM', 'FISV'), 3.1650678643526873),
(('CA', 'FISV'), 3.167146213222451),
(('CA', 'FIS'), 3.2432958740817774),
(('FB', 'GOOGL'), 3.245882311693954),
(('ADP', 'EBAY'), 3.3230907679222694),
(('FIS', 'MA'), 3.324933816903757),
(('MSFT', 'NTAP'), 3.3963282089728937),
(('FIS', 'V'), 3.4082846707821917),
(('NTAP', 'TSS'), 3.4498954116894454),
(('GOOG', 'SNPS'), 3.523679515988144),
(('GOOGL', 'SYMC'), 3.5475571926758573),
(('ORCL', 'SYMC'), 3.565852117237573),
(('CRM', 'NTAP'), 3.5829751108044983),
(('FIS', 'VRSN'), 3.6497308698130544),
(('ADSK', 'NFLX'), 3.651072813726262),
(('FISV', 'V'), 3.683172966223803),
(('NTAP', 'V'), 3.71029345139542),
(('ORCL', 'TSS'), 3.722227037652517),
(('FISV', 'PAYX'), 3.7280432472957137),
(('EA', 'SNPS'), 3.7633171009373787),
(('FIS', 'TSS'), 3.7658130465777226),
(('FIS', 'PAYX'), 3.795920433983037),
(('ADBE', 'SNPS'), 3.811917449321886),
```

```
(('ADP', 'SYMC'), 3.8265242380171927),
(('ADSK', 'VRSN'), 3.838493791946847),
(('NTAP', 'VRSN'), 3.846009116894125),
(('ADSK', 'V'), 3.866485990508485),
(('MA', 'NTAP'), 3.877963044716283),
(('NTAP', 'SNPS'), 3.886666292911035),
(('FISV', 'MA'), 3.9091120301367943),
(('ADP', 'GOOGL'), 3.942300296773336),
(('EBAY', 'TSS'), 3.961557624755921),
(('EBAY', 'FB'), 3.987974429614597),
(('ADP', 'WU'), 4.0021985121418195),
(('GOOG', 'SYMC'), 4.009779642386653),
(('FISV', 'VRSN'), 4.087982318900063),
(('ATVI', 'CDNS'), 4.158582404160264),
(('ADBE', 'ADSK'), 4.158609456332218),
(('ANSS', 'V'), 4.188946306404346),
(('GOOGL', 'SNPS'), 4.190042780132229),
(('CA', 'SYMC'), 4.197878201911511),
(('EBAY', 'NFLX'), 4.234601166058194),
(('INTU', 'NTAP'), 4.281235179423031),
(('ADSK', 'CRM'), 4.2864926579759945),
(('ANSS', 'EA'), 4.295611339857258),
(('ANSS', 'VRSN'), 4.3669750053096985),
(('FISV', 'TSS'), 4.428630778322396),
(('NFLX', 'NTAP'), 4.456918206072289),
(('ANSS', 'NFLX'), 4.472687358212998),
(('ANSS', 'CRM'), 4.534955933384435),
(('ADP', 'GOOG'), 4.618847052101154),
(('ADSK', 'MA'), 4.727316247730728),
(('EA', 'ORCL'), 4.819562449968256),
(('EA', 'V'), 4.845250902788944),
(('FIS', 'NFLX'), 4.858283616529405),
(('CTXS', 'MSFT'), 4.861407447629741),
(('CTXS', 'INTU'), 4.8913709356683865),
(('FB', 'NTAP'), 4.9493478124464785),
(('CA', 'EBAY'), 4.951432360509962),
(('CTXS', 'ORCL'), 4.95844833705484),
(('ADBE', 'FB'), 4.966219754693422),
(('ADSK', 'ORCL'), 5.08922384096354),
(('EBAY', 'SNPS'), 5.11620294624582),
(('FB', 'FIS'), 5.1424069826376915),
(('RHT', 'SNPS'), 5.1797755243532455),
(('EA', 'VRSN'), 5.211672155079992),
(('ANSS', 'MA'), 5.23414096590403),
(('EA', 'NFLX'), 5.278196164196902),
(('ADSK', 'RHT'), 5.3463430031778),
(('GOOG', 'NTAP'), 5.359845564695879),
(('ADSK', 'INTU'), 5.3842964614956),
```

```
(('FISV', 'NFLX'), 5.496221593928438),
(('FB', 'FISV'), 5.50018265466027),
(('CRM', 'CTXS'), 5.520854365718749),
(('ADSK', 'CDNS'), 5.594513994395422),
(('CRM', 'EA'), 5.620478274383742),
(('ANSS', 'TSS'), 5.691196183576592),
(('INTU', 'SYMC'), 5.798862864926634),
(('ADSK', 'MSFT'), 5.841219057017932),
(('ANSS', 'MSFT'), 5.859227648695128),
(('EBAY', 'PAYX'), 5.8822104441942376),
(('PAYX', 'SYMC'), 5.920863136124079),
(('ANSS', 'NTAP'), 5.927731236701019),
(('CRM', 'SYMC'), 6.024650812233192),
(('GOOGL', 'NTAP'), 6.032004586226634),
(('CDNS', 'SNPS'), 6.079412375484493),
(('ANSS', 'INTU'), 6.106299464298635),
(('CA', 'GOOGL'), 6.139275746371604),
(('GOOGL', 'PAYX'), 6.186157455246015),
(('MSFT', 'SYMC'), 6.190732497974761),
(('ADBE', 'EA'), 6.204815344227672),
(('ADBE', 'NFLX'), 6.259575001444054),
(('ADSK', 'TSS'), 6.303920608506605),
(('FB', 'RHT'), 6.3083824442035965),
(('ADP', 'INTU'), 6.380018278467871),
(('ADBE', 'V'), 6.474267995072576),
(('EA', 'MA'), 6.486667219218422),
(('ADP', 'MSFT'), 6.495723204426145),
(('ADBE', 'ATVI'), 6.517351640000458),
(('ADBE', 'VRSN'), 6.529076057006827),
(('SYMC', 'V'), 6.625221625259926),
(('CTXS', 'V'), 6.62745300090062),
(('EA', 'INTU'), 6.68146934576541),
(('FISV', 'SNPS'), 6.727171411308274),
(('FIS', 'SNPS'), 6.766284623296644),
(('GOOG', 'PAYX'), 7.061983751434619),
(('ADBE', 'CRM'), 7.085918839247967),
(('CTXS', 'MA'), 7.086590128788154),
(('CA', 'GOOG'), 7.100471658897913),
(('SYMC', 'VRSN'), 7.133660727502845),
(('ATVI', 'RHT'), 7.152630916743301),
(('ADSK', 'GOOG'), 7.189954599180913),
(('CTXS', 'VRSN'), 7.24828219747269),
(('ADP', 'ORCL'), 7.30246716338635),
(('NFLX', 'RHT'), 7.3574851108681445),
(('CTXS', 'WU'), 7.4085995127082125),
(('ADBE', 'MA'), 7.418980673896577),
(('CTXS', 'TSS'), 7.4228369480356955),
(('CDNS', 'EA'), 7.423172422335358),
```

```
(('RHT', 'VRSN'), 7.470214569170028),
(('EA', 'MSFT'), 7.480453941051254),
(('MA', 'SYMC'), 7.559165147367271),
(('AKAM', 'WU'), 7.571338709860404),
(('ANSS', 'ORCL'), 7.629628521151207),
(('ADBE', 'NTAP'), 7.682165352235385),
(('CDNS', 'FB'), 7.843372412328077),
(('EA', 'RHT'), 7.8529307406098745),
(('NTAP', 'ORCL'), 7.877315748960216),
(('ADSK', 'GOOGL'), 7.974748080038076),
(('RHT', 'V'), 8.008434189190275),
(('ADP', 'MA'), 8.06633351721472),
(('EA', 'TSS'), 8.082116598648454),
(('ADBE', 'TSS'), 8.092865305396348),
(('FB', 'SYMC'), 8.101153424870827),
(('ADP', 'CRM'), 8.147196074456309),
(('ANSS', 'ATVI'), 8.154563717479729),
(('ADP', 'TSS'), 8.179723739097994),
(('EA', 'GOOG'), 8.184044968271582),
(('ANSS', 'GOOG'), 8.30167829382667),
(('EBAY', 'NTAP'), 8.3265036552198),
(('MA', 'RHT'), 8.450039758453155),
(('ADBE', 'MSFT'), 8.58283144569894),
(('NFLX', 'SYMC'), 8.642587930021216),
(('ADP', 'V'), 8.77549584911996),
(('FISV', 'NTAP'), 8.906050140006466),
(('EA', 'GOOGL'), 8.965335998092671),
(('ADP', 'VRSN'), 8.991016449671067),
(('SYMC', 'TSS'), 9.00710896810785),
(('CRM', 'RHT'), 9.013410939134108),
(('CTXS', 'NFLX'), 9.027759466937667),
(('FIS', 'NTAP'), 9.163823903199265),
(('CTXS', 'FB'), 9.18496370039216),
(('NTAP', 'RHT'), 9.196640717701607),
(('EA', 'EBAY'), 9.214843633540891),
(('ADBE', 'INTU'), 9.254211425211263),
(('SNPS', 'SYMC'), 9.25913655540079),
(('ADSK', 'EBAY'), 9.288085300886838),
(('ATVI', 'EA'), 9.292584427519238),
(('RHT', 'TSS'), 9.35679983077451),
(('INTU', 'PAYX'), 9.38788023137246),
(('ANSS', 'GOOGL'), 9.41935164195621),
(('MSFT', 'PAYX'), 9.510533440478504),
(('CA', 'ORCL'), 9.570574032140545),
(('FISV', 'WU'), 9.619206503412636),
(('FIS', 'WU'), 9.662710320419482),
(('CDNS', 'NFLX'), 9.685292569671772),
(('CTXS', 'SNPS'), 9.80419619358976),
```

```
(('CA', 'INTU'), 9.84001653266879),
(('CDNS', 'VRSN'), 9.952639612236819),
(('CDNS', 'V'), 10.053963814709995),
(('CA', 'MSFT'), 10.246127831626062),
(('CTXS', 'NTAP'), 10.444183675011955),
(('MSFT', 'RHT'), 10.450809868954009),
(('ADSK', 'ATVI'), 10.473166879619512),
(('ADP', 'NFLX'), 10.513119519449324),
(('SYMC', 'WU'), 10.572458910322382),
(('ADSK', 'NTAP'), 10.686077929230263),
(('ADBE', 'ORCL'), 10.948462323843168),
(('CDNS', 'CRM'), 11.031947872090864),
(('ADSK', 'FIS'), 11.111437876124937),
(('INTU', 'RHT'), 11.134611012675055),
(('ORCL', 'PAYX'), 11.347323330799314),
(('CDNS', 'MA'), 11.40871375099716),
(('ADSK', 'FISV'), 11.457785548992558),
(('CRM', 'PAYX'), 11.535565085868466),
(('CDNS', 'NTAP'), 11.56312645877254),
(('PAYX', 'TSS'), 11.580753296403191),
(('EA', 'FIS'), 11.70504140967026),
(('ADP', 'FB'), 11.706227210933367),
(('MA', 'PAYX'), 11.709249008900832),
(('EA', 'FISV'), 11.794745443152385),
(('CA', 'CRM'), 11.863681636740417),
(('ADBE', 'GOOG'), 12.213583359818005),
(('ANSS', 'EBAY'), 12.272365956157463),
(('CA', 'MA'), 12.359129781010669),
(('CDNS', 'TSS'), 12.547036771201697),
(('CA', 'TSS'), 12.65449529732265),
(('PAYX', 'V'), 12.677776781877178),
(('EBAY', 'WU'), 12.778796924220387),
(('CA', 'V'), 12.84349435028879),
(('EA', 'NTAP'), 12.948636618694895),
(('PAYX', 'VRSN'), 12.967781798160182),
(('EA', 'SYMC'), 13.155803735417425),
(('CA', 'VRSN'), 13.201898647636582),
(('CDNS', 'MSFT'), 13.254787147719263),
(('ORCL', 'RHT'), 13.280774822778579),
(('ADBE', 'GOOGL'), 13.570674681293951),
(('CDNS', 'INTU'), 13.68628861255111),
(('ADP', 'SNPS'), 13.84130036823279),
(('ADSK', 'SYMC'), 13.912047655573508),
(('ADP', 'NTAP'), 14.013947246377636),
(('GOOGL', 'WU'), 14.147208116409324),
(('ANSS', 'FISV'), 14.165581404285472),
(('NTAP', 'SYMC'), 14.238243898081427),
(('ANSS', 'FIS'), 14.349432684392086),
```

```
(('ATVI', 'SNPS'), 14.395739157136697),
(('GOOG', 'RHT'), 14.870873023159652),
(('NFLX', 'PAYX'), 14.878855391721492),
(('AKAM', 'PAYX'), 15.048243179305434),
(('CDNS', 'ORCL'), 15.148307377550283),
(('CA', 'NFLX'), 15.255154153362918),
(('GOOG', 'WU'), 15.58635530830525),
(('ATVI', 'FB'), 15.839910132367695),
(('CA', 'FB'), 16.12777353971046),
(('GOOGL', 'RHT'), 16.332732227463648),
(('ADBE', 'EBAY'), 16.506262595260367),
(('FB', 'PAYX'), 16.558389369985022),
(('AKAM', 'CA'), 16.636885456824604),
(('ADSK', 'CTXS'), 16.744195321732526),
(('NTAP', 'PAYX'), 16.75099828553493),
(('CTXS', 'EA'), 17.42256764275316),
(('CDNS', 'GOOG'), 17.53980678233555),
(('PAYX', 'SNPS'), 18.258422223208996),
(('CA', 'SNPS'), 18.481677738986455),
(('ANSS', 'CTXS'), 18.552109268627913),
(('EBAY', 'RHT'), 18.774253278083243),
(('ADBE', 'FIS'), 18.847753171034842),
(('ANSS', 'SYMC'), 18.897764721038637),
(('ADBE', 'FISV'), 19.019434180821282),
(('CDNS', 'GOOGL'), 19.139345470598055),
(('INTU', 'WU'), 19.441097299480838),
(('CA', 'NTAP'), 19.46742789414594),
(('ATVI', 'NFLX'), 19.668042415020217),
(('MSFT', 'WU'), 19.7106251657355),
(('ATVI', 'V'), 19.88524201271926),
(('ORCL', 'WU'), 19.993499952513673),
(('ATVI', 'VRSN'), 20.12178071584688),
(('ADP', 'ADSK'), 20.28697737936668),
(('ADP', 'AKAM'), 20.431752323892752),
(('FIS', 'RHT'), 21.064999160442934),
(('ADP', 'EA'), 21.35795561141269),
(('ATVI', 'CRM'), 21.406015165088377),
(('CDNS', 'EBAY'), 21.48987119937285),
(('FISV', 'RHT'), 21.913964868103836),
(('CRM', 'WU'), 22.29733530684016),
(('TSS', 'WU'), 22.493071934956873),
(('MA', 'WU'), 22.543787681119465),
(('ATVI', 'MA'), 22.54702328311286),
(('ATVI', 'ORCL'), 23.358783918184045),
(('V', 'WU'), 23.734044154960266),
(('ADBE', 'SYMC'), 23.828023833314973),
(('ADP', 'ANSS'), 23.84709906933113),
(('ADBE', 'CTXS'), 23.967198766619457),
```

```
(('ATVI', 'NTAP'), 24.090786300249558),
(('VRSN', 'WU'), 24.175810311065415),
(('ATVI', 'TSS'), 24.22966939518591),
(('CDNS', 'FIS'), 24.65439386658227),
(('ATVI', 'INTU'), 24.70908645747999),
(('ATVI', 'MSFT'), 24.91574785973942),
(('ADSK', 'CA'), 25.00253404565265),
(('CDNS', 'FISV'), 25.015929608117617),
(('CA', 'EA'), 25.50215903805071),
(('NFLX', 'WU'), 26.23052590307758),
(('ADSK', 'PAYX'), 26.287252771312566),
(('RHT', 'SYMC'), 26.59712869275919),
(('AKAM', 'CTXS'), 26.681146553488418),
(('CTXS', 'RHT'), 27.659407918969343),
(('EA', 'PAYX'), 28.077198829424677),
(('FB', 'WU'), 28.329607998809706),
(('ATVI', 'GOOG'), 28.839374630162528),
(('CDNS', 'SYMC'), 29.04313870451233),
(('ADBE', 'ADP'), 29.376129693592553),
(('ANSS', 'PAYX'), 29.38047161074107),
(('ANSS', 'CA'), 29.943261681141287),
(('NTAP', 'WU'), 30.38583022136295),
(('ATVI', 'GOOGL'), 30.78521348947124),
(('CDNS', 'CTXS'), 30.874006045052283),
(('SNPS', 'WU'), 31.367380362236766),
(('ADP', 'RHT'), 32.007280537056104),
(('AKAM', 'FISV'), 32.353809778653115),
(('AKAM', 'FIS'), 32.41950857974421),
(('ATVI', 'EBAY'), 32.77265383724729),
(('AKAM', 'SYMC'), 33.97428230347582),
(('ADBE', 'PAYX'), 35.64310708952115),
(('ADBE', 'CA'), 36.511020170463276),
(('ADP', 'CDNS'), 36.91168258814312),
(('ATVI', 'FISV'), 37.52490631785801),
(('AKAM', 'EBAY'), 37.65293431813518),
(('ATVI', 'FIS'), 37.657095348456345),
(('PAYX', 'RHT'), 38.58053388646707),
(('AKAM', 'GOOGL'), 39.397637759370404),
(('ADSK', 'WU'), 39.765895809931905),
(('CA', 'RHT'), 39.76651187557996),
(('ATVI', 'SYMC'), 40.51242765452401),
(('EA', 'WU'), 40.958594678584845),
(('AKAM', 'GOOG'), 41.62565878305333),
(('CDNS', 'PAYX'), 44.03092057783347),
(('CA', 'CDNS'), 44.92324615372572),
(('ATVI', 'CTXS'), 45.44837761131093),
(('ANSS', 'WU'), 45.704650169081155),
(('AKAM', 'MSFT'), 47.223042688254836),
```

```
 (('AKAM', 'INTU'), 47.36271200879837),
 (('AKAM', 'ORCL'), 50.1097858975067),
 (('AKAM', 'TSS'), 50.54386420484589),
 (('AKAM', 'MA'), 51.764804536559545),
 (('AKAM', 'CRM'), 51.8854544044754),
 (('ADBE', 'WU'), 53.5264717931703),
 (('ADP', 'ATVI'), 53.57235979823229),
 (('AKAM', 'V'), 54.19181180072739),
 (('AKAM', 'VRSN'), 54.793869519032775),
 (('RHT', 'WU'), 57.463025756540006),
 (('AKAM', 'NFLX'), 57.59766251382763),
 (('AKAM', 'NTAP'), 58.30983641058923),
 (('ATVI', 'CA'), 61.33711278896739),
 (('AKAM', 'FB'), 61.785387799939116),
 (('ATVI', 'PAYX'), 63.12238407158666),
 (('CDNS', 'WU'), 63.83878338302211),
 (('AKAM', 'SNPS'), 65.45853719125701),
 (('ADSK', 'AKAM'), 79.44756182547066),
 (('AKAM', 'EA'), 81.89300127017108),
 (('ATVI', 'WU'), 84.52634607386977),
 (('AKAM', 'ANSS'), 84.95618664334728),
 (('ADBE', 'AKAM'), 94.87267386981122),
 (('AKAM', 'RHT'), 99.66045731864762),
 (('AKAM', 'CDNS'), 109.05679793343567),
 (('AKAM', 'ATVI'), 138.11864153118768)]
```

2.0.21 Minimized Sum of Squared Deviations within the Related Industry Only - Semiconductors

In [339]:
```
# https://stackoverflow.com/questions/39203662/euclidean-distance-matrix-using-panda
#pd.DataFrame( , columns = df_ticker_closing_for_semiconductors.columns, index = df_

# https://stackoverflow.com/questions/41337316/pandas-compare-all-dataframe-columns-
df = df_ticker_closing_for_semiconductors.copy()

# This is a numpy array
ndarray_a = df.values
column_names = df.columns

# Multiplying every column by every other column
#list_for_matrix_a = []
dict_pairs = {}
for i in range(len(ndarray_a.T)):

    column_name_in_focus = column_names[i]
    column_values_in_focus = ndarray_a[:, i]

    # The other columns
```

```python
        for j in range(len(ndarray_a.T)):

            # Skip comparison to itself
            if j == i:
                continue

            column_name_other = column_names[j]
            column_values_other = ndarray_a[:, j]

            # These are numpy ndarrays
            ndarray_temp_a = column_values_in_focus - column_values_other
            ndarray_temp_b = np.square(ndarray_temp_a)

            sum_squared_deviations = ndarray_temp_b.sum()

            # Put in a list so we can sort it.
            # That way, we will have unique keys in the dictionary.
            # ('ADI', 'XLNX') is the same as ('XLNX', 'ADI') in this case.
            # Yes, the value for the key will be overriden, but it would be the
            # same value.
            #
            # After convert to a tuple. No real reason I can think of except convention.
            list_pair_key = sorted([column_name_in_focus, column_name_other])
            tuple_pair_key = tuple(list_pair_key)

            dict_pairs[tuple_pair_key] = sum_squared_deviations
```

In [340]:
```python
dict_pairs
```

Out[340]:
```
{('ADI', 'AMAT'): 15.409621894273716,
 ('ADI', 'AMD'): 10.270124812815467,
 ('ADI', 'AVGO'): 7.410060308379357,
 ('ADI', 'INTC'): 3.728934838622128,
 ('ADI', 'KLAC'): 2.2691757219751896,
 ('ADI', 'LRCX'): 33.62975145123157,
 ('ADI', 'MCHP'): 4.393596211924288,
 ('ADI', 'MU'): 53.50393464508413,
 ('ADI', 'NVDA'): 52.403051521260295,
 ('ADI', 'QCOM'): 18.563161902462408,
 ('ADI', 'QRVO'): 2.0794069574152445,
 ('ADI', 'SWKS'): 6.976968857740386,
 ('ADI', 'TXN'): 1.9859323681800614,
 ('ADI', 'XLNX'): 0.8472312767165687,
 ('AMAT', 'AMD'): 20.67896462610036,
 ('AMAT', 'AVGO'): 3.7710241391412676,
 ('AMAT', 'INTC'): 26.80535021389236,
 ('AMAT', 'KLAC'): 8.348660336741,
 ('AMAT', 'LRCX'): 3.7335933812111457,
```

```
('AMAT', 'MCHP'): 4.27285769584317,
('AMAT', 'MU'): 13.121547700573586,
('AMAT', 'NVDA'): 13.855923845250377,
('AMAT', 'QCOM'): 61.494638709091895,
('AMAT', 'QRVO'): 13.435834239448647,
('AMAT', 'SWKS'): 7.672281410194768,
('AMAT', 'TXN'): 10.236713850295295,
('AMAT', 'XLNX'): 15.221760878707585,
('AMD', 'AVGO'): 10.4431315585542,
('AMD', 'INTC'): 24.18908167793324,
('AMD', 'KLAC'): 8.841407821806783,
('AMD', 'LRCX'): 36.569508087369414,
('AMD', 'MCHP'): 10.280931387897315,
('AMD', 'MU'): 56.65870410667582,
('AMD', 'NVDA'): 57.77420658558669,
('AMD', 'QCOM'): 45.91631724543036,
('AMD', 'QRVO'): 8.04230801921133,
('AMD', 'SWKS'): 6.032966330410986,
('AMD', 'TXN'): 15.330687075782489,
('AMD', 'XLNX'): 13.25472742746754,
('AVGO', 'INTC'): 17.86423039575638,
('AVGO', 'KLAC'): 3.004284756473002,
('AVGO', 'LRCX'): 12.793031283914337,
('AVGO', 'MCHP'): 1.234849025424745,
('AVGO', 'MU'): 27.133741892778875,
('AVGO', 'NVDA'): 26.56274085731252,
('AVGO', 'QCOM'): 45.49178117187881,
('AVGO', 'QRVO'): 5.011766778799581,
('AVGO', 'SWKS'): 1.822496536249247,
('AVGO', 'TXN'): 5.823712034045354,
('AVGO', 'XLNX'): 7.977478689436387,
('INTC', 'KLAC'): 9.50875360051205,
('INTC', 'LRCX'): 48.5941706790109,
('INTC', 'MCHP'): 13.122870250540794,
('INTC', 'MU'): 69.91548085987642,
('INTC', 'NVDA'): 67.45377571086769,
('INTC', 'QCOM'): 8.319012251502953,
('INTC', 'QRVO'): 8.822870331726378,
('INTC', 'SWKS'): 18.897228638971697,
('INTC', 'TXN'): 4.967780776727633,
('INTC', 'XLNX'): 3.595061680411969,
('KLAC', 'LRCX'): 22.425605324314212,
('KLAC', 'MCHP'): 1.59847575685719,
('KLAC', 'MU'): 39.78112631000559,
('KLAC', 'NVDA'): 41.14624070005211,
('KLAC', 'QCOM'): 30.63221134848983,
('KLAC', 'QRVO'): 1.8528020213566363,
('KLAC', 'SWKS'): 2.8486200736535103,
```

```
 ('KLAC', 'TXN'): 2.5557861098371353,
 ('KLAC', 'XLNX'): 2.9232627425123208,
 ('LRCX', 'MCHP'): 14.88119359096009,
 ('LRCX', 'MU'): 4.769782258097078,
 ('LRCX', 'NVDA'): 6.0678250636180735,
 ('LRCX', 'QCOM'): 93.26870518815028,
 ('LRCX', 'QRVO'): 29.563000037382487,
 ('LRCX', 'SWKS'): 18.63015370135768,
 ('LRCX', 'TXN'): 25.070280770539064,
 ('LRCX', 'XLNX'): 33.06343491474051,
 ('MCHP', 'MU'): 30.61626975853009,
 ('MCHP', 'NVDA'): 29.1033844279511,
 ('MCHP', 'QCOM'): 38.39356241080645,
 ('MCHP', 'QRVO'): 3.5434010166077172,
 ('MCHP', 'SWKS'): 2.457283560344462,
 ('MCHP', 'TXN'): 3.3422108791541403,
 ('MCHP', 'XLNX'): 4.531357897873351,
 ('MU', 'NVDA'): 6.555988006120443,
 ('MU', 'QCOM'): 122.20803404752033,
 ('MU', 'QRVO'): 50.213155565750945,
 ('MU', 'SWKS'): 36.15432210961284,
 ('MU', 'TXN'): 40.86788005521484,
 ('MU', 'XLNX'): 53.95390426700368,
 ('NVDA', 'QCOM'): 116.08385912068772,
 ('NVDA', 'QRVO'): 50.07346203994893,
 ('NVDA', 'SWKS'): 37.20756871720773,
 ('NVDA', 'TXN'): 39.93319524776679,
 ('NVDA', 'XLNX'): 50.390288994795405,
 ('QCOM', 'QRVO'): 26.69438443662439,
 ('QCOM', 'SWKS'): 45.28187095524986,
 ('QCOM', 'TXN'): 23.926189678303295,
 ('QCOM', 'XLNX'): 17.49680478495416,
 ('QRVO', 'SWKS'): 2.9291445432155454,
 ('QRVO', 'TXN'): 4.998382581875542,
 ('QRVO', 'XLNX'): 2.881268035181991,
 ('SWKS', 'TXN'): 8.354541060798446,
 ('SWKS', 'XLNX'): 8.205720456827613,
 ('TXN', 'XLNX'): 2.4415248210604705}
```

In [341]:
```python
# Sort dictionary by value
d = dict_pairs

# Note: Dictionaries prior to Python 3.6 (I believe,) are not ordered.
#       So returning in
list_minimized_sum_of_squared_deviations = [(k, d[k]) for k in sorted(d, key=d.get, r
list_minimized_sum_of_squared_deviations
```

Out[341]:
```
[(('ADI', 'XLNX'), 0.8472312767165687),
 (('AVGO', 'MCHP'), 1.234849025424745),
```

```
(('KLAC', 'MCHP'), 1.59847575685719),
(('AVGO', 'SWKS'), 1.822496536249247),
(('KLAC', 'QRVO'), 1.8528020213566363),
(('ADI', 'TXN'), 1.9859323681800614),
(('ADI', 'QRVO'), 2.0794069574152445),
(('ADI', 'KLAC'), 2.2691757219751896),
(('TXN', 'XLNX'), 2.4415248210604705),
(('MCHP', 'SWKS'), 2.457283560344462),
(('KLAC', 'TXN'), 2.5557861098371353),
(('KLAC', 'SWKS'), 2.8486200736535103),
(('QRVO', 'XLNX'), 2.881268035181991),
(('KLAC', 'XLNX'), 2.9232627425123208),
(('QRVO', 'SWKS'), 2.9291445432155454),
(('AVGO', 'KLAC'), 3.004284756473002),
(('MCHP', 'TXN'), 3.3422108791541403),
(('MCHP', 'QRVO'), 3.5434010166077172),
(('INTC', 'XLNX'), 3.595061680411969),
(('ADI', 'INTC'), 3.728934838622128),
(('AMAT', 'LRCX'), 3.7335933812111457),
(('AMAT', 'AVGO'), 3.7710241391412676),
(('AMAT', 'MCHP'), 4.27285769584317),
(('ADI', 'MCHP'), 4.393596211924288),
(('MCHP', 'XLNX'), 4.531357897873351),
(('LRCX', 'MU'), 4.769782258097078),
(('INTC', 'TXN'), 4.967780776727633),
(('QRVO', 'TXN'), 4.998382581875542),
(('AVGO', 'QRVO'), 5.011766778799581),
(('AVGO', 'TXN'), 5.823712034045354),
(('AMD', 'SWKS'), 6.032966330410986),
(('LRCX', 'NVDA'), 6.0678250636180735),
(('MU', 'NVDA'), 6.555988006120443),
(('ADI', 'SWKS'), 6.976968857740386),
(('ADI', 'AVGO'), 7.410060308379357),
(('AMAT', 'SWKS'), 7.672281410194768),
(('AVGO', 'XLNX'), 7.977478689436387),
(('AMD', 'QRVO'), 8.04230801921133),
(('SWKS', 'XLNX'), 8.205720456827613),
(('INTC', 'QCOM'), 8.319012251502953),
(('AMAT', 'KLAC'), 8.348660336741),
(('SWKS', 'TXN'), 8.354541060798446),
(('INTC', 'QRVO'), 8.822870331726378),
(('AMD', 'KLAC'), 8.841407821806783),
(('INTC', 'KLAC'), 9.50875360051205),
(('AMAT', 'TXN'), 10.236713850295295),
(('ADI', 'AMD'), 10.270124812815467),
(('AMD', 'MCHP'), 10.280931387897315),
(('AMD', 'AVGO'), 10.4431315585542),
(('AVGO', 'LRCX'), 12.793031283914337),
```

```
(('AMAT', 'MU'), 13.121547700573586),
(('INTC', 'MCHP'), 13.122870250540794),
(('AMD', 'XLNX'), 13.25472742746754),
(('AMAT', 'QRVO'), 13.435834239448647),
(('AMAT', 'NVDA'), 13.855923845250377),
(('LRCX', 'MCHP'), 14.88119359096009),
(('AMAT', 'XLNX'), 15.221760878707585),
(('AMD', 'TXN'), 15.330687075782489),
(('ADI', 'AMAT'), 15.409621894273716),
(('QCOM', 'XLNX'), 17.49680478495416),
(('AVGO', 'INTC'), 17.86423039575638),
(('ADI', 'QCOM'), 18.563161902462408),
(('LRCX', 'SWKS'), 18.63015370135768),
(('INTC', 'SWKS'), 18.897228638971697),
(('AMAT', 'AMD'), 20.67896462610036),
(('KLAC', 'LRCX'), 22.425605324314212),
(('QCOM', 'TXN'), 23.926189678303295),
(('AMD', 'INTC'), 24.18908167793324),
(('LRCX', 'TXN'), 25.070280770539064),
(('AVGO', 'NVDA'), 26.56274085731252),
(('QCOM', 'QRVO'), 26.69438443662439),
(('AMAT', 'INTC'), 26.80535021389236),
(('AVGO', 'MU'), 27.133741892778875),
(('MCHP', 'NVDA'), 29.1033844279511),
(('LRCX', 'QRVO'), 29.563000037382487),
(('MCHP', 'MU'), 30.61626975853009),
(('KLAC', 'QCOM'), 30.63221134848983),
(('LRCX', 'XLNX'), 33.06343491474051),
(('ADI', 'LRCX'), 33.62975145123157),
(('MU', 'SWKS'), 36.15432210961284),
(('AMD', 'LRCX'), 36.569508087369414),
(('NVDA', 'SWKS'), 37.20756871720773),
(('MCHP', 'QCOM'), 38.39356241080645),
(('KLAC', 'MU'), 39.78112631000559),
(('NVDA', 'TXN'), 39.93319524776679),
(('MU', 'TXN'), 40.86788005521484),
(('KLAC', 'NVDA'), 41.14624070005211),
(('QCOM', 'SWKS'), 45.28187095524986),
(('AVGO', 'QCOM'), 45.49178117187881),
(('AMD', 'QCOM'), 45.91631724543036),
(('INTC', 'LRCX'), 48.5941706790109),
(('NVDA', 'QRVO'), 50.07346203994893),
(('MU', 'QRVO'), 50.213155565750945),
(('NVDA', 'XLNX'), 50.390288994795405),
(('ADI', 'NVDA'), 52.403051521260295),
(('ADI', 'MU'), 53.50393464508413),
(('MU', 'XLNX'), 53.95390426700368),
(('AMD', 'MU'), 56.65870410667582),
```

```
(('AMD', 'NVDA'), 57.77420658558669),
(('AMAT', 'QCOM'), 61.494638709091895),
(('INTC', 'NVDA'), 67.45377571086769),
(('INTC', 'MU'), 69.91548085987642),
(('LRCX', 'QCOM'), 93.26870518815028),
(('NVDA', 'QCOM'), 116.08385912068772),
(('MU', 'QCOM'), 122.20803404752033)]
```

2.0.22 Minimized Sum of Squared Deviations between Main industry and Related industry - Software and Semiconductors

In [342]:
```
# https://stackoverflow.com/questions/39203662/euclidean-distance-matrix-using-panda
#pd.DataFrame( , columns = df_ticker_closing_for_semiconductors.columns, index = df_

# https://stackoverflow.com/questions/41337316/pandas-compare-all-dataframe-columns-
df_a = df_ticker_closing_for_software.copy()
df_b = df_ticker_closing_for_semiconductors.copy()

# This is a numpy array
ndarray_a = df_a.values
column_names_a = df_a.columns
#
ndarray_b = df_b.values
column_names_b = df_b.columns

# Multiplying every column by every other column
#list_for_matrix_a = []
dict_pairs = {}
for i in range(len(ndarray_a.T)):

    column_name_in_focus = column_names_a[i]
    column_values_in_focus = ndarray_a[:, i]

    # The other columns
    for j in range(len(ndarray_b.T)):

#         # Skip comparison to itself
#         if j == i:
#             continue

        column_name_other = column_names_b[j]
        column_values_other = ndarray_b[:, j]

        # These are numpy ndarrays
        ndarray_temp_a = column_values_in_focus - column_values_other
        ndarray_temp_b = np.square(ndarray_temp_a)

        sum_squared_deviations = ndarray_temp_b.sum()
```

```python
                # Put in a list so we can sort it.
                # That way, we will have unique keys in the dictionary.
                # ('ADI', 'XLNX') is the same as ('XLNX', 'ADI') in this case.
                # Yes, the value for the key will be overriden, but it would be the
                # same value.
                #
                # After convert to a tuple. No real reason I can think of except convention.
                #list_pair_key = sorted([column_name_in_focus, column_name_other])
                tuple_pair_key = (column_name_in_focus, column_name_other)

                dict_pairs[tuple_pair_key] = sum_squared_deviations
```

In [343]: dict_pairs

```
Out[343]: {('ADBE', 'ADI'): 15.641497452290585,
           ('ADBE', 'AMAT'): 1.2800980451057407,
           ('ADBE', 'AMD'): 21.141148492516677,
           ('ADBE', 'AVGO'): 3.229031816092198,
           ('ADBE', 'INTC'): 26.603769805898253,
           ('ADBE', 'KLAC'): 8.983868763624907,
           ('ADBE', 'LRCX'): 4.947437744266467,
           ('ADBE', 'MCHP'): 4.7534664617861475,
           ('ADBE', 'MU'): 14.015114483544183,
           ('ADBE', 'NVDA'): 14.090597686215157,
           ('ADBE', 'QCOM'): 60.97321968260145,
           ('ADBE', 'QRVO'): 13.91965734214309,
           ('ADBE', 'SWKS'): 8.013527852182303,
           ('ADBE', 'TXN'): 9.743496085765168,
           ('ADBE', 'XLNX'): 15.995232558511132,
           ('ADP', 'ADI'): 3.3399481439441305,
           ('ADP', 'AMAT'): 29.55695293245271,
           ('ADP', 'AMD'): 19.534769843996934,
           ('ADP', 'AVGO'): 18.21057535276634,
           ('ADP', 'INTC'): 1.513227017157409,
           ('ADP', 'KLAC'): 9.546610453832688,
           ('ADP', 'LRCX'): 52.833379866900344,
           ('ADP', 'MCHP'): 13.297060351310222,
           ('ADP', 'MU'): 77.29454456617162,
           ('ADP', 'NVDA'): 72.71541042059124,
           ('ADP', 'QCOM'): 7.667910811503187,
           ('ADP', 'QRVO'): 7.965328007054493,
           ('ADP', 'SWKS'): 17.82225497072124,
           ('ADP', 'TXN'): 6.608582231432001,
           ('ADP', 'XLNX'): 3.1367306779098536,
           ('ADSK', 'ADI'): 9.550995428644642,
           ('ADSK', 'AMAT'): 3.1305312474565037,
           ('ADSK', 'AMD'): 13.965940041283234,
```

('ADSK', 'AVGO'): 1.8915725819144973,
('ADSK', 'INTC'): 20.185431992966514,
('ADSK', 'KLAC'): 5.179947441698206,
('ADSK', 'LRCX'): 10.712682283806249,
('ADSK', 'MCHP'): 1.7208553397356319,
('ADSK', 'MU'): 26.27962743127616,
('ADSK', 'NVDA'): 22.735921675969003,
('ADSK', 'QCOM'): 48.4727188419712,
('ADSK', 'QRVO'): 6.559853702580115,
('ADSK', 'SWKS'): 3.47605743380632,
('ADSK', 'TXN'): 8.245649450023652,
('ADSK', 'XLNX'): 8.688691070489895,
('AKAM', 'ADI'): 36.80614763565199,
('AKAM', 'AMAT'): 95.54572746289244,
('AKAM', 'AMD'): 65.99217019899946,
('AKAM', 'AVGO'): 74.57691760935943,
('AKAM', 'INTC'): 22.18151190769331,
('AKAM', 'KLAC'): 53.65149329006273,
('AKAM', 'LRCX'): 134.67021174233633,
('AKAM', 'MCHP'): 64.97435841816045,
('AKAM', 'MU'): 168.02243780617792,
('AKAM', 'NVDA'): 162.55528571903338,
('AKAM', 'QCOM'): 5.397042405948808,
('AKAM', 'QRVO'): 48.090892214536396,
('AKAM', 'SWKS'): 71.71002966893637,
('AKAM', 'TXN'): 45.15239789357456,
('AKAM', 'XLNX'): 36.77129923927621,
('ANSS', 'ADI'): 11.705766345503283,
('ANSS', 'AMAT'): 1.8386290662095512,
('ANSS', 'AMD'): 18.06570273178554,
('ANSS', 'AVGO'): 1.5230308747745935,
('ANSS', 'INTC'): 21.74741223760421,
('ANSS', 'KLAC'): 5.995556859729725,
('ANSS', 'LRCX'): 7.936326621368339,
('ANSS', 'MCHP'): 2.971177160670713,
('ANSS', 'MU'): 19.198469575621022,
('ANSS', 'NVDA'): 18.617339876833864,
('ANSS', 'QCOM'): 52.42964327275603,
('ANSS', 'QRVO'): 9.963992834470323,
('ANSS', 'SWKS'): 5.708933420619753,
('ANSS', 'TXN'): 7.2347660308521835,
('ANSS', 'XLNX'): 11.830091672421048,
('ATVI', 'ADI'): 33.61372729988623,
('ATVI', 'AMAT'): 7.122805862797942,
('ATVI', 'AMD'): 28.10278074203699,
('ATVI', 'AVGO'): 10.584802766357736,
('ATVI', 'INTC'): 53.19064463347061,
('ATVI', 'KLAC'): 21.551867317891215,

('ATVI', 'LRCX'): 4.964722388916429,
('ATVI', 'MCHP'): 14.488160438511093,
('ATVI', 'MU'): 12.654621517069684,
('ATVI', 'NVDA'): 12.450756584249177,
('ATVI', 'QCOM'): 96.58766474067346,
('ATVI', 'QRVO'): 27.41276718406751,
('ATVI', 'SWKS'): 14.613960720782337,
('ATVI', 'TXN'): 27.81844331361608,
('ATVI', 'XLNX'): 33.92204891825767,
('CA', 'ADI'): 5.445553025342331,
('CA', 'AMAT'): 36.49507269746307,
('CA', 'AMD'): 20.73605408420914,
('CA', 'AVGO'): 22.746763651316854,
('CA', 'INTC'): 3.445512618519095,
('CA', 'KLAC'): 12.489029290514786,
('CA', 'LRCX'): 62.2387275194252,
('CA', 'MCHP'): 17.589221451485724,
('CA', 'MU'): 89.04651743061828,
('CA', 'NVDA'): 84.39443333912233,
('CA', 'QCOM'): 5.947465216980399,
('CA', 'QRVO'): 9.818295962133433,
('CA', 'SWKS'): 21.009716682953986,
('CA', 'TXN'): 10.752124544886751,
('CA', 'XLNX'): 4.962389858707814,
('CDNS', 'ADI'): 20.64316526927612,
('CDNS', 'AMAT'): 1.4409675059472304,
('CDNS', 'AMD'): 22.751348532462618,
('CDNS', 'AVGO'): 5.161811092239464,
('CDNS', 'INTC'): 34.19982518433737,
('CDNS', 'KLAC'): 12.486160085666137,
('CDNS', 'LRCX'): 2.786202384120178,
('CDNS', 'MCHP'): 7.015705353059896,
('CDNS', 'MU'): 10.003787537836825,
('CDNS', 'NVDA'): 11.428964086685394,
('CDNS', 'QCOM'): 72.89573389114403,
('CDNS', 'QRVO'): 17.888684971584798,
('CDNS', 'SWKS'): 9.810187459426341,
('CDNS', 'TXN'): 14.4872169329341,
('CDNS', 'XLNX'): 21.40032723887533,
('CRM', 'ADI'): 1.997656016697197,
('CRM', 'AMAT'): 7.6193332473852555,
('CRM', 'AMD'): 11.724097423669896,
('CRM', 'AVGO'): 2.6179215903960937,
('CRM', 'INTC'): 7.446315317339748,
('CRM', 'KLAC'): 1.3086781125145386,
('CRM', 'LRCX'): 20.941885192615864,
('CRM', 'MCHP'): 1.20869169961224,
('CRM', 'MU'): 37.57257377897888,

```
('CRM', 'NVDA'): 36.19835380367905,
('CRM', 'QCOM'): 28.24636354741675,
('CRM', 'QRVO'): 2.6604460084357497,
('CRM', 'SWKS'): 4.313964227859302,
('CRM', 'TXN'): 1.0856977608231528,
('CRM', 'XLNX'): 2.2146502472736547,
('CTXS', 'ADI'): 1.7111418190556513,
('CTXS', 'AMAT'): 24.355743725844047,
('CTXS', 'AMD'): 14.091955561162326,
('CTXS', 'AVGO'): 13.109353439176544,
('CTXS', 'INTC'): 2.7353165635824075,
('CTXS', 'KLAC'): 5.217057967076028,
('CTXS', 'LRCX'): 46.31669601666305,
('CTXS', 'MCHP'): 10.00839196776698,
('CTXS', 'MU'): 68.9365041143938,
('CTXS', 'NVDA'): 68.519632570599,
('CTXS', 'QCOM'): 11.870586808569865,
('CTXS', 'QRVO'): 4.297245266774303,
('CTXS', 'SWKS'): 11.917604880251496,
('CTXS', 'TXN'): 4.868710949397263,
('CTXS', 'XLNX'): 2.368944712068714,
('EA', 'ADI'): 10.632559611184648,
('EA', 'AMAT'): 5.67038492363221,
('EA', 'AMD'): 11.369723342340796,
('EA', 'AVGO'): 2.226087984222831,
('EA', 'INTC'): 23.2365422708407,
('EA', 'KLAC'): 5.603884866043956,
('EA', 'LRCX'): 13.922809590464553,
('EA', 'MCHP'): 2.381116610056907,
('EA', 'MU'): 31.16913827811288,
('EA', 'NVDA'): 26.652673080545902,
('EA', 'QCOM'): 51.25816088996295,
('EA', 'QRVO'): 7.080377861068116,
('EA', 'SWKS'): 3.0018709650252138,
('EA', 'TXN'): 10.291513338129036,
('EA', 'XLNX'): 9.964181785582714,
('EBAY', 'ADI'): 0.6669700061807349,
('EBAY', 'AMAT'): 16.369847413421255,
('EBAY', 'AMD'): 9.470492419121397,
('EBAY', 'AVGO'): 7.3766072626476955,
('EBAY', 'INTC'): 4.844847353878416,
('EBAY', 'KLAC'): 2.5157133544305084,
('EBAY', 'LRCX'): 34.66476798597868,
('EBAY', 'MCHP'): 4.4358271633060635,
('EBAY', 'MU'): 55.78390231749456,
('EBAY', 'NVDA'): 53.122504526800135,
('EBAY', 'QCOM'): 19.3054523972676,
('EBAY', 'QRVO'): 2.277695702694797,
```

```
('EBAY', 'SWKS'): 6.773082305246333,
('EBAY', 'TXN'): 3.0875436703530985,
('EBAY', 'XLNX'): 0.9172979723184711,
('FB', 'ADI'): 4.310549648198803,
('FB', 'AMAT'): 5.355019798959194,
('FB', 'AMD'): 11.381937533598368,
('FB', 'AVGO'): 1.3819220813233803,
('FB', 'INTC'): 11.62931972616743,
('FB', 'KLAC'): 2.2368750663683565,
('FB', 'LRCX'): 16.302038846448006,
('FB', 'MCHP'): 0.6189008071597084,
('FB', 'MU'): 32.511727562217175,
('FB', 'NVDA'): 29.91482373000176,
('FB', 'QCOM'): 35.70497897079473,
('FB', 'QRVO'): 4.048544380735673,
('FB', 'SWKS'): 3.4153550499609855,
('FB', 'TXN'): 2.877096600387234,
('FB', 'XLNX'): 4.22918299708272,
('FIS', 'ADI'): 0.9721460842565901,
('FIS', 'AMAT'): 18.883272204506873,
('FIS', 'AMD'): 13.099156992395116,
('FIS', 'AVGO'): 9.65874003260073,
('FIS', 'INTC'): 2.948774328347077,
('FIS', 'KLAC'): 4.200469503274639,
('FIS', 'LRCX'): 38.08838038691995,
('FIS', 'MCHP'): 6.127472044001941,
('FIS', 'MU'): 60.3922441819914,
('FIS', 'NVDA'): 56.141610858453,
('FIS', 'QCOM'): 15.050310033931911,
('FIS', 'QRVO'): 3.4764696998930686,
('FIS', 'SWKS'): 9.631628850034886,
('FIS', 'TXN'): 3.241467900080252,
('FIS', 'XLNX'): 0.6465886000734158,
('FISV', 'ADI'): 0.8126773010290321,
('FISV', 'AMAT'): 19.123786755704163,
('FISV', 'AMD'): 12.297532416462612,
('FISV', 'AVGO'): 9.366093956502034,
('FISV', 'INTC'): 3.22238329364667,
('FISV', 'KLAC'): 3.5291831245484273,
('FISV', 'LRCX'): 38.787350179774954,
('FISV', 'MCHP'): 6.276913817659961,
('FISV', 'MU'): 61.0502194903199,
('FISV', 'NVDA'): 58.00668154104851,
('FISV', 'QCOM'): 14.854789074063332,
('FISV', 'QRVO'): 2.8842466377820197,
('FISV', 'SWKS'): 8.990832337676242,
('FISV', 'TXN'): 3.326089525756251,
('FISV', 'XLNX'): 0.7275039920196817,
```

('GOOG', 'ADI'): 1.1283331094134528,
('GOOG', 'AMAT'): 12.387602835021475,
('GOOG', 'AMD'): 13.05014283097869,
('GOOG', 'AVGO'): 5.433618853208897,
('GOOG', 'INTC'): 4.572231565891804,
('GOOG', 'KLAC'): 1.7678068660222688,
('GOOG', 'LRCX'): 28.827379638643123,
('GOOG', 'MCHP'): 3.263660244777946,
('GOOG', 'MU'): 48.13536028587864,
('GOOG', 'NVDA'): 45.37886778456493,
('GOOG', 'QCOM'): 20.56211397569321,
('GOOG', 'QRVO'): 2.6492552874095923,
('GOOG', 'SWKS'): 6.6086921281843525,
('GOOG', 'TXN'): 1.399497980887719,
('GOOG', 'XLNX'): 0.6938522819778221,
('GOOGL', 'ADI'): 0.9991659273680565,
('GOOGL', 'AMAT'): 13.6722252827118,
('GOOGL', 'AMD'): 13.200857208214854,
('GOOGL', 'AVGO'): 6.221321566606663,
('GOOGL', 'INTC'): 4.051446129603078,
('GOOGL', 'KLAC'): 2.0612747556407918,
('GOOGL', 'LRCX'): 30.80466655990735,
('GOOGL', 'MCHP'): 3.865271615523935,
('GOOGL', 'MU'): 50.74060670955242,
('GOOGL', 'NVDA'): 47.857521548210244,
('GOOGL', 'QCOM'): 19.031407198249152,
('GOOGL', 'QRVO'): 2.6353344814422806,
('GOOGL', 'SWKS'): 7.124572235900465,
('GOOGL', 'TXN'): 1.6840029125995648,
('GOOGL', 'XLNX'): 0.5245130065173396,
('INTU', 'ADI'): 1.751296011542319,
('INTU', 'AMAT'): 9.586137473219715,
('INTU', 'AMD'): 13.539439140899896,
('INTU', 'AVGO'): 4.071252478781896,
('INTU', 'INTC'): 5.927393857176819,
('INTU', 'KLAC'): 1.9821576343612868,
('INTU', 'LRCX'): 24.105139605998502,
('INTU', 'MCHP'): 2.022870270040184,
('INTU', 'MU'): 41.5971311228647,
('INTU', 'NVDA'): 38.44431668259838,
('INTU', 'QCOM'): 24.550801037955857,
('INTU', 'QRVO'): 3.299520483719693,
('INTU', 'SWKS'): 6.186094457908639,
('INTU', 'TXN'): 1.0660962802519576,
('INTU', 'XLNX'): 1.4400592840000366,
('MA', 'ADI'): 2.935080982147751,
('MA', 'AMAT'): 7.618397641012901,
('MA', 'AMD'): 15.096972970089018,

```
('MA', 'AVGO'): 4.074457220782927,
('MA', 'INTC'): 7.175343403904266,
('MA', 'KLAC'): 2.778926360724334,
('MA', 'LRCX'): 20.448689367332328,
('MA', 'MCHP'): 1.7521076086846996,
('MA', 'MU'): 36.66557007389974,
('MA', 'NVDA'): 33.3619426658736,
('MA', 'QCOM'): 27.85879720917311,
('MA', 'QRVO'): 4.865818609764919,
('MA', 'SWKS'): 6.9578593037274254,
('MA', 'TXN'): 1.0191323715355078,
('MA', 'XLNX'): 2.4274168830795517,
('MSFT', 'ADI'): 1.8803364815253043,
('MSFT', 'AMAT'): 9.217085500885672,
('MSFT', 'AMD'): 14.129795736113532,
('MSFT', 'AVGO'): 4.122631744688235,
('MSFT', 'INTC'): 5.500999430446283,
('MSFT', 'KLAC'): 2.053888970457047,
('MSFT', 'LRCX'): 23.43085212265243,
('MSFT', 'MCHP'): 2.220780241314992,
('MSFT', 'MU'): 40.37932016642564,
('MSFT', 'NVDA'): 37.92775234110368,
('MSFT', 'QCOM'): 24.666532088192184,
('MSFT', 'QRVO'): 3.7170582173486455,
('MSFT', 'SWKS'): 6.482207717384746,
('MSFT', 'TXN'): 0.5759653537096523,
('MSFT', 'XLNX'): 1.75269278207508,
('NFLX', 'ADI'): 4.17577528613352,
('NFLX', 'AMAT'): 6.076876469767402,
('NFLX', 'AMD'): 14.343703256756674,
('NFLX', 'AVGO'): 3.4856634117804353,
('NFLX', 'INTC'): 9.802561115281504,
('NFLX', 'KLAC'): 3.0468563044287507,
('NFLX', 'LRCX'): 17.660767634811247,
('NFLX', 'MCHP'): 1.4663045721183574,
('NFLX', 'MU'): 33.9442862602299,
('NFLX', 'NVDA'): 29.983751208738926,
('NFLX', 'QCOM'): 32.61305961899776,
('NFLX', 'QRVO'): 5.465229138847734,
('NFLX', 'SWKS'): 6.08352444983263,
('NFLX', 'TXN'): 1.9400265053862158,
('NFLX', 'XLNX'): 3.6519196840191097,
('NTAP', 'ADI'): 6.955848572814682,
('NTAP', 'AMAT'): 9.534159756510446,
('NTAP', 'AMD'): 18.791851847270415,
('NTAP', 'AVGO'): 6.141731750548153,
('NTAP', 'INTC'): 11.313983355275052,
('NTAP', 'KLAC'): 5.622385025555119,
```

```
('NTAP', 'LRCX'): 21.859420192094536,
('NTAP', 'MCHP'): 5.934168240322908,
('NTAP', 'MU'): 33.359871632141434,
('NTAP', 'NVDA'): 34.49513158308581,
('NTAP', 'QCOM'): 34.06382149429426,
('NTAP', 'QRVO'): 9.900726491281485,
('NTAP', 'SWKS'): 10.95204457640072,
('NTAP', 'TXN'): 2.4964700617261997,
('NTAP', 'XLNX'): 8.196438205872427,
('ORCL', 'ADI'): 2.237859080294177,
('ORCL', 'AMAT'): 11.36725985738814,
('ORCL', 'AMD'): 7.744808769555508,
('ORCL', 'AVGO'): 3.628280724780087,
('ORCL', 'INTC'): 9.127402312076551,
('ORCL', 'KLAC'): 1.9666625510388887,
('ORCL', 'LRCX'): 26.3380801544425,
('ORCL', 'MCHP'): 2.3649409870550215,
('ORCL', 'MU'): 46.32812837336788,
('ORCL', 'NVDA'): 43.57909500699902,
('ORCL', 'QCOM'): 27.748789920419252,
('ORCL', 'QRVO'): 2.092417050465725,
('ORCL', 'SWKS'): 3.4639988164450752,
('ORCL', 'TXN'): 3.8927966052404903,
('ORCL', 'XLNX'): 2.3265285796604918,
('PAYX', 'ADI'): 5.273245677432621,
('PAYX', 'AMAT'): 35.66660509503371,
('PAYX', 'AMD'): 24.34207539586413,
('PAYX', 'AVGO'): 23.43063128659174,
('PAYX', 'INTC'): 1.4694648674078978,
('PAYX', 'KLAC'): 12.674199363197296,
('PAYX', 'LRCX'): 61.19587107091768,
('PAYX', 'MCHP'): 17.994460326854544,
('PAYX', 'MU'): 85.7109809119928,
('PAYX', 'NVDA'): 82.68973605434279,
('PAYX', 'QCOM'): 4.575606421432423,
('PAYX', 'QRVO'): 10.843215219243742,
('PAYX', 'SWKS'): 22.880415869794682,
('PAYX', 'TXN'): 9.008339362042072,
('PAYX', 'XLNX'): 5.247483808508269,
('RHT', 'ADI'): 18.009576455547833,
('RHT', 'AMAT'): 1.3902993520505271,
('RHT', 'AMD'): 23.50427092505474,
('RHT', 'AVGO'): 5.235385460682404,
('RHT', 'INTC'): 28.99042315171751,
('RHT', 'KLAC'): 11.462190235741506,
('RHT', 'LRCX'): 3.9347514482060673,
('RHT', 'MCHP'): 6.109404247131482,
('RHT', 'MU'): 11.193967715622387,
```

('RHT', 'NVDA'): 11.629685197164665,
('RHT', 'QCOM'): 65.23114304984585,
('RHT', 'QRVO'): 17.13480950272999,
('RHT', 'SWKS'): 10.794234160554456,
('RHT', 'TXN'): 11.531388438403074,
('RHT', 'XLNX'): 18.163783871728363,
('SNPS', 'ADI'): 4.665030522378919,
('SNPS', 'AMAT'): 4.2111509211811295,
('SNPS', 'AMD'): 10.63366920282201,
('SNPS', 'AVGO'): 0.9195603825634844,
('SNPS', 'INTC'): 13.039591357224342,
('SNPS', 'KLAC'): 1.7330298854192816,
('SNPS', 'LRCX'): 14.612695806918683,
('SNPS', 'MCHP'): 0.5986578446501867,
('SNPS', 'MU'): 28.968260437077348,
('SNPS', 'NVDA'): 29.40447854573659,
('SNPS', 'QCOM'): 38.778548066039626,
('SNPS', 'QRVO'): 4.085032654272124,
('SNPS', 'SWKS'): 2.766974459468903,
('SNPS', 'TXN'): 2.9403693350312694,
('SNPS', 'XLNX'): 5.42967222082347,
('SYMC', 'ADI'): 2.2853977924112274,
('SYMC', 'AMAT'): 22.871462838930405,
('SYMC', 'AMD'): 9.402337134118063,
('SYMC', 'AVGO'): 11.911479971425239,
('SYMC', 'INTC'): 6.557964616085342,
('SYMC', 'KLAC'): 4.687166171826333,
('SYMC', 'LRCX'): 43.40338461552502,
('SYMC', 'MCHP'): 8.299938487398158,
('SYMC', 'MU'): 67.68017770371281,
('SYMC', 'NVDA'): 66.75494128562804,
('SYMC', 'QCOM'): 18.49612606499751,
('SYMC', 'QRVO'): 3.1271746259948285,
('SYMC', 'SWKS'): 8.6933873801736,
('SYMC', 'TXN'): 7.075507223722063,
('SYMC', 'XLNX'): 2.766619347766248,
('TSS', 'ADI'): 3.6725300573839,
('TSS', 'AMAT'): 9.131078389941319,
('TSS', 'AMD'): 17.737619708919453,
('TSS', 'AVGO'): 5.100992735044828,
('TSS', 'INTC'): 6.8381829885920435,
('TSS', 'KLAC'): 3.96344097435693,
('TSS', 'LRCX'): 22.172386708097605,
('TSS', 'MCHP'): 2.929578489308252,
('TSS', 'MU'): 38.311821236585786,
('TSS', 'NVDA'): 34.25145354298562,
('TSS', 'QCOM'): 26.827964552473546,
('TSS', 'QRVO'): 6.234948323089551,

('TSS', 'SWKS'): 8.900870412229509,
('TSS', 'TXN'): 1.1467625205748755,
('TSS', 'XLNX'): 3.167828834732221,
('V', 'ADI'): 2.517895089161541,
('V', 'AMAT'): 6.996183783066659,
('V', 'AMD'): 11.675134951725434,
('V', 'AVGO'): 2.4480915503312657,
('V', 'INTC'): 8.405913278598494,
('V', 'KLAC'): 1.6613098801675854,
('V', 'LRCX'): 19.694890290000835,
('V', 'MCHP'): 0.9116461786887379,
('V', 'MU'): 36.16409052659907,
('V', 'NVDA'): 33.86405899286408,
('V', 'QCOM'): 30.091350717807885,
('V', 'QRVO'): 3.423067256173825,
('V', 'SWKS'): 4.4877857185558785,
('V', 'TXN'): 1.2447344545367391,
('V', 'XLNX'): 2.652737851307526,
('VRSN', 'ADI'): 3.06579512064087,
('VRSN', 'AMAT'): 6.7414531805595095,
('VRSN', 'AMD'): 12.83430263635474,
('VRSN', 'AVGO'): 2.738503779686982,
('VRSN', 'INTC'): 8.449443787143437,
('VRSN', 'KLAC'): 2.1684262007897592,
('VRSN', 'LRCX'): 19.08250883933981,
('VRSN', 'MCHP'): 1.2180349493319742,
('VRSN', 'MU'): 35.06684092556682,
('VRSN', 'NVDA'): 32.907270434259715,
('VRSN', 'QCOM'): 30.26595065139195,
('VRSN', 'QRVO'): 4.0625107553148485,
('VRSN', 'SWKS'): 5.213495731194438,
('VRSN', 'TXN'): 1.5696648850894575,
('VRSN', 'XLNX'): 2.8965990255744365,
('WU', 'ADI'): 12.83749513294439,
('WU', 'AMAT'): 53.43619825071505,
('WU', 'AMD'): 33.18207365023986,
('WU', 'AVGO'): 37.30204098086463,
('WU', 'INTC'): 6.444176386626403,
('WU', 'KLAC'): 23.33664694696038,
('WU', 'LRCX'): 83.75062683089261,
('WU', 'MCHP'): 30.394019960623993,
('WU', 'MU'): 113.82915436034669,
('WU', 'NVDA'): 108.20750814688978,
('WU', 'QCOM'): 2.0307105617280126,
('WU', 'QRVO'): 19.073607208988577,
('WU', 'SWKS'): 34.76551732113953,
('WU', 'TXN'): 19.515134819031534,
('WU', 'XLNX'): 12.255954726885394}

```
In [344]: # Sort dictionary by value
          d = dict_pairs

          # Note: Dictionaries prior to Python 3.6 (I believe,) are not ordered.
          #       So returning in
          list_minimized_sum_of_squared_deviations = [(k, d[k]) for k in sorted(d, key=d.get, r
          list_minimized_sum_of_squared_deviations

Out[344]: [(('GOOGL', 'XLNX'), 0.5245130065173396),
           (('MSFT', 'TXN'), 0.5759653537096523),
           (('SNPS', 'MCHP'), 0.5986578446501867),
           (('FB', 'MCHP'), 0.6189008071597084),
           (('FIS', 'XLNX'), 0.6465886000734158),
           (('EBAY', 'ADI'), 0.6669700061807349),
           (('GOOG', 'XLNX'), 0.6938522819778221),
           (('FISV', 'XLNX'), 0.7275039920196817),
           (('FISV', 'ADI'), 0.8126773010290321),
           (('V', 'MCHP'), 0.9116461786887379),
           (('EBAY', 'XLNX'), 0.9172979723184711),
           (('SNPS', 'AVGO'), 0.9195603825634844),
           (('FIS', 'ADI'), 0.9721460842565901),
           (('GOOGL', 'ADI'), 0.9991659273680565),
           (('MA', 'TXN'), 1.0191323715355078),
           (('INTU', 'TXN'), 1.0660962802519576),
           (('CRM', 'TXN'), 1.0856977608231528),
           (('GOOG', 'ADI'), 1.1283331094134528),
           (('TSS', 'TXN'), 1.1467625205748755),
           (('CRM', 'MCHP'), 1.20869169961224),
           (('VRSN', 'MCHP'), 1.2180349493319742),
           (('V', 'TXN'), 1.2447344545367391),
           (('ADBE', 'AMAT'), 1.2800980451057407),
           (('CRM', 'KLAC'), 1.3086781125145386),
           (('FB', 'AVGO'), 1.3819220813233803),
           (('RHT', 'AMAT'), 1.3902993520505271),
           (('GOOG', 'TXN'), 1.399497980887719),
           (('INTU', 'XLNX'), 1.4400592840000366),
           (('CDNS', 'AMAT'), 1.4409675059472304),
           (('NFLX', 'MCHP'), 1.4663045721183574),
           (('PAYX', 'INTC'), 1.4694648674078978),
           (('ADP', 'INTC'), 1.513227017157409),
           (('ANSS', 'AVGO'), 1.5230308747745935),
           (('VRSN', 'TXN'), 1.5696648850894575),
           (('V', 'KLAC'), 1.6613098801675854),
           (('GOOGL', 'TXN'), 1.6840029125995648),
           (('CTXS', 'ADI'), 1.7111418190556513),
           (('ADSK', 'MCHP'), 1.7208553397356319),
           (('SNPS', 'KLAC'), 1.7330298854192816),
           (('INTU', 'ADI'), 1.751296011542319),
```

```
(('MA', 'MCHP'), 1.7521076086846996),
(('MSFT', 'XLNX'), 1.75269278207508),
(('GOOG', 'KLAC'), 1.7678068660222688),
(('ANSS', 'AMAT'), 1.8386290662095512),
(('MSFT', 'ADI'), 1.8803364815253043),
(('ADSK', 'AVGO'), 1.8915725819144973),
(('NFLX', 'TXN'), 1.9400265053862158),
(('ORCL', 'KLAC'), 1.9666625510388887),
(('INTU', 'KLAC'), 1.9821576343612868),
(('CRM', 'ADI'), 1.997656016697197),
(('INTU', 'MCHP'), 2.022870270040184),
(('WU', 'QCOM'), 2.0307105617280126),
(('MSFT', 'KLAC'), 2.053888970457047),
(('GOOGL', 'KLAC'), 2.0612747556407918),
(('ORCL', 'QRVO'), 2.092417050465725),
(('VRSN', 'KLAC'), 2.1684262007897592),
(('CRM', 'XLNX'), 2.2146502472736547),
(('MSFT', 'MCHP'), 2.220780241314992),
(('EA', 'AVGO'), 2.226087984222831),
(('FB', 'KLAC'), 2.2368750663683565),
(('ORCL', 'ADI'), 2.237859080294177),
(('EBAY', 'QRVO'), 2.277695702694797),
(('SYMC', 'ADI'), 2.2853977924112274),
(('ORCL', 'XLNX'), 2.3265285796604918),
(('ORCL', 'MCHP'), 2.3649409870550215),
(('CTXS', 'XLNX'), 2.368944712068714),
(('EA', 'MCHP'), 2.381116610056907),
(('MA', 'XLNX'), 2.4274168830795517),
(('V', 'AVGO'), 2.4480915503312657),
(('NTAP', 'TXN'), 2.4964700617261997),
(('EBAY', 'KLAC'), 2.5157133544305084),
(('V', 'ADI'), 2.517895089161541),
(('CRM', 'AVGO'), 2.6179215903960937),
(('GOOGL', 'QRVO'), 2.6353344814422806),
(('GOOG', 'QRVO'), 2.6492552874095923),
(('V', 'XLNX'), 2.652737851307526),
(('CRM', 'QRVO'), 2.6604460084357497),
(('CTXS', 'INTC'), 2.7353165635824075),
(('VRSN', 'AVGO'), 2.738503779686982),
(('SYMC', 'XLNX'), 2.766619347766248),
(('SNPS', 'SWKS'), 2.766974459468903),
(('MA', 'KLAC'), 2.778926360724334),
(('CDNS', 'LRCX'), 2.786202384120178),
(('FB', 'TXN'), 2.877096600387234),
(('FISV', 'QRVO'), 2.8842466377820197),
(('VRSN', 'XLNX'), 2.8965990255744365),
(('TSS', 'MCHP'), 2.929578489308252),
(('MA', 'ADI'), 2.935080982147751),
```

```
(('SNPS', 'TXN'), 2.9403693350312694),
(('FIS', 'INTC'), 2.948774328347077),
(('ANSS', 'MCHP'), 2.971177160670713),
(('EA', 'SWKS'), 3.0018709650252138),
(('NFLX', 'KLAC'), 3.0468563044287507),
(('VRSN', 'ADI'), 3.06579512064087),
(('EBAY', 'TXN'), 3.0875436703530985),
(('SYMC', 'QRVO'), 3.1271746259948285),
(('ADSK', 'AMAT'), 3.1305312474565037),
(('ADP', 'XLNX'), 3.1367306779098536),
(('TSS', 'XLNX'), 3.167828834732221),
(('FISV', 'INTC'), 3.22238329364667),
(('ADBE', 'AVGO'), 3.229031816092198),
(('FIS', 'TXN'), 3.241467900080252),
(('GOOG', 'MCHP'), 3.263660244777946),
(('INTU', 'QRVO'), 3.299520483719693),
(('FISV', 'TXN'), 3.326089525756251),
(('ADP', 'ADI'), 3.3399481439441305),
(('FB', 'SWKS'), 3.4153550499609855),
(('V', 'QRVO'), 3.423067256173825),
(('CA', 'INTC'), 3.445512618519095),
(('ORCL', 'SWKS'), 3.4639988164450752),
(('ADSK', 'SWKS'), 3.47605743380632),
(('FIS', 'QRVO'), 3.4764696998930686),
(('NFLX', 'AVGO'), 3.4856634117804353),
(('FISV', 'KLAC'), 3.5291831245484273),
(('ORCL', 'AVGO'), 3.628280724780087),
(('NFLX', 'XLNX'), 3.6519196840191097),
(('TSS', 'ADI'), 3.6725300573839),
(('MSFT', 'QRVO'), 3.7170582173486455),
(('GOOGL', 'MCHP'), 3.865271615523935),
(('ORCL', 'TXN'), 3.8927966052404903),
(('RHT', 'LRCX'), 3.9347514482060673),
(('TSS', 'KLAC'), 3.96344097435693),
(('FB', 'QRVO'), 4.048544380735673),
(('GOOGL', 'INTC'), 4.051446129603078),
(('VRSN', 'QRVO'), 4.0625107553148485),
(('INTU', 'AVGO'), 4.071252478781896),
(('MA', 'AVGO'), 4.074457220782927),
(('SNPS', 'QRVO'), 4.085032654272124),
(('MSFT', 'AVGO'), 4.122631744688235),
(('NFLX', 'ADI'), 4.17577528613352),
(('FIS', 'KLAC'), 4.200469503274639),
(('SNPS', 'AMAT'), 4.2111509211811295),
(('FB', 'XLNX'), 4.22918299708272),
(('CTXS', 'QRVO'), 4.297245266774303),
(('FB', 'ADI'), 4.310549648198803),
(('CRM', 'SWKS'), 4.313964227859302),
```

```
(('EBAY', 'MCHP'), 4.4358271633060635),
(('V', 'SWKS'), 4.4877857185558785),
(('GOOG', 'INTC'), 4.572231565891804),
(('PAYX', 'QCOM'), 4.575606421432423),
(('SNPS', 'ADI'), 4.665030522378919),
(('SYMC', 'KLAC'), 4.687166171826333),
(('ADBE', 'MCHP'), 4.7534664617861475),
(('EBAY', 'INTC'), 4.844847353878416),
(('MA', 'QRVO'), 4.865818609764919),
(('CTXS', 'TXN'), 4.868710949397263),
(('ADBE', 'LRCX'), 4.947437744266467),
(('CA', 'XLNX'), 4.962389858707814),
(('ATVI', 'LRCX'), 4.964722388916429),
(('TSS', 'AVGO'), 5.100992735044828),
(('CDNS', 'AVGO'), 5.161811092239464),
(('ADSK', 'KLAC'), 5.179947441698206),
(('VRSN', 'SWKS'), 5.213495731194438),
(('CTXS', 'KLAC'), 5.217057967076028),
(('RHT', 'AVGO'), 5.235385460682404),
(('PAYX', 'XLNX'), 5.247483808508269),
(('PAYX', 'ADI'), 5.273245677432621),
(('FB', 'AMAT'), 5.355019798959194),
(('AKAM', 'QCOM'), 5.397042405948808),
(('SNPS', 'XLNX'), 5.42967222082347),
(('GOOG', 'AVGO'), 5.433618853208897),
(('CA', 'ADI'), 5.445553025342331),
(('NFLX', 'QRVO'), 5.465229138847734),
(('MSFT', 'INTC'), 5.500999430446283),
(('EA', 'KLAC'), 5.603884866043956),
(('NTAP', 'KLAC'), 5.622385025555119),
(('EA', 'AMAT'), 5.67038492363221),
(('ANSS', 'SWKS'), 5.708933420619753),
(('INTU', 'INTC'), 5.927393857176819),
(('NTAP', 'MCHP'), 5.934168240322908),
(('CA', 'QCOM'), 5.947465216980399),
(('ANSS', 'KLAC'), 5.995556859729725),
(('NFLX', 'AMAT'), 6.076876469767402),
(('NFLX', 'SWKS'), 6.08352444983263),
(('RHT', 'MCHP'), 6.109404247131482),
(('FIS', 'MCHP'), 6.127472044001941),
(('NTAP', 'AVGO'), 6.141731750548153),
(('INTU', 'SWKS'), 6.186094457908639),
(('GOOGL', 'AVGO'), 6.221321566606663),
(('TSS', 'QRVO'), 6.234948323089551),
(('FISV', 'MCHP'), 6.276913817659961),
(('WU', 'INTC'), 6.444176386626403),
(('MSFT', 'SWKS'), 6.482207717384746),
(('SYMC', 'INTC'), 6.557964616085342),
```

```
(('ADSK', 'QRVO'), 6.559853702580115),
(('ADP', 'TXN'), 6.608582231432001),
(('GOOG', 'SWKS'), 6.6086921281843525),
(('VRSN', 'AMAT'), 6.7414531805595095),
(('EBAY', 'SWKS'), 6.773082305246333),
(('TSS', 'INTC'), 6.8381829885920435),
(('NTAP', 'ADI'), 6.955848572814682),
(('MA', 'SWKS'), 6.9578593037274254),
(('V', 'AMAT'), 6.996183783066659),
(('CDNS', 'MCHP'), 7.015705353059896),
(('SYMC', 'TXN'), 7.075507223722063),
(('EA', 'QRVO'), 7.080377861068116),
(('ATVI', 'AMAT'), 7.122805862797942),
(('GOOGL', 'SWKS'), 7.124572235900465),
(('MA', 'INTC'), 7.175343403904266),
(('ANSS', 'TXN'), 7.2347660308521835),
(('EBAY', 'AVGO'), 7.3766072626476955),
(('CRM', 'INTC'), 7.446315317339748),
(('MA', 'AMAT'), 7.618397641012901),
(('CRM', 'AMAT'), 7.6193332473852555),
(('ADP', 'QCOM'), 7.667910811503187),
(('ORCL', 'AMD'), 7.744808769555508),
(('ANSS', 'LRCX'), 7.936326621368339),
(('ADP', 'QRVO'), 7.965328007054493),
(('ADBE', 'SWKS'), 8.013527852182303),
(('NTAP', 'XLNX'), 8.196438205872427),
(('ADSK', 'TXN'), 8.245649450023652),
(('SYMC', 'MCHP'), 8.299938487398158),
(('V', 'INTC'), 8.405913278598494),
(('VRSN', 'INTC'), 8.449443787143437),
(('ADSK', 'XLNX'), 8.688691070489895),
(('SYMC', 'SWKS'), 8.6933873801736),
(('TSS', 'SWKS'), 8.900870412229509),
(('ADBE', 'KLAC'), 8.983868763624907),
(('FISV', 'SWKS'), 8.990832337676242),
(('PAYX', 'TXN'), 9.008339362042072),
(('ORCL', 'INTC'), 9.127402312076551),
(('TSS', 'AMAT'), 9.131078389941319),
(('MSFT', 'AMAT'), 9.217085500885672),
(('FISV', 'AVGO'), 9.366093956502034),
(('SYMC', 'AMD'), 9.402337134118063),
(('EBAY', 'AMD'), 9.470492419121397),
(('NTAP', 'AMAT'), 9.534159756510446),
(('ADP', 'KLAC'), 9.546610453832688),
(('ADSK', 'ADI'), 9.550995428644642),
(('INTU', 'AMAT'), 9.586137473219715),
(('FIS', 'SWKS'), 9.631628850034886),
(('FIS', 'AVGO'), 9.65874003260073),
```

```
(('ADBE', 'TXN'), 9.743496085765168),
(('NFLX', 'INTC'), 9.802561115281504),
(('CDNS', 'SWKS'), 9.810187459426341),
(('CA', 'QRVO'), 9.818295962133433),
(('NTAP', 'QRVO'), 9.900726491281485),
(('ANSS', 'QRVO'), 9.963992834470323),
(('EA', 'XLNX'), 9.964181785582714),
(('CDNS', 'MU'), 10.003787537836825),
(('CTXS', 'MCHP'), 10.00839196776698),
(('EA', 'TXN'), 10.291513338129036),
(('ATVI', 'AVGO'), 10.584802766357736),
(('EA', 'ADI'), 10.632559611184648),
(('SNPS', 'AMD'), 10.63366920282201),
(('ADSK', 'LRCX'), 10.712682283806249),
(('CA', 'TXN'), 10.752124544886751),
(('RHT', 'SWKS'), 10.794234160554456),
(('PAYX', 'QRVO'), 10.843215219243742),
(('NTAP', 'SWKS'), 10.95204457640072),
(('RHT', 'MU'), 11.193967715622387),
(('NTAP', 'INTC'), 11.313983355275052),
(('ORCL', 'AMAT'), 11.36725985738814),
(('EA', 'AMD'), 11.369723342340796),
(('FB', 'AMD'), 11.381937533598368),
(('CDNS', 'NVDA'), 11.428964086685394),
(('RHT', 'KLAC'), 11.462190235741506),
(('RHT', 'TXN'), 11.531388438403074),
(('FB', 'INTC'), 11.62931972616743),
(('RHT', 'NVDA'), 11.629685197164665),
(('V', 'AMD'), 11.675134951725434),
(('ANSS', 'ADI'), 11.705766345503283),
(('CRM', 'AMD'), 11.724097423669896),
(('ANSS', 'XLNX'), 11.830091672421048),
(('CTXS', 'QCOM'), 11.870586808569865),
(('SYMC', 'AVGO'), 11.911479971425239),
(('CTXS', 'SWKS'), 11.917604880251496),
(('WU', 'XLNX'), 12.255954726885394),
(('FISV', 'AMD'), 12.297532416462612),
(('GOOG', 'AMAT'), 12.387602835021475),
(('ATVI', 'NVDA'), 12.450756584249177),
(('CDNS', 'KLAC'), 12.486160085666137),
(('CA', 'KLAC'), 12.489029290514786),
(('ATVI', 'MU'), 12.654621517069684),
(('PAYX', 'KLAC'), 12.674199363197296),
(('VRSN', 'AMD'), 12.83430263635474),
(('WU', 'ADI'), 12.83749513294439),
(('SNPS', 'INTC'), 13.039591357224342),
(('GOOG', 'AMD'), 13.05014283097869),
(('FIS', 'AMD'), 13.099156992395116),
```

```
(('CTXS', 'AVGO'), 13.109353439176544),
(('GOOGL', 'AMD'), 13.200857208214854),
(('ADP', 'MCHP'), 13.297060351310222),
(('INTU', 'AMD'), 13.539439140899896),
(('GOOGL', 'AMAT'), 13.6722252827118),
(('ADBE', 'QRVO'), 13.91965734214309),
(('EA', 'LRCX'), 13.922809590464553),
(('ADSK', 'AMD'), 13.965940041283234),
(('ADBE', 'MU'), 14.015114483544183),
(('ADBE', 'NVDA'), 14.090597686215157),
(('CTXS', 'AMD'), 14.091955561162326),
(('MSFT', 'AMD'), 14.129795736113532),
(('NFLX', 'AMD'), 14.343703256756674),
(('CDNS', 'TXN'), 14.4872169329341),
(('ATVI', 'MCHP'), 14.488160438511093),
(('SNPS', 'LRCX'), 14.612695806918683),
(('ATVI', 'SWKS'), 14.613960720782337),
(('FISV', 'QCOM'), 14.854789074063332),
(('FIS', 'QCOM'), 15.050310033931911),
(('MA', 'AMD'), 15.096972970089018),
(('ADBE', 'ADI'), 15.641497452290585),
(('ADBE', 'XLNX'), 15.995232558511132),
(('FB', 'LRCX'), 16.302038846448006),
(('EBAY', 'AMAT'), 16.369847413421255),
(('RHT', 'QRVO'), 17.13480950272999),
(('CA', 'MCHP'), 17.589221451485724),
(('NFLX', 'LRCX'), 17.660767634811247),
(('TSS', 'AMD'), 17.737619708919453),
(('ADP', 'SWKS'), 17.82225497072124),
(('CDNS', 'QRVO'), 17.888684971584798),
(('PAYX', 'MCHP'), 17.994460326854544),
(('RHT', 'ADI'), 18.009576455547833),
(('ANSS', 'AMD'), 18.06570273178554),
(('RHT', 'XLNX'), 18.163783871728363),
(('ADP', 'AVGO'), 18.21057535276634),
(('SYMC', 'QCOM'), 18.49612606499751),
(('ANSS', 'NVDA'), 18.617339876833864),
(('NTAP', 'AMD'), 18.791851847270415),
(('FIS', 'AMAT'), 18.883272204506873),
(('GOOGL', 'QCOM'), 19.031407198249152),
(('WU', 'QRVO'), 19.073607208988577),
(('VRSN', 'LRCX'), 19.08250883933981),
(('FISV', 'AMAT'), 19.123786755704163),
(('ANSS', 'MU'), 19.198469575621022),
(('EBAY', 'QCOM'), 19.3054523972676),
(('WU', 'TXN'), 19.515134819031534),
(('ADP', 'AMD'), 19.534769843996934),
(('V', 'LRCX'), 19.694890290000835),
```

```
(('ADSK', 'INTC'), 20.185431992966514),
(('MA', 'LRCX'), 20.448689367332328),
(('GOOG', 'QCOM'), 20.56211397569321),
(('CDNS', 'ADI'), 20.64316526927612),
(('CA', 'AMD'), 20.73605408420914),
(('CRM', 'LRCX'), 20.941885192615864),
(('CA', 'SWKS'), 21.009716682953986),
(('ADBE', 'AMD'), 21.141148492516677),
(('CDNS', 'XLNX'), 21.40032723887533),
(('ATVI', 'KLAC'), 21.551867317891215),
(('ANSS', 'INTC'), 21.74741223760421),
(('NTAP', 'LRCX'), 21.859420192094536),
(('TSS', 'LRCX'), 22.172386708097605),
(('AKAM', 'INTC'), 22.18151190769331),
(('ADSK', 'NVDA'), 22.735921675969003),
(('CA', 'AVGO'), 22.746763651316854),
(('CDNS', 'AMD'), 22.751348532462618),
(('SYMC', 'AMAT'), 22.871462838930405),
(('PAYX', 'SWKS'), 22.880415869794682),
(('EA', 'INTC'), 23.2365422708407),
(('WU', 'KLAC'), 23.33664694696038),
(('PAYX', 'AVGO'), 23.43063128659174),
(('MSFT', 'LRCX'), 23.43085212265243),
(('RHT', 'AMD'), 23.50427092505474),
(('INTU', 'LRCX'), 24.105139605998502),
(('PAYX', 'AMD'), 24.34207539586413),
(('CTXS', 'AMAT'), 24.355743725844047),
(('INTU', 'QCOM'), 24.550801037955857),
(('MSFT', 'QCOM'), 24.666532088192184),
(('ADSK', 'MU'), 26.27962743127616),
(('ORCL', 'LRCX'), 26.3380801544425),
(('ADBE', 'INTC'), 26.603769805898253),
(('EA', 'NVDA'), 26.652673080545902),
(('TSS', 'QCOM'), 26.827964552473546),
(('ATVI', 'QRVO'), 27.41276718406751),
(('ORCL', 'QCOM'), 27.748789920419252),
(('ATVI', 'TXN'), 27.81844331361608),
(('MA', 'QCOM'), 27.85879720917311),
(('ATVI', 'AMD'), 28.10278074203699),
(('CRM', 'QCOM'), 28.24636354741675),
(('GOOG', 'LRCX'), 28.827379638643123),
(('SNPS', 'MU'), 28.968260437077348),
(('RHT', 'INTC'), 28.99042315171751),
(('SNPS', 'NVDA'), 29.40447854573659),
(('ADP', 'AMAT'), 29.55695293245271),
(('FB', 'NVDA'), 29.91482373000176),
(('NFLX', 'NVDA'), 29.983751208738926),
(('V', 'QCOM'), 30.091350717807885),
```

```
(('VRSN', 'QCOM'), 30.26595065139195),
(('WU', 'MCHP'), 30.394019960623993),
(('GOOGL', 'LRCX'), 30.80466655990735),
(('EA', 'MU'), 31.16913827811288),
(('FB', 'MU'), 32.511727562217175),
(('NFLX', 'QCOM'), 32.61305961899776),
(('VRSN', 'NVDA'), 32.907270434259715),
(('WU', 'AMD'), 33.18207365023986),
(('NTAP', 'MU'), 33.359871632141434),
(('MA', 'NVDA'), 33.3619426658736),
(('ATVI', 'ADI'), 33.61372729988623),
(('V', 'NVDA'), 33.86405899286408),
(('ATVI', 'XLNX'), 33.92204891825767),
(('NFLX', 'MU'), 33.9442862602299),
(('NTAP', 'QCOM'), 34.06382149429426),
(('CDNS', 'INTC'), 34.19982518433737),
(('TSS', 'NVDA'), 34.25145354298562),
(('NTAP', 'NVDA'), 34.49513158308581),
(('EBAY', 'LRCX'), 34.66476798597868),
(('WU', 'SWKS'), 34.76551732113953),
(('VRSN', 'MU'), 35.06684092556682),
(('PAYX', 'AMAT'), 35.66660509503371),
(('FB', 'QCOM'), 35.70497897079473),
(('V', 'MU'), 36.16409052659907),
(('CRM', 'NVDA'), 36.19835380367905),
(('CA', 'AMAT'), 36.49507269746307),
(('MA', 'MU'), 36.66557007389974),
(('AKAM', 'XLNX'), 36.77129923927621),
(('AKAM', 'ADI'), 36.80614763565199),
(('WU', 'AVGO'), 37.30204098086463),
(('CRM', 'MU'), 37.57257377897888),
(('MSFT', 'NVDA'), 37.92775234110368),
(('FIS', 'LRCX'), 38.08838038691995),
(('TSS', 'MU'), 38.311821236585786),
(('INTU', 'NVDA'), 38.44431668259838),
(('SNPS', 'QCOM'), 38.778548066039626),
(('FISV', 'LRCX'), 38.787350179774954),
(('MSFT', 'MU'), 40.37932016642564),
(('INTU', 'MU'), 41.5971311228647),
(('SYMC', 'LRCX'), 43.40338461552502),
(('ORCL', 'NVDA'), 43.57909500699902),
(('AKAM', 'TXN'), 45.15239789357456),
(('GOOG', 'NVDA'), 45.37886778456493),
(('CTXS', 'LRCX'), 46.31669601666305),
(('ORCL', 'MU'), 46.32812837336788),
(('GOOGL', 'NVDA'), 47.857521548210244),
(('AKAM', 'QRVO'), 48.090892214536396),
(('GOOG', 'MU'), 48.13536028587864),
```

```
(('ADSK', 'QCOM'), 48.4727188419712),
(('GOOGL', 'MU'), 50.74060670955242),
(('EA', 'QCOM'), 51.25816088996295),
(('ANSS', 'QCOM'), 52.42964327275603),
(('ADP', 'LRCX'), 52.833379866900344),
(('EBAY', 'NVDA'), 53.122504526800135),
(('ATVI', 'INTC'), 53.19064463347061),
(('WU', 'AMAT'), 53.43619825071505),
(('AKAM', 'KLAC'), 53.65149329006273),
(('EBAY', 'MU'), 55.78390231749456),
(('FIS', 'NVDA'), 56.141610858453),
(('FISV', 'NVDA'), 58.00668154104851),
(('FIS', 'MU'), 60.3922441819914),
(('ADBE', 'QCOM'), 60.97321968260145),
(('FISV', 'MU'), 61.0502194903199),
(('PAYX', 'LRCX'), 61.19587107091768),
(('CA', 'LRCX'), 62.2387275194252),
(('AKAM', 'MCHP'), 64.97435841816045),
(('RHT', 'QCOM'), 65.23114304984585),
(('AKAM', 'AMD'), 65.99217019899946),
(('SYMC', 'NVDA'), 66.75494128562804),
(('SYMC', 'MU'), 67.68017770371281),
(('CTXS', 'NVDA'), 68.519632570599),
(('CTXS', 'MU'), 68.9365041143938),
(('AKAM', 'SWKS'), 71.71002966893637),
(('ADP', 'NVDA'), 72.71541042059124),
(('CDNS', 'QCOM'), 72.89573389114403),
(('AKAM', 'AVGO'), 74.57691760935943),
(('ADP', 'MU'), 77.29454456617162),
(('PAYX', 'NVDA'), 82.68973605434279),
(('WU', 'LRCX'), 83.75062683089261),
(('CA', 'NVDA'), 84.39443333912233),
(('PAYX', 'MU'), 85.7109809119928),
(('CA', 'MU'), 89.04651743061828),
(('AKAM', 'AMAT'), 95.54572746289244),
(('ATVI', 'QCOM'), 96.58766474067346),
(('WU', 'NVDA'), 108.20750814688978),
(('WU', 'MU'), 113.82915436034669),
(('AKAM', 'LRCX'), 134.67021174233633),
(('AKAM', 'NVDA'), 162.55528571903338),
(('AKAM', 'MU'), 168.02243780617792)]
```

```
In [347]: len(list_minimized_sum_of_squared_deviations)
```

```
Out[347]: 465
```

2.0.23 Excess Return Computation

Because pairs may open and close at various points during the six-month trading period, the calculation of the excess return on a portfolio of pairs is a non-trivial issue. Pairs that open and converge during the trading interval will have positive cash flows. Because pairs can reopen after initial convergence, they can have multiple positive cash flows during the trading interval. Pairs that open but do not converge will only have cash flows on the last day of the trading interval when all positions are closed out. Therefore, the payoffs to pairs trading strategies are a set of positive cash flows that are randomly distributed throughout the trading period, and a set of cash flows at the end of the trading interval which can either be positive or negative.

Main Industry (Software) ((’GOOG’, ’GOOGL’), 0.04045301814067839) #### Main Industry and Related Industry (Software and Semiconductors) ((’GOOGL’, ’XLNX’), 0.5245130065173396)

2.0.24 Get the daily returns for these industries (GOOG, GOOGL, XLNX)

```
In [116]: # Get just the column for the normalized price (which is of type Pandas Series)
          # and convert to Pandas DataFrame.
          df_goog = df_ticker_closing_for_software['GOOG'].to_frame()
          df_googl = df_ticker_closing_for_software['GOOGL'].to_frame()
          df_xlnx = df_ticker_closing_for_semiconductors['XLNX'].to_frame()

          # Change the names of the columns.
          df_goog.columns = ['normalized_price']
          df_googl.columns = ['normalized_price']
          df_xlnx.columns = ['normalized_price']

          # Calculate the mean average
          df_goog_mean_average = df_goog['normalized_price'].mean()
          df_googl_mean_average = df_goog['normalized_price'].mean()
          df_xlnx_mean_average = df_goog['normalized_price'].mean()

          # daily returns
          # https://www.fool.com/knowledge-center/how-to-convert-daily-returns-to-annual-retur
          # amount
          df_goog['daily_return'] = (df_goog['normalized_price'] / df_goog['normalized_price']
          df_googl['daily_return'] = (df_googl['normalized_price'] / df_googl['normalized_pric
          df_xlnx['daily_return'] = (df_xlnx['normalized_price'] / df_xlnx['normalized_price']

          # Average return
          df_goog_average_return = df_goog['daily_return'].mean()
          df_googl_average_return = df_googl['daily_return'].mean()
          df_xlnx_average_return = df_xlnx['daily_return'].mean()

          # Standard Deviation of return
          goog_std_return= df_goog['daily_return'].std()
          googl_std_return = df_googl['daily_return'].std()
          xlnx_std_return = df_xlnx['daily_return'].std()
```

```
# Get the mean of pairs
#
# Main Industry pairs
mean_pair_goog_googl = (df_goog['normalized_price'] + df_googl['normalized_price']).m
mean_pair_googl_googl = (df_googl['normalized_price'] + df_goog['normalized_price'])
#
# Related Industry pairs
mean_pair_googl_xlnx = (df_googl['normalized_price'] + df_xlnx['normalized_price']).m
mean_pair_xlnx_googl = (df_xlnx['normalized_price'] + df_googl['normalized_price']).m

# Get the standard deviation between pairs
#
# Main Industry pairs
std_pair_goog_googl = (df_goog['normalized_price'] + df_googl['normalized_price']).st
std_pair_googl_googl = (df_googl['normalized_price'] + df_goog['normalized_price']).s
#
# Related Industry pairs
std_pair_googl_xlnx = (df_googl['normalized_price'] + df_xlnx['normalized_price']).st
std_pair_xlnx_googl = (df_xlnx['normalized_price'] + df_googl['normalized_price']).st

# Find whenever the normalized price was 2 times over the standard deviation.
#df_temp_g = df_xlnx[np.abs(df_xlnx['normalized_price'] - mean_pair_googl_xlnx) > 2.

# Holding period return
df_goog_holding_period_return =  (df_goog.iloc[-1] / df_goog.iloc[0]) - 1
df_googl_holding_period_return =  (df_googl.iloc[-1] / df_googl.iloc[0]) - 1
df_xlnx_holding_period_return =  (df_xlnx.iloc[-1] / df_xlnx.iloc[0]) - 1

# Sharpe measure
#
# Sharpe measure is on total risk.
# Sharpe Raito = (Rp - Rf)/std
# Rp => Average return
# Rf => Risk free rate
# std => standard deviation
#
# https://www.investopedia.com/terms/r/risk-freerate.asp#ixzz55B5Vc4ly
# the interest rate on a three-month U.S. Treasury bill is often used as the risk-fr
#
# Risk free rate: https://ycharts.com/indicators/3_month_t_bill
Rp = df_goog_average_return
Rf = 0.0133
std = goog_std_return
sharpe_goog = (Rp - Rf) / std
#
Rp = df_googl_average_return
Rf = 0.0133
```

```
std = googl_std_return
sharpe_googl = (Rp - Rf) / std
#
Rp = df_xlnx_average_return
Rf = 0.0133
std = xlnx_std_return
sharpe_xlnx = (Rp - Rf) / std

# Treynor measure
#
# Similar to Sharpe except it only considers systematic risks
# not total risk.
#
# Treynor Raito = (Rp - Rf)/Beta
# Rp => Average return
# Rf => Risk free rate
# Beta => Beta
#
# https://www.investopedia.com/terms/r/risk-freerate.asp#ixzz55B5Vc4ly
# the interest rate on a three-month U.S. Treasury bill is often used as the risk-fr
#
# Risk free rate: https://ycharts.com/indicators/3_month_t_bill
#
# Beta - https://finance.yahoo.com/quote/GOOG?p=GOOG
#        https://finance.yahoo.com/quote/GOOGL?p=GOOGL
Rp = df_goog_average_return
Rf = 0.0133
Beta = 1.04
treynor_goog = (Rp - Rf) / Beta
#
Rp = df_googl_average_return
Rf = 0.0133
Beta = 1.01
treynor_googl = (Rp - Rf) / Beta
#
Rp = df_goog_average_return
Rf = 0.0133
Beta = 0.88
treynor_xlnx = (Rp - Rf) / Beta


# Jensen's Measure
#
# Excess returns
# alpha = Rp - [Rf - Beta(Rm - Rf)]
# Rm is Market return. The market is the S&P 500 so using ticker symbol
# SPX which follows the S&P 500.
df_spx['daily_return'] = (df_spx['Adj Close'] / (df_spx['Adj Close'][1]))
```

```
df_spx_average_return = df_spx['daily_return'].mean()
#
Rp = df_goog_average_return
Rf = 0.0133
Beta = 0.88
Rm = df_spx_average_return
goog_jensen_alpha = Rp - (Rf - Beta*(Rm - Rf))
#
Rp = df_googl_average_return
Rf = 0.0133
Beta = 0.88
Rm = df_spx_average_return
googl_jensen_alpha = Rp - (Rf - Beta*(Rm - Rf))
#
Rp = df_xlnx_average_return
Rf = 0.0133
Beta = 0.88
Rm = df_spx_average_return
xlnx_jensen_alpha = Rp - (Rf - Beta*(Rm - Rf))
```

```
In [117]: goog_jensen_alpha
```

```
Out[117]: 0.882301658875199
```

```
In [118]: googl_jensen_alpha
```

```
Out[118]: 0.8719089729169318
```

```
In [119]: xlnx_jensen_alpha
```

```
Out[119]: 0.8544309763096805
```

2.0.25 DEBUG

```
In [59]: df_goog_holding_period_return
```

```
Out[59]: normalized_price    0.418032
         daily_return             inf
         dtype: float64
```

```
In [53]: df_googl_holding_period_return
```

```
Out[53]: normalized_price    0.38712
         dtype: float64
```

```
In [54]: df_xlnx_holding_period_return
```

```
Out[54]: normalized_price    0.32766
         dtype: float64
```

```
In [66]: sharpe_goog

Out[66]: 1.4782398792731872

In [68]: sharpe_googl

Out[68]: 1.4620040368557

In [69]: sharpe_xlnx

Out[69]: 1.2894166002250353

In [71]: treynor_goog

Out[71]: 0.14717192208828575

In [72]: treynor_googl

Out[72]: 0.14125357724113863

In [73]: treynor_xlnx

Out[73]: 0.17393045337706498
```