FIN3210 Week 3 Assignment

Ma Kexuan 120090651

October 5, 2023

[1]:
```python
import pandas as pd
import numpy as np
from linearmodels import PanelOLS
import matplotlib.pyplot as plt
from matplotlib.pyplot import MultipleLocator
```

[2]:
```python
df = pd.read_excel('FIN3210 Week 3 Stock returns.xlsx', sheet_name='data')
```

[3]:
```python
data = df.drop(['stknme','conme'], axis=1).copy()
data['month'] = pd.to_datetime(data['month'], format='%Y-%m').dt.to_period('M')␣
 ↪# Transfer to datetime type with month based
data['year_quarter'] = data['month'].dt.strftime('%Y-Q%q') # Transfer to format␣
 ↪year-quarter
data.head()
```

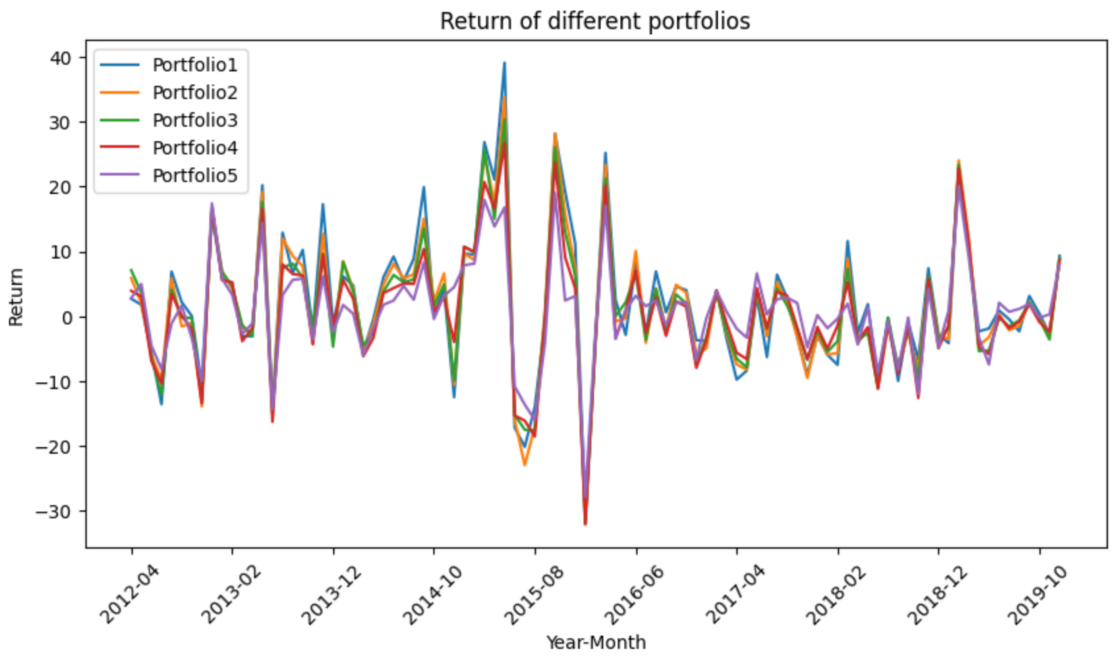
[3]:
```
   stkcd    month    retrf   mktrf     smb     hml     umd      size  \
0      9  2013-07   9.0756  1.8833  5.1896 -0.1922  6.1829  23.10871
1      9  2013-08  -5.1363  4.5833  5.9407  0.0976 -5.5041  23.10871
2      9  2013-09   3.5840  3.3833  0.5848  2.8554  8.8693  23.10871
3      9  2014-01  -1.0932 -3.6167  5.6921  1.1126  8.9357  23.28204
4      9  2014-02  19.0704  0.9833  3.8899  0.6902 -1.8764  23.28204

         bm  return12       roa       lev       ppe    intang  numanalyst  \
0  0.438860  13.37412  0.007949  0.430605  0.086185  0.031575           0
1  0.438860  13.37412  0.007949  0.430605  0.086185  0.031575           0
2  0.438860  13.37412  0.007949  0.430605  0.086185  0.031575           0
3  0.394303  46.59159  0.025973  0.432246  0.098660  0.029798           0
4  0.394303  46.59159  0.025973  0.432246  0.098660  0.029798           0

   instown           mv year_quarter
0   6.4646  11874173952      2013-Q3
1   6.4646  11874173952      2013-Q3
2   6.4646  11874173952      2013-Q3
3   5.6741  11731143680      2014-Q1
4   5.6741  11731143680      2014-Q1
```

0.1 1) Using the data set of stock returns, sort stocks into quintiles by size every quarter, hold stocks over the quarter, and calculate monthly portfolio returns

[4]:
```
# Divide into 5 groups
data['size_label'] = data.groupby('year_quarter')['size'].transform(lambda x:␣
↪pd.qcut(x, 5, labels=[1,2,3,4,5]))
# Use the size of last month to predict next month
data['last_size_label'] = data.groupby(['stkcd','year_quarter'])['size_label'].
↪shift(1)
# fill in NAN value
data.loc[data['last_size_label'].isnull(),'last_size_label'] = data.
↪loc[data['last_size_label'].isnull(),'size_label']
# Equal weighted result of portfolio return
port_res = data.groupby(['month','last_size_label'])['retrf'].mean().
↪reset_index()
port_res.head(10)
```

[4]:
```
     month last_size_label     retrf
0  2012-04               1  2.688905
1  2012-04               2  5.861874
2  2012-04               3  7.099519
3  2012-04               4  3.920090
4  2012-04               5  2.800216
5  2012-05               1  1.778694
6  2012-05               2  2.056422
7  2012-05               3  3.618288
8  2012-05               4  3.030542
9  2012-05               5  4.962478
```

[5]:
```
port_res['month'] = port_res['month'].astype(str)
plt.figure(figsize = (10,5))
for i in [1,2,3,4,5]:
    plt.plot(port_res.loc[port_res['last_size_label']==i, 'month'],
            port_res.loc[port_res['last_size_label']==i, 'retrf'],
            label = 'Portfolio{}'.format(i))
plt.xlabel('Year-Month')
plt.ylabel('Return')
plt.title('Return of different portfolios')
plt.xticks(rotation = 45)
x_major_locator=MultipleLocator(10)
ax = plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.legend()
plt.show()
```

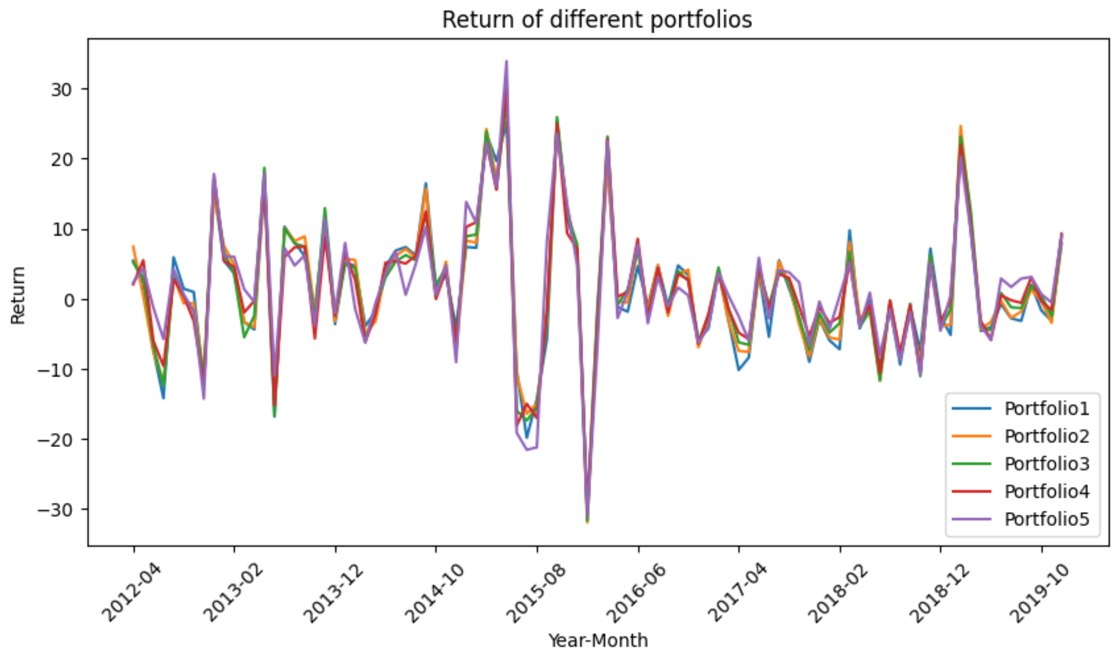


0.2 2) Using the data set of stock returns, sort stocks into quintiles by institutional ownership every quarter, hold stocks over the quarter, and calculate monthly portfolio returns

[6]:
```
data['inst_label'] = data.groupby('year_quarter')['instown'].transform(lambda x:
  ↪ pd.qcut(x, 5, labels=[1,2,3,4,5]))
data['last_inst_label'] = data.groupby(['stkcd','year_quarter'])['inst_label'].
  ↪shift(1)
data.loc[data['last_inst_label'].isnull(),'last_inst_label'] = data.
  ↪loc[data['last_inst_label'].isnull(),'inst_label']
port_res = data.groupby(['month','last_inst_label'])['retrf'].mean().
  ↪reset_index()
port_res.head(10)
```

[6]:
```
     month last_inst_label     retrf
0  2012-04               1  5.411529
1  2012-04               2  7.414722
2  2012-04               3  5.176190
3  2012-04               4  2.049471
4  2012-04               5  2.287461
5  2012-05               1  2.237273
6  2012-05               2  0.351318
7  2012-05               3  3.046009
8  2012-05               4  5.482549
9  2012-05               5  4.313619
```

```
[7]: port_res['month'] = port_res['month'].astype(str)
     plt.figure(figsize = (10,5))
     for i in [1,2,3,4,5]:
         plt.plot(port_res.loc[port_res['last_inst_label']==i, 'month'],
                  port_res.loc[port_res['last_inst_label']==i, 'retrf'],
                  label = 'Portfolio{}'.format(i))
     plt.xlabel('Year-Month')
     plt.ylabel('Return')
     plt.title('Return of different portfolios')
     plt.xticks(rotation = 45)
     x_major_locator=MultipleLocator(10)
     ax = plt.gca()
     ax.xaxis.set_major_locator(x_major_locator)
     plt.legend()
     plt.show()
```



0.3 3) Using the data set of stock returns, perform panel regression, and regress stock returns on firm characteristics such as size, book-to-market ratio, return12, roa, leverage, ppe, intang, number of analysts, institutional ownership, controlling for or not for firm and year-month fixed effects. Cluster standard errors by firm and year-month (double clustering)

[8]:
```
panel_data = data[['stkcd','month','retrf','size','bm','return12',
                   'roa','lev','ppe','intang','numanalyst','instown']].copy()
panel_data[['size','bm','return12','roa','lev','ppe','intang','numanalyst','instown']]␣
  ↪= panel_data.groupby('stkcd')[['size',
             'bm','return12','roa','lev','ppe','intang','numanalyst','instown']].
  ↪shift(1)
panel_data.dropna(inplace = True)
panel_data
```

[8]:
```
        stkcd    month    retrf      size        bm   return12       roa  \
1           9  2013-08  -5.1363  23.10871  0.438860  13.374120  0.007949
2           9  2013-09   3.5840  23.10871  0.438860  13.374120  0.007949
3           9  2014-01  -1.0932  23.10871  0.438860  13.374120  0.007949
4           9  2014-02  19.0704  23.28204  0.394303  46.591590  0.025973
5           9  2014-03  -5.9711  23.28204  0.394303  46.591590  0.025973
…         …       …        …         …         …          …
87427  900956  2019-08  -5.3373  19.54512  5.124264  -4.492293  0.005443
87428  900956  2019-09  11.0761  19.54512  5.124264  -4.492293  0.005443
87429  900956  2019-10  12.3759  19.54512  5.124264  -4.492293  0.005443
87430  900956  2019-11   0.5954  19.41050  5.950344 -14.776860  0.010888
87431  900956  2019-12   7.2701  19.41050  5.950344 -14.776860  0.010888

            lev       ppe    intang  numanalyst  instown
1      0.430605  0.086185  0.031575         0.0   6.4646
2      0.430605  0.086185  0.031575         0.0   6.4646
3      0.430605  0.086185  0.031575         0.0   6.4646
4      0.432246  0.098660  0.029798         0.0   5.6741
5      0.432246  0.098660  0.029798         0.0   5.6741
…          …         …         …          …         …
87427  0.598459  0.267594  0.025879         0.0   0.0000
87428  0.598459  0.267594  0.025879         0.0   0.0000
87429  0.598459  0.267594  0.025879         0.0   0.0000
87430  0.425125  0.269671  0.025286         0.0   0.0000
87431  0.425125  0.269671  0.025286         0.0   0.0000

[85397 rows x 12 columns]
```

[9]:
```
panel_data['month'] = pd.to_numeric(panel_data['month'].dt.strftime('%Y%m'))
panel_data.set_index(['stkcd','month'], inplace=True) # Control for firm and␣
  ↪year-month fixed effects
model = PanelOLS(panel_data['retrf'], panel_data[['size','bm','return12','roa',
        'lev','ppe','intang','numanalyst','instown']], entity_effects=True,␣
 ↪time_effects=True)
res = model.fit(cov_type='clustered', cluster_entity=True, cluster_time=True) #␣
 ↪Cluster standard errors
res.summary
```

```
[9]: <class 'linearmodels.compat.statsmodels.Summary'>
"""
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:                  retrf   R-squared:                        0.0133
Estimator:                   PanelOLS   R-squared (Between):             -1228.5
No. Observations:               85397   R-squared (Within):               0.0171
Date:                Thu, Oct 05 2023   R-squared (Overall):             -45.250
Time:                        15:22:52   Log-likelihood                  -3.2e+05
Cov. Estimator:             Clustered
                                        F-statistic:                      124.83
Entities:                        2035   P-value                           0.0000
Avg Obs:                       41.964   Distribution:                 F(9,83262)
Min Obs:                       2.0000
Max Obs:                       92.000   F-statistic (robust):             14.436
                                        P-value                           0.0000
Time periods:                      92   Distribution:                 F(9,83262)
Avg Obs:                       928.23
Min Obs:                       377.00
Max Obs:                       1825.0

                             Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
size          -4.0216     0.4473    -8.9917     0.0000     -4.8983     -3.1450
bm             0.2309     0.8461     0.2729     0.7849     -1.4274      1.8893
return12      -0.0023     0.0030    -0.7629     0.4455     -0.0081      0.0036
roa            4.6914     3.1983     1.4668     0.1424     -1.5772      10.960
lev           -1.3608     0.6284    -2.1655     0.0304     -2.5925     -0.1291
ppe           -0.0489     1.0345    -0.0473     0.9623     -2.0765      1.9788
intang         3.0578     2.5080     1.2192     0.2228     -1.8579      7.9736
numanalyst     0.0113     0.0163     0.6901     0.4901     -0.0207      0.0432
instown        0.0086     0.0160     0.5349     0.5927     -0.0228      0.0399
==============================================================================

F-test for Poolability: 28.922
P-value: 0.0000
Distribution: F(2125,83262)

Included effects: Entity, Time
```

```
"""
```

[10]:
```python
model = PanelOLS(panel_data['retrf'], panel_data[['size','bm','return12',
        'roa','lev','ppe','intang','numanalyst','instown']],
        entity_effects=False, time_effects=False) # Not control for these two␣
↪effects
# Cluster standard errors
res = model.fit(cov_type='clustered', cluster_entity=True, cluster_time=True)
res.summary
```

[10]:
```
<class 'linearmodels.compat.statsmodels.Summary'>
"""
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:                  retrf   R-squared:                        0.0062
Estimator:                   PanelOLS   R-squared (Between):              0.0578
No. Observations:               85397   R-squared (Within):               0.0027
Date:                Thu, Oct 05 2023   R-squared (Overall):              0.0062
Time:                        15:22:53   Log-likelihood                -3.438e+05
Cov. Estimator:             Clustered
                                        F-statistic:                      59.151
Entities:                        2035   P-value                           0.0000
Avg Obs:                       41.964   Distribution:                F(9,85388)
Min Obs:                       2.0000
Max Obs:                       92.000   F-statistic (robust):             2.1815
                                        P-value                           0.0203
Time periods:                      92   Distribution:                F(9,85388)
Avg Obs:                       928.23
Min Obs:                       377.00
Max Obs:                       1825.0

                             Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
size          -0.0107     0.0508    -0.2100     0.8337     -0.1102      0.0888
bm             1.4906     0.8773     1.6991     0.0893     -0.2288      3.2101
return12       0.0036     0.0141     0.2555     0.7983     -0.0241      0.0313
roa            5.3103     7.6867     0.6908     0.4897     -9.7556      20.376
lev            0.5546     1.8904     0.2934     0.7692     -3.1504      4.2597
ppe            0.1788     1.2655     0.1413     0.8877     -2.3016      2.6592
intang         0.3477     1.2780     0.2721     0.7855     -2.1571      2.8526
numanalyst    -0.0084     0.0157    -0.5371     0.5912     -0.0392      0.0224
instown        0.0436     0.0165     2.6450     0.0082      0.0113      0.0759
==============================================================================
"""
```

0.4 4) Using the data set of Online sales, aggregate monthly online sales over quarters, download reported quarterly total sales from CSMAR, and plot figures including both online sales and reported quarterly sales.

[11]:
```
sales = pd.read_excel('FIN3210 Week 3 Online sales.xlsx', sheet_name='  ')
# Change the columns into rows in the dataframe
sales = sales.melt(id_vars=[' '], var_name='Brand', value_name='online_sales')
sales.rename({' ':'month'}, axis=1, inplace=True)
sales
```

[11]:
```
          month          Brand  online_sales
0   2016-01-01     603899.SH    3888967.80
1   2016-02-01     603899.SH    3983190.45
2   2016-03-01     603899.SH    6395686.99
3   2016-04-01     603899.SH    4968614.43
4   2016-05-01     603899.SH    6566980.86
..         …            …             …
545 2020-03-01     002956.SZ    8621983.82
546 2020-04-01     002956.SZ    8476046.53
547 2020-05-01     002956.SZ    8540964.70
548 2020-06-01     002956.SZ   11706599.62
549 2020-07-01     002956.SZ    6767988.81

[550 rows x 3 columns]
```

[12]:
```
sales['month'] = pd.to_datetime(sales['month'], format='%Y-%m').dt.
  ↪to_period('M')
sales['year_quarter'] = sales['month'].dt.strftime('%Y-Q%q')
sales['stkcd'] = sales['Brand'].str.extract('(\d+)').astype('int') # Extract␣
  ↪stock id
sales
```

[12]:
```
       month          Brand  online_sales year_quarter   stkcd
0    2016-01     603899.SH    3888967.80      2016-Q1  603899
1    2016-02     603899.SH    3983190.45      2016-Q1  603899
2    2016-03     603899.SH    6395686.99      2016-Q1  603899
3    2016-04     603899.SH    4968614.43      2016-Q2  603899
4    2016-05     603899.SH    6566980.86      2016-Q2  603899
..       …            …             …           …       …
545  2020-03     002956.SZ    8621983.82      2020-Q1    2956
546  2020-04     002956.SZ    8476046.53      2020-Q2    2956
547  2020-05     002956.SZ    8540964.70      2020-Q2    2956
548  2020-06     002956.SZ   11706599.62      2020-Q2    2956
549  2020-07     002956.SZ    6767988.81      2020-Q3    2956
```

```
[550 rows x 5 columns]
```

[13]:
```
sales = sales.groupby(['stkcd','year_quarter'])['online_sales'].sum().
  ↪reset_index() # Aggregate to get the sum of sales in each quarter
sales
```

[13]:
```
      stkcd year_quarter  online_sales
0      2511      2016-Q1  3.384803e+07
1      2511      2016-Q2  4.424781e+07
2      2511      2016-Q3  3.921140e+07
3      2511      2016-Q4  3.867866e+07
4      2511      2017-Q1  5.079266e+07
..      …          …            …
185  603899      2019-Q3  8.817731e+07
186  603899      2019-Q4  1.169147e+08
187  603899      2020-Q1  9.600548e+07
188  603899      2020-Q2  1.338863e+08
189  603899      2020-Q3  4.121578e+07

[190 rows x 3 columns]
```

[14]:
```
report_sales = pd.read_csv('FS_Comins.csv')
report_sales['Accper'] = pd.to_datetime(report_sales['Accper'],␣
  ↪format='%Y-%m-%d').dt.to_period('M')
report_sales['Accper'] = report_sales['Accper'].dt.strftime('%Y-Q%q')
report_sales = report_sales.loc[report_sales['Typrep']=='A',:] # Count for all␣
  ↪of the relevant companies
report_sales.drop(['B001101000','Typrep'], axis=1, inplace=True)
report_sales.reset_index(drop=True, inplace=True)
report_sales.rename({'B001100000':'report_sales_cum'}, axis=1, inplace=True)
report_sales['year'] = pd.to_datetime(report_sales['Accper']).dt.year
# Remove the first line, because it's the data of last year
report_sales = report_sales.groupby(['Stkcd','year']).apply(lambda x: x.iloc[1:
  ↪])
report_sales.reset_index(drop=True, inplace=True)
report_sales['report_sales_sft'] = report_sales.
  ↪groupby(['Stkcd','year'])['report_sales_cum'].shift(1)
# Calculate the quartly sales since the original data is cumulative
report_sales['report_sales'] = report_sales['report_sales_cum'] -␣
  ↪report_sales['report_sales_sft']
# Fill in the NAN values of the first line using the original sales
report_sales.loc[report_sales['report_sales'].isnull(),
     'report_sales'] = report_sales.loc[report_sales['report_sales'].isnull(),␣
  ↪'report_sales_cum']
report_sales.drop(['report_sales_sft'], axis = 1, inplace = True)
report_sales.reset_index(drop=True, inplace=True)
report_sales
```

```
[14]:       Stkcd ShortName   Accper  report_sales_cum  year  report_sales
      0      2511           2016-Q1      8.540908e+08  2016  8.540908e+08
      1      2511           2016-Q2      1.771553e+09  2016  9.174619e+08
      2      2511           2016-Q3      2.740959e+09  2016  9.694064e+08
      3      2511           2016-Q4      3.809349e+09  2016  1.068390e+09
      4      2511           2017-Q1      1.032247e+09  2017  1.032247e+09
      ..      ...       ...      ...               ...   ...           ...
      120  603899           2019-Q2      4.838623e+09  2019  2.483009e+09
      121  603899           2019-Q3      7.947344e+09  2019  3.108721e+09
      122  603899           2019-Q4      1.114110e+10  2019  3.193757e+09
      123  603899           2020-Q1      2.083587e+09  2020  2.083587e+09
      124  603899           2020-Q2      4.761424e+09  2020  2.677836e+09

      [125 rows x 6 columns]
```

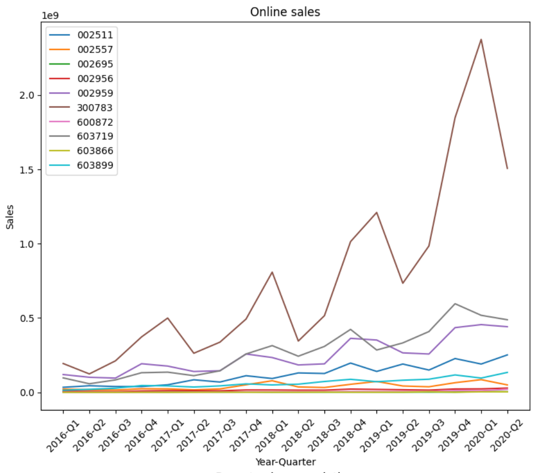
```
[15]: online_report = pd.merge(sales, report_sales, how='left',␣
        ↪left_on=['stkcd','year_quarter'], right_on=['Stkcd','Accper'])
      online_report = online_report.loc[online_report['year_quarter']!='2020-Q3']
      online_report
```

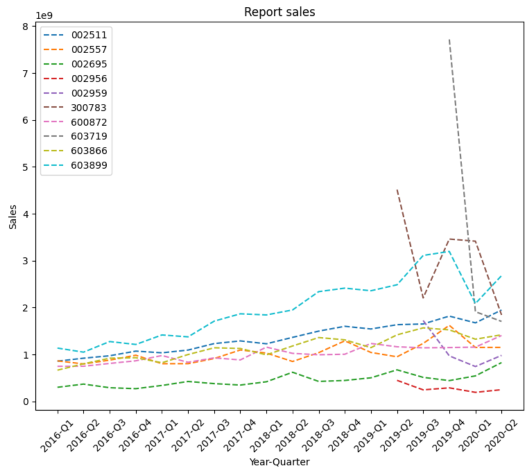
```
[15]:       stkcd year_quarter  online_sales     Stkcd ShortName   Accper  \
      0      2511      2016-Q1  3.384803e+07    2511.0           2016-Q1
      1      2511      2016-Q2  4.424781e+07    2511.0           2016-Q2
      2      2511      2016-Q3  3.921140e+07    2511.0           2016-Q3
      3      2511      2016-Q4  3.867866e+07    2511.0           2016-Q4
      4      2511      2017-Q1  5.079266e+07    2511.0           2017-Q1
      ..      ...          ...           ...       ...       ...      ...
      184  603899      2019-Q2  8.132643e+07  603899.0           2019-Q2
      185  603899      2019-Q3  8.817731e+07  603899.0           2019-Q3
      186  603899      2019-Q4  1.169147e+08  603899.0           2019-Q4
      187  603899      2020-Q1  9.600548e+07  603899.0           2020-Q1
      188  603899      2020-Q2  1.338863e+08  603899.0           2020-Q2

          report_sales_cum    year  report_sales
      0       8.540908e+08  2016.0  8.540908e+08
      1       1.771553e+09  2016.0  9.174619e+08
      2       2.740959e+09  2016.0  9.694064e+08
      3       3.809349e+09  2016.0  1.068390e+09
      4       1.032247e+09  2017.0  1.032247e+09
      ..               ...     ...           ...
      184     4.838623e+09  2019.0  2.483009e+09
      185     7.947344e+09  2019.0  3.108721e+09
      186     1.114110e+10  2019.0  3.193757e+09
      187     2.083587e+09  2020.0  2.083587e+09
      188     4.761424e+09  2020.0  2.677836e+09

      [180 rows x 9 columns]
```

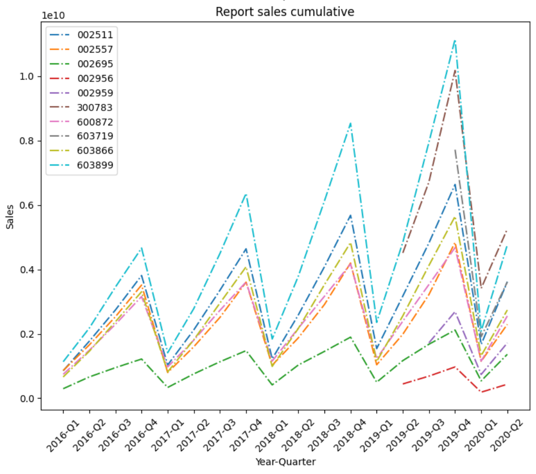
[16]:
```
stkcd_list = online_report['stkcd'].unique().astype('str').tolist()
for i in range(len(stkcd_list)):
    if len(stkcd_list[i])==4:
        stkcd_list[i] = '00'+stkcd_list[i]
stkcd_list
```

[16]:
```
['002511',
 '002557',
 '002695',
 '002956',
 '002959',
 '300783',
 '600872',
 '603719',
 '603866',
 '603899']
```

[17]:
```
plt.figure(figsize=(20,16))
plt.subplot(2,2,1)
for stkcd in stkcd_list:
    int_stk = int(stkcd)
    plt.plot(online_report.loc[online_report['stkcd']==int_stk, 'year_quarter'],
             online_report.loc[online_report['stkcd']==int_stk,␣
 ↪'online_sales'], label = '{}'.format(stkcd))
plt.title('Online sales')
plt.xticks(rotation=45)
plt.xlabel('Year-Quarter')
plt.ylabel('Sales')
plt.legend()

plt.subplot(2,2,2)
for stkcd in stkcd_list:
    int_stk = int(stkcd)
    plt.plot(online_report.loc[online_report['stkcd']==int_stk, 'year_quarter'],
             online_report.loc[online_report['stkcd']==int_stk, 'report_sales'],
             label = '{}'.format(stkcd), linestyle = 'dashed')
plt.title('Report sales')
plt.xticks(rotation=45)
plt.xlabel('Year-Quarter')
plt.ylabel('Sales')
plt.legend()

plt.subplot(2,2,3)
for stkcd in stkcd_list:
    int_stk = int(stkcd)
    plt.plot(online_report.loc[online_report['stkcd']==int_stk, 'year_quarter'],
            online_report.loc[online_report['stkcd']==int_stk,␣
↪'report_sales_cum'],
            label = '{}'.format(stkcd), linestyle = 'dashdot')
plt.title('Report sales cumulative')
plt.xticks(rotation=45)
plt.xlabel('Year-Quarter')
plt.ylabel('Sales')
plt.legend()
plt.show()
```


Online sales
1e9
Sales
Year-Quarter
2016-Q1
2016-Q2
2016-Q3
2016-Q4
2017-Q1
2017-Q2
2017-Q3
2017-Q4
2018-Q1
2018-Q2
2018-Q3
2018-Q4
2019-Q1
2019-Q2
2019-Q3
2019-Q4
2020-Q1
2020-Q2


Report sales
1e9
Sales
Year-Quarter
2016-Q1
2016-Q2
2016-Q3
2016-Q4
2017-Q1
2017-Q2
2017-Q3
2017-Q4
2018-Q1
2018-Q2
2018-Q3
2018-Q4
2019-Q1
2019-Q2
2019-Q3
2019-Q4
2020-Q1
2020-Q2


Report sales cumulative
1e10
Sales
Year-Quarter
2016-Q1
2016-Q2
2016-Q3
2016-Q4
2017-Q1
2017-Q2
2017-Q3
2017-Q4
2018-Q1
2018-Q2
2018-Q3
2018-Q4
2019-Q1
2019-Q2
2019-Q3
2019-Q4
2020-Q1
2020-Q2