06_conditional_autoencoder_for_asset_pricing_model

September 29, 2021

1 Conditional Autoencoder for Asset Pricing - Part 2: The Model

This notebook uses a dataset created using `yfinance` in the notebook conditional_autoencoder_for_asset_pricing_data. The results will vary depending on which ticker downloads succeeded.

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import sys, os
from time import time
from pathlib import Path
from itertools import product
from tqdm import tqdm

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Dot, Reshape,␣
 ↪BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import TensorBoard

from sklearn.preprocessing import quantile_transform

from scipy.stats import spearmanr
```

[3]:
```
gpu_devices = tf.config.experimental.list_physical_devices('GPU')
if gpu_devices:
    print('Using GPU')
    tf.config.experimental.set_memory_growth(gpu_devices[0], True)
else:
    print('Using CPU')
```

```
Using GPU
```

[4]:
```
sys.path.insert(1, os.path.join(sys.path[0], '..'))
from utils import MultipleTimeSeriesCV, format_time
```

[5]:
```
idx = pd.IndexSlice
sns.set_style('whitegrid')
np.random.seed(42)
```

[6]:
```
results_path = Path('results', 'asset_pricing')
if not results_path.exists():
    results_path.mkdir(parents=True)
```

[7]:
```
characteristics = ['beta', 'betasq', 'chmom', 'dolvol', 'idiovol', 'ill',␣
 ↪'indmom',
                   'maxret', 'mom12m', 'mom1m', 'mom36m', 'mvel', 'retvol',␣
 ↪'turn', 'turn_std']
```

1.1 Load Data

[8]:
```
with pd.HDFStore(results_path / 'autoencoder.h5') as store:
    print(store.info())
```

```
<class 'pandas.io.pytables.HDFStore'>
File path: results/asset_pricing/autoencoder.h5
/close                      frame        (shape->[7559,4420])
/factor/beta                frame        (shape->[2969406,1])
/factor/betasq              frame        (shape->[2969406,1])
/factor/chmom               frame        (shape->[3375489,1])
/factor/dolvol              frame        (shape->[3534960,1])
/factor/idiovol             frame        (shape->[2969406,1])
/factor/ill                 frame        (shape->[3210773,1])
/factor/indmom              frame        (shape->[3551199,1])
/factor/maxret              frame        (shape->[3562402,1])
/factor/mom12m              frame        (shape->[3375489,1])
/factor/mom1m               series       (shape->[3580621])
/factor/mom36m              frame        (shape->[2967391,1])
/factor/mvel                frame        (shape->[3597636,1])
/factor/retvol              frame        (shape->[3580621,1])
/factor/turn                frame        (shape->[3506569,1])
/factor/turn_std            frame        (shape->[3552216,1])
/metadata                   frame        (shape->[1,3])
/returns                    frame        (shape->[1565,4420])
/volume                     frame        (shape->[7559,4420])
```

1.1.1 Weekly returns

[20]:
```
data = (pd.read_hdf(results_path / 'autoencoder.h5', 'returns')
        .stack(dropna=False)
        .to_frame('returns')
        .loc[idx['1993':, :], :])
```

[22]:
```
with pd.HDFStore(results_path / 'autoencoder.h5') as store:
    keys = [k[1:] for k in store.keys() if k[1:].startswith('factor')]
    for key in keys:
        data[key.split('/')[-1]] = store[key].squeeze()
```

[23]:
```
characteristics = data.drop('returns', axis=1).columns.tolist()
```

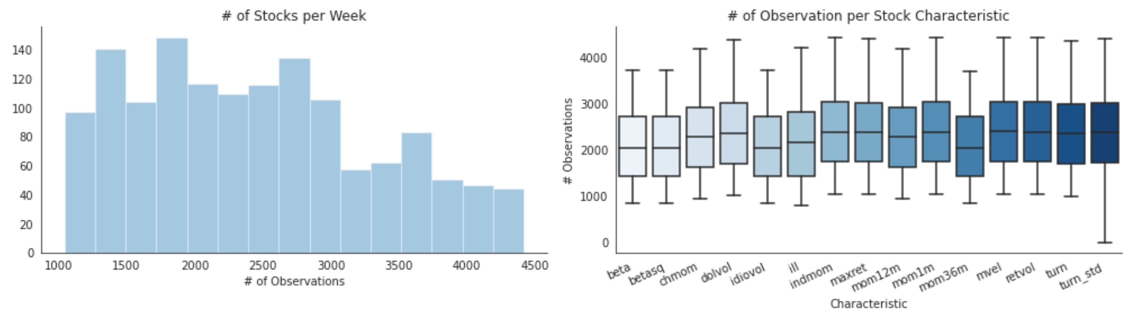
[24]:
```
data['returns_fwd'] = data.returns.unstack('ticker').shift(-1).stack()
```

[25]:
```
data.info(null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 6232200 entries, (Timestamp('1993-01-01 00:00:00', freq='W-FRI'),
'A') to (Timestamp('2020-01-03 00:00:00', freq='W-FRI'), 'ZYXI')
Data columns (total 17 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   returns      3452579 non-null  float64
 1   beta         2969406 non-null  float64
 2   betasq       2969406 non-null  float64
 3   chmom        3283334 non-null  float64
 4   dolvol       3403423 non-null  float64
 5   idiovol      2969406 non-null  float64
 6   ill          3108429 non-null  float64
 7   indmom       3452527 non-null  float64
 8   maxret       3426881 non-null  float64
 9   mom12m       3283334 non-null  float64
 10  mom1m        3440945 non-null  float64
 11  mom36m       2967391 non-null  float64
 12  mvel         3454030 non-null  float64
 13  retvol       3440945 non-null  float64
 14  turn         3380001 non-null  float64
 15  turn_std     3413256 non-null  float64
 16  returns_fwd  3451536 non-null  float64
dtypes: float64(17)
memory usage: 832.3+ MB
```

[14]:
```
nobs_by_date = data.groupby(level='date').count().max(1)
nobs_by_characteristic = pd.melt(data[characteristics].groupby(level='date').
 ↪count(),
                                    value_name='# Observations',
                                    var_name=['Characteristic'])
```

[15]:
```
with sns.axes_style("white"):
    fig, axes = plt.subplots(ncols=2, figsize=(14, 4))
    sns.distplot(nobs_by_date, kde=False, ax=axes[0])
    axes[0].set_title('# of Stocks per Week')
    axes[0].set_xlabel('# of Observations')
    sns.boxplot(x='Characteristic',
                y='# Observations',
                data=nobs_by_characteristic,
                ax=axes[1],
                palette='Blues')
    axes[1].set_xticklabels(axes[1].get_xticklabels(),
                            rotation=25,
                            ha='right')
    axes[1].set_title('# of Observation per Stock Characteristic')
    sns.despine()
    fig.tight_layout()
```



1.1.2 Rank-normalize characteristics

[16]:
```
data.loc[:, characteristics] = (data.loc[:, characteristics]
                                .groupby(level='date')
                                .apply(lambda x: pd.
 ↪DataFrame(quantile_transform(x,
                                                                                    ␣
 ↪ copy=True,
                                                                                    ␣
 ↪ n_quantiles=x.shape[0]),
                                                                  ␣
 ↪columns=characteristics,
                                                                   index=x.index.
 ↪get_level_values('ticker')))
                               .mul(2).sub(1))
```

[17]:
```
data.info(null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 6232200 entries, (Timestamp('1993-01-01 00:00:00', freq='W-FRI'),
'A') to (Timestamp('2020-01-03 00:00:00', freq='W-FRI'), 'ZYXI')
Data columns (total 17 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   returns      3452579 non-null  float64
 1   beta         2969406 non-null  float64
 2   betasq       2969406 non-null  float64
 3   chmom        3283334 non-null  float64
 4   dolvol       3403423 non-null  float64
 5   idiovol      2969406 non-null  float64
 6   ill          3108429 non-null  float64
 7   indmom       3452527 non-null  float64
 8   maxret       3426881 non-null  float64
 9   mom12m       3283334 non-null  float64
 10  mom1m        3440945 non-null  float64
 11  mom36m       2967391 non-null  float64
 12  mvel         3454030 non-null  float64
 13  retvol       3440945 non-null  float64
 14  turn         3380001 non-null  float64
 15  turn_std     3413256 non-null  float64
 16  returns_fwd  3451536 non-null  float64
dtypes: float64(17)
memory usage: 832.3+ MB
```

[18]:
```
data.index.names
```

[18]:
```
FrozenList(['date', 'ticker'])
```

[19]:
```
data.describe()
```

```
[19]:              returns          beta        betasq         chmom        dolvol  \
count   3.452579e+06  2.969406e+06  2.969406e+06  3.283334e+06  3.403423e+06
mean    3.011444e-03 -4.514118e-09 -3.661858e-07 -4.961806e-08 -8.404166e-07
std     6.176189e-02  5.776241e-01  5.776246e-01  5.775977e-01  5.775907e-01
min    -9.269350e-01 -1.000000e+00 -1.000000e+00 -1.000000e+00 -1.000000e+00
25%    -2.151944e-02 -5.002862e-01 -5.002779e-01 -5.002653e-01 -5.002520e-01
50%     9.756321e-04  4.103460e-06  3.428143e-06  5.635024e-06 -6.761061e-06
75%     2.491691e-02  5.002871e-01  5.002862e-01  5.002657e-01  5.002522e-01
max     4.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

             idiovol           ill        indmom        maxret        mom12m  \
count   2.969406e+06  3.108429e+06  3.452527e+06  3.426881e+06  3.283334e+06
mean   -9.121839e-08 -4.094780e-07  1.003525e-03 -7.020948e-08 -1.223854e-07
std     5.776242e-01  5.776119e-01  5.859634e-01  5.775870e-01  5.775978e-01
min    -1.000000e+00 -1.000000e+00 -1.000000e+00 -1.000000e+00 -1.000000e+00
25%    -5.002830e-01 -5.002665e-01 -4.969450e-01 -5.002388e-01 -5.002763e-01
50%    -5.194775e-06  7.698840e-06  0.000000e+00  8.172791e-06  6.175095e-06
75%     5.002879e-01  5.002743e-01  4.774836e-01  5.002266e-01  5.002604e-01
max     1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

               mom1m        mom36m          mvel        retvol          turn  \
count   3.440945e+06  2.967391e+06  3.454030e+06  3.440945e+06  3.380001e+06
mean   -2.300930e-08 -1.665202e-07 -2.744138e-08 -1.153290e-06 -3.940610e-07
std     5.775841e-01  5.776243e-01  5.775857e-01  5.775884e-01  5.775911e-01
min    -1.000000e+00 -1.000000e+00 -1.000000e+00 -1.000000e+00 -1.000000e+00
25%    -5.002454e-01 -5.002837e-01 -5.002486e-01 -5.002370e-01 -5.002599e-01
50%    -1.165672e-04 -7.247569e-06  2.850846e-06 -2.794473e-06  1.038184e-05
75%     5.002571e-01  5.002797e-01  5.002506e-01  5.002554e-01  5.002418e-01
max     1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

            turn_std   returns_fwd
count   3.413256e+06  3.451536e+06
mean   -1.342879e-06  3.008981e-03
std     5.775899e-01  6.176569e-02
min    -1.000000e+00 -9.269350e-01
25%    -5.002387e-01 -2.152460e-02
50%     2.845388e-06  9.756585e-04
75%     5.002498e-01  2.491509e-02
max     1.000000e+00  4.000000e+00
```

```
[20]: data = data.loc[idx[:'2019', :], :]
```

```
[21]: data.loc[:, ['returns', 'returns_fwd']] = data.loc[:, ['returns',␣
       ↪'returns_fwd']].clip(lower=-1, upper=1.0)
```

```
[22]: data = data.fillna(-2)
```

[23]:
```
data.to_hdf(results_path / 'autoencoder.h5', 'model_data')
```

1.2 Architecture

[8]:
```
data = pd.read_hdf(results_path / 'autoencoder.h5', 'model_data')
```

1.2.1 Key parameters

[9]:
```
n_factors = 3
n_characteristics = len(characteristics)
n_tickers = len(data.index.unique('ticker'))
```

[10]:
```
n_tickers
```

[10]:
```
4420
```

[11]:
```
n_characteristics
```

[11]:
```
15
```

1.2.2 Input Layer

[28]:
```
input_beta = Input((n_tickers, n_characteristics), name='input_beta')
input_factor = Input((n_tickers,), name='input_factor')
```

1.2.3 Stock Characteristics Network

[29]:
```
hidden_layer = Dense(units=8, activation='relu',␣
 ↪name='hidden_layer')(input_beta)
batch_norm = BatchNormalization(name='batch_norm')(hidden_layer)
output_beta = Dense(units=n_factors, name='output_beta')(batch_norm)
```

1.2.4 Factor Network

[30]:
```
output_factor = Dense(units=n_factors, name='output_factor')(input_factor)
```

1.2.5 Output Layer

[31]:
```
output = Dot(axes=(2,1), name='output_layer')([output_beta, output_factor])
```

1.2.6 Compile Layer

[32]:
```
model = Model(inputs=[input_beta, input_factor], outputs=output)
model.compile(loss='mse', optimizer='adam')
```

1.2.7 Automate model generation

[12]:
```python
def make_model(hidden_units=8, n_factors=3):
    input_beta = Input((n_tickers, n_characteristics), name='input_beta')
    input_factor = Input((n_tickers,), name='input_factor')

    hidden_layer = Dense(units=hidden_units, activation='relu',␣
 ↪name='hidden_layer')(input_beta)
    batch_norm = BatchNormalization(name='batch_norm')(hidden_layer)

    output_beta = Dense(units=n_factors, name='output_beta')(batch_norm)

    output_factor = Dense(units=n_factors, name='output_factor')(input_factor)

    output = Dot(axes=(2,1), name='output_layer')([output_beta, output_factor])

    model = Model(inputs=[input_beta, input_factor], outputs=output)
    model.compile(loss='mse', optimizer='adam')
    return model
```

1.2.8 Model Summary

[34]:
```python
model.summary()
```

```
Model: "model"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_beta (InputLayer)         [(None, 4420, 15)]   0
________________________________________________________________________________
__________________
hidden_layer (Dense)            (None, 4420, 8)      128
input_beta[0][0]
________________________________________________________________________________
__________________
batch_norm (BatchNormalization) (None, 4420, 8)      32
hidden_layer[0][0]
________________________________________________________________________________
__________________
input_factor (InputLayer)       [(None, 4420)]       0
________________________________________________________________________________
__________________
output_beta (Dense)             (None, 4420, 3)      27
batch_norm[0][0]
________________________________________________________________________________
```

```
------------------
output_factor (Dense)          (None, 3)            13263
input_factor[0][0]
________________________________________________________________________________
__________________
output_layer (Dot)             (None, 4420)         0
output_beta[0][0]
output_factor[0][0]
================================================================================
==================
Total params: 13,450
Trainable params: 13,434
Non-trainable params: 16
________________________________________________________________________________
__________________
```

1.3 Train Model

1.3.1 Cross-validation parameters

[13]:
```python
YEAR = 52
```

[14]:
```python
cv = MultipleTimeSeriesCV(n_splits=5,
                          train_period_length=20*YEAR,
                          test_period_length=1*YEAR,
                          lookahead=1)
```

[15]:
```python
def get_train_valid_data(data, train_idx, val_idx):
    train, val = data.iloc[train_idx], data.iloc[val_idx]
    X1_train = train.loc[:, characteristics].values.reshape(-1, n_tickers,␣
 ↪n_characteristics)
    X1_val = val.loc[:, characteristics].values.reshape(-1, n_tickers,␣
 ↪n_characteristics)
    X2_train = train.loc[:, 'returns'].unstack('ticker')
    X2_val = val.loc[:, 'returns'].unstack('ticker')
    y_train = train.returns_fwd.unstack('ticker')
    y_val = val.returns_fwd.unstack('ticker')
    return X1_train, X2_train, y_train, X1_val, X2_val, y_val
```

1.3.2 Hyperparameter Options

[16]:
```python
factor_opts = [2, 3, 4, 5, 6]
unit_opts = [8, 16, 32]
```

[17]:
```python
param_grid = list(product(unit_opts, factor_opts))
```

1.3.3 Run Cross-Validation

[40]:
```
batch_size = 32
```

[41]:
```
cols = ['units', 'n_factors', 'fold', 'epoch', 'ic_mean',
        'ic_daily_mean', 'ic_daily_std', 'ic_daily_median']
```

[42]:
```
start = time()
for units, n_factors in param_grid:
    scores = []
    model = make_model(hidden_units=units, n_factors=n_factors)
    for fold, (train_idx, val_idx) in enumerate(cv.split(data)):
        X1_train, X2_train, y_train, X1_val, X2_val, y_val =␣
 ↪get_train_valid_data(data,
                                                                                     ␣
 ↪   train_idx,
                                                                                     ␣
 ↪   val_idx)
        for epoch in range(250):
            model.fit([X1_train, X2_train], y_train,
                      batch_size=batch_size,
                      validation_data=([X1_val, X2_val], y_val),
                      epochs=epoch + 1,
                      initial_epoch=epoch,
                      verbose=0, shuffle=True)
            result = (pd.DataFrame({'y_pred': model.predict([X1_val,
                                                              X2_val]).
 ↪reshape(-1),
                                    'y_true': y_val.stack().values},
                                   index=y_val.stack().index)
                      .replace(-2, np.nan).dropna())
            r0 = spearmanr(result.y_true, result.y_pred)[0]
            r1 = result.groupby(level='date').apply(lambda x: spearmanr(x.
 ↪y_pred,
                                                                        x.
 ↪y_true)[0])

            scores.append([units, n_factors, fold, epoch, r0,
                           r1.mean(), r1.std(), r1.median()])
            if epoch % 50 == 0:
                print(f'{format_time(time()-start)} | {n_factors} | {units:02}␣
 ↪| {fold:02}-{epoch:03} | {r0:6.2%} | '
                      f'{r1.mean():6.2%} | {r1.median():6.2%}')
    scores = pd.DataFrame(scores, columns=cols)
    scores.to_hdf(results_path / 'scores.h5', f'{units}/{n_factors}')
```

```
00:00:03 | 2 08 | 00-000 |  1.24% |  0.24% | -0.25%
```

```
00:00:32 | 2 08 | 00-050 | -0.26% | -0.38% |  0.10%
00:01:01 | 2 08 | 00-100 | -1.46% |  0.22% | -0.62%
00:01:30 | 2 08 | 00-150 | -2.23% | -0.19% | -0.46%
00:02:00 | 2 08 | 00-200 | -3.28% |  0.42% | -1.34%
00:02:31 | 2 08 | 01-000 | -1.09% |  1.13% |  1.28%
00:03:01 | 2 08 | 01-050 |  0.19% |  1.12% |  1.57%
00:03:31 | 2 08 | 01-100 |  0.16% | -1.05% | -2.39%
00:04:00 | 2 08 | 01-150 |  0.82% |  0.28% | -0.02%
00:04:31 | 2 08 | 01-200 |  1.13% |  0.06% |  0.22%
00:05:01 | 2 08 | 02-000 | -0.21% | -0.34% | -0.56%
00:05:30 | 2 08 | 02-050 |  0.14% | -0.09% |  0.25%
00:05:59 | 2 08 | 02-100 |  1.96% |  0.99% |  2.66%
00:06:26 | 2 08 | 02-150 |  0.96% | -0.08% | -0.98%
00:06:55 | 2 08 | 02-200 |  0.87% |  0.19% |  1.90%
00:07:25 | 2 08 | 03-000 |  0.75% | -0.15% |  1.57%
00:07:53 | 2 08 | 03-050 |  1.59% |  1.50% |  2.23%
00:08:22 | 2 08 | 03-100 |  0.42% |  0.75% | -0.60%
00:08:51 | 2 08 | 03-150 | -1.20% | -1.20% | -1.41%
00:09:19 | 2 08 | 03-200 | -1.13% | -1.58% | -1.78%
00:09:50 | 2 08 | 04-000 |  0.23% | -0.42% |  2.12%
00:10:17 | 2 08 | 04-050 |  1.08% |  0.69% |  0.34%
00:10:45 | 2 08 | 04-100 | -0.48% | -0.85% |  1.53%
00:11:14 | 2 08 | 04-150 | -0.29% | -0.60% | -0.04%
00:11:41 | 2 08 | 04-200 | -0.53% | -0.33% |  2.27%
00:12:12 | 3 08 | 00-000 | -1.61% |  0.60% | -0.13%
00:12:40 | 3 08 | 00-050 |  6.10% |  4.07% |  2.67%
00:13:09 | 3 08 | 00-100 |  5.82% |  4.28% |  2.81%
00:13:38 | 3 08 | 00-150 |  3.56% |  3.06% |  3.63%
00:14:06 | 3 08 | 00-200 |  3.54% |  2.25% |  2.33%
00:14:36 | 3 08 | 01-000 |  0.40% |  0.94% |  0.16%
00:15:05 | 3 08 | 01-050 |  0.54% | -0.31% | -1.41%
00:15:33 | 3 08 | 01-100 |  2.20% |  0.73% |  0.70%
00:16:03 | 3 08 | 01-150 | -3.09% | -1.88% | -4.01%
00:16:30 | 3 08 | 01-200 | -1.28% | -1.94% | -1.92%
00:16:58 | 3 08 | 02-000 | -1.02% | -1.18% | -2.59%
00:17:25 | 3 08 | 02-050 |  1.02% |  1.15% |  2.03%
00:17:50 | 3 08 | 02-100 |  0.44% |  0.78% |  1.78%
00:18:15 | 3 08 | 02-150 |  1.77% |  2.10% |  4.52%
00:18:40 | 3 08 | 02-200 |  1.15% |  1.77% |  3.96%
00:19:05 | 3 08 | 03-000 |  0.32% |  1.93% |  0.77%
00:19:32 | 3 08 | 03-050 | -1.11% | -1.34% |  0.27%
00:19:60 | 3 08 | 03-100 |  0.15% | -0.56% |  0.36%
00:20:27 | 3 08 | 03-150 |  2.07% |  1.65% | -0.87%
00:20:55 | 3 08 | 03-200 |  2.14% | -0.05% |  0.29%
00:21:24 | 3 08 | 04-000 |  0.42% | -0.05% | -0.59%
00:21:47 | 3 08 | 04-050 |  1.66% | -0.80% | -1.21%
00:22:14 | 3 08 | 04-100 |  2.71% |  1.66% |  2.87%
00:22:41 | 3 08 | 04-150 |  3.04% |  1.63% |  1.03%
```

```
00:23:07 | 3 08 | 04-200 | -0.62% | -0.41% | -1.59%
00:23:35 | 4 08 | 00-000 | -5.39% | -3.54% | -4.08%
00:24:00 | 4 08 | 00-050 |  0.20% | -1.82% | -0.84%
00:24:26 | 4 08 | 00-100 |  0.48% | -1.11% | -0.64%
00:24:52 | 4 08 | 00-150 |  0.11% | -0.55% |  0.89%
00:25:18 | 4 08 | 00-200 | -1.10% | -1.00% |  0.32%
00:25:45 | 4 08 | 01-000 |  2.23% |  0.06% | -0.44%
00:26:11 | 4 08 | 01-050 |  0.26% |  0.54% |  1.93%
00:26:39 | 4 08 | 01-100 |  0.68% |  2.06% |  1.37%
00:27:05 | 4 08 | 01-150 |  0.50% |  1.27% |  2.51%
00:27:32 | 4 08 | 01-200 |  3.78% | -1.46% | -0.49%
00:28:00 | 4 08 | 02-000 |  0.78% |  0.56% |  0.78%
00:28:26 | 4 08 | 02-050 | -1.11% | -1.83% | -2.84%
00:28:52 | 4 08 | 02-100 |  0.27% |  0.84% |  0.11%
00:29:19 | 4 08 | 02-150 |  1.86% |  2.07% |  3.07%
00:29:44 | 4 08 | 02-200 |  0.60% |  0.89% |  1.13%
00:30:12 | 4 08 | 03-000 | -2.19% | -2.51% | -1.79%
00:30:40 | 4 08 | 03-050 | -0.05% | -0.84% | -0.86%
00:31:08 | 4 08 | 03-100 | -1.88% |  2.02% |  0.47%
00:31:36 | 4 08 | 03-150 |  2.85% | -0.29% | -1.73%
00:32:04 | 4 08 | 03-200 |  3.04% | -1.25% | -1.50%
00:32:34 | 4 08 | 04-000 |  4.31% | -2.77% | -2.50%
00:33:01 | 4 08 | 04-050 |  9.06% | -1.24% | -0.18%
00:33:27 | 4 08 | 04-100 | 18.58% |  4.69% |  1.39%
00:33:52 | 4 08 | 04-150 | 16.30% |  2.64% |  1.76%
00:34:18 | 4 08 | 04-200 | -0.71% |  0.88% | -0.68%
00:34:45 | 5 08 | 00-000 |  1.11% | -1.80% | -2.43%
00:35:12 | 5 08 | 00-050 | -0.70% |  0.29% | -0.39%
00:35:38 | 5 08 | 00-100 | -1.26% | -0.85% | -1.12%
00:36:04 | 5 08 | 00-150 | -1.52% | -1.14% | -2.67%
00:36:31 | 5 08 | 00-200 | -2.29% | -1.53% | -2.25%
00:36:58 | 5 08 | 01-000 |  2.09% |  1.87% |  1.71%
00:37:25 | 5 08 | 01-050 |  0.09% | -1.26% | -1.22%
00:37:52 | 5 08 | 01-100 |  1.38% | -0.33% | -1.52%
00:38:18 | 5 08 | 01-150 | -1.30% | -1.31% | -1.85%
00:38:44 | 5 08 | 01-200 |  0.33% | -1.28% | -3.28%
00:39:12 | 5 08 | 02-000 |  1.99% |  0.77% | -0.35%
00:39:37 | 5 08 | 02-050 |  0.72% |  0.04% |  0.34%
00:40:04 | 5 08 | 02-100 |  1.22% |  0.94% |  3.31%
00:40:30 | 5 08 | 02-150 |  0.34% | -0.13% | -1.37%
00:40:56 | 5 08 | 02-200 |  0.02% | -0.37% |  2.42%
00:41:24 | 5 08 | 03-000 |  2.11% |  0.76% |  1.18%
00:41:50 | 5 08 | 03-050 |  1.49% |  0.27% |  0.43%
00:42:19 | 5 08 | 03-100 |  3.72% |  0.68% |  2.20%
00:42:46 | 5 08 | 03-150 |  4.07% |  2.47% |  2.21%
00:43:13 | 5 08 | 03-200 |  2.07% |  1.49% |  3.06%
00:43:43 | 5 08 | 04-000 |  4.18% |  2.87% |  2.12%
00:44:10 | 5 08 | 04-050 |  2.26% |  0.11% |  1.83%
```

```
00:44:38 | 5 08 | 04-100 |  5.73% |  0.20% | -1.13%
00:45:06 | 5 08 | 04-150 |  1.38% |  0.04% | -1.19%
00:45:32 | 5 08 | 04-200 |  0.99% |  0.29% |  0.27%
00:46:01 | 6 08 | 00-000 |  0.02% |  1.34% |  1.52%
00:46:27 | 6 08 | 00-050 |  1.21% | -0.64% | -1.59%
00:46:52 | 6 08 | 00-100 |  0.77% | -0.15% | -0.21%
00:47:18 | 6 08 | 00-150 |  0.45% |  0.02% |  0.28%
00:47:44 | 6 08 | 00-200 |  0.37% |  0.19% |  0.45%
00:48:11 | 6 08 | 01-000 |  8.19% |  5.48% |  5.04%
00:48:38 | 6 08 | 01-050 |  2.35% | -0.39% | -2.14%
00:49:03 | 6 08 | 01-100 |  3.09% |  2.57% |  1.20%
00:49:29 | 6 08 | 01-150 | -0.40% |  0.68% |  0.56%
00:49:55 | 6 08 | 01-200 |  2.05% | -1.43% | -1.61%
00:50:21 | 6 08 | 02-000 |  9.24% | -2.02% | -3.29%
00:50:47 | 6 08 | 02-050 |  4.14% | -1.39% | -3.72%
00:51:12 | 6 08 | 02-100 | -2.82% |  1.26% |  2.54%
00:51:37 | 6 08 | 02-150 |  0.07% | -1.11% | -2.55%
00:52:03 | 6 08 | 02-200 |  5.57% |  0.83% |  1.13%
00:52:28 | 6 08 | 03-000 | 13.18% |  0.01% |  1.09%
00:52:54 | 6 08 | 03-050 | -1.73% |  1.05% |  1.09%
00:53:20 | 6 08 | 03-100 | 11.19% | -0.18% | -0.25%
00:53:44 | 6 08 | 03-150 |  4.35% |  0.81% |  2.71%
00:54:09 | 6 08 | 03-200 |  3.66% |  0.21% |  0.29%
00:54:35 | 6 08 | 04-000 |  4.74% |  3.01% |  3.89%
00:54:58 | 6 08 | 04-050 | 13.84% | -0.68% | -0.79%
00:55:21 | 6 08 | 04-100 | 22.41% | -0.29% | -1.37%
00:55:46 | 6 08 | 04-150 | 14.08% |  0.31% | -0.37%
00:56:11 | 6 08 | 04-200 | 12.63% |  1.73% |  2.01%
00:56:38 | 2 16 | 00-000 |  1.39% |  1.33% |  1.60%
00:57:02 | 2 16 | 00-050 |  0.52% | -0.81% | -0.57%
00:57:28 | 2 16 | 00-100 |  0.22% | -0.16% |  0.37%
00:57:54 | 2 16 | 00-150 |  0.49% |  1.10% |  1.21%
00:58:18 | 2 16 | 00-200 | -1.89% | -2.14% | -0.89%
00:58:45 | 2 16 | 01-000 |  1.84% |  2.40% |  4.04%
00:59:11 | 2 16 | 01-050 |  0.73% |  0.76% |  1.22%
00:59:37 | 2 16 | 01-100 | -0.36% | -0.18% | -0.13%
01:00:03 | 2 16 | 01-150 | -0.13% | -0.74% | -2.32%
01:00:28 | 2 16 | 01-200 | -0.16% |  0.30% |  0.64%
01:00:53 | 2 16 | 02-000 | -1.92% | -2.24% | -3.58%
01:01:20 | 2 16 | 02-050 |  2.80% |  3.08% |  4.62%
01:01:46 | 2 16 | 02-100 | -1.99% | -2.05% | -3.30%
01:02:12 | 2 16 | 02-150 | -1.12% | -1.39% | -0.19%
01:02:40 | 2 16 | 02-200 |  1.24% |  1.48% |  2.06%
01:03:08 | 2 16 | 03-000 |  1.71% |  1.13% |  5.00%
01:03:33 | 2 16 | 03-050 |  0.61% |  1.95% |  1.67%
01:04:00 | 2 16 | 03-100 | -0.78% | -0.66% | -0.18%
01:04:27 | 2 16 | 03-150 | -0.09% | -0.71% | -2.86%
01:04:55 | 2 16 | 03-200 |  1.06% |  1.25% | -0.52%
```

```
01:05:23 | 2 16 | 04-000 |  1.91% | -0.71% | -0.32%
01:05:49 | 2 16 | 04-050 |  0.82% |  1.67% | -0.02%
01:06:16 | 2 16 | 04-100 |  4.10% |  3.32% |  3.54%
01:06:42 | 2 16 | 04-150 |  3.91% |  4.46% |  4.54%
01:07:08 | 2 16 | 04-200 |  6.34% |  1.59% | -0.54%
01:07:37 | 3 16 | 00-000 | -0.68% | -0.74% | -0.36%
01:08:00 | 3 16 | 00-050 |  0.28% |  0.42% |  0.83%
01:08:24 | 3 16 | 00-100 |  0.25% |  0.39% |  0.85%
01:08:49 | 3 16 | 00-150 | -0.74% | -0.12% |  0.43%
01:09:12 | 3 16 | 00-200 | -1.40% |  0.01% | -0.01%
01:09:38 | 3 16 | 01-000 | -0.45% |  0.54% |  1.45%
01:10:04 | 3 16 | 01-050 | -0.97% |  1.37% |  1.68%
01:10:30 | 3 16 | 01-100 | -1.34% |  0.20% | -0.05%
01:10:55 | 3 16 | 01-150 | -1.81% | -0.54% | -1.85%
01:11:21 | 3 16 | 01-200 |  1.01% |  0.87% |  0.37%
01:11:47 | 3 16 | 02-000 |  1.36% |  1.28% |  1.63%
01:12:11 | 3 16 | 02-050 | -0.43% | -0.41% |  0.44%
01:12:36 | 3 16 | 02-100 |  1.22% |  0.97% |  2.21%
01:13:02 | 3 16 | 02-150 | -0.32% |  0.22% | -0.21%
01:13:28 | 3 16 | 02-200 |  1.78% |  1.42% |  2.20%
01:13:54 | 3 16 | 03-000 | -0.45% |  0.11% |  0.31%
01:14:20 | 3 16 | 03-050 |  1.84% |  1.43% |  0.68%
01:14:45 | 3 16 | 03-100 |  1.91% |  0.82% |  2.21%
01:15:10 | 3 16 | 03-150 |  1.30% |  1.01% |  1.47%
01:15:36 | 3 16 | 03-200 | -0.58% | -0.00% |  0.59%
01:16:04 | 3 16 | 04-000 | -1.66% | -1.20% | -2.58%
01:16:31 | 3 16 | 04-050 |  0.63% |  1.13% |  1.47%
01:16:59 | 3 16 | 04-100 | -2.45% |  0.45% |  0.75%
01:17:24 | 3 16 | 04-150 |  0.42% |  0.96% | -0.03%
01:17:49 | 3 16 | 04-200 | -0.09% |  0.22% |  0.55%
01:18:18 | 4 16 | 00-000 |  0.68% | -0.68% | -0.80%
01:18:44 | 4 16 | 00-050 |  1.10% |  0.53% |  0.15%
01:19:11 | 4 16 | 00-100 |  0.36% |  0.37% | -0.64%
01:19:35 | 4 16 | 00-150 | -0.50% | -0.43% | -0.98%
01:20:00 | 4 16 | 00-200 | -1.08% | -0.49% | -1.01%
01:20:27 | 4 16 | 01-000 |  2.04% |  1.67% |  1.98%
01:20:52 | 4 16 | 01-050 |  1.39% |  2.31% |  3.22%
01:21:17 | 4 16 | 01-100 |  1.88% |  1.19% |  2.16%
01:21:43 | 4 16 | 01-150 |  2.52% |  2.46% |  2.41%
01:22:07 | 4 16 | 01-200 |  2.00% |  0.86% |  2.30%
01:22:34 | 4 16 | 02-000 |  2.23% |  2.04% |  1.97%
01:22:59 | 4 16 | 02-050 | -1.92% | -2.05% | -2.65%
01:23:24 | 4 16 | 02-100 |  0.48% |  0.33% |  0.91%
01:23:48 | 4 16 | 02-150 |  0.82% | -0.25% | -0.92%
01:24:13 | 4 16 | 02-200 |  0.97% |  0.72% |  0.85%
01:24:39 | 4 16 | 03-000 |  2.11% |  1.04% |  0.77%
01:25:05 | 4 16 | 03-050 |  2.62% |  1.16% |  0.95%
01:25:29 | 4 16 | 03-100 |  0.95% |  0.68% |  1.40%
```

```
01:25:53 | 4 16 | 03-150 |  2.33% |  1.66% |  1.54%
01:26:17 | 4 16 | 03-200 | -0.43% | -0.85% | -3.98%
01:26:43 | 4 16 | 04-000 |  0.65% | -0.93% | -0.80%
01:27:07 | 4 16 | 04-050 | -6.39% | -1.51% | -1.40%
01:27:34 | 4 16 | 04-100 |  3.52% |  3.93% |  4.99%
01:27:57 | 4 16 | 04-150 |  3.27% |  2.02% | -0.64%
01:28:23 | 4 16 | 04-200 |  4.98% |  1.22% | -0.10%
01:28:53 | 5 16 | 00-000 | -4.46% | -2.67% | -1.41%
01:29:19 | 5 16 | 00-050 | -0.04% | -0.02% | -0.20%
01:29:46 | 5 16 | 00-100 | -0.03% |  0.01% |  0.33%
01:30:10 | 5 16 | 00-150 |  0.45% |  0.63% |  1.44%
01:30:35 | 5 16 | 00-200 | -0.41% |  0.01% |  0.40%
01:31:04 | 5 16 | 01-000 |  1.08% |  1.82% |  2.12%
01:31:27 | 5 16 | 01-050 | -1.49% | -1.16% | -1.01%
01:31:52 | 5 16 | 01-100 |  0.91% |  1.30% |  1.96%
01:32:19 | 5 16 | 01-150 | -1.41% |  1.26% | -0.33%
01:32:43 | 5 16 | 01-200 |  0.86% |  1.32% |  0.61%
01:33:09 | 5 16 | 02-000 |  0.86% |  0.88% |  0.97%
01:33:34 | 5 16 | 02-050 |  0.66% |  1.01% |  2.18%
01:33:58 | 5 16 | 02-100 |  1.67% |  1.35% |  1.39%
01:34:22 | 5 16 | 02-150 |  1.22% |  0.20% |  0.68%
01:34:47 | 5 16 | 02-200 |  1.37% |  0.87% |  1.37%
01:35:12 | 5 16 | 03-000 |  3.34% |  2.01% |  2.41%
01:35:36 | 5 16 | 03-050 |  4.10% |  1.40% |  1.56%
01:36:00 | 5 16 | 03-100 |  2.88% |  1.13% | -0.05%
01:36:24 | 5 16 | 03-150 |  2.42% |  1.00% |  1.51%
01:36:49 | 5 16 | 03-200 | -0.05% | -0.58% | -0.49%
01:37:15 | 5 16 | 04-000 |  4.69% | -1.29% | -2.87%
01:37:40 | 5 16 | 04-050 |  3.93% | -0.88% | -1.13%
01:38:06 | 5 16 | 04-100 |  4.50% | -0.13% | -0.36%
01:38:31 | 5 16 | 04-150 | -0.83% | -0.37% | -0.76%
01:38:56 | 5 16 | 04-200 |  4.49% |  1.64% |  0.96%
01:39:25 | 6 16 | 00-000 | -0.89% | -0.23% |  0.94%
01:39:49 | 6 16 | 00-050 | -1.46% | -2.07% | -0.79%
01:40:14 | 6 16 | 00-100 | -1.39% | -2.06% | -1.56%
01:40:39 | 6 16 | 00-150 | -1.28% | -2.00% | -1.03%
01:41:03 | 6 16 | 00-200 | -0.53% | -0.84% |  0.40%
01:41:29 | 6 16 | 01-000 |  3.73% | -0.31% | -0.15%
01:41:53 | 6 16 | 01-050 | -0.25% | -0.65% | -0.83%
01:42:17 | 6 16 | 01-100 | -0.75% | -1.12% | -2.30%
01:42:41 | 6 16 | 01-150 | -0.46% | -1.54% | -2.92%
01:43:06 | 6 16 | 01-200 |  2.09% |  2.66% |  3.64%
01:43:31 | 6 16 | 02-000 | -1.48% | -1.65% | -2.24%
01:43:56 | 6 16 | 02-050 |  1.19% |  0.95% |  0.42%
01:44:20 | 6 16 | 02-100 |  0.13% |  0.01% | -0.51%
01:44:44 | 6 16 | 02-150 |  3.89% |  2.86% |  3.74%
01:45:07 | 6 16 | 02-200 |  1.51% |  2.29% |  2.13%
01:45:32 | 6 16 | 03-000 |  3.69% |  1.69% |  1.89%
```

```
01:45:56 | 6 16 | 03-050 |  1.37% |  1.17% | -0.86%
01:46:22 | 6 16 | 03-100 |  4.53% |  3.29% |  1.85%
01:46:46 | 6 16 | 03-150 |  2.82% |  0.77% |  0.58%
01:47:09 | 6 16 | 03-200 |  4.97% |  2.62% |  3.63%
01:47:36 | 6 16 | 04-000 |  2.36% | -0.47% | -1.58%
01:48:00 | 6 16 | 04-050 | -0.16% |  0.97% | -2.02%
01:48:25 | 6 16 | 04-100 |  3.43% |  0.77% |  2.32%
01:48:51 | 6 16 | 04-150 |  1.46% | -0.60% | -4.10%
01:49:17 | 6 16 | 04-200 | -0.42% |  0.11% | -1.86%
01:49:45 | 2 32 | 00-000 | -0.66% |  0.65% | -0.32%
01:50:11 | 2 32 | 00-050 |  1.19% |  0.49% | -0.37%
01:50:37 | 2 32 | 00-100 |  0.92% |  0.25% | -0.06%
01:51:02 | 2 32 | 00-150 |  0.54% | -0.27% | -0.80%
01:51:28 | 2 32 | 00-200 |  1.00% | -0.08% | -1.02%
01:51:55 | 2 32 | 01-000 |  0.86% |  0.16% | -0.47%
01:52:22 | 2 32 | 01-050 | -0.46% | -1.09% | -1.38%
01:52:50 | 2 32 | 01-100 | -0.33% | -1.42% | -2.72%
01:53:17 | 2 32 | 01-150 | -0.74% | -1.61% | -1.07%
01:53:45 | 2 32 | 01-200 | -1.00% | -0.80% | -0.81%
01:54:14 | 2 32 | 02-000 | -0.48% | -0.44% |  0.66%
01:54:42 | 2 32 | 02-050 |  1.01% |  0.99% |  0.69%
01:55:11 | 2 32 | 02-100 | -0.80% | -0.94% | -2.37%
01:55:36 | 2 32 | 02-150 | -0.12% |  0.20% |  0.66%
01:56:04 | 2 32 | 02-200 | -0.03% | -0.12% |  0.19%
01:56:32 | 2 32 | 03-000 | -1.76% | -0.79% | -2.85%
01:56:57 | 2 32 | 03-050 |  0.88% |  1.77% |  2.87%
01:57:22 | 2 32 | 03-100 |  1.34% |  1.30% |  2.70%
01:57:50 | 2 32 | 03-150 | -0.46% | -0.85% | -0.27%
01:58:17 | 2 32 | 03-200 |  1.22% |  0.50% |  0.64%
01:58:45 | 2 32 | 04-000 |  6.54% |  2.84% |  1.42%
01:59:13 | 2 32 | 04-050 |  5.43% |  3.02% |  2.35%
01:59:40 | 2 32 | 04-100 | -0.69% | -0.33% |  0.99%
02:00:08 | 2 32 | 04-150 |  1.20% |  2.19% |  1.70%
02:00:35 | 2 32 | 04-200 |  4.05% |  3.04% |  3.46%
02:01:05 | 3 32 | 00-000 | -0.87% | -0.60% |  1.23%
02:01:33 | 3 32 | 00-050 |  0.16% | -0.39% | -0.13%
02:01:59 | 3 32 | 00-100 |  1.27% | -0.97% |  0.10%
02:02:26 | 3 32 | 00-150 |  2.01% |  0.13% |  0.28%
02:02:54 | 3 32 | 00-200 | -0.41% | -0.35% | -0.82%
02:03:21 | 3 32 | 01-000 |  0.38% | -0.36% |  1.00%
02:03:47 | 3 32 | 01-050 | -1.15% | -1.65% | -0.69%
02:04:15 | 3 32 | 01-100 | -0.23% |  1.07% |  0.58%
02:04:42 | 3 32 | 01-150 | -0.10% |  0.38% | -0.39%
02:05:09 | 3 32 | 01-200 |  1.51% |  1.96% |  1.51%
02:05:37 | 3 32 | 02-000 | -0.91% | -0.62% | -0.91%
02:06:02 | 3 32 | 02-050 |  0.56% |  0.61% |  0.54%
02:06:27 | 3 32 | 02-100 |  0.78% |  0.88% |  0.33%
02:06:53 | 3 32 | 02-150 | -0.39% | -0.69% | -1.18%
```

```
02:07:19 | 3 32 | 02-200 |  0.88% |  0.59% |  0.31%
02:07:46 | 3 32 | 03-000 |  0.28% |  0.41% |  1.00%
02:08:11 | 3 32 | 03-050 |  1.52% |  0.79% | -3.27%
02:08:36 | 3 32 | 03-100 |  0.86% |  1.25% |  1.69%
02:09:02 | 3 32 | 03-150 | -1.15% | -1.02% | -0.82%
02:09:28 | 3 32 | 03-200 |  2.27% |  1.80% | -0.74%
02:09:54 | 3 32 | 04-000 |  1.61% |  1.76% |  1.93%
02:10:20 | 3 32 | 04-050 | -0.68% | -0.89% | -1.53%
02:10:46 | 3 32 | 04-100 |  2.21% | -0.99% | -1.91%
02:11:12 | 3 32 | 04-150 |  2.55% |  2.79% |  4.55%
02:11:39 | 3 32 | 04-200 | -1.55% | -2.58% | -1.72%
02:12:07 | 4 32 | 00-000 | -4.30% | -1.99% | -2.26%
02:12:36 | 4 32 | 00-050 | -1.16% |  0.11% |  0.17%
02:13:03 | 4 32 | 00-100 | -1.17% |  0.15% |  0.68%
02:13:30 | 4 32 | 00-150 |  0.47% |  0.41% | -0.19%
02:13:58 | 4 32 | 00-200 | -1.31% | -0.29% | -0.80%
02:14:27 | 4 32 | 01-000 | -1.40% | -1.37% | -3.22%
02:14:54 | 4 32 | 01-050 |  0.28% | -0.27% | -0.09%
02:15:23 | 4 32 | 01-100 | -0.17% | -0.76% | -0.29%
02:15:51 | 4 32 | 01-150 |  0.45% | -0.39% |  1.01%
02:16:18 | 4 32 | 01-200 |  0.63% | -0.07% |  0.91%
02:16:49 | 4 32 | 02-000 |  1.64% |  1.63% |  1.31%
02:17:15 | 4 32 | 02-050 | -1.15% | -1.04% | -1.21%
02:17:42 | 4 32 | 02-100 |  1.29% |  1.24% |  1.88%
02:18:09 | 4 32 | 02-150 | -0.69% | -1.08% | -1.47%
02:18:36 | 4 32 | 02-200 |  1.73% |  1.56% |  2.26%
02:19:04 | 4 32 | 03-000 |  1.14% |  0.60% |  1.14%
02:19:33 | 4 32 | 03-050 |  0.95% |  0.68% |  1.37%
02:19:59 | 4 32 | 03-100 | -0.66% | -0.18% | -1.67%
02:20:26 | 4 32 | 03-150 |  1.85% |  1.16% |  2.21%
02:20:53 | 4 32 | 03-200 | -0.52% |  0.35% |  0.30%
02:21:21 | 4 32 | 04-000 |  5.03% |  2.61% |  0.80%
02:21:49 | 4 32 | 04-050 |  0.61% | -0.71% |  1.09%
02:22:17 | 4 32 | 04-100 |  5.45% |  4.32% |  3.05%
02:22:43 | 4 32 | 04-150 |  1.40% |  3.85% |  3.64%
02:23:11 | 4 32 | 04-200 | -4.90% | -2.73% | -2.85%
02:23:42 | 5 32 | 00-000 | -2.62% | -2.92% | -3.29%
02:24:13 | 5 32 | 00-050 |  1.15% | -0.38% | -1.95%
02:24:44 | 5 32 | 00-100 |  0.85% | -0.85% | -1.90%
02:25:10 | 5 32 | 00-150 |  0.69% | -0.97% | -1.53%
02:25:38 | 5 32 | 00-200 | -0.01% | -1.20% | -1.89%
02:26:09 | 5 32 | 01-000 |  0.18% |  0.27% |  0.69%
02:26:35 | 5 32 | 01-050 | -0.01% | -0.29% |  0.40%
02:27:01 | 5 32 | 01-100 |  0.57% |  0.25% |  0.96%
02:27:29 | 5 32 | 01-150 | -0.22% | -0.05% | -1.12%
02:27:57 | 5 32 | 01-200 | -0.71% | -0.35% | -2.00%
02:28:27 | 5 32 | 02-000 | -1.30% | -1.37% | -1.80%
02:28:57 | 5 32 | 02-050 |  1.89% |  1.75% |  2.58%
```

```
02:29:24 | 5 32 | 02-100 |  0.50% |  0.10% | -0.60%
02:29:50 | 5 32 | 02-150 |  2.07% |  2.22% |  2.48%
02:30:20 | 5 32 | 02-200 |  1.69% |  1.67% |  3.78%
02:30:50 | 5 32 | 03-000 |  1.42% |  2.16% |  2.78%
02:31:18 | 5 32 | 03-050 | -1.75% | -0.87% | -2.64%
02:31:48 | 5 32 | 03-100 |  1.81% |  1.50% |  2.32%
02:32:16 | 5 32 | 03-150 |  1.30% |  1.57% |  1.06%
02:32:44 | 5 32 | 03-200 | -0.62% | -0.89% | -1.82%
02:33:15 | 5 32 | 04-000 |  0.89% | -1.07% |  2.75%
02:33:43 | 5 32 | 04-050 |  6.00% |  1.67% |  0.86%
02:34:12 | 5 32 | 04-100 |  4.22% |  0.55% | -1.03%
02:34:39 | 5 32 | 04-150 |  2.87% |  1.12% |  1.46%
02:35:05 | 5 32 | 04-200 |  0.61% |  0.60% |  0.72%
02:35:34 | 6 32 | 00-000 | -0.85% | -1.69% | -2.30%
02:36:01 | 6 32 | 00-050 | -2.71% | -2.04% | -2.99%
02:36:30 | 6 32 | 00-100 | -1.64% | -1.25% | -1.89%
02:36:60 | 6 32 | 00-150 | -0.90% | -0.36% | -1.10%
02:37:28 | 6 32 | 00-200 |  0.14% |  0.53% |  0.22%
02:37:58 | 6 32 | 01-000 |  2.33% |  1.58% |  0.90%
02:38:27 | 6 32 | 01-050 | -0.13% |  1.08% |  1.54%
02:38:53 | 6 32 | 01-100 |  2.18% |  2.60% |  2.78%
02:39:21 | 6 32 | 01-150 | -1.04% | -0.74% | -2.07%
02:39:49 | 6 32 | 01-200 |  1.44% |  0.56% |  2.34%
02:40:16 | 6 32 | 02-000 |  1.31% |  1.15% |  1.55%
02:40:43 | 6 32 | 02-050 |  0.88% |  1.13% |  1.59%
02:41:10 | 6 32 | 02-100 |  2.18% |  2.22% |  2.92%
02:41:37 | 6 32 | 02-150 | -0.22% |  0.07% |  0.84%
02:42:06 | 6 32 | 02-200 |  0.11% | -0.06% |  1.13%
02:42:37 | 6 32 | 03-000 |  0.54% |  0.45% |  0.10%
02:43:03 | 6 32 | 03-050 |  1.34% |  0.95% |  0.91%
02:43:30 | 6 32 | 03-100 |  2.19% |  1.67% |  3.98%
02:43:57 | 6 32 | 03-150 |  2.29% |  1.72% |  0.53%
02:44:26 | 6 32 | 03-200 | -1.13% | -2.00% | -2.74%
02:44:55 | 6 32 | 04-000 | -1.44% | -2.56% | -2.89%
02:45:22 | 6 32 | 04-050 |  0.20% |  0.55% |  0.76%
02:45:47 | 6 32 | 04-100 |  2.50% |  2.63% |  1.59%
02:46:16 | 6 32 | 04-150 | -4.57% | -2.51% | -2.70%
02:46:43 | 6 32 | 04-200 |  2.58% |  1.61% | -0.67%
```

1.3.4 Evaluate Results

[13]:
```
scores = []
with pd.HDFStore(results_path / 'scores.h5') as store:
    for key in store.keys():
        scores.append(store[key])
scores = pd.concat(scores)
```

[14]:
```
scores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18750 entries, 0 to 1249
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   units            18750 non-null  int64
 1   n_factors        18750 non-null  int64
 2   fold             18750 non-null  int64
 3   epoch            18750 non-null  int64
 4   ic_mean          18750 non-null  float64
 5   ic_daily_mean    18750 non-null  float64
 6   ic_daily_std     18750 non-null  float64
 7   ic_daily_median  18750 non-null  float64
dtypes: float64(4), int64(4)
memory usage: 1.3 MB
```

[15]:
```
avg = (scores.groupby(['n_factors', 'units', 'epoch'])
       ['ic_mean', 'ic_daily_mean', 'ic_daily_median']
        .mean()
       .reset_index())
```

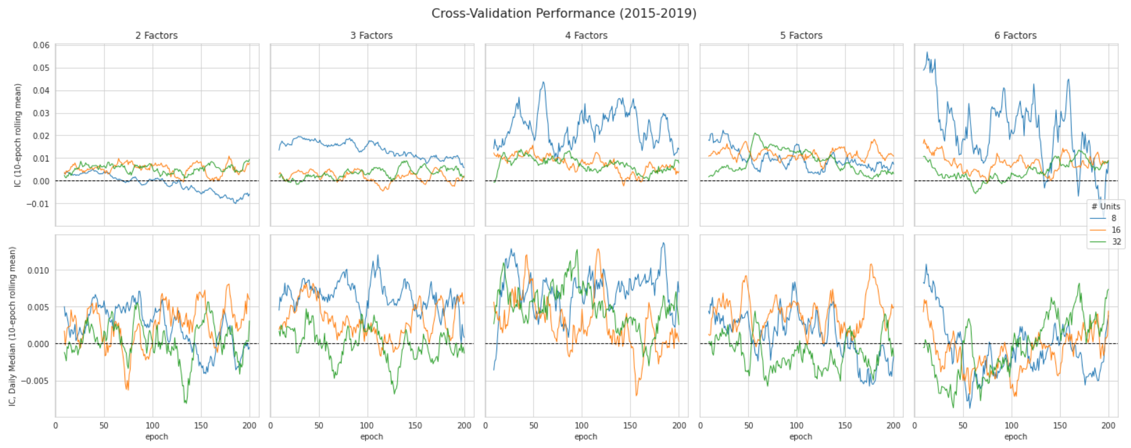
[16]:
```
avg.nlargest(n=20, columns=['ic_daily_median'])
```

[16]:
```
      n_factors  units  epoch   ic_mean  ic_daily_mean  ic_daily_median
2079          4     32     79  0.026611       0.023304         0.028009
2218          4     32    218  0.019487       0.015941         0.027230
2052          4     32     52  0.023268       0.019379         0.027194
1681          4      8    181  0.056288       0.015536         0.027112
2234          4     32    234  0.026894       0.016454         0.026352
1614          4      8    114  0.037274       0.018129         0.025588
1608          4      8    108  0.030997       0.019158         0.025526
765           3      8     15  0.015636       0.014492         0.024900
1716          4      8    216  0.003554       0.016880         0.024367
1712          4      8    212  0.020408       0.019991         0.024052
2094          4     32     94  0.018744       0.013401         0.023730
2087          4     32     87  0.013595       0.013018         0.023570
471           2     16    221  0.013744       0.012241         0.023094
1719          4      8    219  0.031323       0.017167         0.022970
2104          4     32    104  0.006138       0.014180         0.022912
2637          5     16    137  0.022596       0.017144         0.022794
1866          4     16    116  0.017252       0.016993         0.022738
1676          4      8    176  0.019320       0.021859         0.022262
395           2     16    145  0.019981       0.018661         0.022152
852           3      8    102  0.025021       0.020112         0.022113
```

```
[17]: top = (avg.groupby(['n_factors', 'units'])
             .apply(lambda x: x.nlargest(n=5, columns=['ic_daily_median']))
             .reset_index(-1, drop=True))

      top.nlargest(n=5, columns=['ic_daily_median'])
```

```
[17]:                  n_factors  units  epoch   ic_mean  ic_daily_mean  \
      n_factors units
      4         32             4     32     79  0.026611       0.023304
                32             4     32    218  0.019487       0.015941
                32             4     32     52  0.023268       0.019379
                8              4      8    181  0.056288       0.015536
                32             4     32    234  0.026894       0.016454

                       ic_daily_median
      n_factors units
      4         32            0.028009
                32            0.027230
                32            0.027194
                8             0.027112
                32            0.026352
```

```
[48]: fig, axes = plt.subplots(ncols=5, nrows=2, figsize=(20, 8), sharey='row',␣
       ↪sharex=True)

      for n in range(2, 7):
          df = avg[avg.n_factors==n].pivot(index='epoch', columns='units',␣
       ↪values='ic_mean')
          df.rolling(10).mean().loc[:200].plot(ax=axes[0][n-2], lw=1, title=f'{n}␣
       ↪Factors')
          axes[0][n-2].axhline(0, ls='--', c='k', lw=1)
          axes[0][n-2].get_legend().remove()
          axes[0][n-2].set_ylabel('IC (10-epoch rolling mean)')

          df = avg[avg.n_factors==n].pivot(index='epoch', columns='units',␣
       ↪values='ic_daily_median')
          df.rolling(10).mean().loc[:200].plot(ax=axes[1][n-2], lw=1)
          axes[1][n-2].axhline(0, ls='--', c='k', lw=1)
          axes[1][n-2].get_legend().remove()
          axes[1][n-2].set_ylabel('IC, Daily Median (10-epoch rolling mean)')

      handles, labels = axes[0][0].get_legend_handles_labels()
      fig.legend(handles, labels, loc='center right', title='# Units')
      fig.suptitle('Cross-Validation Performance (2015-2019)', fontsize=16)
      fig.tight_layout()
      fig.subplots_adjust(top=.9)
      fig.savefig(results_path / 'cv_performance', dpi=300);
```



1.4 Generate Predictions

We'll average over a range of epochs that appears to deliver good predictions.

[18]:
```
n_factors = 4
units = 32
batch_size = 32
first_epoch = 50
last_epoch = 80
```

[19]:
```
predictions = []
for epoch in tqdm(list(range(first_epoch, last_epoch))):
    epoch_preds = []
    for fold, (train_idx, val_idx) in enumerate(cv.split(data)):
        X1_train, X2_train, y_train, X1_val, X2_val, y_val =␣
 ↪get_train_valid_data(data,
                                                                              ␣
 ↪   train_idx,
                                                                              ␣
 ↪   val_idx)

        model = make_model(n_factors=n_factors, hidden_units=units)
        model.fit([X1_train, X2_train], y_train,
                  batch_size=batch_size,
                  epochs=epoch,
                  verbose=0,
                  shuffle=True)
        epoch_preds.append(pd.Series(model.predict([X1_val, X2_val]).
 ↪reshape(-1),
                                     index=y_val.stack().index).to_frame(epoch))

    predictions.append(pd.concat(epoch_preds))
```

```
100%|          | 30/30 [32:27<00:00, 64.92s/it]
```

[51]:
```
predictions_combined = pd.concat(predictions, axis=1).sort_index()
```

[52]:
```
predictions_combined.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 1149200 entries, (Timestamp('2015-01-09 00:00:00'), 'A') to
(Timestamp('2019-12-27 00:00:00'), 'ZYXI')
Data columns (total 40 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   130     1149200 non-null  float32
 1   131     1149200 non-null  float32
 2   132     1149200 non-null  float32
 3   133     1149200 non-null  float32
 4   134     1149200 non-null  float32
 5   135     1149200 non-null  float32
 6   136     1149200 non-null  float32
 7   137     1149200 non-null  float32
 8   138     1149200 non-null  float32
 9   139     1149200 non-null  float32
 10  140     1149200 non-null  float32
 11  141     1149200 non-null  float32
 12  142     1149200 non-null  float32
 13  143     1149200 non-null  float32
 14  144     1149200 non-null  float32
 15  145     1149200 non-null  float32
 16  146     1149200 non-null  float32
 17  147     1149200 non-null  float32
 18  148     1149200 non-null  float32
 19  149     1149200 non-null  float32
 20  150     1149200 non-null  float32
 21  151     1149200 non-null  float32
 22  152     1149200 non-null  float32
 23  153     1149200 non-null  float32
 24  154     1149200 non-null  float32
 25  155     1149200 non-null  float32
 26  156     1149200 non-null  float32
 27  157     1149200 non-null  float32
 28  158     1149200 non-null  float32
 29  159     1149200 non-null  float32
 30  160     1149200 non-null  float32
 31  161     1149200 non-null  float32
 32  162     1149200 non-null  float32
 33  163     1149200 non-null  float32
 34  164     1149200 non-null  float32
 35  165     1149200 non-null  float32
 36  166      1149200 non-null  float32
 37  167      1149200 non-null  float32
 38  168      1149200 non-null  float32
 39  169      1149200 non-null  float32
dtypes: float32(40)
memory usage: 179.9+ MB
```

[53]:
```
predictions_combined.to_hdf(results_path / 'predictions.h5', 'predictions')
```