

03_101_formulaic_alphas

September 29, 2021

1 101 Formulaic Alphas

Based on [101 Formulaic Alphas](#), Zura Kakushadze, arxiv, 2015

1.1 Imports & Settings

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

```
[5]: %matplotlib inline

import numpy as np
import pandas as pd
from sklearn.feature_selection import mutual_info_regression
from scipy.stats import spearmanr
import matplotlib.pyplot as plt
import seaborn as sns
from talib import WMA
```

```
[6]: idx= pd.IndexSlice
      sns.set_style('whitegrid')
```

“An alpha is a combination of mathematical expressions, computer source code, and configuration parameters that can be used, in combination with historical data, to make predictions about future movements of various financial instruments”

[Finding Alphas: A Quantitative Approach to Building Trading Strategies](#), Igor Tulchinsky, 2019

1.2 Functions

The expressions below that define the 101 formulaic alphas contain functions for both time-series and cross-sectional computations.

1.2.1 Cross-section

Function	Definition
rank(x)	Cross-sectional rank
scale(x, a)	Rescaled x such that $\text{sum}(\text{abs}(x)) = a$ (the default is $a = 1$)

Function	Definition
indneutralize(x, g)	x cross-sectionally demeaned within groups g (subindustries, industries, etc.)

```
[7]: def rank(df):
    """Return the cross-sectional percentile rank

    Args:
        :param df: tickers in columns, sorted dates in rows.

    Returns:
        pd.DataFrame: the ranked values
    """
    return df.rank(axis=1, pct=True)
```

```
[8]: def scale(df):
    """
    Scaling time serie.
    :param df: a pandas DataFrame.
    :param k: scaling factor.
    :return: a pandas DataFrame rescaled df such that sum(abs(df)) = k
    """
    return df.div(df.abs().sum(axis=1), axis=0)
```

1.2.2 Operators

- $\text{abs}(x)$, $\log(x)$, $\text{sign}(x)$, $\text{power}(x, a)$ = standard definitions
- same for the operators “+”, “-”, “*”, “/”, “>”, “<”, “==”, “||”, “x ? y : z”

```
[9]: def log(df):
    return np.log1p(df)
```

```
[10]: def sign(df):
    return np.sign(df)
```

```
[11]: def power(df, exp):
    return df.pow(exp)
```

1.2.3 Time Series

Function	Definition
ts_{O}(x, d)	Operator O applied to the time-series for the past d days; non-integer number of days d is converted to floor(d)
ts_lag(x, d)	Value of x d days ago

Function	Definition
<code>ts_delta(x, d)</code>	Difference between the value of x today and d days ago
<code>ts_weighted_mean(x, d)</code>	Weighted moving average over the past d days with linearly decaying weights d, d - 1, ..., 1 (rescaled to sum up to 1)
<code>ts_sum(x, d)</code>	Rolling sum over the past d days
<code>ts_product(x, d)</code>	Rolling product over the past d days
<code>ts_stddev(x, d)</code>	Moving standard deviation over the past d days
<code>ts_rank(x, d)</code>	Rank over the past d days
<code>ts_min(x, d)</code>	Rolling min over the past d days [alias: <code>min(x, d)</code>]
<code>ts_max(x, d)</code>	Rolling max over the past d days [alias: <code>max(x, d)</code>]
<code>ts_argmax(x, d)</code>	Day of <code>ts_max(x, d)</code>
<code>ts_argmin(x, d)</code>	Day of <code>ts_min(x, d)</code>
<code>ts_correlation(x, y, d)</code>	Correlation of x and y for the past d days
<code>ts_covariance(x, y, d)</code>	Covariance of x and y for the past d days

Pandas Implementation

```
[12]: def ts_lag(df: pd.DataFrame, t: int = 1) -> pd.DataFrame:
```

```
    """Return the lagged values t periods ago.
```

```
    Args:
```

```
        :param df: tickers in columns, sorted dates in rows.
```

```
        :param t: lag
```

```
    Returns:
```

```
        pd.DataFrame: the lagged values
```

```
    """
```

```
    return df.shift(t)
```

```
[13]: def ts_delta(df, period=1):
```

```
    """
```

```
    Wrapper function to estimate difference.
```

```
    :param df: a pandas DataFrame.
```

```
    :param period: the difference grade.
```

```
    :return: a pandas DataFrame with today's value minus the value 'period'□
```

```
    ↪ days ago.
```

```
    """
```

```
    return df.diff(period)
```

```
[14]: def ts_sum(df: pd.DataFrame, window: int = 10) -> pd.DataFrame:
```

```
    """Computes the rolling ts_sum for the given window size.
```

```

Args:
    df (pd.DataFrame): tickers in columns, dates in rows.
    window      (int): size of rolling window.

Returns:
    pd.DataFrame: the ts_sum over the last 'window' days.
    """
return df.rolling(window).sum()

```

```

[15]: def ts_mean(df, window=10):
    """Computes the rolling mean for the given window size.

    Args:
        df (pd.DataFrame): tickers in columns, dates in rows.
        window      (int): size of rolling window.

    Returns:
        pd.DataFrame: the mean over the last 'window' days.
        """
    return df.rolling(window).mean()

```

```

[16]: def ts_weighted_mean(df, period=10):
    """
    Linear weighted moving average implementation.
    :param df: a pandas DataFrame.
    :param period: the LWMA period
    :return: a pandas DataFrame with the LWMA.
    """
    return (df.apply(lambda x: WMA(x, timeperiod=period)))

```

```

[17]: def ts_std(df, window=10):
    """
    Wrapper function to estimate rolling standard deviation.
    :param df: a pandas DataFrame.
    :param window: the rolling window.
    :return: a pandas DataFrame with the time-series min over the past 'window'
    ↪ days.
    """
    return (df
            .rolling(window)
            .std())

```

```

[18]: def ts_rank(df, window=10):
    """
    Wrapper function to estimate rolling rank.
    :param df: a pandas DataFrame.
    :param window: the rolling window.

```

```

        :return: a pandas DataFrame with the time-series rank over the past window_
        ↪ days.
        """
        return (df
                .rolling(window)
                .apply(lambda x: x.rank().iloc[-1]))

```

```

[19]: def ts_product(df, window=10):
        """
        Wrapper function to estimate rolling ts_product.
        :param df: a pandas DataFrame.
        :param window: the rolling window.
        :return: a pandas DataFrame with the time-series ts_product over the past_
        ↪ 'window' days.
        """
        return (df
                .rolling(window)
                .apply(np.prod))

```

```

[20]: def ts_min(df, window=10):
        """
        Wrapper function to estimate rolling min.
        :param df: a pandas DataFrame.
        :param window: the rolling window.
        :return: a pandas DataFrame with the time-series min over the past 'window'_
        ↪ days.
        """
        return df.rolling(window).min()

```

```

[21]: def ts_max(df, window=10):
        """
        Wrapper function to estimate rolling min.
        :param df: a pandas DataFrame.
        :param window: the rolling window.
        :return: a pandas DataFrame with the time-series max over the past 'window'_
        ↪ days.
        """
        return df.rolling(window).max()

```

```

[22]: def ts_argmax(df, window=10):
        """
        Wrapper function to estimate which day ts_max(df, window) occurred on
        :param df: a pandas DataFrame.
        :param window: the rolling window.
        :return: well.. that :)
        """
        return df.rolling(window).apply(np.argmax).add(1)

```

```
[23]: def ts_argmin(df, window=10):
      """
      Wrapper function to estimate which day ts_min(df, window) occurred on
      :param df: a pandas DataFrame.
      :param window: the rolling window.
      :return: well.. that :)
      """
      return (df.rolling(window)
              .apply(np.argmin)
              .add(1))
```

```
[24]: def ts_corr(x, y, window=10):
      """
      Wrapper function to estimate rolling correlations.
      :param x, y: pandas DataFrames.
      :param window: the rolling window.
      :return: a pandas DataFrame with the time-series min over the past 'window'
      ↪ days.
      """
      return x.rolling(window).corr(y)
```

```
[25]: def ts_cov(x, y, window=10):
      """
      Wrapper function to estimate rolling covariance.
      :param df: a pandas DataFrame.
      :param window: the rolling window.
      :return: a pandas DataFrame with the time-series min over the past 'window'
      ↪ days.
      """
      return x.rolling(window).cov(y)
```

1.3 Load Data

1.3.1 500 most-traded stocks

```
[26]: ohlcv = ['open', 'high', 'low', 'close', 'volume']
      data = (pd.read_hdf('data.h5', 'data/top500')
              .loc[:, ohlcv + ['ret_01', 'sector', 'ret_fwd']]
              .rename(columns={'ret_01': 'returns'})
              .sort_index())
```

```
[27]: adv20 = data.groupby('ticker').rolling(20).volume.mean().reset_index(0,
      ↪ drop=True)
```

```
[28]: data = data.assign(adv20=adv20)
```

```
[29]: data = data.join(data.groupby('date')[ohlcv].rank(axis=1, pct=True),
    ↪rsuffix='_rank')
```

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

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 1255093 entries, ('A', Timestamp('2007-01-04 00:00:00')) to ('ZION',
Timestamp('2016-12-29 00:00:00'))
Data columns (total 14 columns):
#   Column          Non-Null Count  Dtype
---  -
0   open            1255093 non-null  float64
1   high            1255093 non-null  float64
2   low             1255093 non-null  float64
3   close           1255093 non-null  float64
4   volume          1255093 non-null  float64
5   returns         1254593 non-null  float64
6   sector          1255093 non-null  float64
7   ret_fwd         1255093 non-null  float64
8   adv20           0 non-null        float64
9   open_rank       1255093 non-null  float64
10  high_rank       1255093 non-null  float64
11  low_rank        1255093 non-null  float64
12  close_rank      1255093 non-null  float64
13  volume_rank     1255093 non-null  float64
dtypes: float64(14)
memory usage: 171.9+ MB
```

```
[31]: # data.to_hdf('factors.h5', 'data')
```

1.3.2 Input Data

Variable	Description
returns	daily close-to-close returns
open, close, high, low, volume	standard definitions for daily price and volume data
vwap	daily volume-weighted average price
cap	market cap
adv{d}	average daily dollar volume for the past d days

Variable	Description
IndClass	a generic placeholder for a binary industry classification such as GICS, BICS, NAICS, SIC, etc., in <code>indneutralize(x, IndClass.level)</code> , where <code>level</code> = sector, industry, subindustry, etc. Multiple IndClass in the same alpha need not correspond to the same industry classification.

```
[32]: o = data.open.unstack('ticker')
      h = data.high.unstack('ticker')
      l = data.low.unstack('ticker')
      c = data.close.unstack('ticker')
      v = data.volume.unstack('ticker')
      vwap = o.add(h).add(l).add(c).div(4)
      adv20 = v.rolling(20).mean()
      r = data.returns.unstack('ticker')
```

1.4 Evaluate Alphas

```
[33]: alphas = data[['returns', 'ret_fwd']].copy()
      mi, ic = {}, {}
```

```
[34]: def get_mutual_info_score(returns, alpha, n=100000):
      df = pd.DataFrame({'y': returns, 'alpha': alpha}).dropna().sample(n=n)
      return mutual_info_regression(y=df.y, X=df[['alpha']])[0]
```

1.5 Alpha 001

```
rank(ts_argmax(power(((returns < 0) ? ts_std(returns, 20) : close), 2.), 5))
```

```
[32]: def alpha001(c, r):
      """(rank(ts_argmax(power(((returns < 0)
      ? ts_std(returns, 20)
      : close), 2.), 5)) - 0.5)"""
      c[r < 0] = ts_std(r, 20)
      return (rank(ts_argmax(power(c, 2), 5)).mul(-.5)
              .stack().swaplevel())
```

```
[33]: alpha = 1
```

```
[34]: %%time
      alphas[f'{alpha:03}'] = alpha001(c, r)
```

```
CPU times: user 1min 57s, sys: 334 ms, total: 1min 57s
Wall time: 1min 58s
```

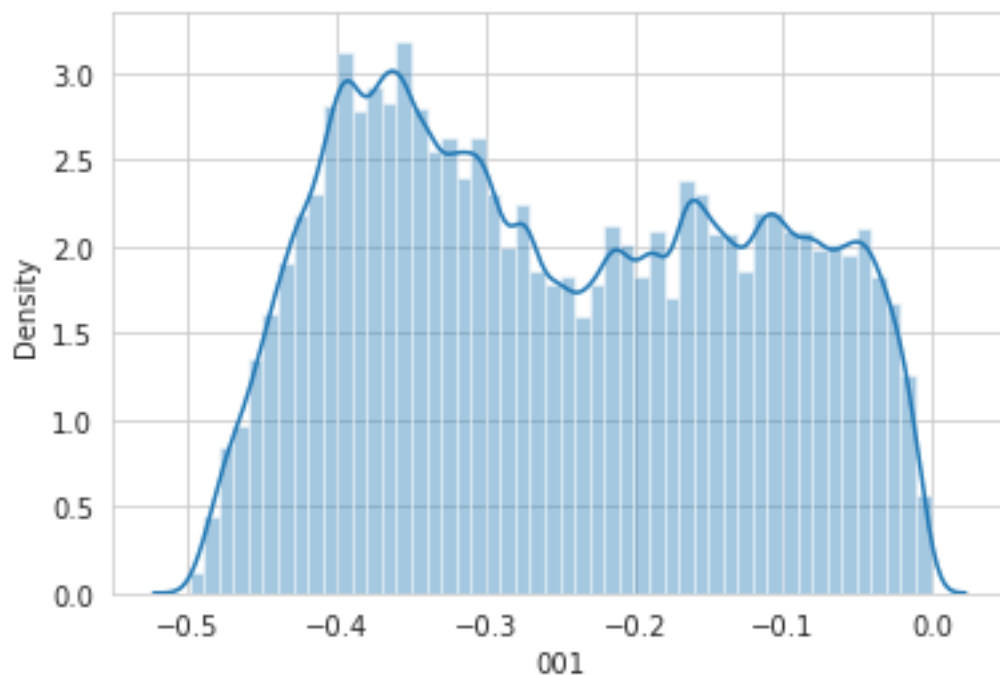


```
[35]: alphas.info()
```

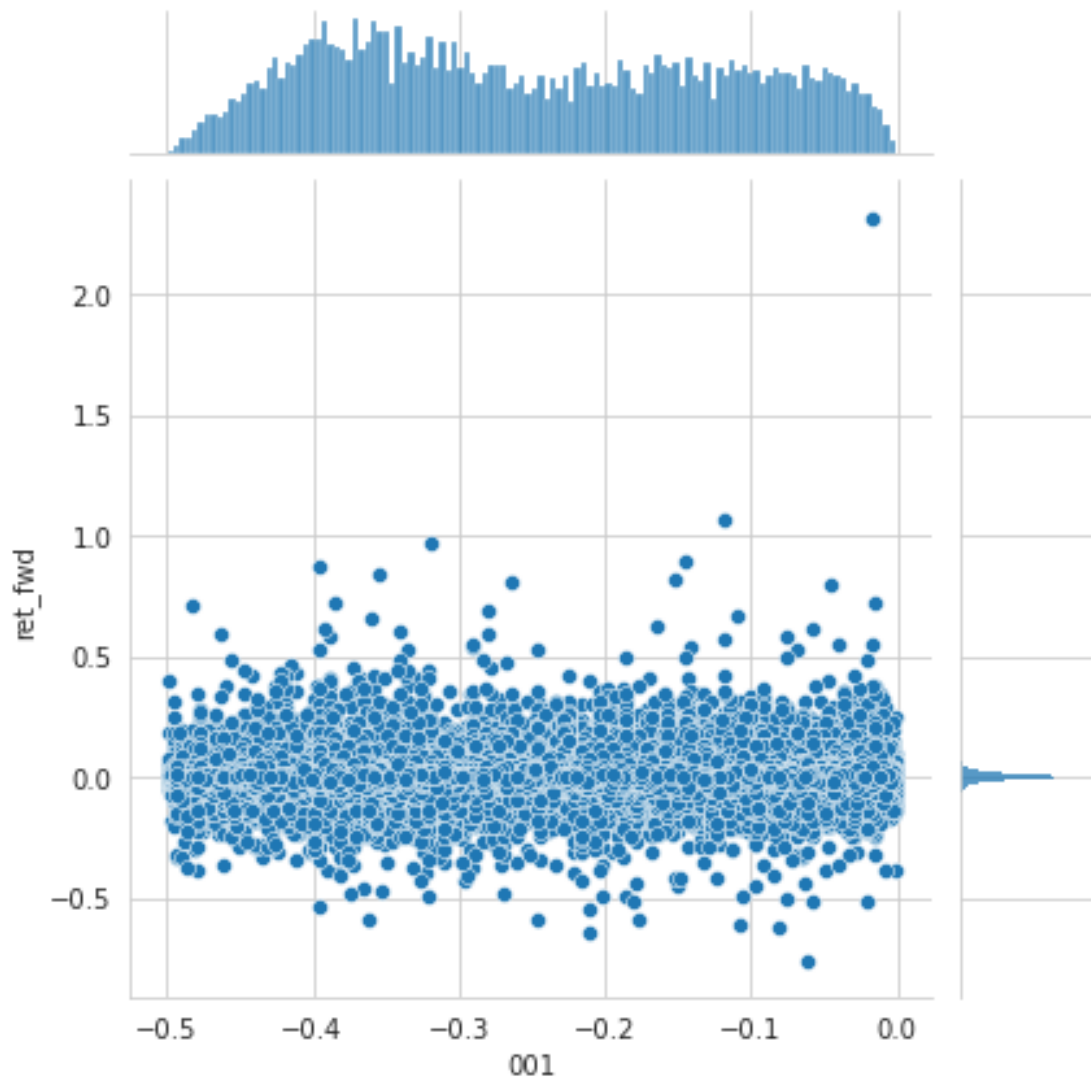
```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 1255093 entries, ('A', Timestamp('2007-01-04 00:00:00')) to ('ZION',
Timestamp('2016-12-29 00:00:00'))
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   returns     1254593 non-null  float64
1   ret_fwd     1255093 non-null  float64
2   001         1243849 non-null  float64
dtypes: float64(3)
memory usage: 66.5+ MB
```

```
[36]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[37]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[38]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas)
```



```
[39]: mi[1] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
      mi[1]
```

```
[39]: 0.01888595802975246
```

1.6 Alpha 002

```
correlation(rank(delta(log(volume), 2)), rank(((close - open) / open)), 6))
```

```
[40]: def alpha002(o, c, v):
      """(-1 * ts_corr(rank(ts_delta(log(volume), 2)), rank(((close - open) /
      ↪open)), 6))"""
      s1 = rank(ts_delta(log(v), 2))
      s2 = rank((c / o) - 1)
```

```
alpha = -ts_corr(s1, s2, 6)
return alpha.stack('ticker').swaplevel().replace([-np.inf, np.inf], np.nan)
```

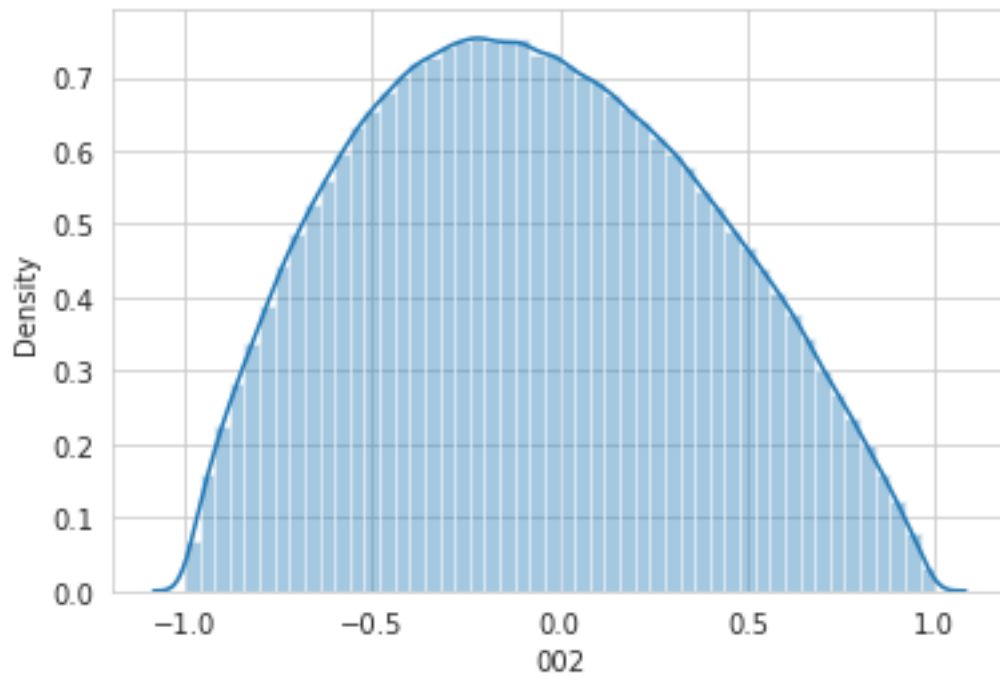
```
[41]: alpha = 2
```

```
[42]: %%time
alphas[f'{alpha:03}'] = alpha002(o, c, v)
```

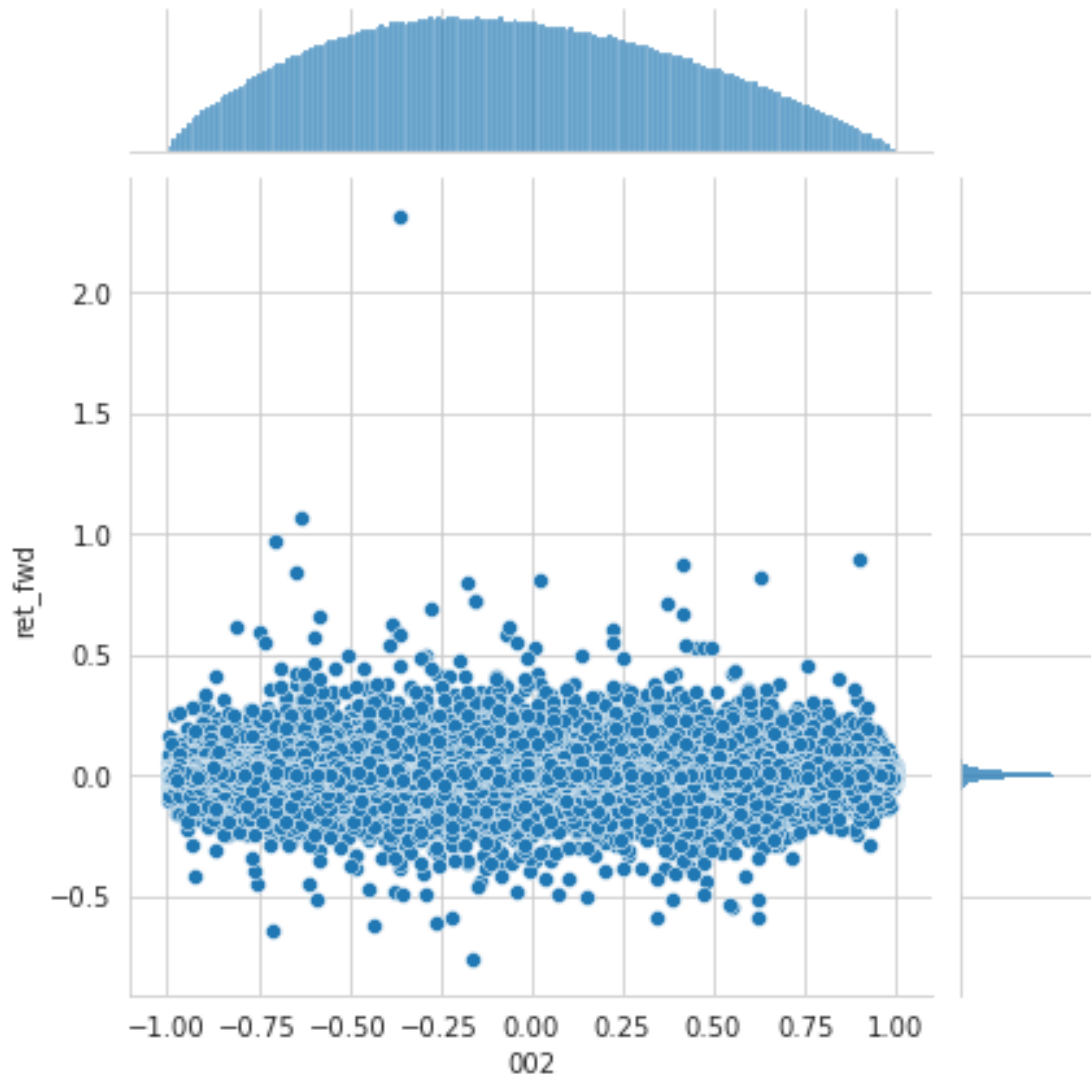
CPU times: user 3.86 s, sys: 28 ms, total: 3.88 s
Wall time: 3.84 s

```
[43]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[44]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[45]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas)
```



```
[46]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
      mi[2]
```

```
[46]: 0.0012399516229262275
```

1.7 Alpha 003

```
(-1 * correlation(rank(open), rank(volume), 10))
```

```
[47]: def alpha003(o, v):
      """(-1 * ts_corr(rank(open), rank(volume), 10))"""

      return (-ts_corr(rank(o), rank(v), 10)
              .stack('ticker'))
```

```
.swaplevel()  
.replace([-np.inf, np.inf], np.nan))
```

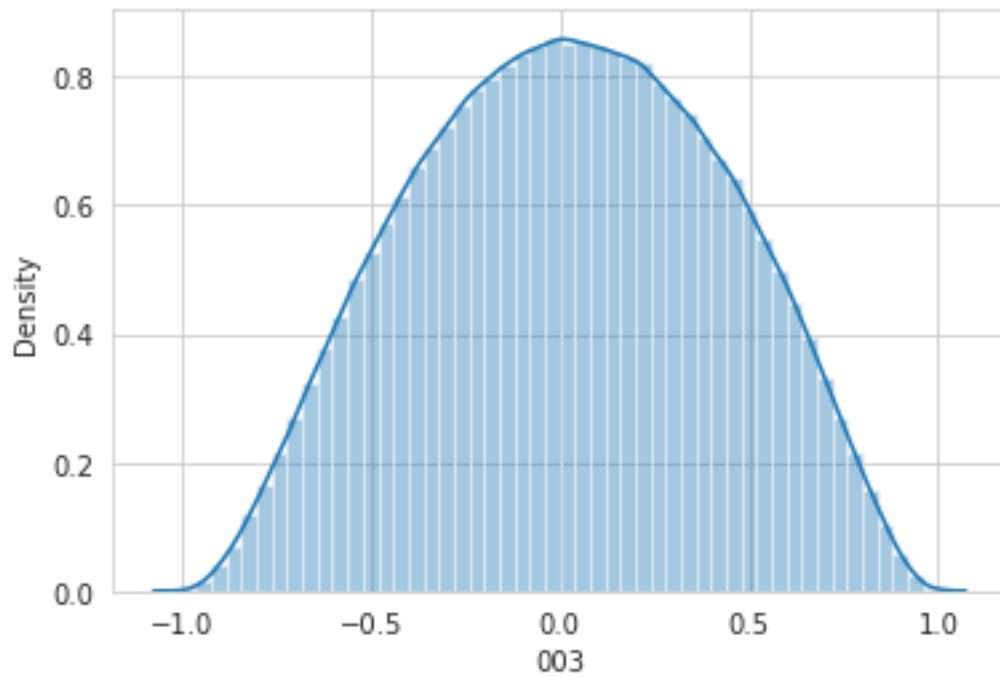
```
[48]: alpha = 3
```

```
[49]: %%time  
alphas[f'{alpha:03}'] = alpha003(o, v)
```

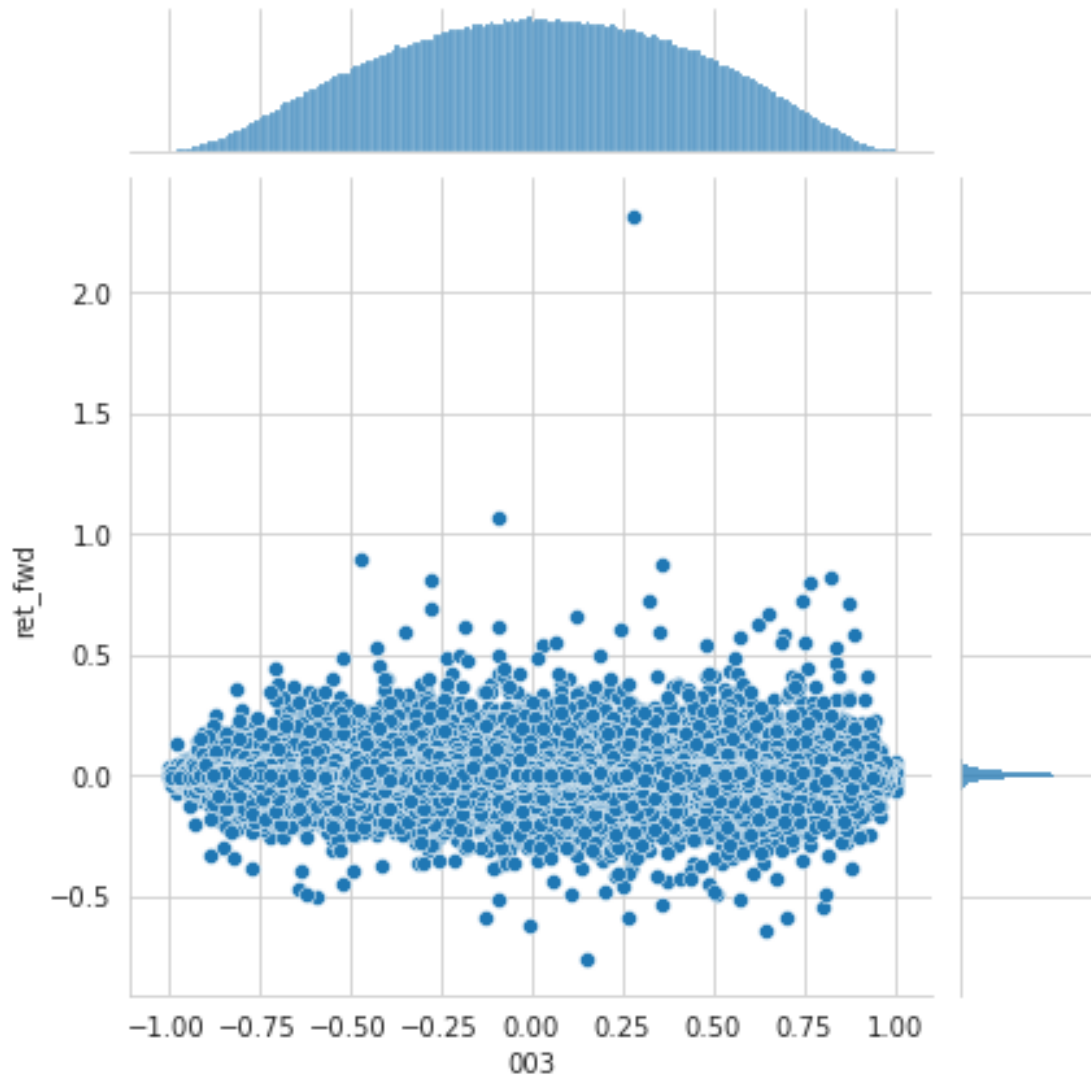
CPU times: user 4.03 s, sys: 112 ms, total: 4.14 s
Wall time: 4.17 s

```
[50]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[51]: sns.distplot(alphas[f'{alpha:03}'].clip(lower=-1));
```



```
[52]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[53]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

[53]: 0

1.8 Alpha 004

```
(-1 * Ts_Rank(rank(low), 9))
```

```
[54]: def alpha004(l):
    """(-1 * Ts_Rank(rank(low), 9))"""
    return (-ts_rank(rank(l), 9)
            .stack('ticker')
            .swaplevel())
```

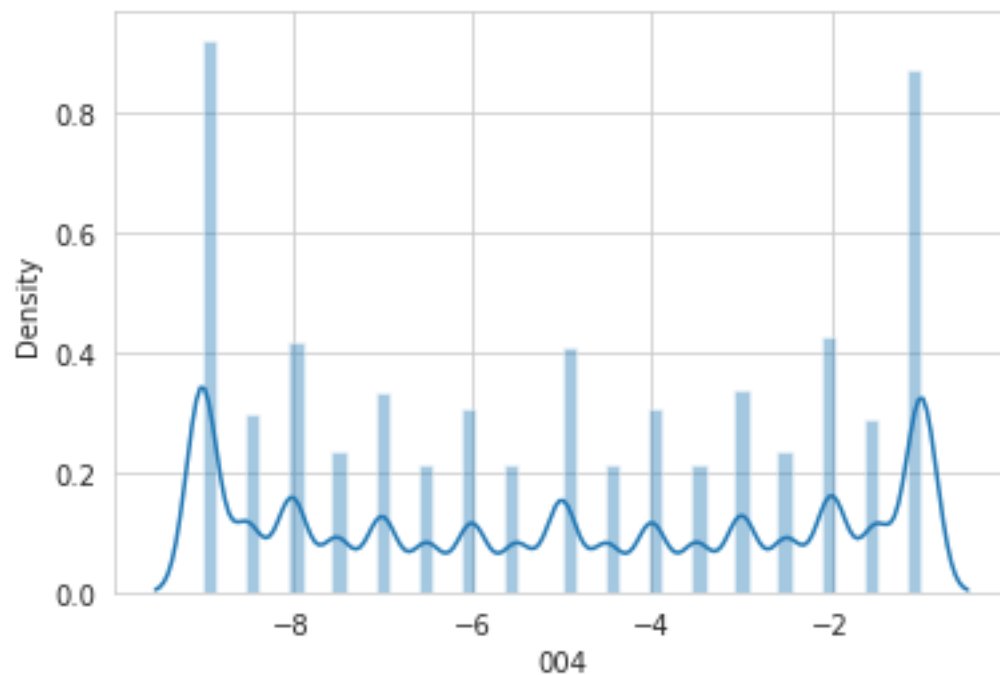
```
[55]: alpha = 4
```

```
[56]: %%time  
alphas[f'{alpha:03}'] = alpha004(1)
```

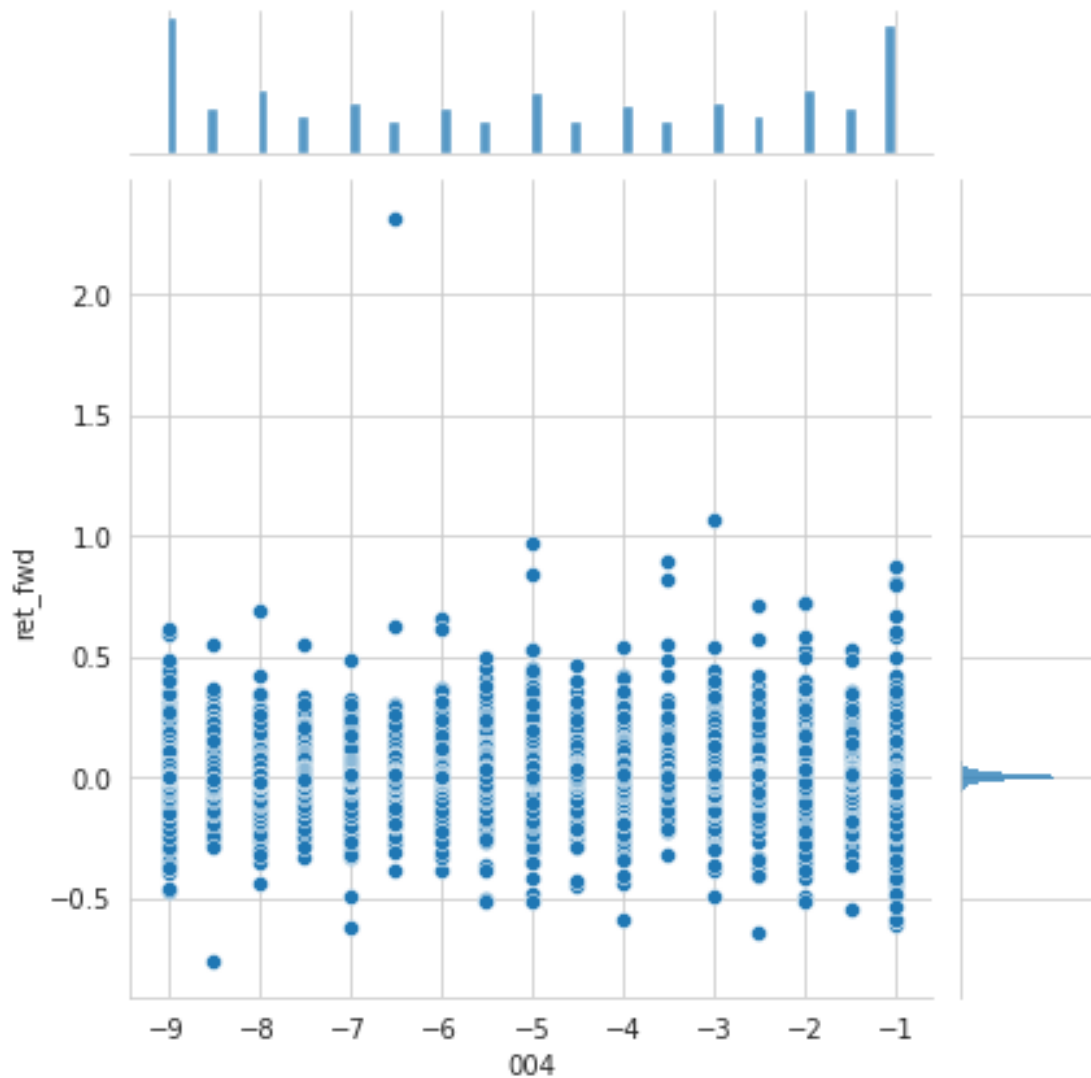
CPU times: user 3min 3s, sys: 62.4 ms, total: 3min 3s
Wall time: 3min 3s

```
[57]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[58]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[59]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[60]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[60]: 0
```

1.9 Alpha 005

Very roughly approximating wvap as average of OHLC.

```
(rank((open - (sum(vwap, 10) / 10))) * (-1 * abs(rank((close - vwap)))))
```

```
[61]: def alpha005(o, vwap, c):
        """(rank((open - ts_mean(vwap, 10))) * (-1 * abs(rank((close - vwap)))))"""
        return (rank(o.sub(ts_mean(vwap, 10)))
```



```
.mul(rank(c.sub(vwap)).mul(-1).abs())
.stack('ticker')
.swaplevel()
```

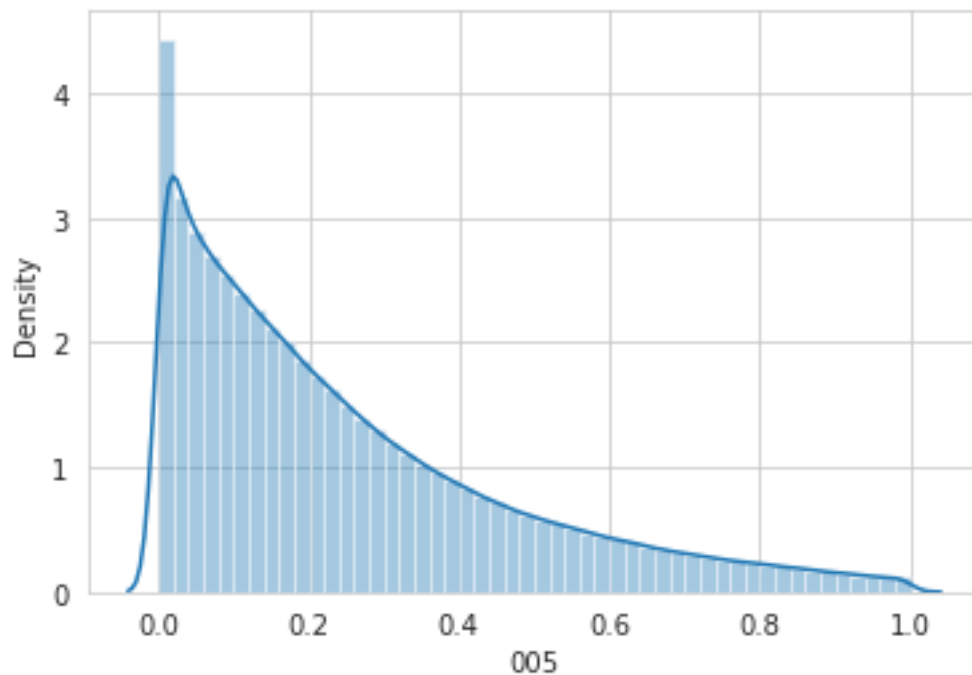
```
[62]: alpha = 5
```

```
[63]: %%time
alphas[f'{alpha:03}'] = alpha005(o, vwap, c)
```

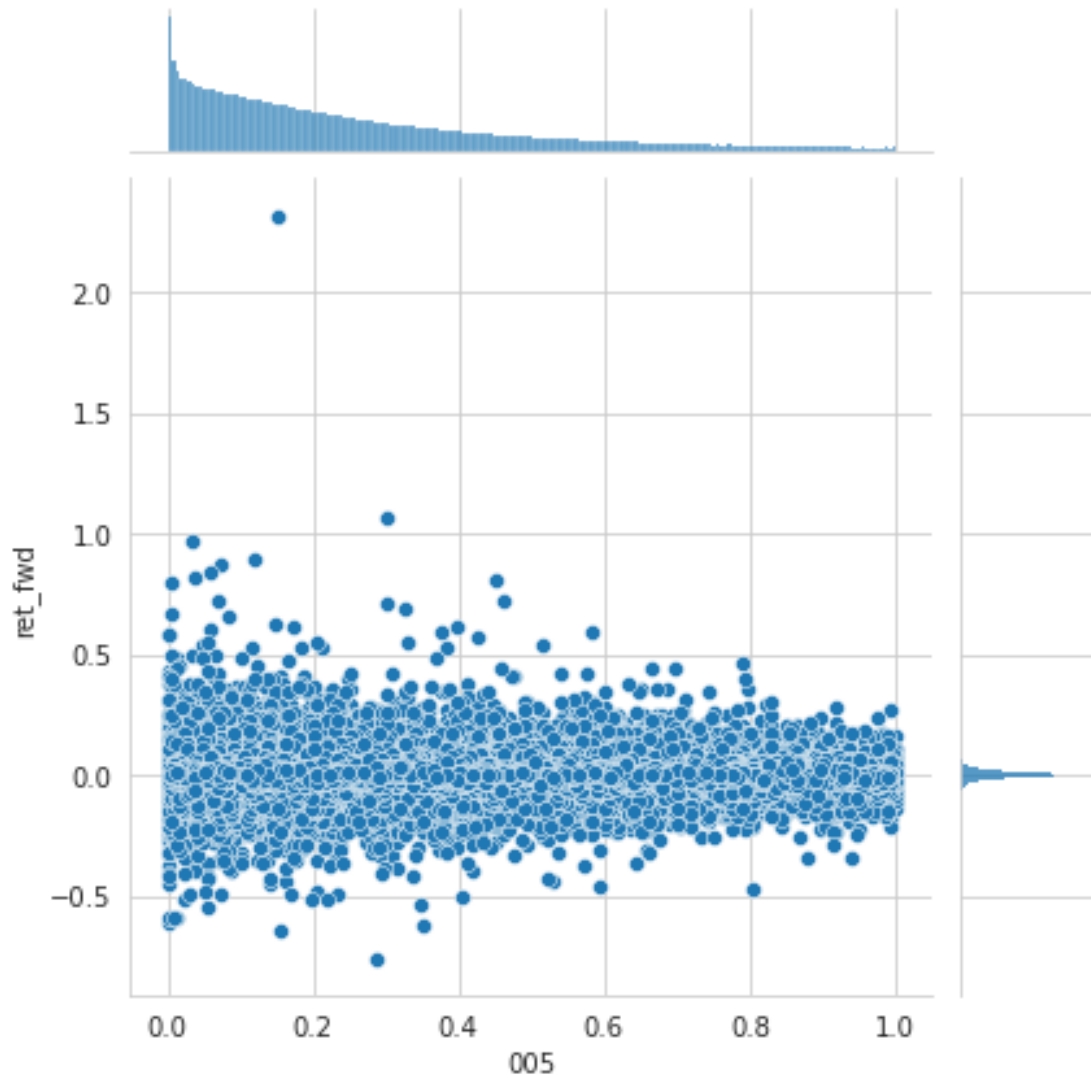
CPU times: user 2.21 s, sys: 12 ms, total: 2.22 s
Wall time: 2.18 s

```
[64]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[65]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[66]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[67]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[67]: 0
```

1.10 Alpha 006

```
-ts_corr(open, volume, 10)
```

```
[68]: def alpha006(o, v):
        """(-ts_corr(open, volume, 10))"""
        return (-ts_corr(o, v, 10)
                .stack('ticker')
                .swaplevel())
```

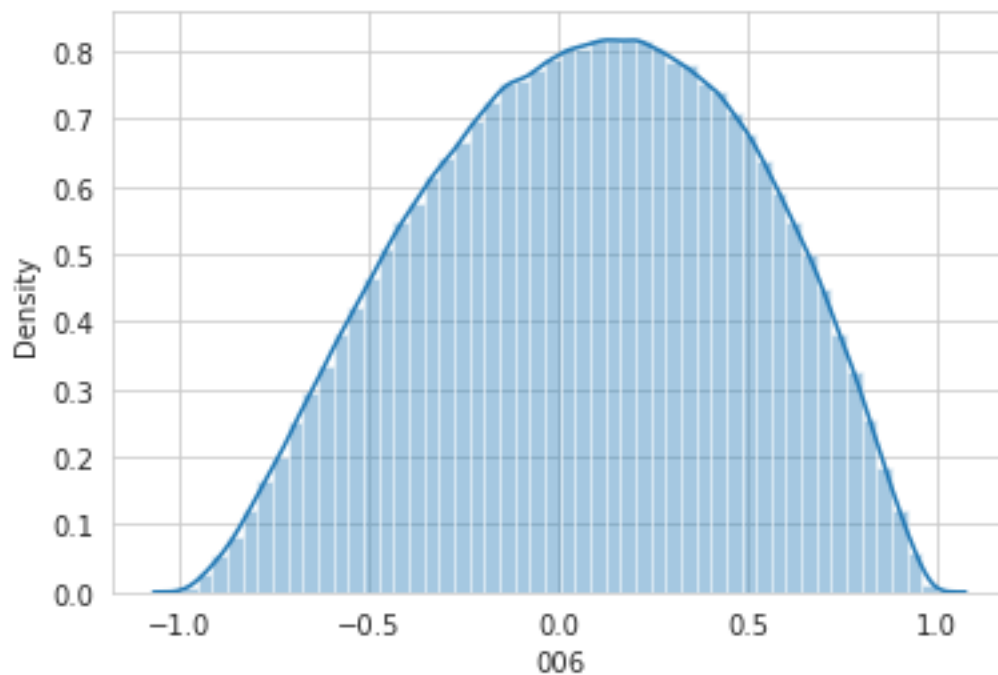
```
[69]: alpha = 6
```

```
[70]: %%time  
alphas[f'{alpha:03}'] = alpha006(o, v)
```

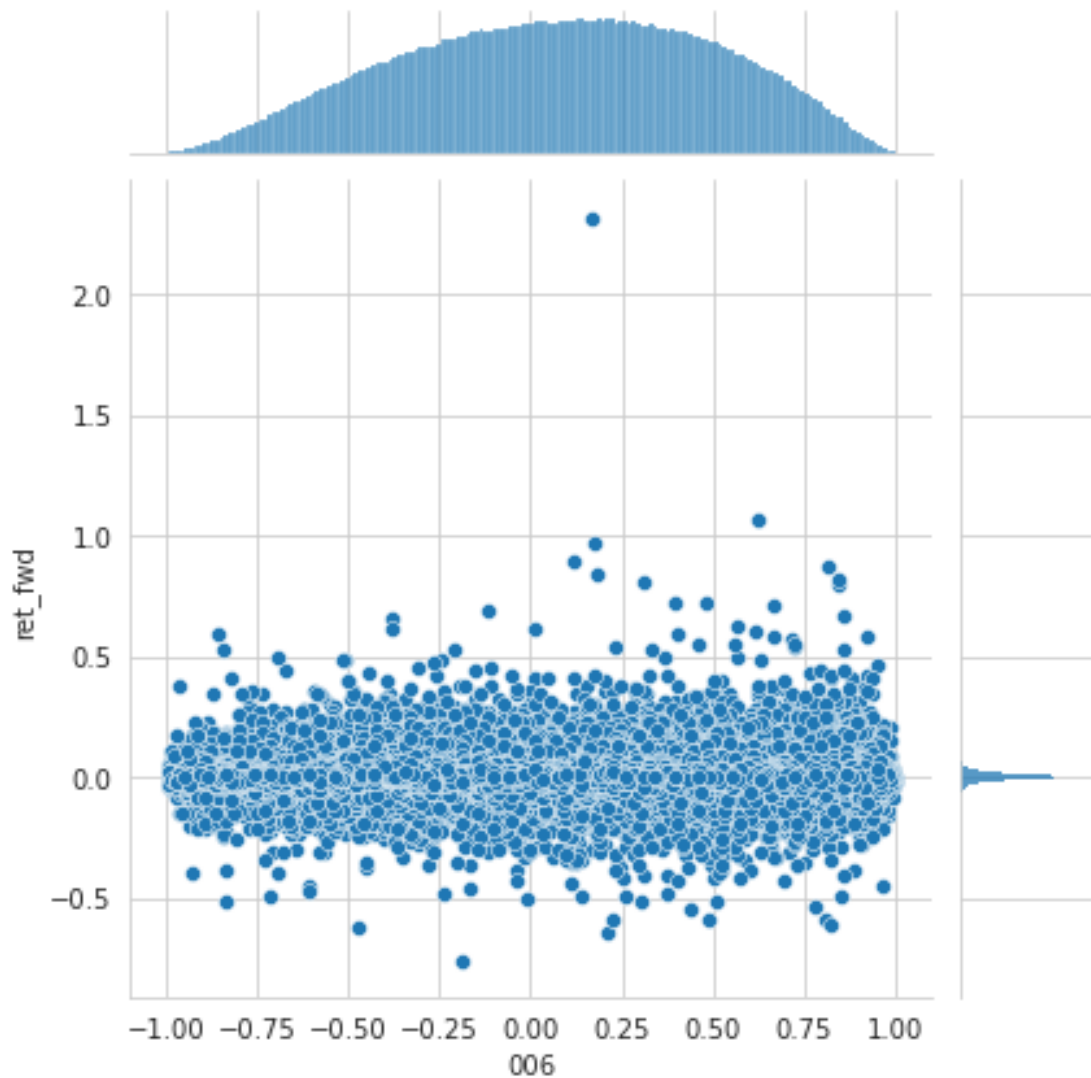
CPU times: user 3.54 s, sys: 48.2 ms, total: 3.59 s
Wall time: 3.56 s

```
[71]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[72]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[73]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[74]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[75]: mi[alpha]
```

```
[75]: 0.0026892751534788317
```

1.11 Alpha 007

```
(adv20 < volume)
? ((-1 * ts_rank(abs(ts_delta(close, 7)), 60)) * sign(ts_delta(close, 7)))
: -1
```

```
[76]: def alpha007(c, v, adv20):
      """(adv20 < volume)
```

```

        ? ((-ts_rank(abs(ts_delta(close, 7)), 60)) * sign(ts_delta(close, 7)))
        : -1
    """

    delta7 = ts_delta(c, 7)
    return (-ts_rank(abs(delta7), 60)
            .mul(sign(delta7))
            .where(adv20<v, -1)
            .stack('ticker')
            .swaplevel())

```

```
[77]: alpha = 7
```

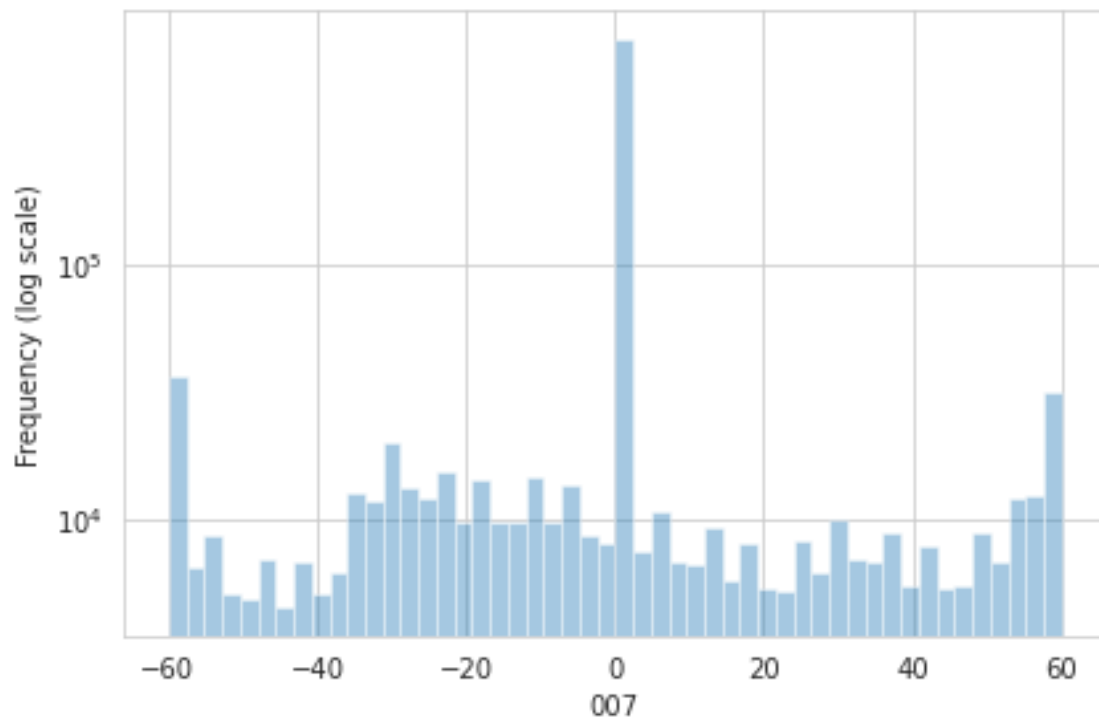
```
[78]: %%time
      alphas[f'{alpha:03}'] = alpha007(c, v, adv20)
```

CPU times: user 3min 1s, sys: 83.5 ms, total: 3min 1s

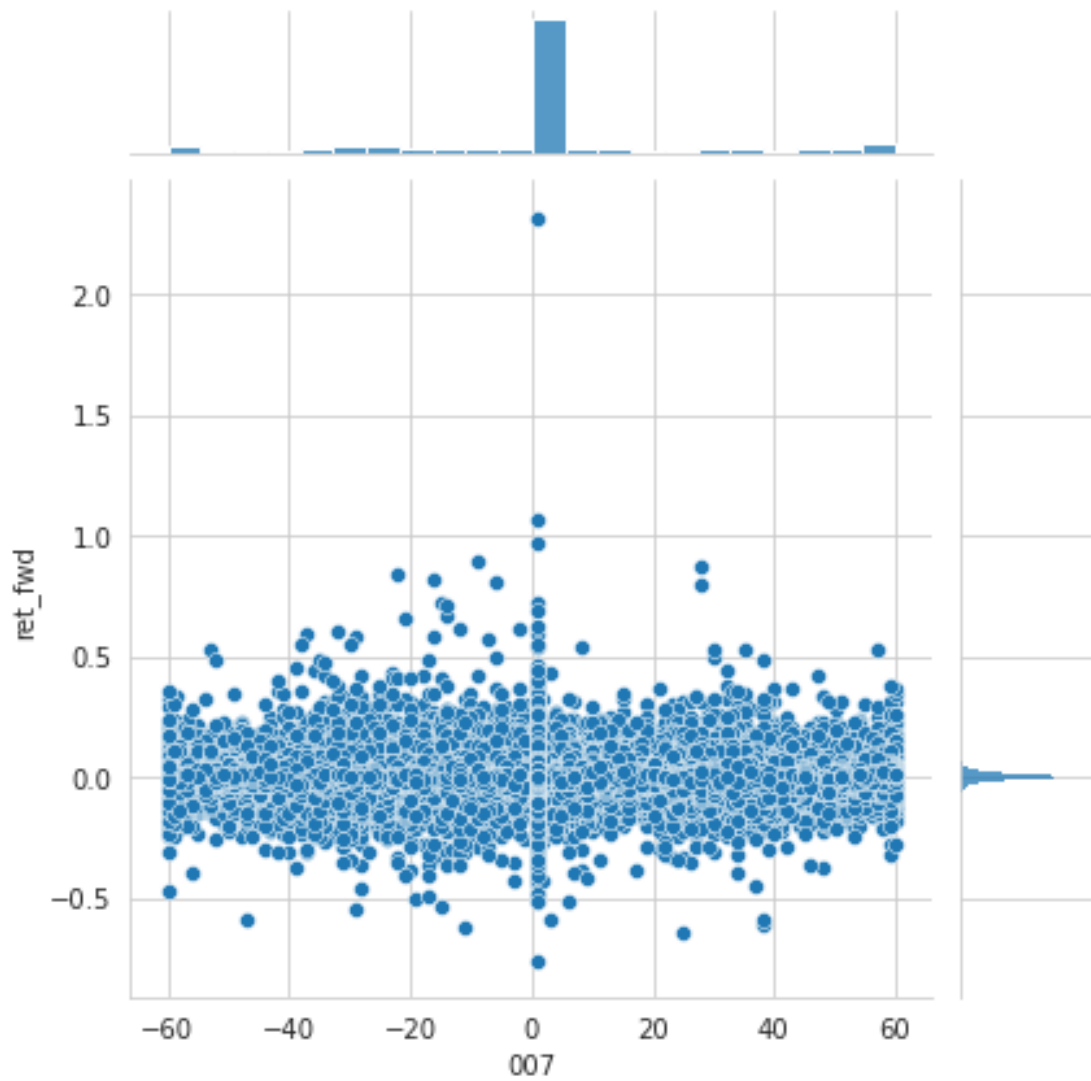
Wall time: 3min 1s

```
[79]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[80]: ax = sns.distplot(alphas[f'{alpha:03}'], kde=False)
      ax.set_yscale('log')
      ax.set_ylabel('Frequency (log scale)')
      plt.tight_layout();
```



```
[81]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[82]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[83]: # mi[alpha]
```

1.12 Alpha 008

```
-rank(((ts_sum(open, 5) * ts_sum(returns, 5)) - ts_lag((ts_sum(open, 5) * ts_sum(returns, 5))),
```

```
[84]: def alpha008(o, r):
        """-rank(((ts_sum(open, 5) * ts_sum(returns, 5)) -
                    ts_lag((ts_sum(open, 5) * ts_sum(returns, 5)),10)))
        """
        return (-(rank(((ts_sum(o, 5) * ts_sum(r, 5)) -
                        ts_lag((ts_sum(o, 5) * ts_sum(r, 5)), 10))))
                .stack('ticker')
                .swaplevel())
```

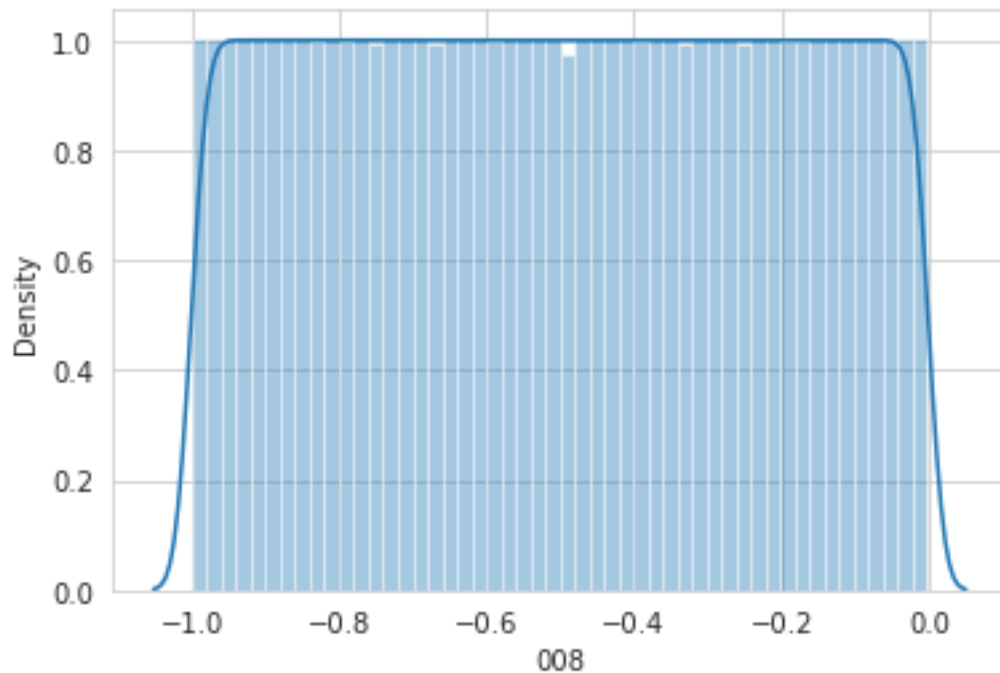
```
[85]: alpha = 8
```

```
[86]: %%time
        alphas[f'{alpha:03}'] = alpha008(o, r)
```

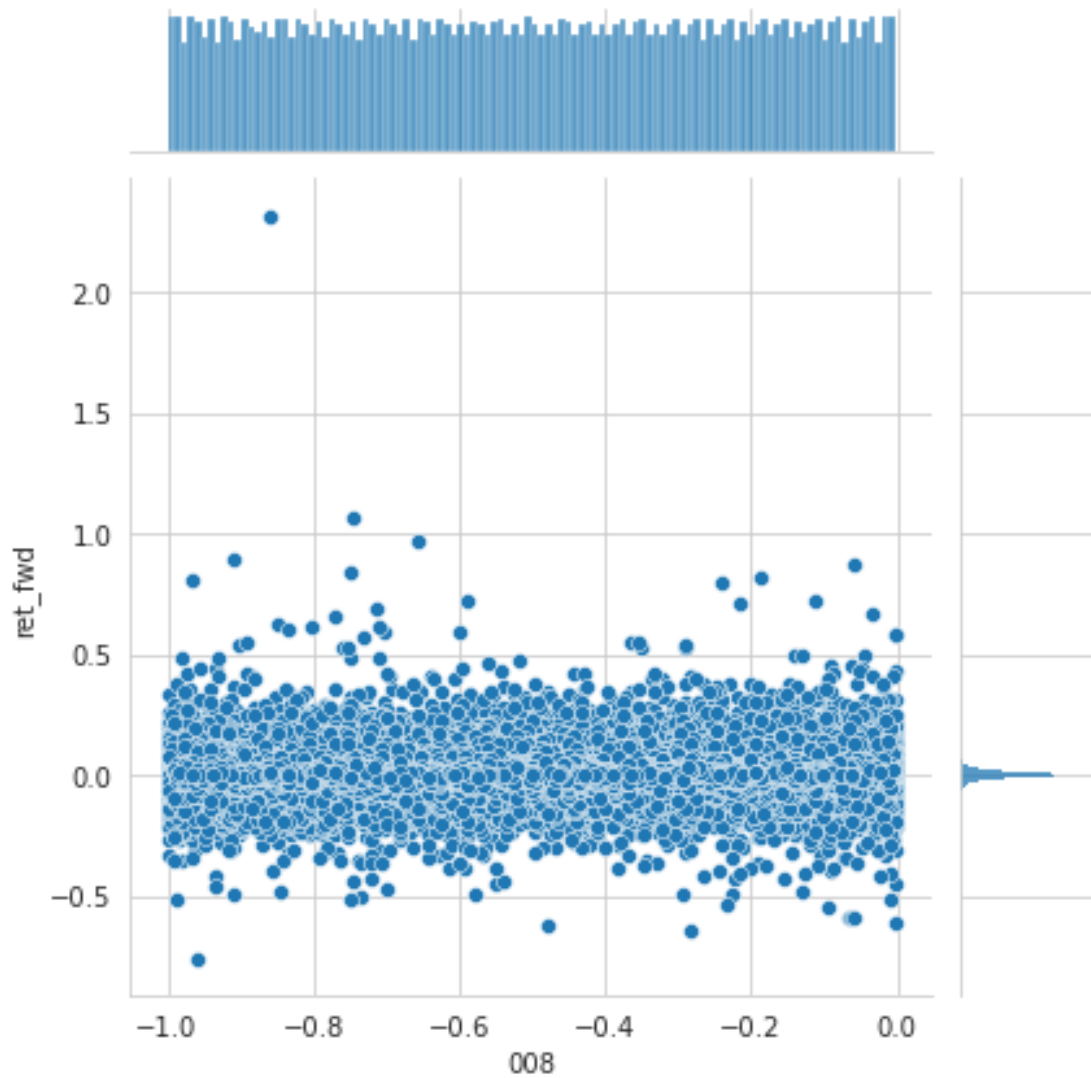
CPU times: user 2.15 s, sys: 12 ms, total: 2.16 s
Wall time: 2.13 s

```
[87]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[88]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[89]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[90]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[91]: mi[alpha]
```

```
[91]: 0
```

1.13 Alpha 009

```
(0 < ts_min(ts_delta(close, 1), 5)) ? ts_delta(close, 1)
: ((ts_max(ts_delta(close, 1), 5) < 0)
? ts_delta(close, 1) : (-1 * ts_delta(close, 1)))
```



```
[92]: def alpha009(c):
      """(0 < ts_min(ts_delta(close, 1), 5)) ? ts_delta(close, 1)
      : ((ts_max(ts_delta(close, 1), 5) < 0)
      ? ts_delta(close, 1) : (-1 * ts_delta(close, 1)))
      """
      close_diff = ts_delta(c, 1)
      alpha = close_diff.where(ts_min(close_diff, 5) > 0,
                              close_diff.where(ts_max(close_diff, 5) < 0,
                                                    -close_diff))

      return (alpha
              .stack('ticker')
              .swaplevel())
```

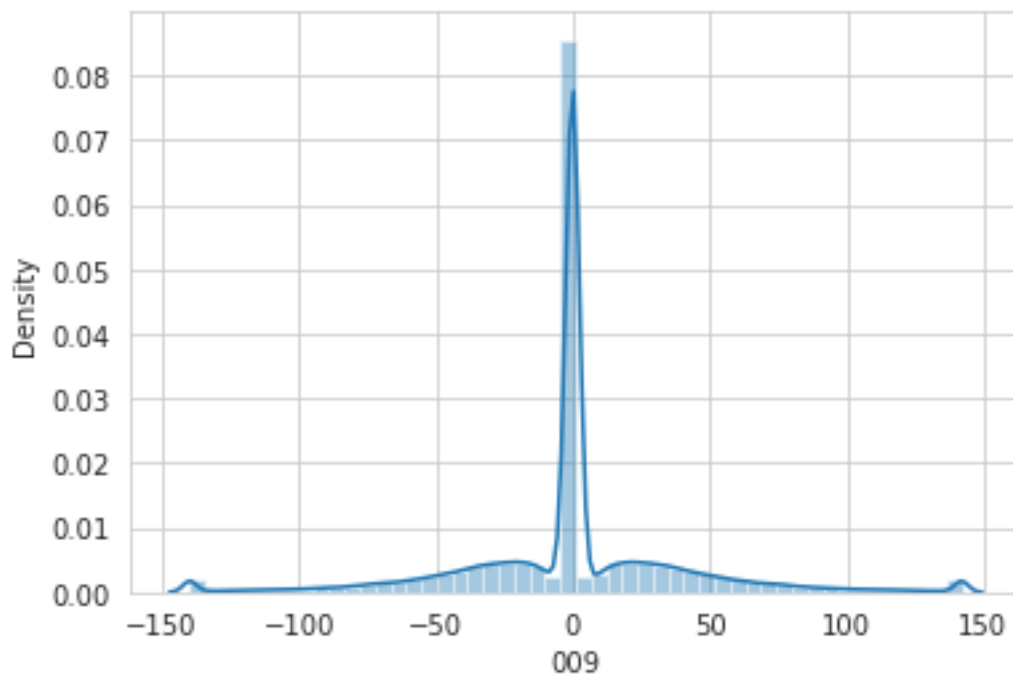
```
[93]: alpha = 9
```

```
[94]: %%time
      alphas[f'{alpha:03}'] = alpha009(c)
```

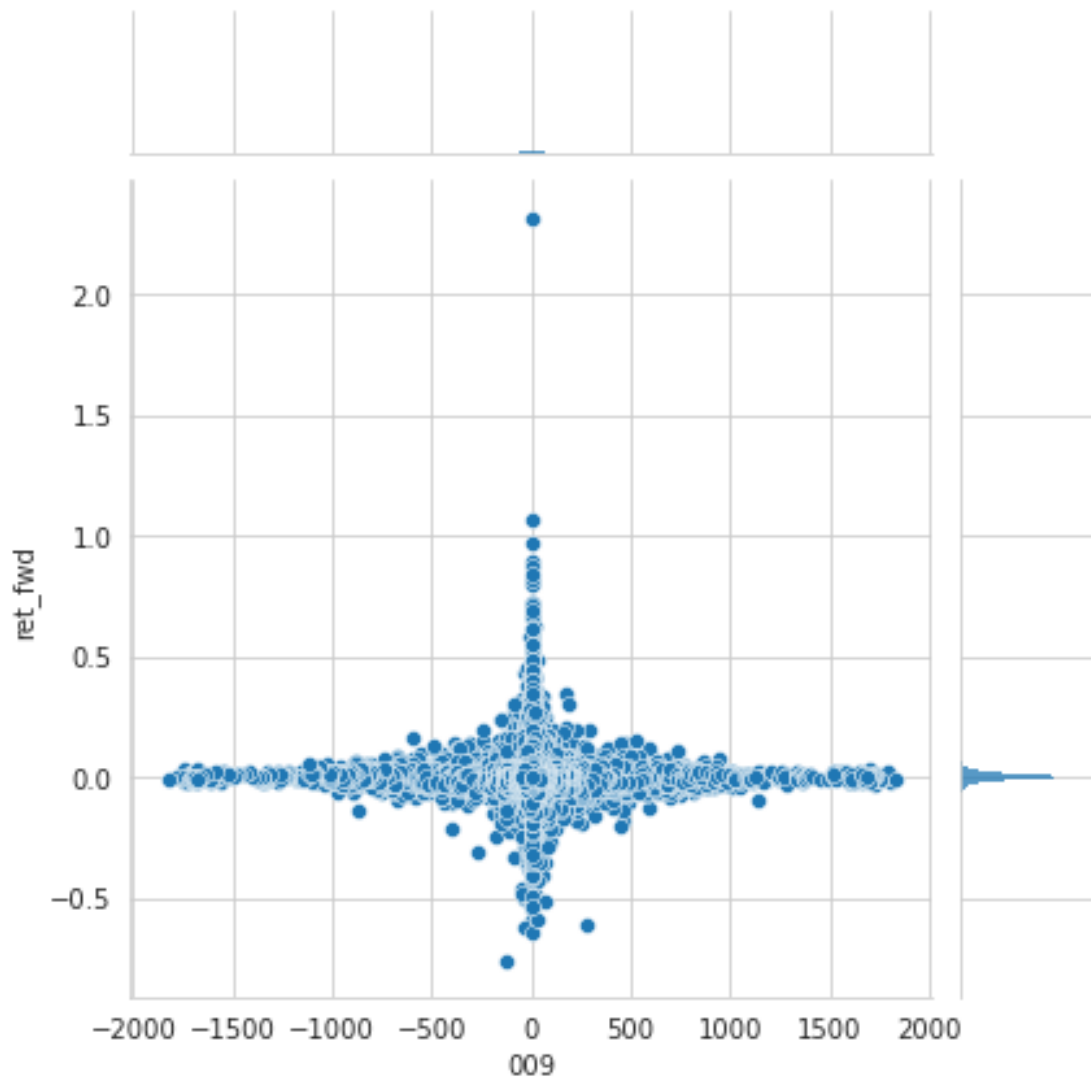
CPU times: user 2.01 s, sys: 20.1 ms, total: 2.03 s
 Wall time: 2.01 s

```
[95]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[96]: q = 0.01
      sns.distplot(alphas[f'{alpha:03}'].clip(lower=alphas[f'{alpha:03}'].quantile(q),
                                                upper=alphas[f'{alpha:03}'].
        ↪ quantile(1-q)));
```



```
[97]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[98]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[99]: mi[alpha]
```

```
[99]: 0.02205990873050201
```

```
[100]: pd.Series(mi)
```

```
[100]: 1    0.018886
      2    0.001240
      3    0.000000
      4    0.000000
      5    0.000000
      6    0.002689
      8    0.000000
      9    0.022060
      dtype: float64
```

1.14 Alpha 010

```
rank(((0 < ts_min(ts_delta(close, 1), 4))
? ts_delta(close, 1)
: ((ts_max(ts_delta(close, 1), 4) < 0)
? ts_delta(close, 1)
: (-1 * ts_delta(close, 1)))))
```

```
[101]: def alpha010(c):
        """rank(((0 < ts_min(ts_delta(close, 1), 4))
        ? ts_delta(close, 1)
        : ((ts_max(ts_delta(close, 1), 4) < 0)
        ? ts_delta(close, 1)
        : (-1 * ts_delta(close, 1)))))
        """
        close_diff = ts_delta(c, 1)
        alpha = close_diff.where(ts_min(close_diff, 4) > 0,
                                close_diff.where(ts_min(close_diff, 4) > 0,
                                                    -close_diff))

        return (rank(alpha)
                .stack('ticker')
                .swaplevel())
```

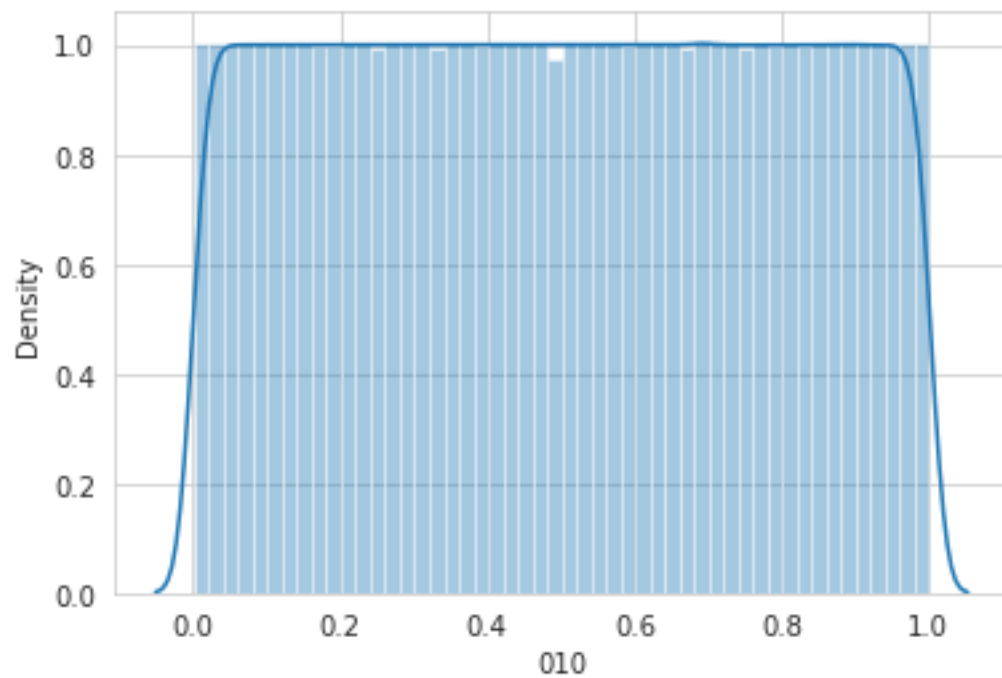
```
[102]: alpha = 10
```

```
[103]: %%time
        alphas[f'{alpha:03}'] = alpha010(c)
```

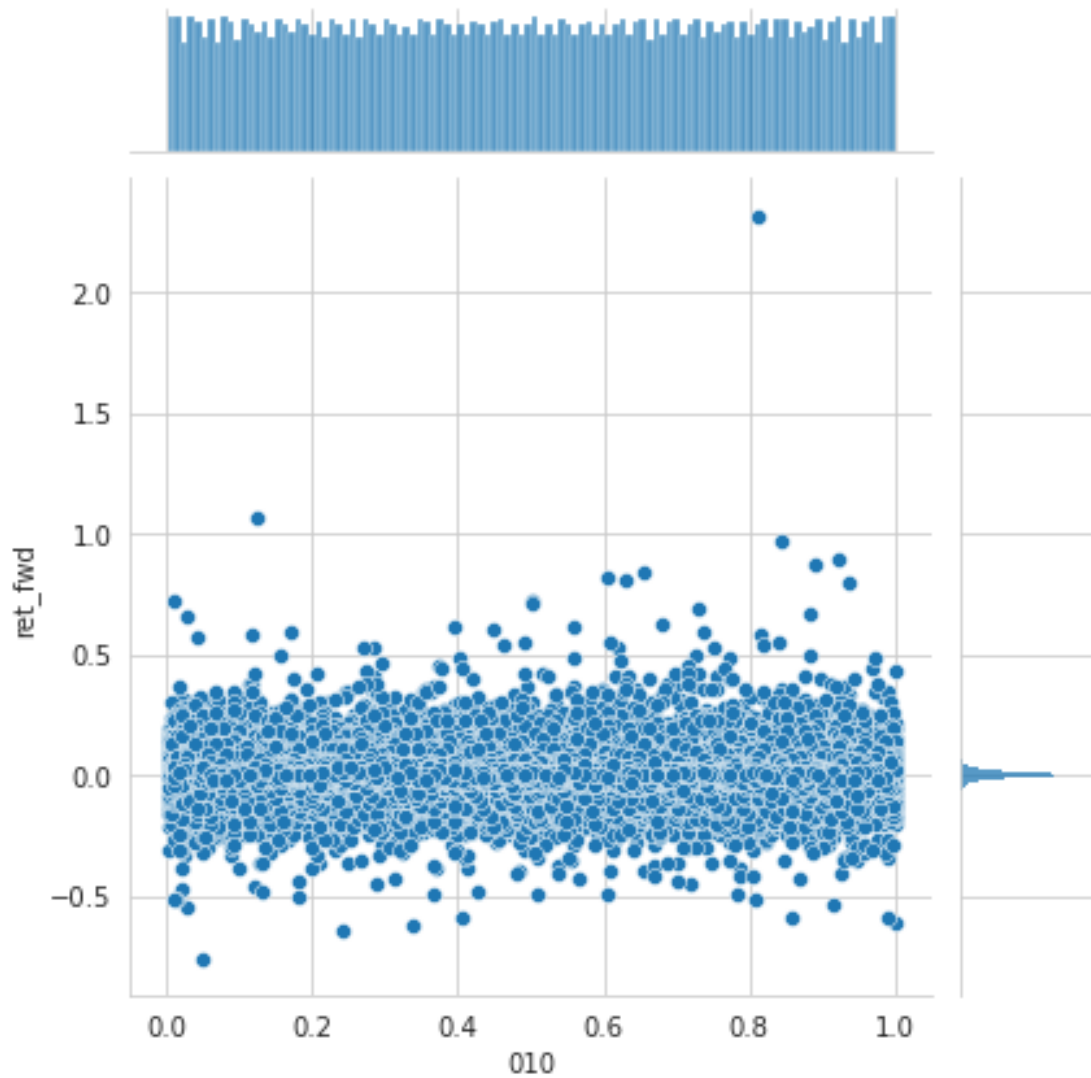
```
CPU times: user 2.67 s, sys: 24.1 ms, total: 2.7 s
Wall time: 2.67 s
```

```
[104]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[105]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[106]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[107]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[108]: mi[alpha]
```

```
[108]: 0
```

```
[109]: pd.Series(mi).to_csv('mi.csv')
```

1.15 Alpha 011

```
((rank(ts_max((vwap - close), 3)) + rank(ts_min((vwap - close), 3))) * rank(ts_delta(volume, 3)))
```

```
[110]: def alpha011(c, vwap, v):
        """(rank(ts_max((vwap - close), 3)) +
            rank(ts_min(vwap - close), 3)) *
            rank(ts_delta(volume, 3))
        """
        return (rank(ts_max(vwap.sub(c), 3))
                .add(rank(ts_min(vwap.sub(c), 3)))
                .mul(rank(ts_delta(v, 3)))
                .stack('ticker')
                .swaplevel())
```

```
[111]: alpha = 11
```

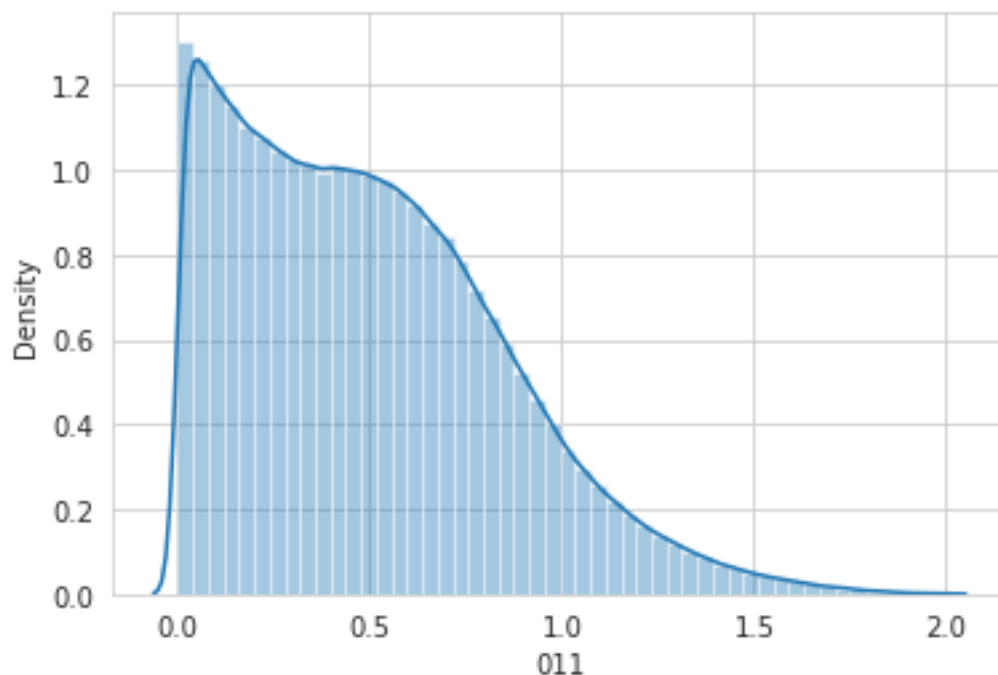
```
[112]: %%time
        alphas[f'{alpha:03}'] = alpha011(c, vwap, v)
```

CPU times: user 2.48 s, sys: 64.2 ms, total: 2.55 s

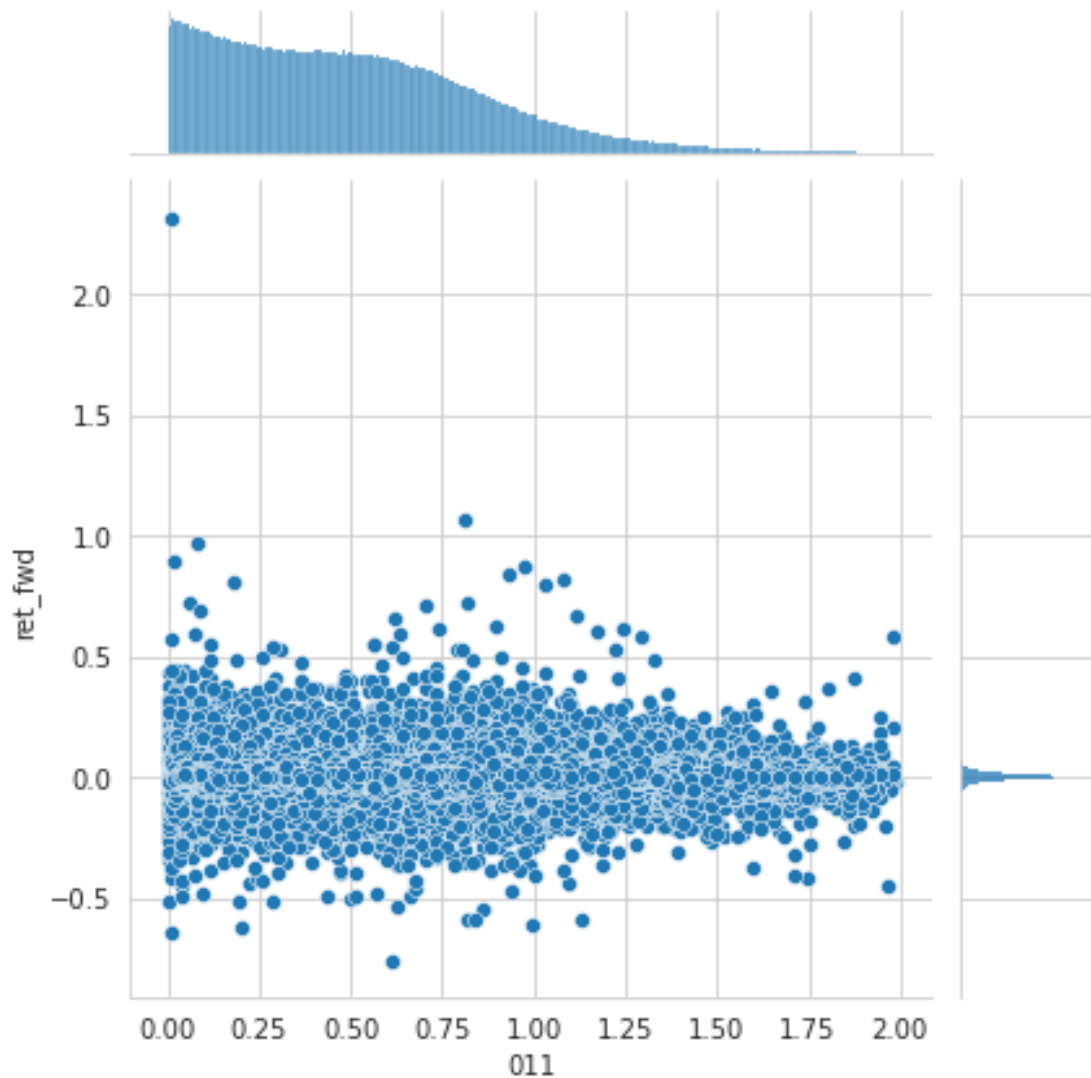
Wall time: 2.49 s

```
[113]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[114]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[115]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[116]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[117]: mi[alpha]
```

```
[117]: 0.0032785883763768453
```

1.16 Alpha 012

```
sign(ts_delta(volume, 1)) * -ts_delta(close, 1)
```

```
[118]: def alpha012(v, c):
        """(sign(ts_delta(volume, 1)) *
            (-1 * ts_delta(close, 1)))
        """
```

```

return (sign(ts_delta(v, 1)).mul(-ts_delta(c, 1))
        .stack('ticker')
        .swaplevel())

```

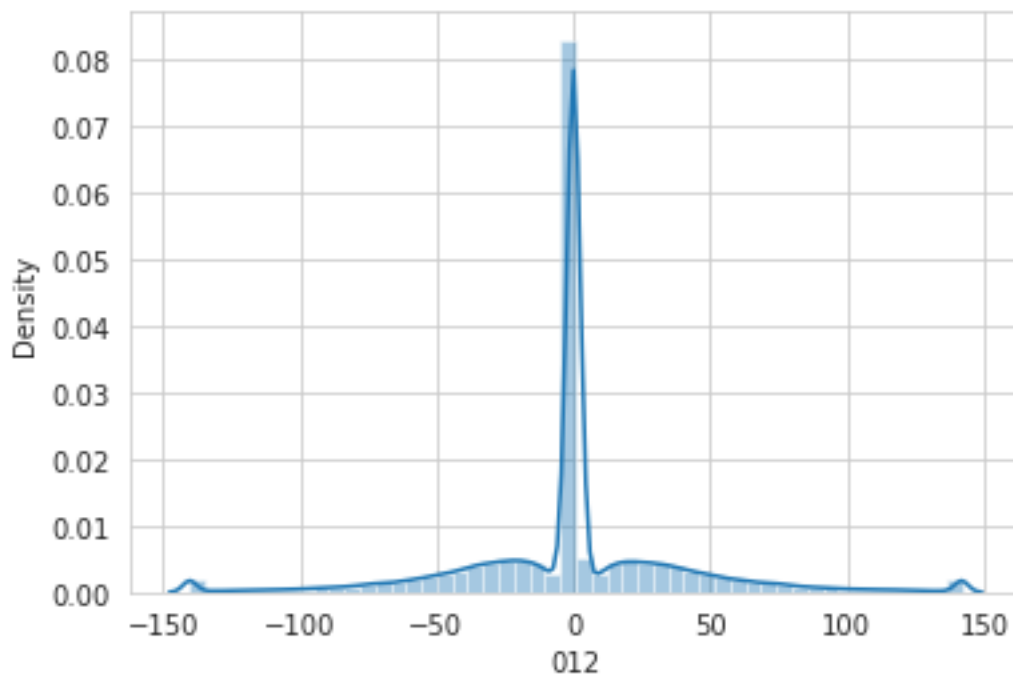
```
[119]: alpha = 12
```

```
[120]: %%time
alphas[f'{alpha:03}'] = alpha012(v, c)
```

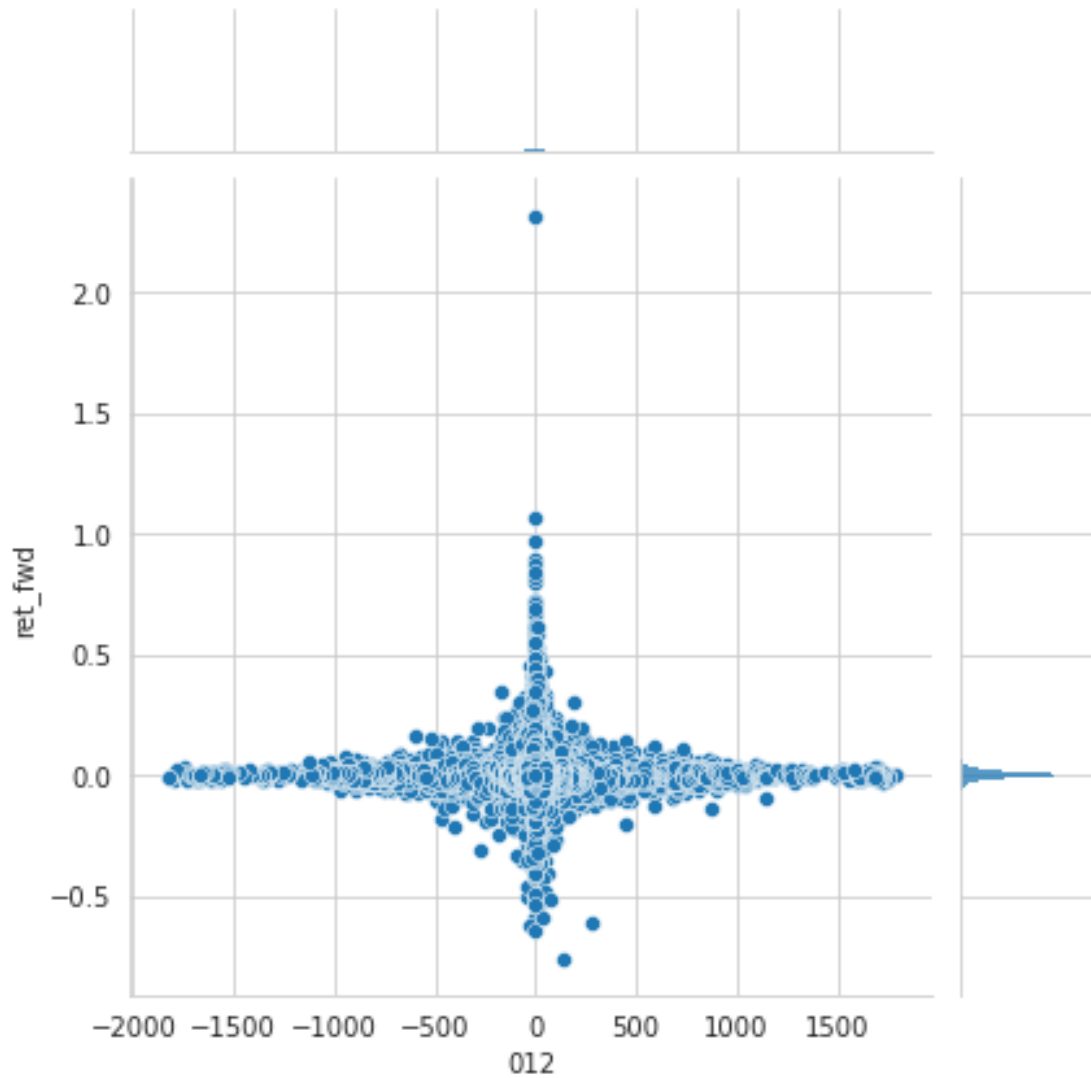
CPU times: user 2.03 s, sys: 8.03 ms, total: 2.04 s
Wall time: 2.03 s

```
[121]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[122]: q = 0.01
sns.distplot(alphas[f'{alpha:03}'].clip(lower=alphas[f'{alpha:03}'].quantile(q),
        upper=alphas[f'{alpha:03}'].
        ↪quantile(1-q)));
```



```
[123]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```

```
[124]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[125]: mi[alpha]
```

```
[125]: 0.015514105621370788
```

1.17 Alpha 013

```
-rank(ts_cov(rank(close), rank(volume), 5))
```

```
[126]: def alpha013(c, v):
        """-rank(ts_cov(rank(close), rank(volume), 5))"""
        return (-rank(ts_cov(rank(c), rank(v), 5))
                .stack('ticker'))
```

```
.swaplevel()
```

```
[127]: alpha = 13
```

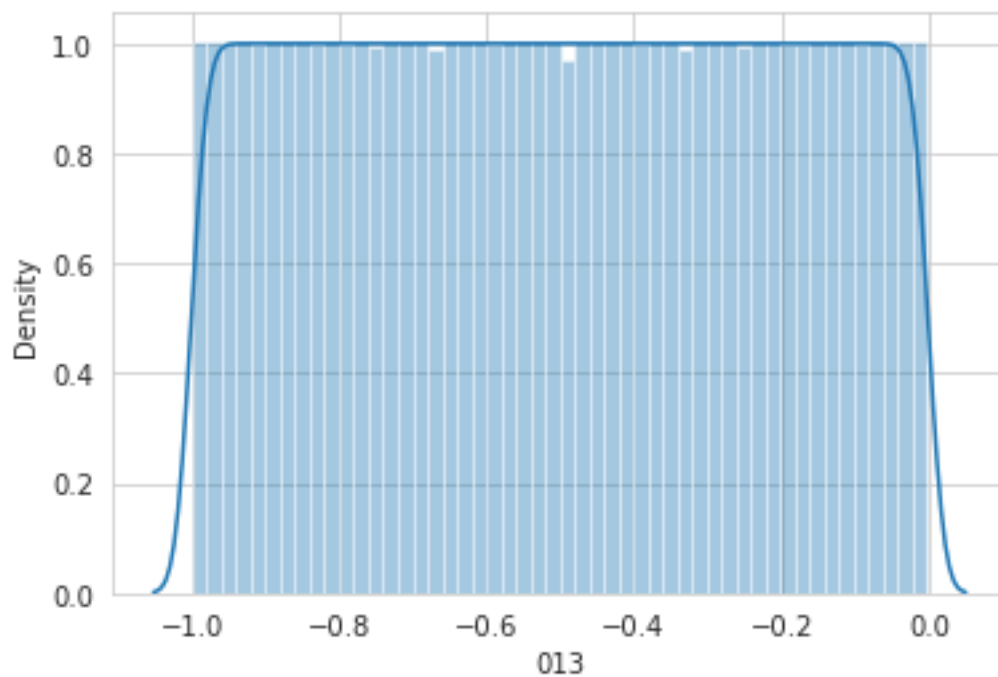
```
[128]: %%time  
alphas[f'{alpha:03}'] = alpha013(c, v)
```

CPU times: user 3.64 s, sys: 48.1 ms, total: 3.69 s

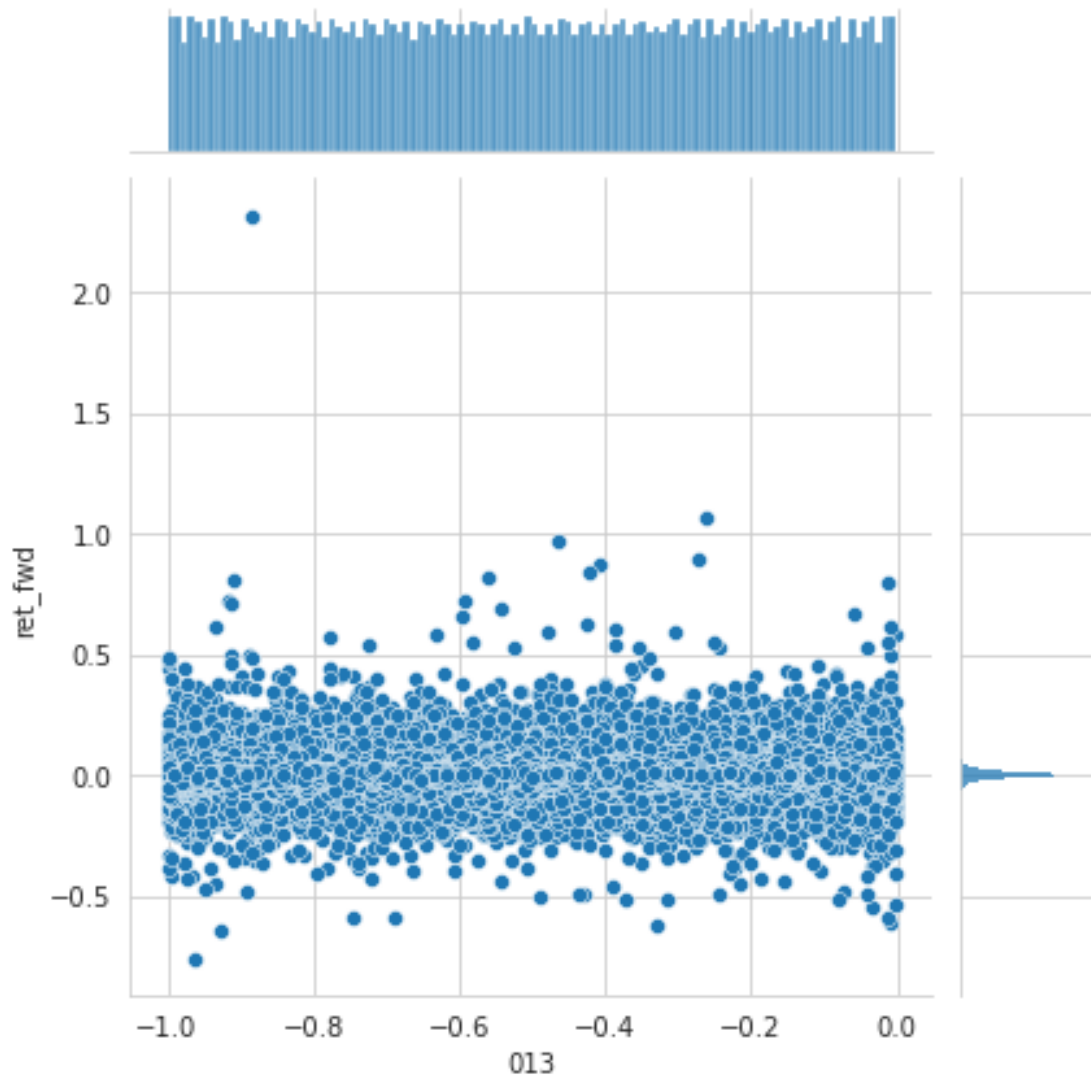
Wall time: 3.64 s

```
[129]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[130]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[131]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[132]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[133]: mi[alpha]
```

```
[133]: 0.0019222442390711691
```

```
[134]: pd.Series(mi).to_csv('mi.csv')
```

1.18 Alpha 014

```
(-rank(ts_delta(returns, 3))) * ts_corr(open, volume, 10))
```

```
[135]: def alpha014(o, v, r):
        """
        (-rank(ts_delta(returns, 3))) * ts_corr(open, volume, 10))
        """

        alpha = -rank(ts_delta(r, 3)).mul(ts_corr(o, v, 10)
                                           .replace([-np.inf,
                                                    np.inf],
                                                    np.nan))

        return (alpha
                .stack('ticker')
                .swaplevel())
```

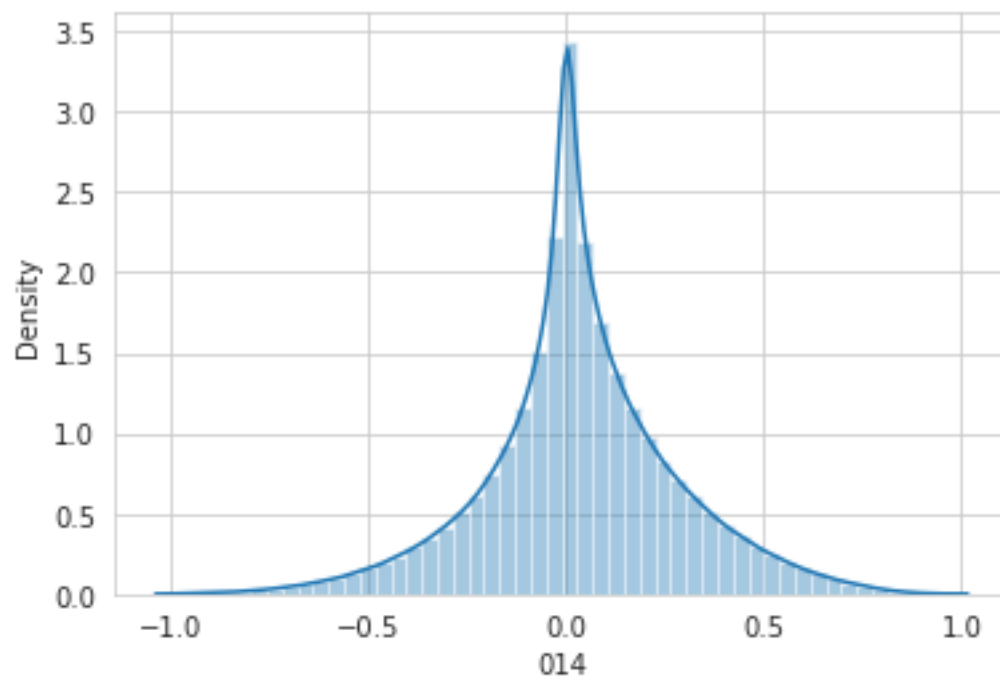
```
[136]: alpha = 14
```

```
[137]: %%time
        alphas[f'{alpha:03}'] = alpha014(o, v, r)
```

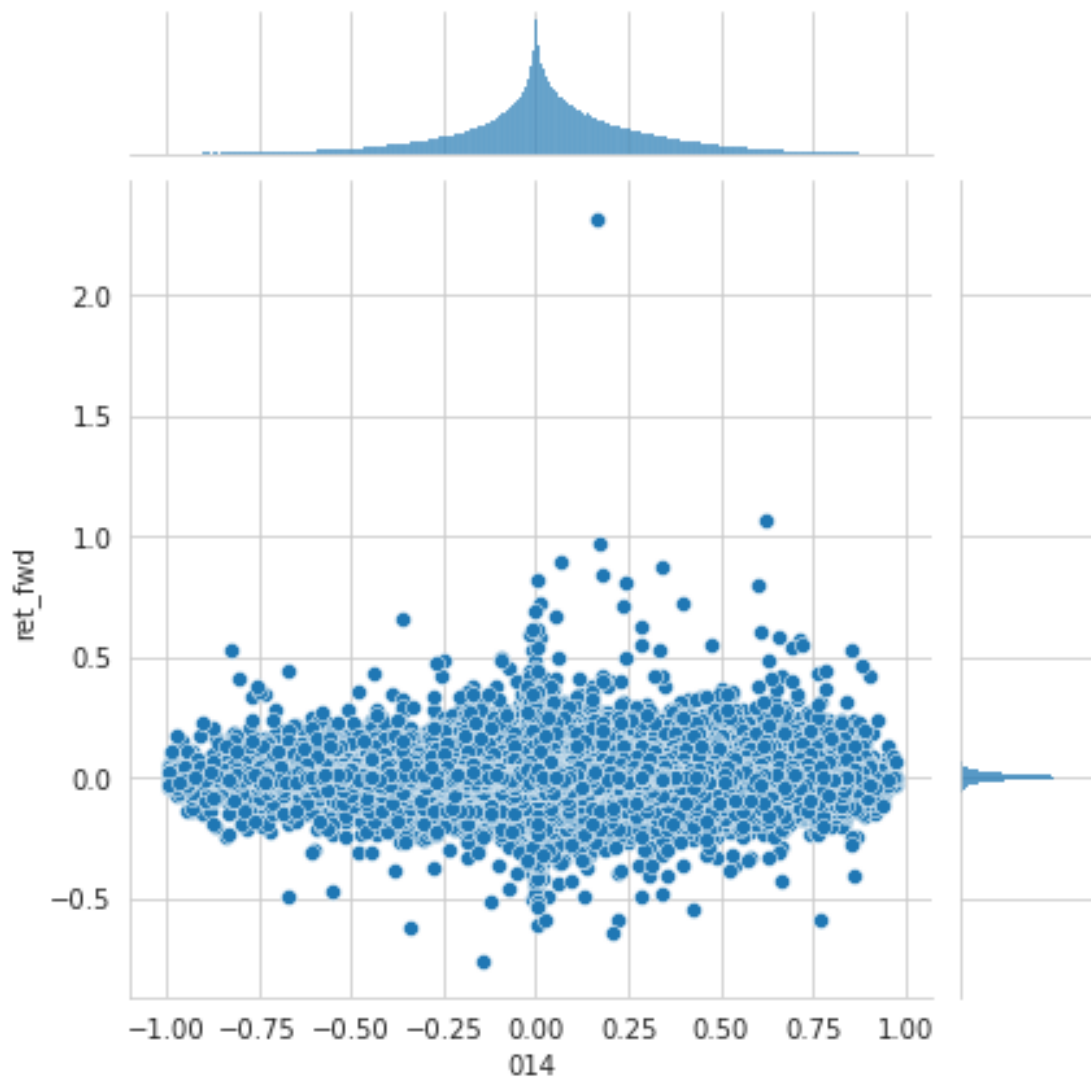
CPU times: user 3.58 s, sys: 36 ms, total: 3.62 s
Wall time: 3.58 s

```
[138]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[139]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[140]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[141]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[142]: mi[alpha]
```

```
[142]: 0.0009324039861251521
```

1.19 Alpha 015

```
(-1 * ts_sum(rank(ts_corr(rank(high), rank(volume), 3)), 3))
```

```
[143]: def alpha015(h, v):
        """(-1 * ts_sum(rank(ts_corr(rank(high), rank(volume), 3)), 3))"""
        alpha = (-ts_sum(rank(ts_corr(rank(h), rank(v), 3)
                           .replace([-np.inf, np.inf], np.nan)), 3))
        return (alpha
                .stack('ticker')
                .swaplevel())
```

```
[144]: alpha = 15
```

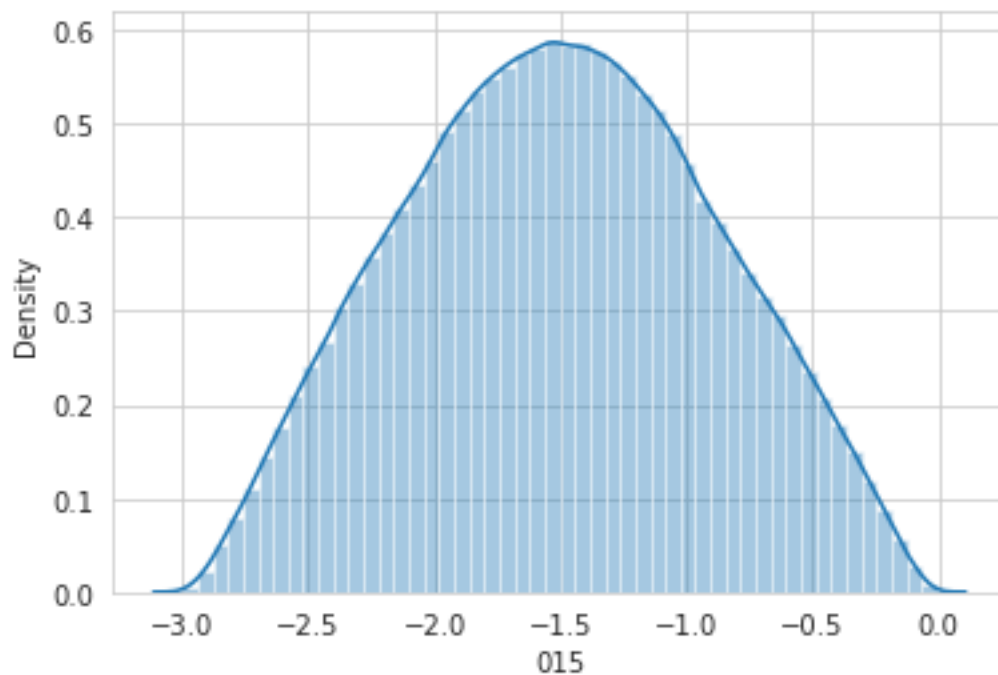
```
[145]: %%time
        alphas[f'{alpha:03}'] = alpha015(h, v)
```

CPU times: user 3.44 s, sys: 24.1 ms, total: 3.47 s

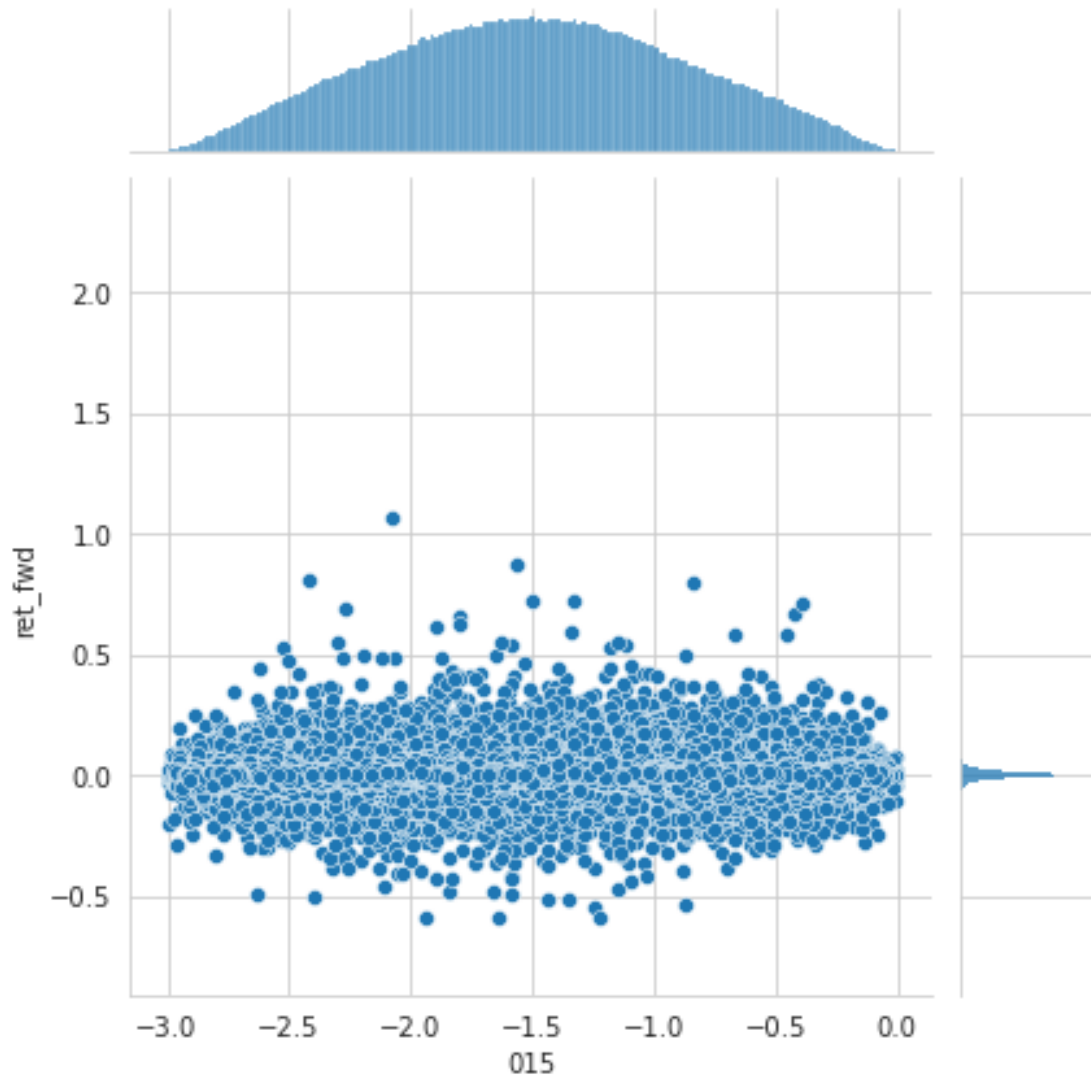
Wall time: 3.44 s

```
[146]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[147]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[148]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[149]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[150]: mi[alpha]
```

```
[150]: 0.0011393971145832182
```

1.20 Alpha 016

```
(-1 * rank(ts_cov(rank(high), rank(volume), 5)))
```

```
[151]: def alpha016(h, v):
        """(-1 * rank(ts_cov(rank(high), rank(volume), 5)))"""
        return (-rank(ts_cov(rank(h), rank(v), 5))
                .stack('ticker'))
```

```
.swaplevel()
```

```
[152]: alpha = 16
```

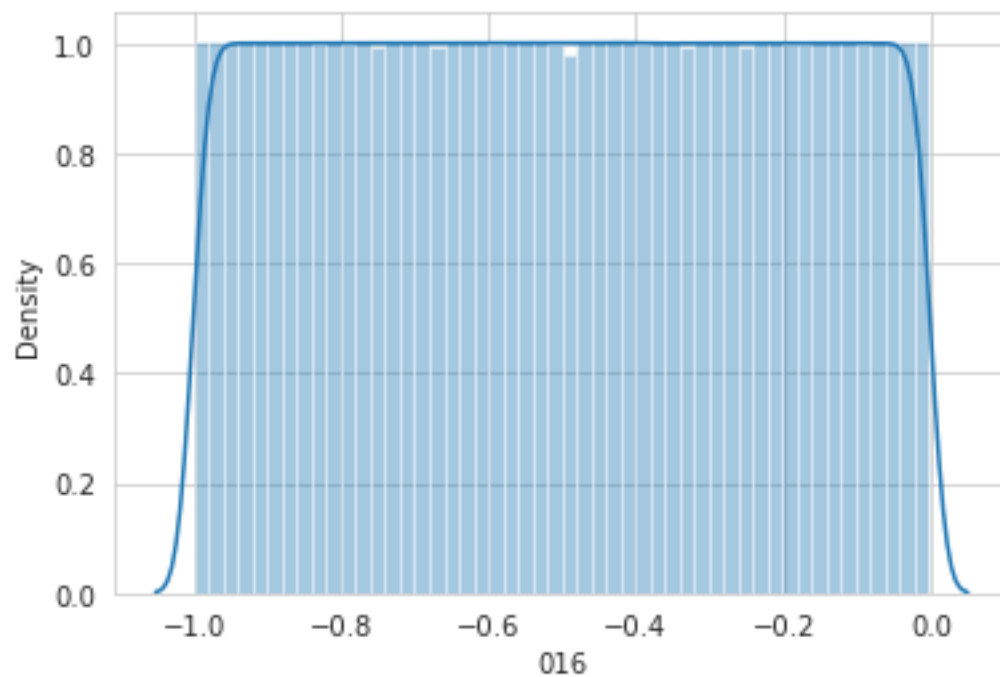
```
[153]: %%time  
alphas[f'{alpha:03}'] = alpha016(h, v)
```

CPU times: user 2.89 s, sys: 44.1 ms, total: 2.94 s

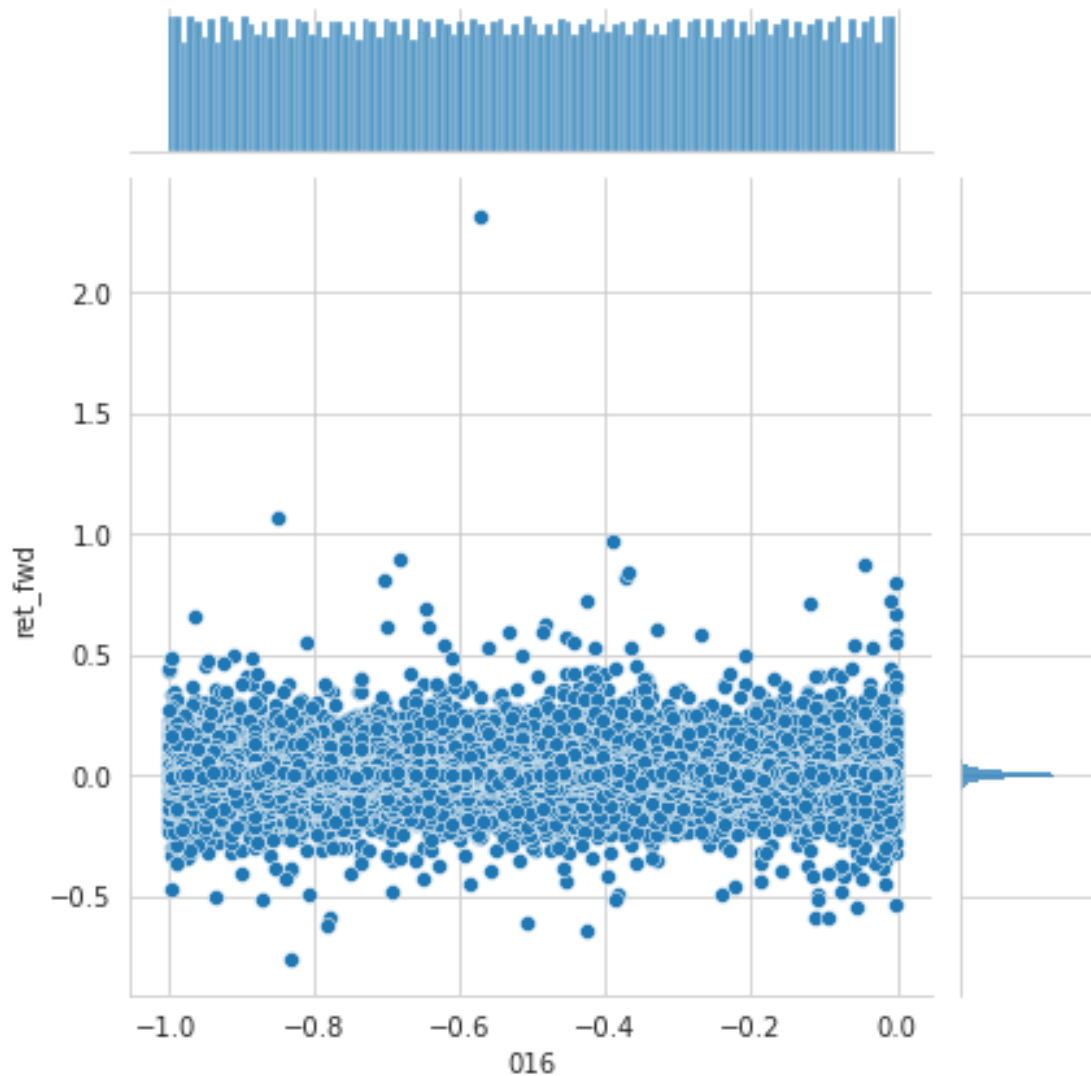
Wall time: 2.91 s

```
[154]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[155]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[156]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```

```
[157]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[158]: mi[alpha]
```

```
[158]: 0
```

```
[159]: pd.Series(mi).to_csv('mi.csv')
```

1.21 Alpha 017

```
rank(((0 < ts_min(ts_delta(close, 1), 4))
? ts_delta(close, 1)
: ((ts_max(ts_delta(close, 1), 4) < 0)
? ts_delta(close, 1)
```

```
: (-1 * ts_delta(close, 1))))
```

```
[160]: def alpha017(c, v):
        """((-1 * rank(ts_rank(close, 10))) * rank(ts_delta(ts_delta(close, 1),
        →1))) * rank(ts_rank((volume / adv20), 5)))
        """
        adv20 = ts_mean(v, 20)
        return (-rank(ts_rank(c, 10))
                .mul(rank(ts_delta(ts_delta(c, 1), 1)))
                .mul(rank(ts_rank(v.div(adv20), 5)))
                .stack('ticker')
                .swaplevel())
```

```
[161]: alpha = 17
```

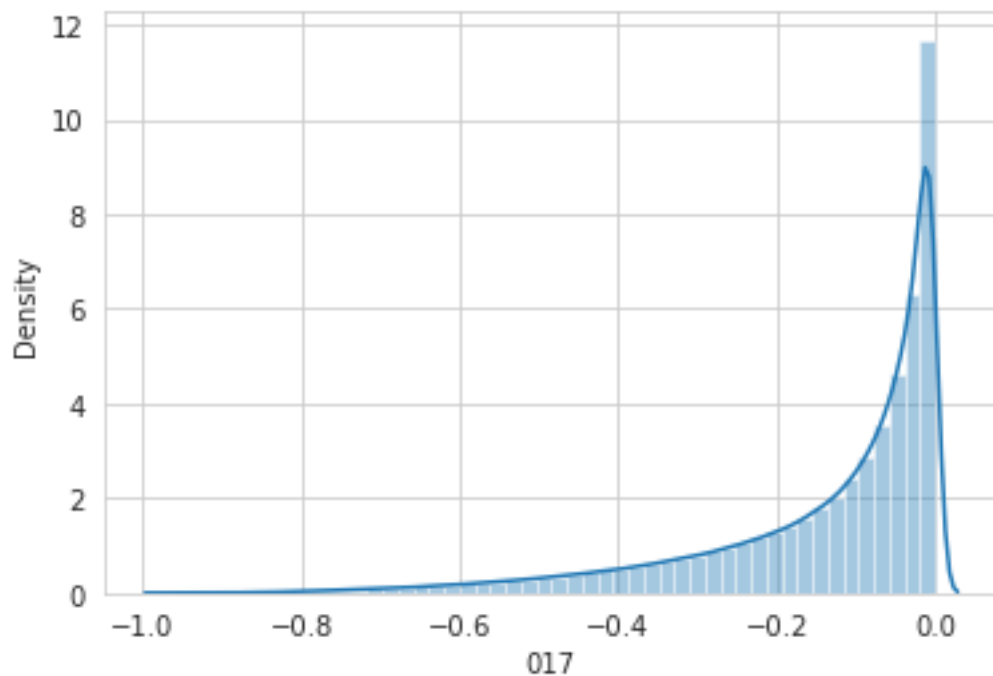
```
[162]: %%time
        alphas[f'{alpha:03}'] = alpha017(c, v)
```

CPU times: user 5min 59s, sys: 127 ms, total: 5min 59s

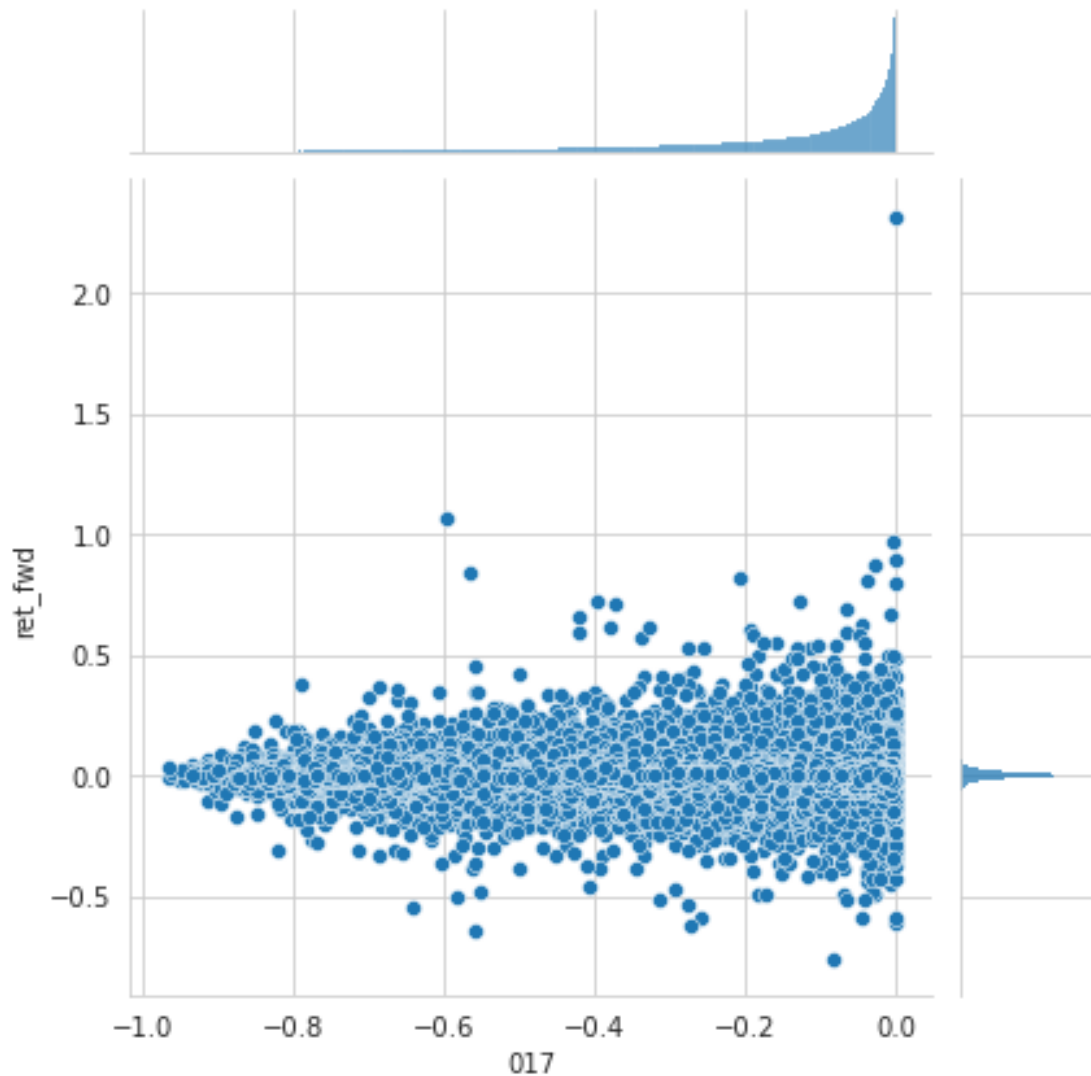
Wall time: 5min 59s

```
[163]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[164]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[165]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[166]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[167]: mi[alpha]
```

```
[167]: 0
```

1.22 Alpha 018

```
-rank((ts_std(abs((close - open)), 5) + (close - open)) +  
      ts_corr(close, open, 10))
```

```
[168]: def alpha018(o, c):
        """-rank((ts_std(abs((close - open)), 5) + (close - open)) +
            ts_corr(close, open, 10))
        """
        return (-rank(ts_std(c.sub(o).abs(), 5)
            .add(c.sub(o))
            .add(ts_corr(c, o, 10)
                .replace([-np.inf,
                    np.inf],
                    np.nan)))
            .stack('ticker')
            .swaplevel())
```

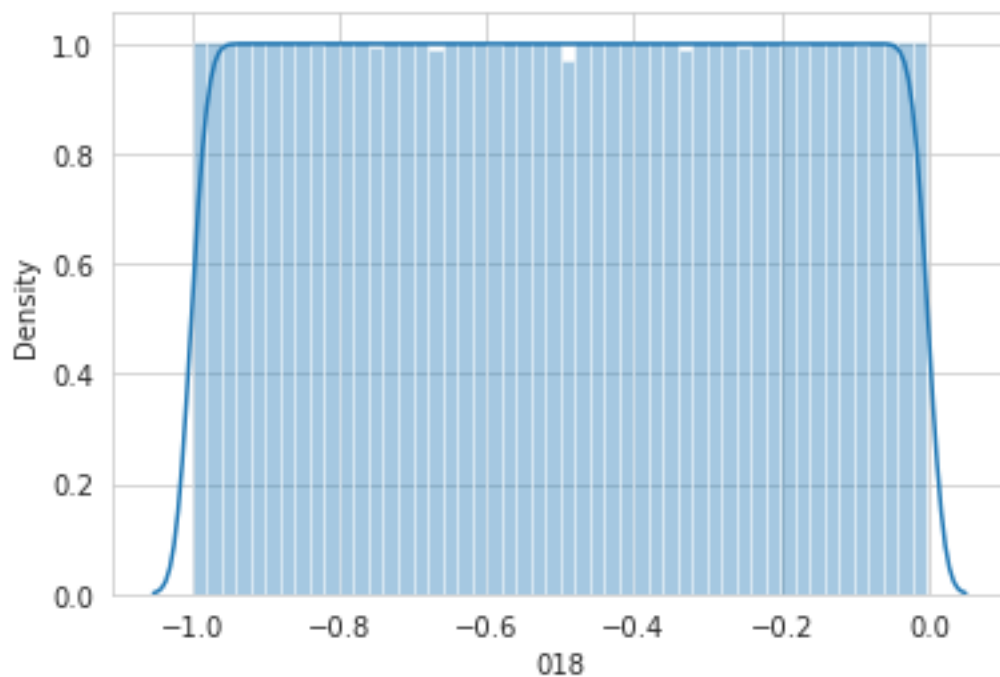
```
[169]: alpha = 18
```

```
[170]: %%time
        alphas[f'{alpha:03}'] = alpha018(o, c)
```

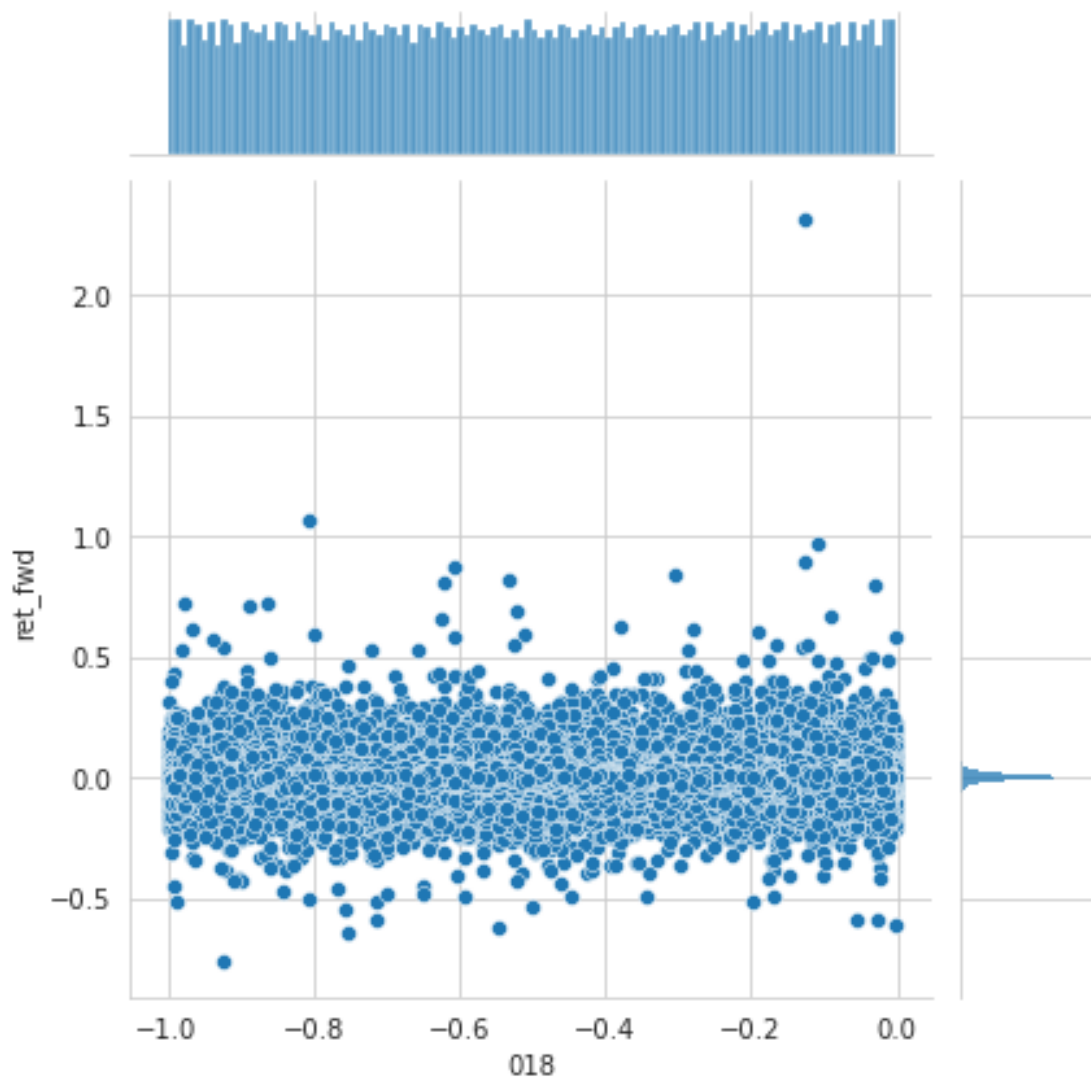
CPU times: user 3.64 s, sys: 84.1 ms, total: 3.72 s
Wall time: 3.64 s

```
[171]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[172]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[173]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[174]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[175]: mi[alpha]
```

```
[175]: 0
```

1.23 Alpha 019

```
rank(((0 < ts_min(ts_delta(close, 1), 4))
? ts_delta(close, 1)
: ((ts_max(ts_delta(close, 1), 4) < 0)
? ts_delta(close, 1)
```

```
: (-1 * ts_delta(close, 1))))
```

```
[176]: def alpha019(c, r):  
        """((-1 * sign(((close - ts_lag(close, 7)) + ts_delta(close, 7)))) *  
        (1 + rank((1 + ts_sum(returns, 250)))))  
        """  
        return (-sign(ts_delta(c, 7) + ts_delta(c, 7))  
                .mul(1 + rank(1 + ts_sum(r, 250)))  
                .stack('ticker')  
                .swaplevel())
```

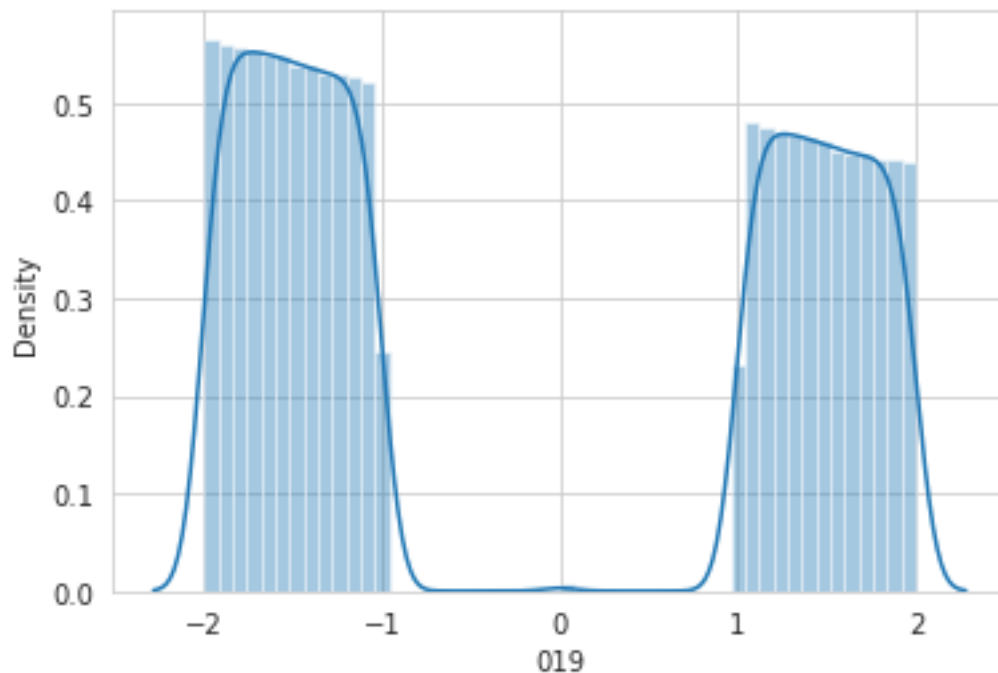
```
[177]: alpha = 19
```

```
[178]: %%time  
alphas[f'{alpha:03}'] = alpha019(c, r)
```

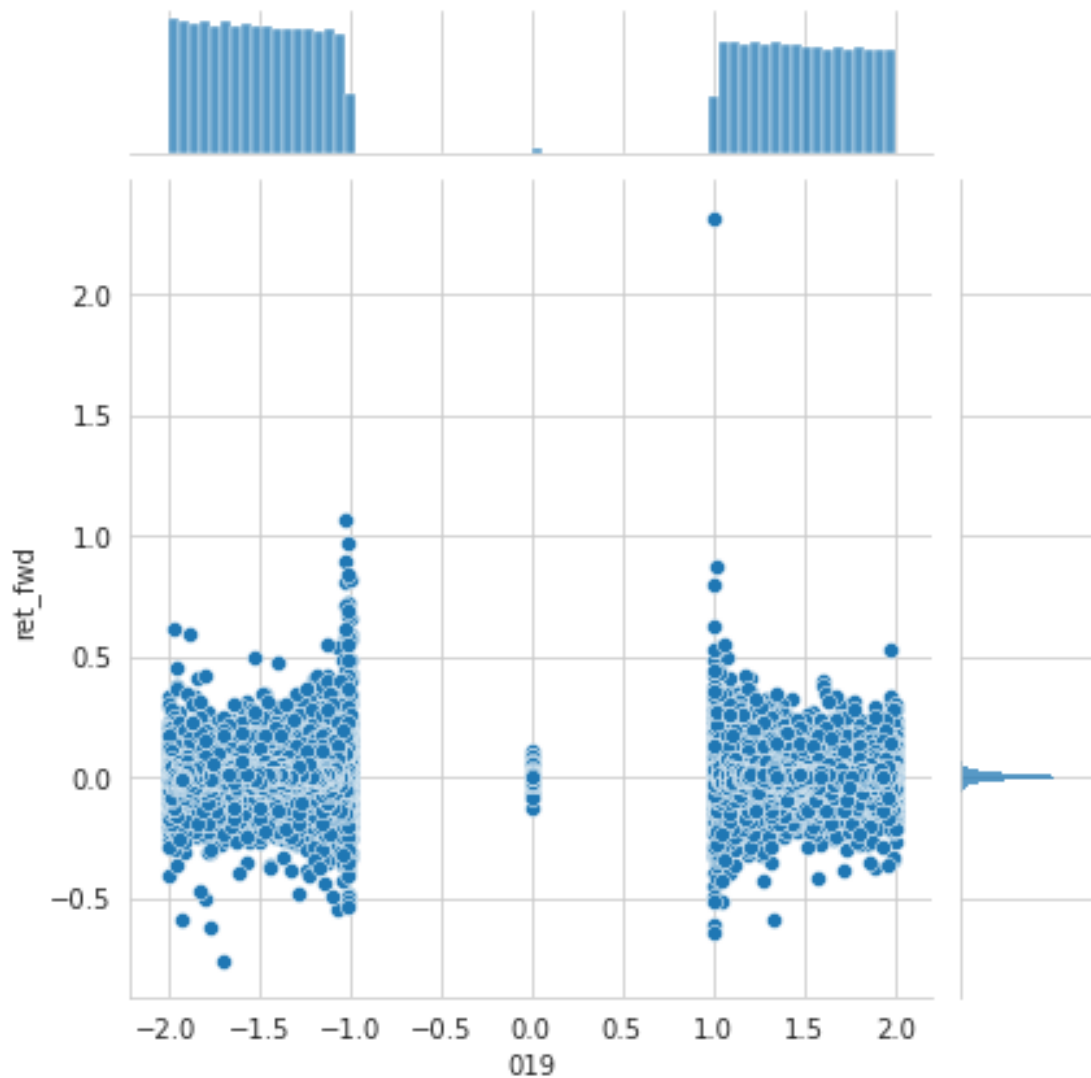
CPU times: user 2.33 s, sys: 24 ms, total: 2.36 s
Wall time: 2.33 s

```
[179]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[180]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[181]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[182]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[183]: mi[alpha]
```

```
[183]: 0.010082898712334476
```

```
[184]: pd.Series(mi).to_csv('mi.csv')
```

1.24 Alpha 020

```
-rank(open - ts_lag(high, 1)) *  
rank(open - ts_lag(close, 1)) *  
rank(open - ts_lag(low, 1))
```

```
[185]: def alpha020(o, h, l, c):
        """-rank(open - ts_lag(high, 1)) *
            rank(open - ts_lag(close, 1)) *
            rank(open - ts_lag(low, 1))"""
        return (rank(o - ts_lag(h, 1))
                .mul(rank(o - ts_lag(c, 1)))
                .mul(rank(o - ts_lag(l, 1)))
                .mul(-1)
                .stack('ticker')
                .swaplevel())
```

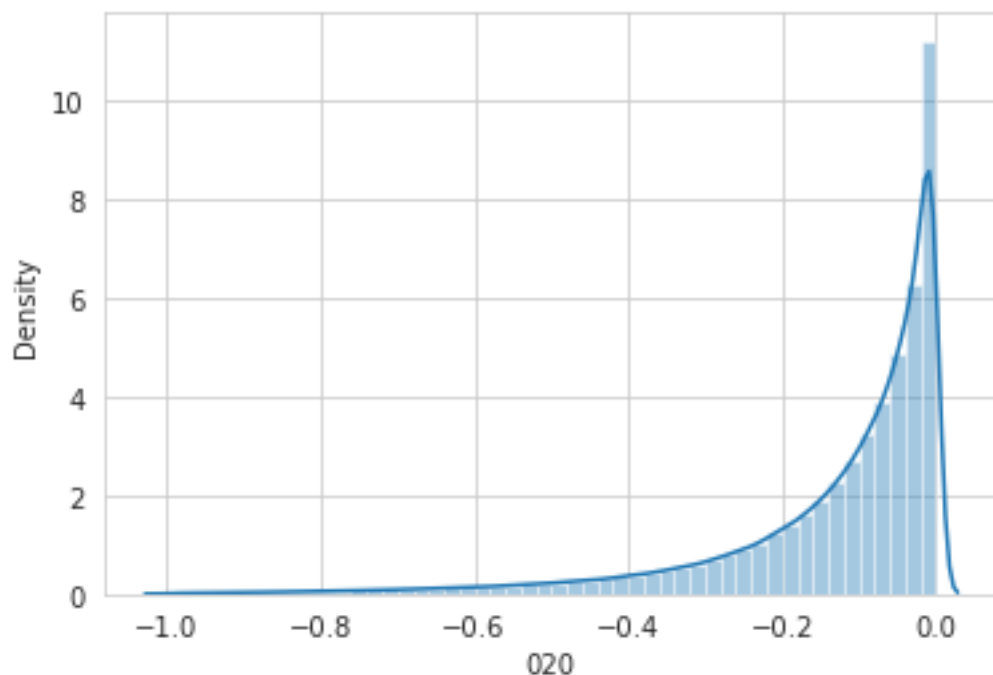
```
[186]: alpha = 20
```

```
[187]: %%time
        alphas[f'{alpha:03}'] = alpha020(o, h, l, c)
```

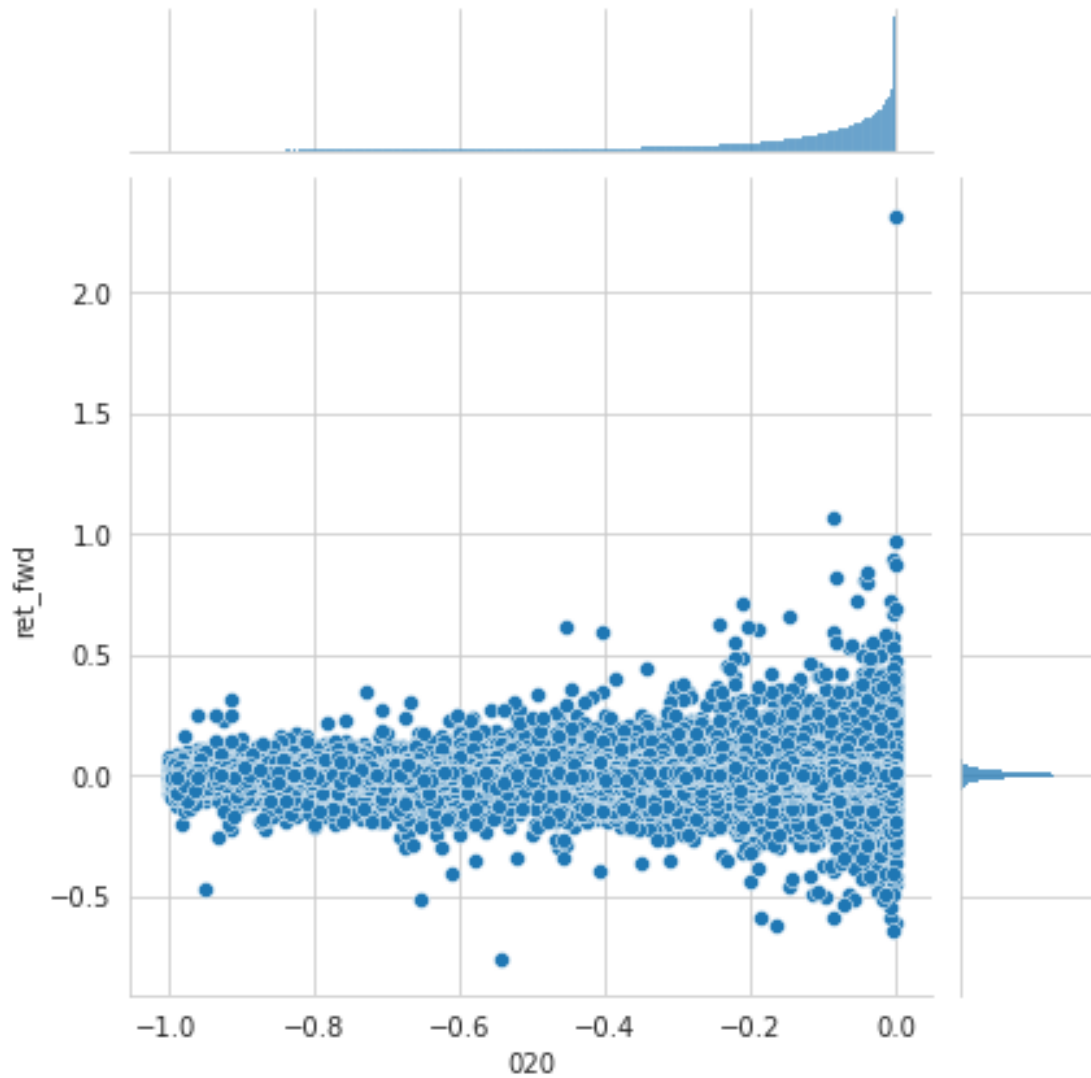
CPU times: user 2.4 s, sys: 64 ms, total: 2.47 s
Wall time: 2.4 s

```
[188]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[189]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[190]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```

```
[191]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[192]: mi[alpha]
```

```
[192]: 0.002095780517723078
```

1.25 Alpha 021

```
ts_mean(close, 8) + ts_std(close, 8) < ts_mean(close, 2)
? -1
: (ts_mean(close,2) < ts_mean(close, 8) - ts_std(close, 8)
? 1
: (volume / adv20 < 1
? -1
```

```
: 1))
```

```
[193]: def alpha021(c, v):
        """ts_mean(close, 8) + ts_std(close, 8) < ts_mean(close, 2)
        ? -1
        : (ts_mean(close, 2) < ts_mean(close, 8) - ts_std(close, 8)
        ? 1
        : (volume / adv20 < 1
        ? -1
        : 1))

        """
        sma2 = ts_mean(c, 2)
        sma8 = ts_mean(c, 8)
        std8 = ts_std(c, 8)

        cond_1 = sma8.add(std8) < sma2
        cond_2 = sma8.add(std8) > sma2
        cond_3 = v.div(ts_mean(v, 20)) < 1

        val = np.ones_like(c)
        alpha = pd.DataFrame(np.select(condlist=[cond_1, cond_2, cond_3],
                                         choicelist=[-1, 1, -1], default=1),
                              index=c.index,
                              columns=c.columns)

        return (alpha
                .stack('ticker')
                .swaplevel())
```

```
[194]: alpha = 21
```

```
[195]: %%time
        alphas[f'{alpha:03}'] = alpha021(c, v)
```

CPU times: user 2.15 s, sys: 16 ms, total: 2.17 s

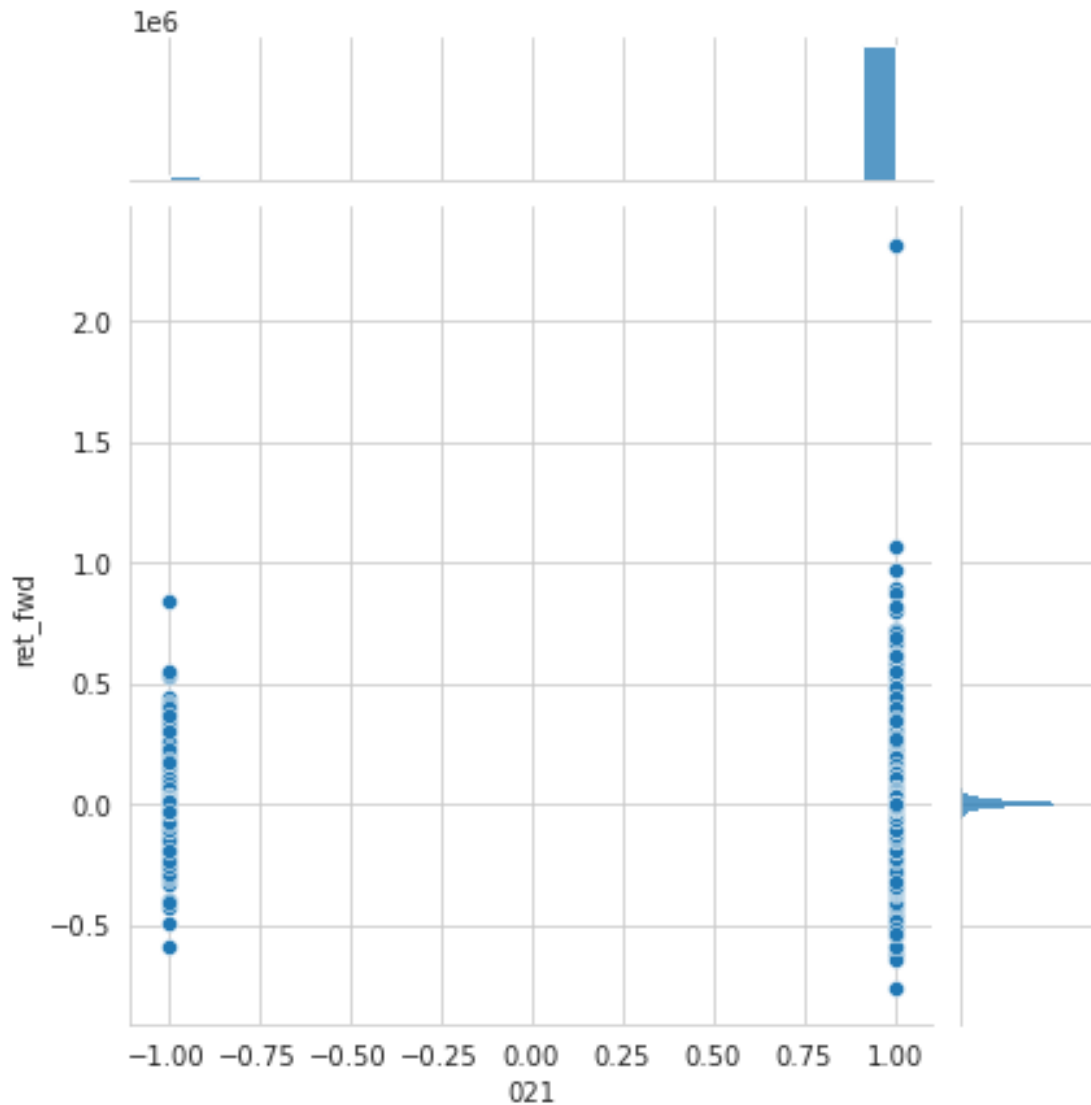
Wall time: 2.13 s

```
[196]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[197]: alphas[f'{alpha:03}'].value_counts()
```

```
[197]: 1      1211187
      -1       43906
      Name: 021, dtype: int64
```

```
[198]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[199]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[200]: # mi[alpha]
```

1.26 Alpha 022

```
-(ts_delta(ts_corr(high, volume, 5), 5) *  
    rank(ts_std(close, 20)))
```

```
[201]: def alpha022(h, c, v):  
    """-(ts_delta(ts_corr(high, volume, 5), 5) *  
        rank(ts_std(close, 20)))  
    """
```

```

return (ts_delta(ts_corr(h, v, 5)
                    .replace([-np.inf,
                             np.inf],
                             np.nan), 5)
        .mul(rank(ts_std(c, 20)))
        .mul(-1)
        .stack('ticker')
        .swaplevel())

```

```
[202]: alpha = 22
```

```

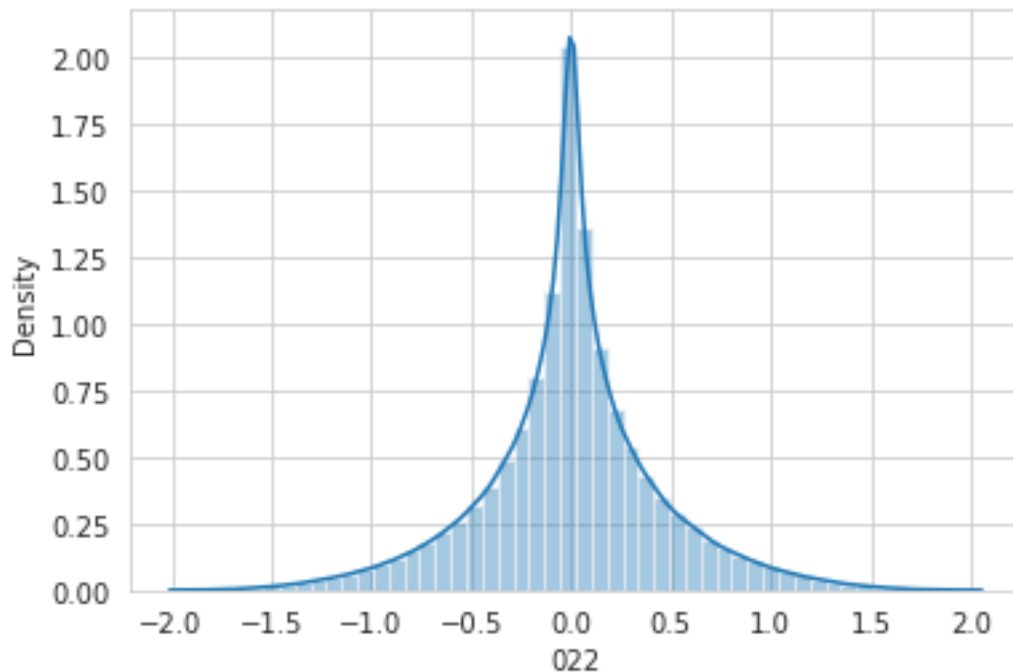
[203]: %%time
alphas[f'{alpha:03}'] = alpha022(h, c, v)

```

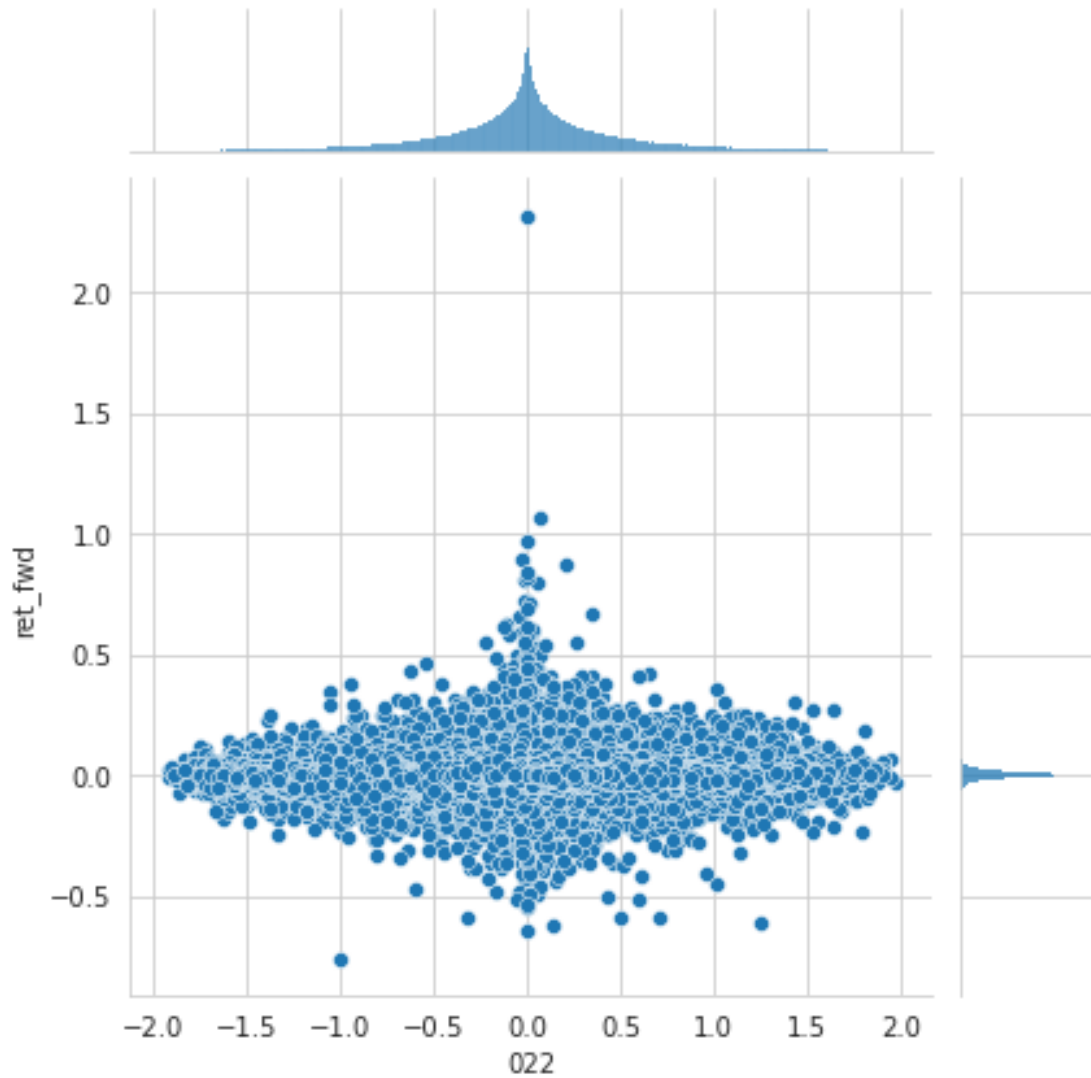
CPU times: user 3.47 s, sys: 52.1 ms, total: 3.52 s
Wall time: 3.45 s

```
[204]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[205]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[206]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[207]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[208]: mi[alpha]
```

```
[208]: 0.0025639008286280074
```

```
[209]: pd.Series(mi).to_csv('mi.csv')
```

1.27 Alpha 023

```
((ts_sum(high, 20) / 20) < high)
    ? (-1 * ts_delta(high, 2))
    : 0
```

```
[210]: def alpha023(h, c):
        """((ts_mean(high, 20) < high)
            ? (-1 * ts_delta(high, 2))
            : 0
            """

        return (ts_delta(h, 2)
                .mul(-1)
                .where(ts_mean(h, 20) < h, 0)
                .stack('ticker')
                .swaplevel())
```

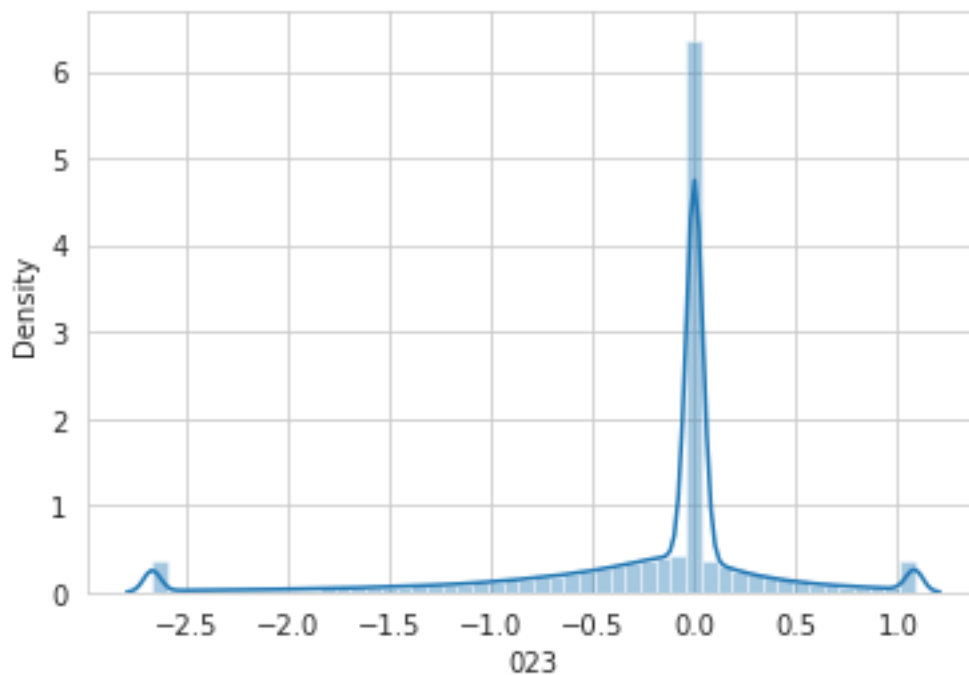
```
[211]: alpha = 23
```

```
[212]: %%time
        alphas[f'{alpha:03}'] = alpha023(h, c)
```

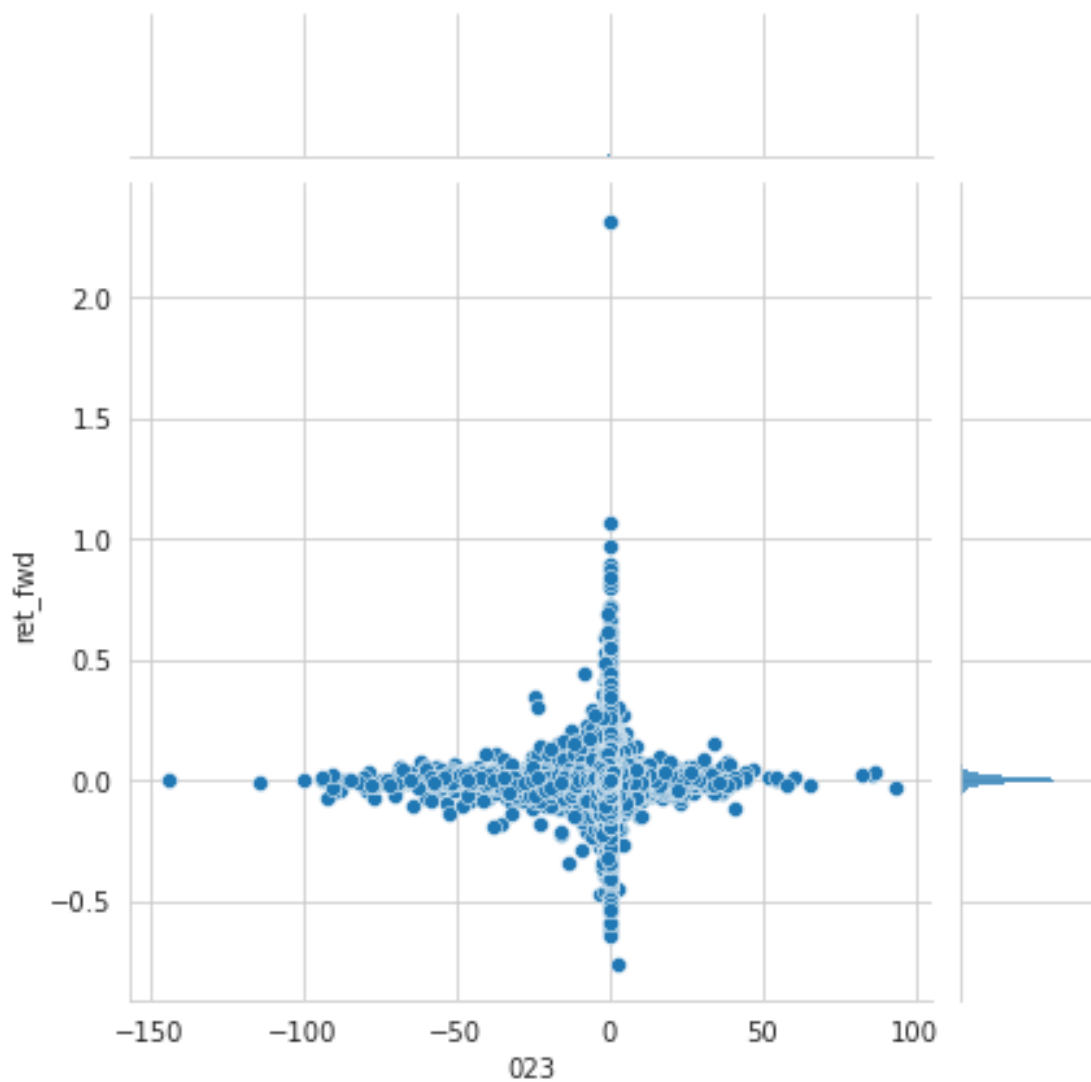
CPU times: user 2.49 s, sys: 8.05 ms, total: 2.5 s
Wall time: 2.48 s

```
[213]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[214]: q = 0.025
        sns.distplot(alphas[f'{alpha:03}'].clip(lower=alphas[f'{alpha:03}'].quantile(q),
            upper=alphas[f'{alpha:03}'].
            ↪quantile(1-q)));
```



```
[215]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[216]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[217]: mi[alpha]
```

```
[217]: 0.010802888976805036
```

1.28 Alpha 024

```
((((ts_delta((ts_mean(close, 100)), 100) / ts_lag(close, 100)) <= 0.05)
  ? (-1 * (close - ts_min(close, 100)))
```

```
: (-1 * ts_delta(close, 3)))
```

```
[218]: def alpha024(c):
        """(((ts_delta((ts_mean(close, 100)), 100) / ts_lag(close, 100)) <= 0.05)
            ? (-1 * (close - ts_min(close, 100)))
            : (-1 * ts_delta(close, 3)))
        """
        cond = ts_delta(ts_mean(c, 100), 100) / ts_lag(c, 100) <= 0.05

        return (c.sub(ts_min(c, 100)).mul(-1).where(cond, -ts_delta(c, 3))
                .stack('ticker')
                .swaplevel())
```

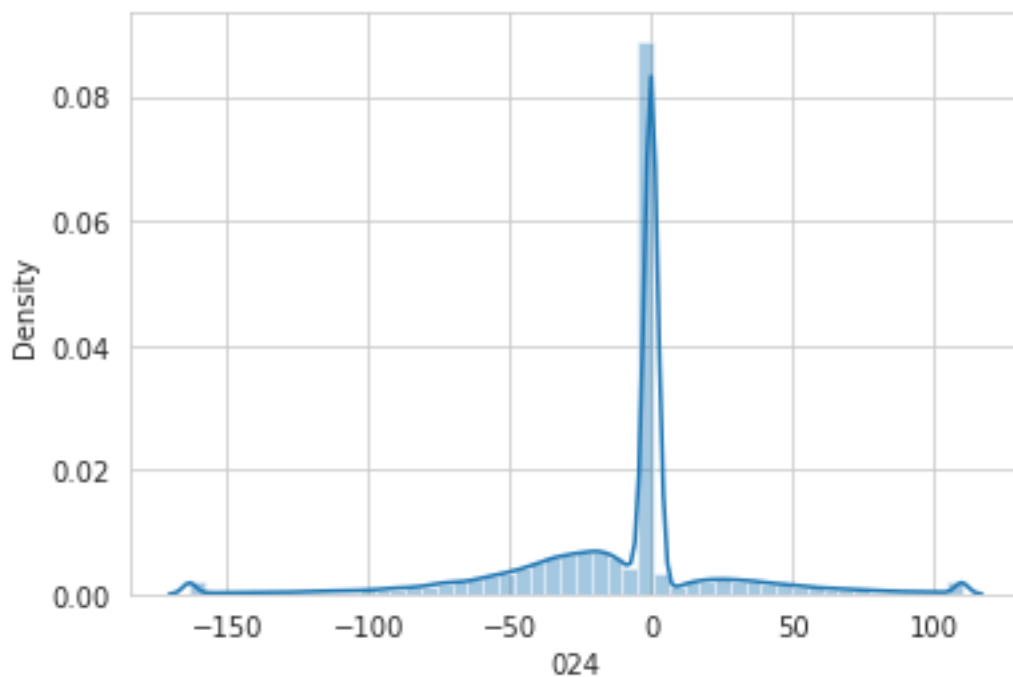
```
[219]: alpha = 24
```

```
[220]: %%time
        alphas[f'{alpha:03}'] = alpha024(c)
```

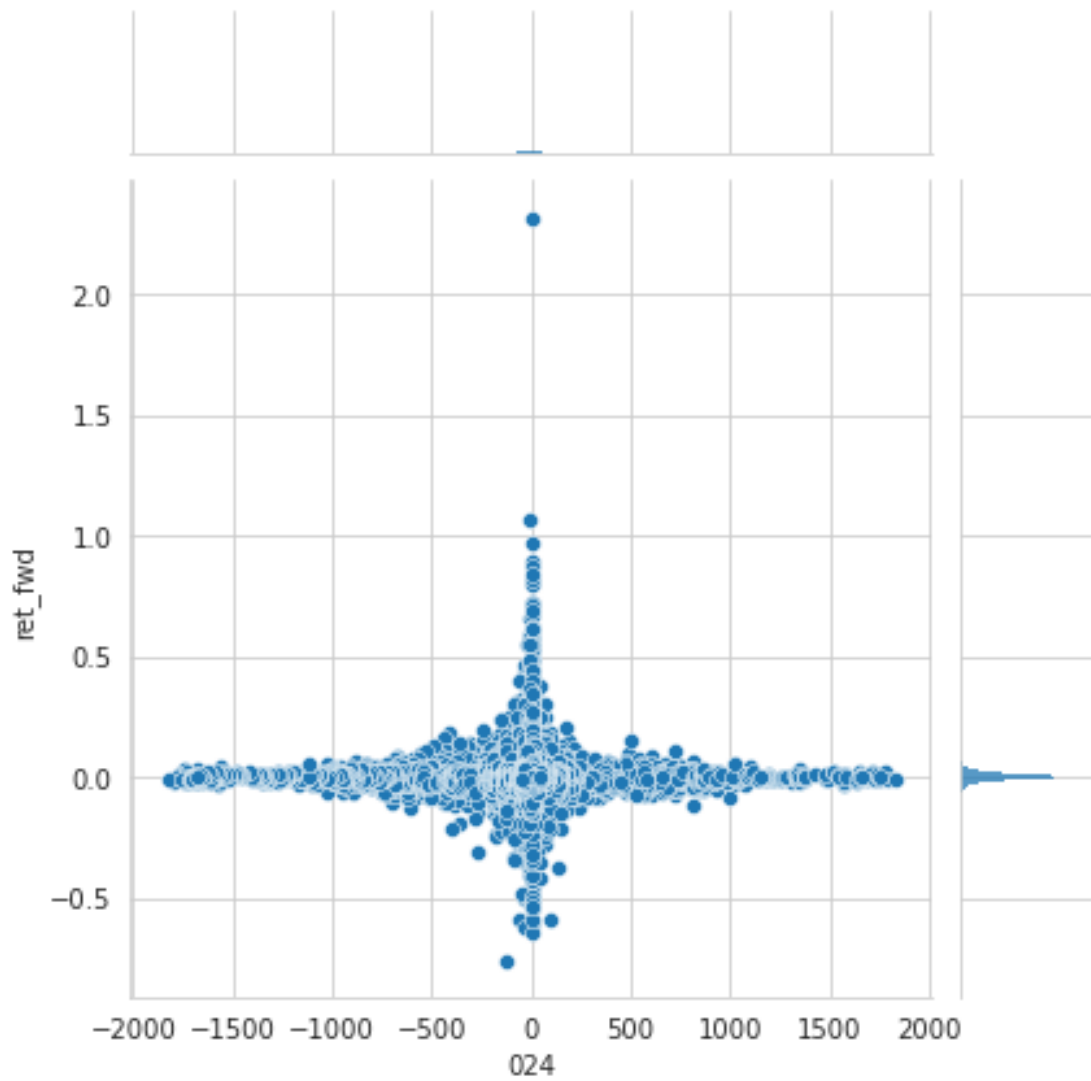
CPU times: user 2.45 s, sys: 48 ms, total: 2.5 s
Wall time: 2.44 s

```
[221]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[222]: q = 0.01
        sns.distplot(alphas[f'{alpha:03}'].clip(lower=alphas[f'{alpha:03}'].quantile(q),
                                                    upper=alphas[f'{alpha:03}'].
                                                    ↪quantile(1-q)));
```




```
[223]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[224]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[225]: mi[alpha]
```

```
[225]: 0.031433111296676586
```

1.29 Alpha 025

```
rank((-1 * returns) * adv20 * vwap * (high - close))
```

```
[226]: def alpha025(h, c, r, vwap, adv20):
        """rank((-1 * returns) * adv20 * vwap * (high - close))"""
        return (rank(-r.mul(adv20)
                        .mul(vwap)
                        .mul(h.sub(c)))
                .stack('ticker')
                .swaplevel())
```

```
[227]: alpha = 25
```

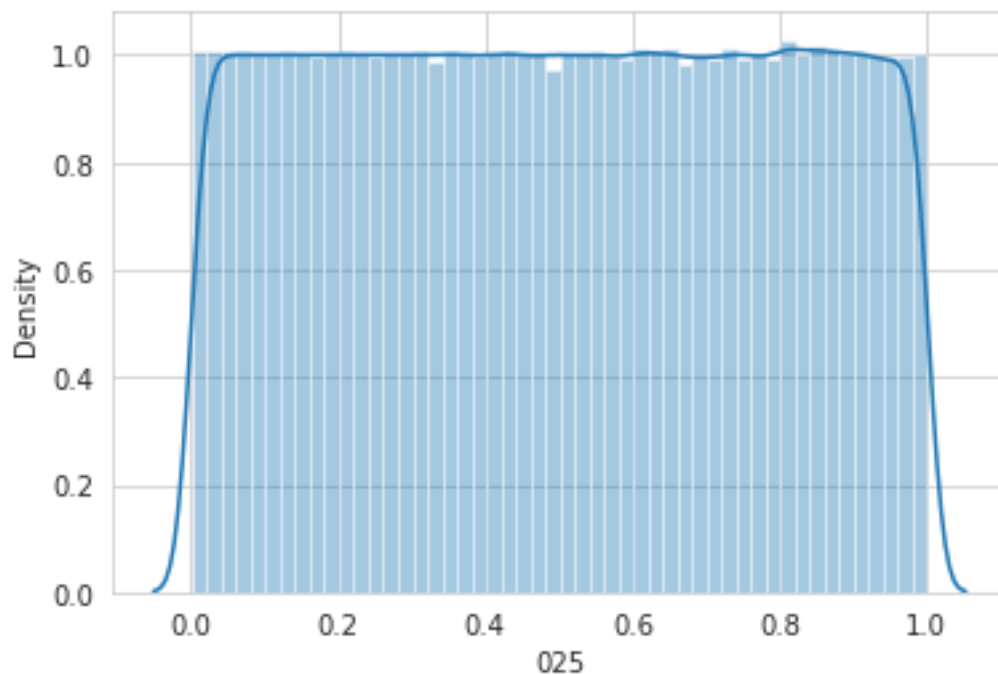
```
[228]: %%time
        alphas[f'{alpha:03}'] = alpha025(h, c, r, vwap, adv20)
```

CPU times: user 2.9 s, sys: 36 ms, total: 2.93 s

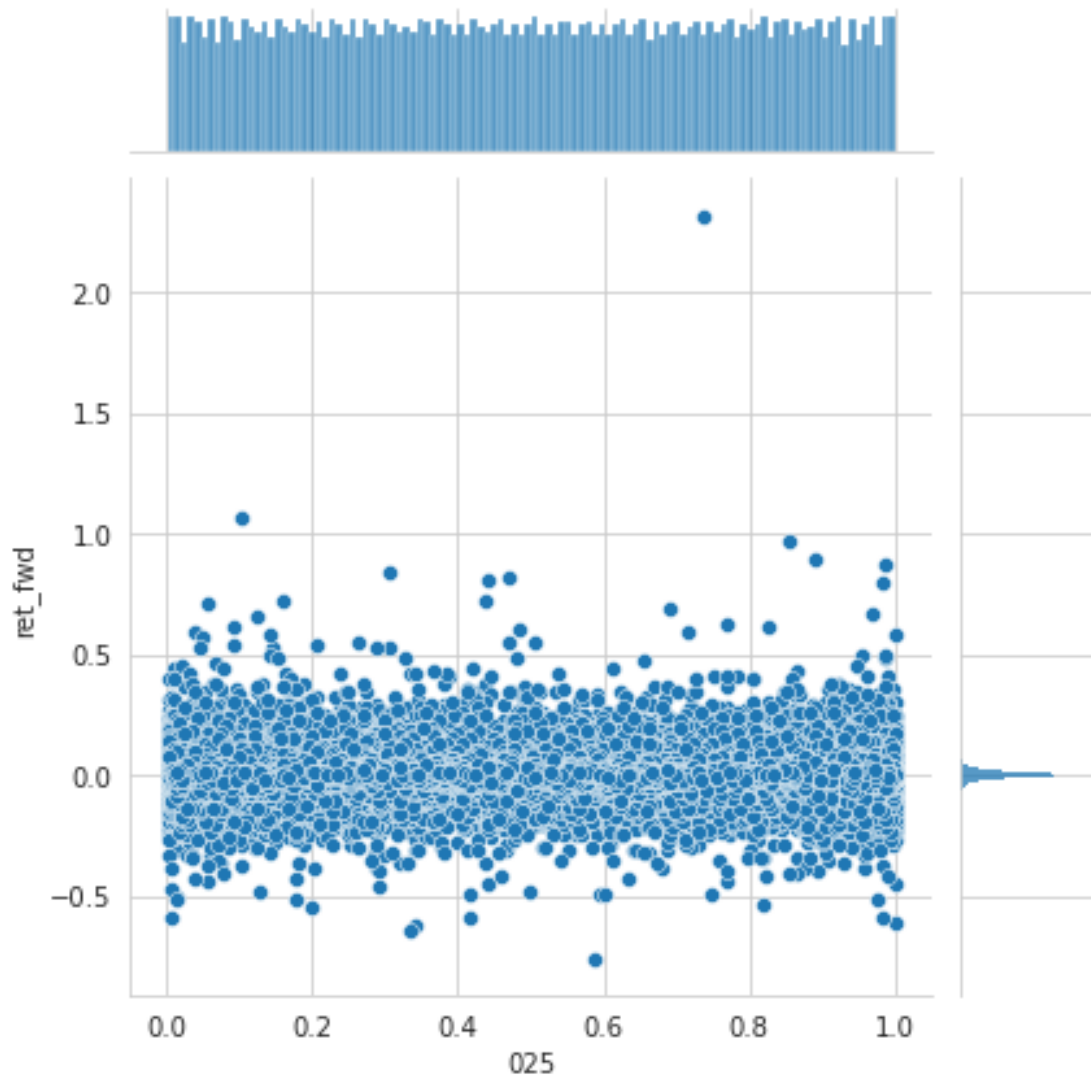
Wall time: 2.9 s

```
[229]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[230]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[231]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[232]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[233]: mi[alpha]
```

```
[233]: 0.002047811727091897
```

```
[234]: pd.Series(mi).to_csv('mi.csv')
```

1.30 Alpha 026

```
(-1 * rank(ts_cov(rank(high), rank(volume), 5)))
```

```
[235]: def alpha026(h, v):
        """(-1 * ts_max(ts_corr(ts_rank(volume, 5), ts_rank(high, 5), 5), 3))"""
        return (ts_max(ts_corr(ts_rank(v, 5),
                                ts_rank(h, 5), 5)
                        .replace([-np.inf, np.inf], np.nan), 3)
                .mul(-1)
                .stack('ticker')
                .swaplevel())
```

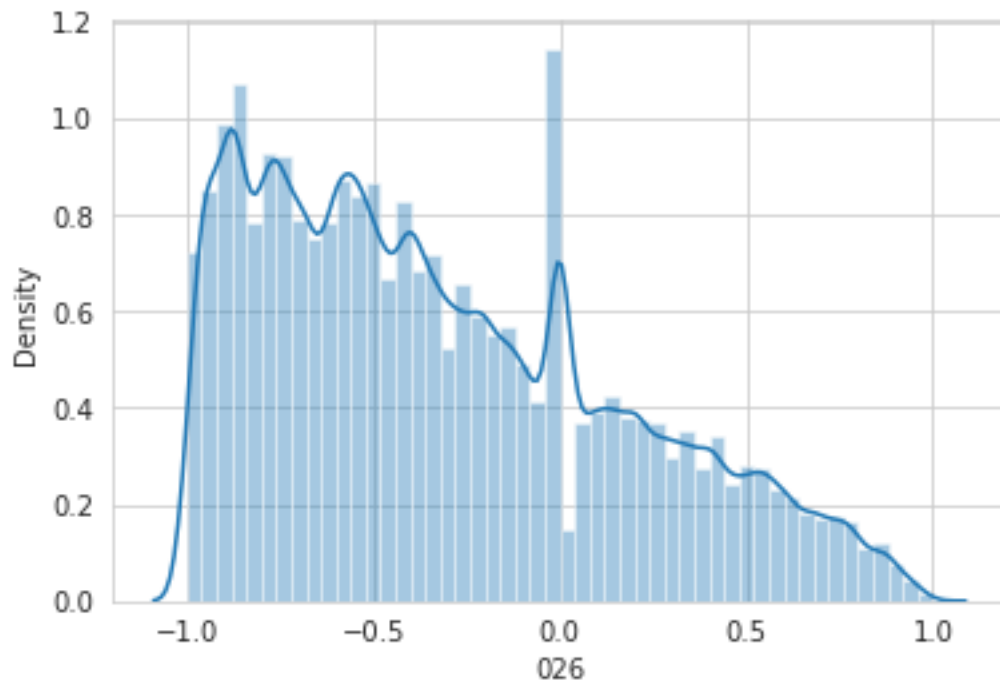
```
[236]: alpha = 26
```

```
[237]: %%time
        alphas[f'{alpha:03}'] = alpha026(h, v)
```

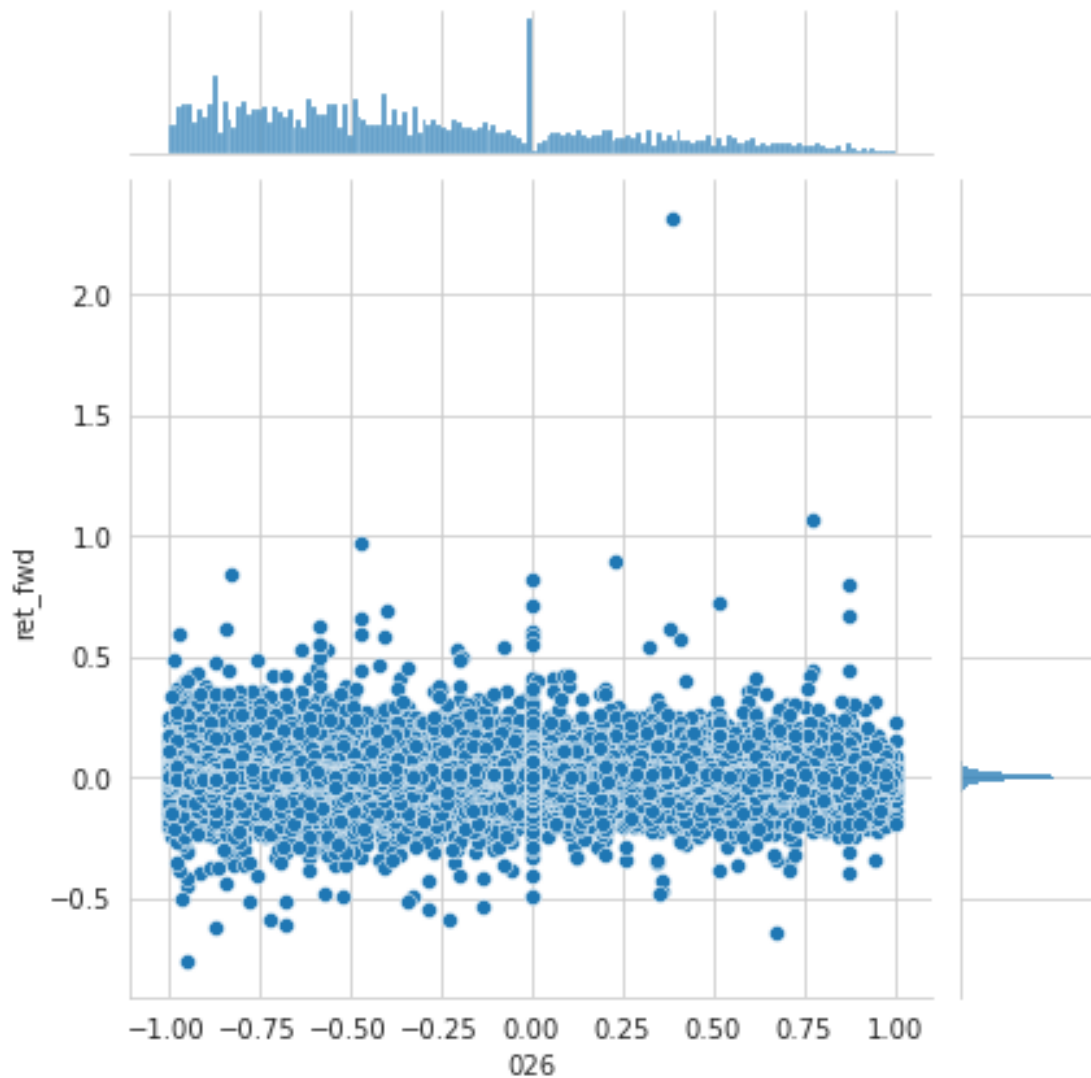
CPU times: user 6min 3s, sys: 148 ms, total: 6min 3s
Wall time: 6min 3s

```
[238]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[239]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[240]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[241]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[242]: mi[alpha]
```

```
[242]: 0.006628205346157046
```

1.31 Alpha 027

```
rank(((0 < ts_min(ts_delta(close, 1), 4))
? ts_delta(close, 1)
: ((ts_max(ts_delta(close, 1), 4) < 0)
? ts_delta(close, 1)
: (-1 * ts_delta(close, 1)))))
```

```
[243]: def alpha027(v, vwap):
        """((0.5 < rank(ts_mean(ts_corr(rank(volume), rank(vwap), 6), 2)))
           ? -1
           : 1)"""
        cond = rank(ts_mean(ts_corr(rank(v),
                                     rank(vwap), 6), 2))
        alpha = cond.notnull().astype(float)
        return (alpha.where(cond <= 0.5, -alpha)
                .stack('ticker')
                .swaplevel())
```

```
[244]: alpha = 27
```

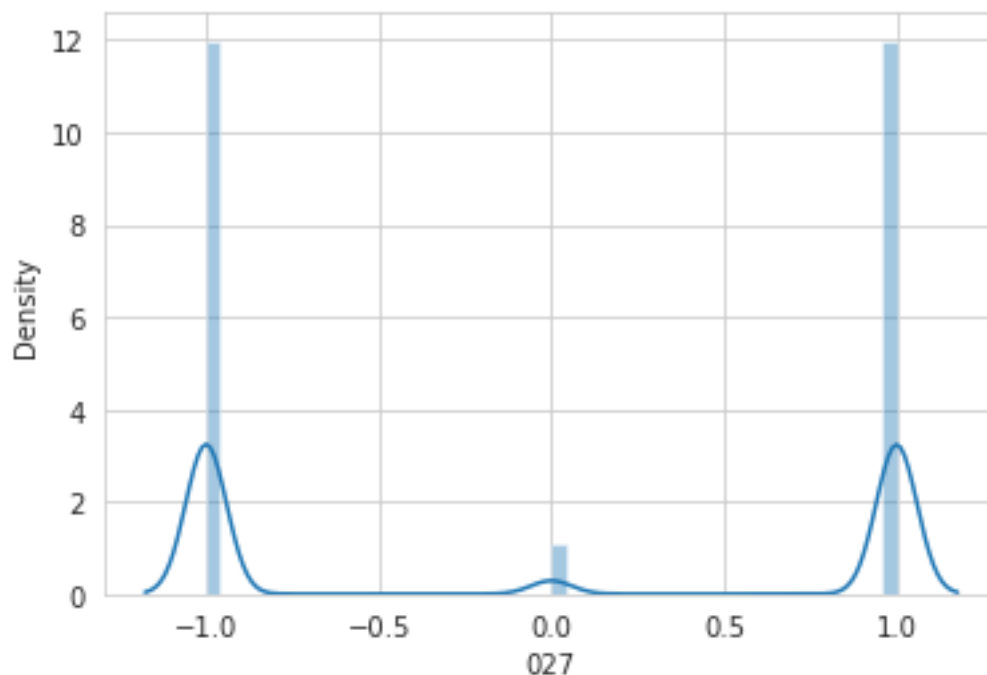
```
[245]: %%time
        alphas[f'{alpha:03}'] = alpha027(v, vwap)
```

CPU times: user 3.65 s, sys: 28 ms, total: 3.68 s

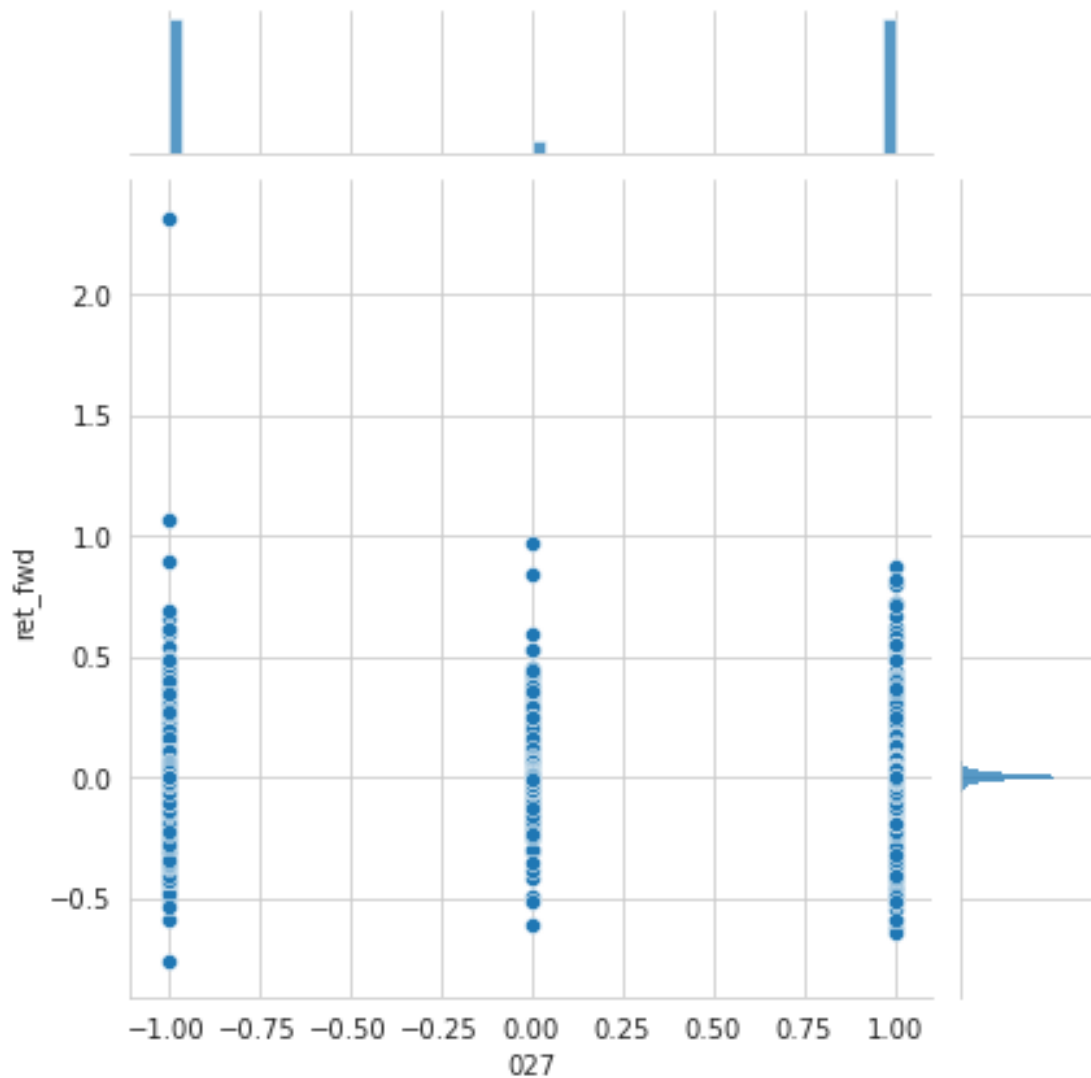
Wall time: 3.64 s

```
[246]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[247]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[248]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[249]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[250]: # mi[alpha]
```

1.32 Alpha 028

```
-rank((ts_std(abs((close - open)), 5) + (close - open)) +  
      ts_corr(close, open, 10))
```

```
[251]: def alpha028(h, l, c, v, adv20):  
        """scale(((ts_corr(adv20, low, 5) + (high + low) / 2) - close))"""  
        return (scale(ts_corr(adv20, l, 5)  
                      .replace([-np.inf, np.inf], 0)  
                      .add(h.add(1).div(2).sub(c)))
```

```
.stack('ticker')  
.swaplevel()
```

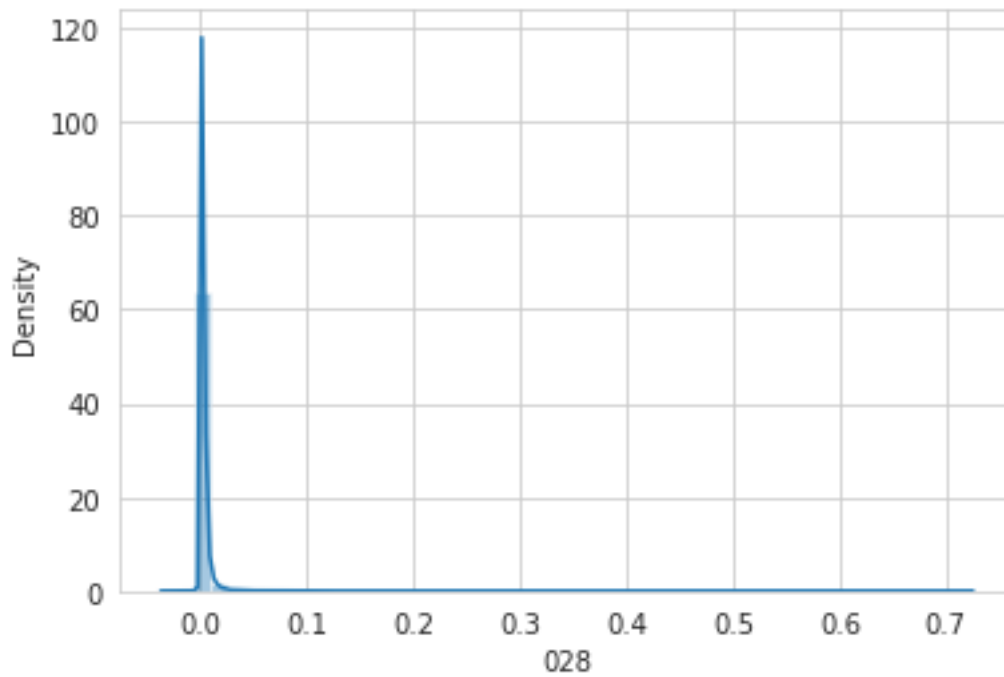
```
[252]: alpha = 28
```

```
[253]: %%time  
alphas[f'{alpha:03}'] = alpha028(h, l, c, v, adv20)
```

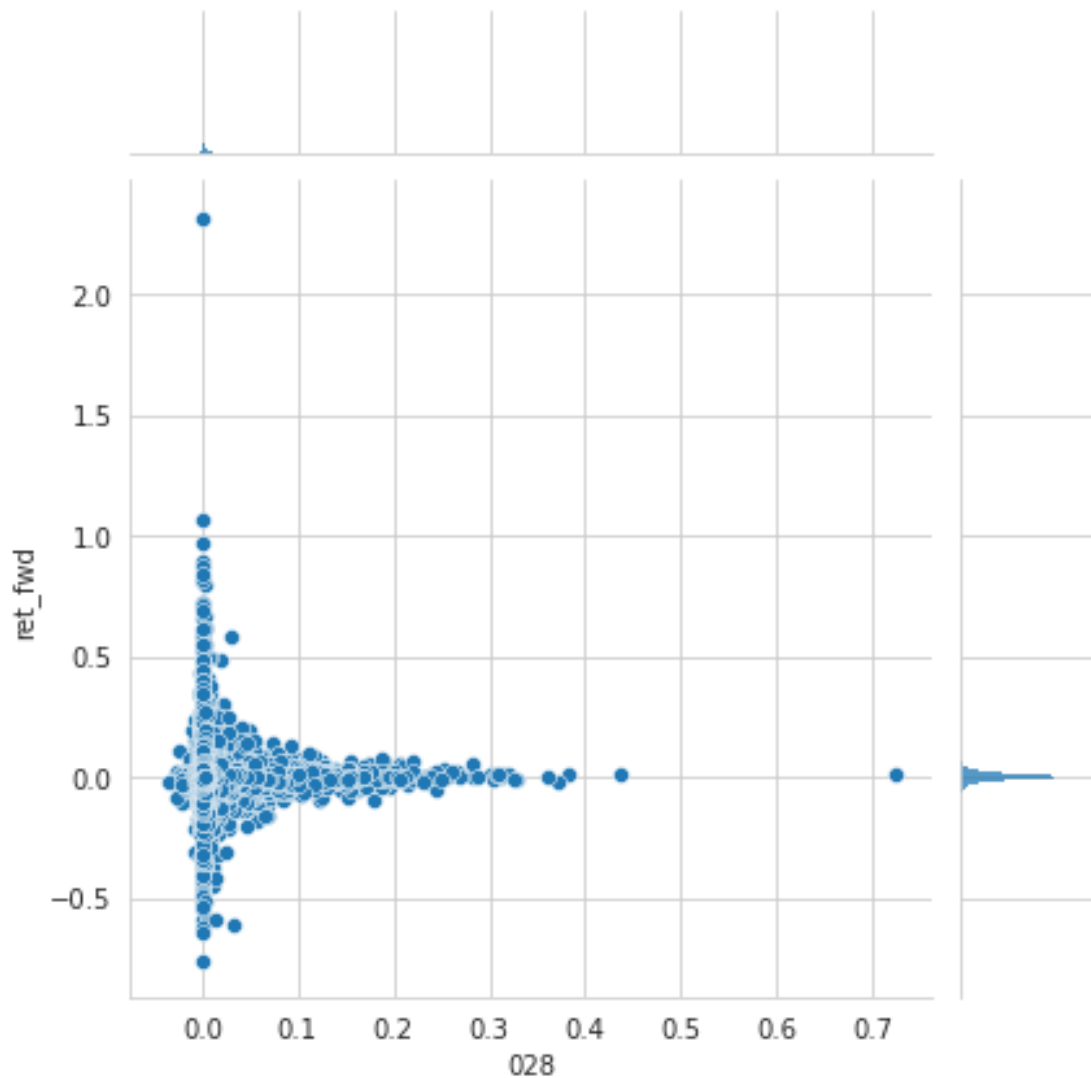
CPU times: user 3.3 s, sys: 24 ms, total: 3.33 s
Wall time: 3.27 s

```
[254]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[255]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[256]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```

```
[257]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[258]: mi[alpha]
```

```
[258]: 0.0064583435249003784
```

```
[259]: pd.Series(mi).to_csv('mi.csv')
```

1.33 Alpha 029

```
rank(((0 < ts_min(ts_delta(close, 1), 4))
? ts_delta(close, 1)
: ((ts_max(ts_delta(close, 1), 4) < 0)
? ts_delta(close, 1)
```

```
: (-1 * ts_delta(close, 1))))
```

```
[260]: def alpha029(c, r):
        """(ts_min(ts_product(rank(rank(scale(log(ts_sum(ts_min(rank(rank((-1 *
            rank(ts_delta((close - 1), 5))))), 2), 1))))), 1), 5)
            + ts_rank(ts_lag((-1 * returns), 6), 5))
        """
        return (ts_min(rank(rank(scale(log(ts_sum(rank(rank(-rank(ts_delta((c - 1),
        ↪5))))), 2))))), 5)
            .add(ts_rank(ts_lag((-1 * r), 6), 5))
            .stack('ticker')
            .swaplevel())
```

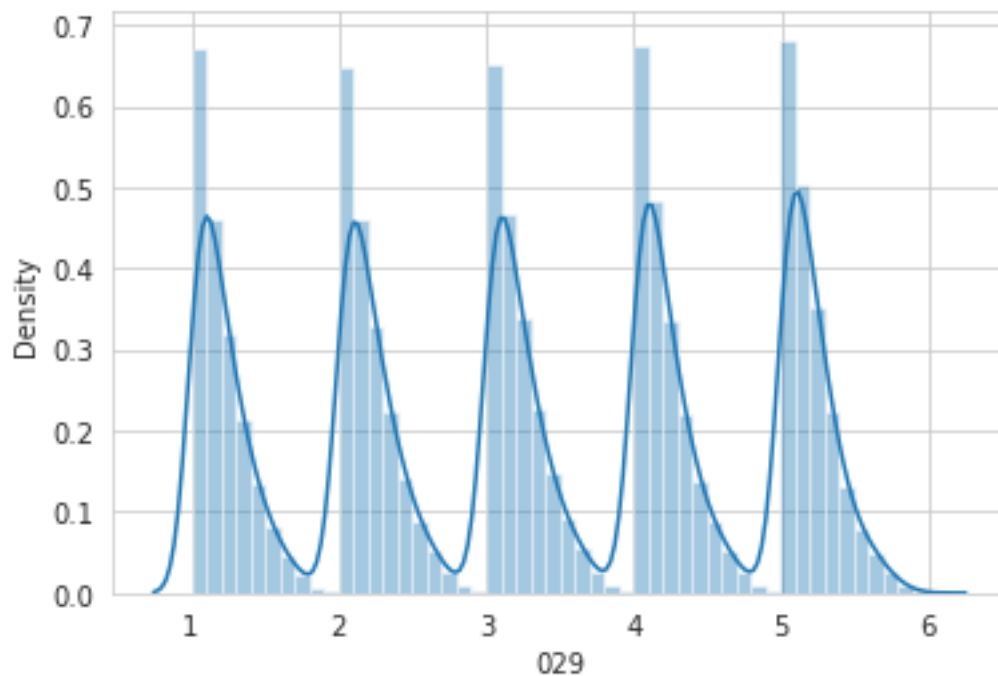
```
[261]: alpha = 29
```

```
[262]: %%time
        alphas[f'{alpha:03}'] = alpha029(c, r)
```

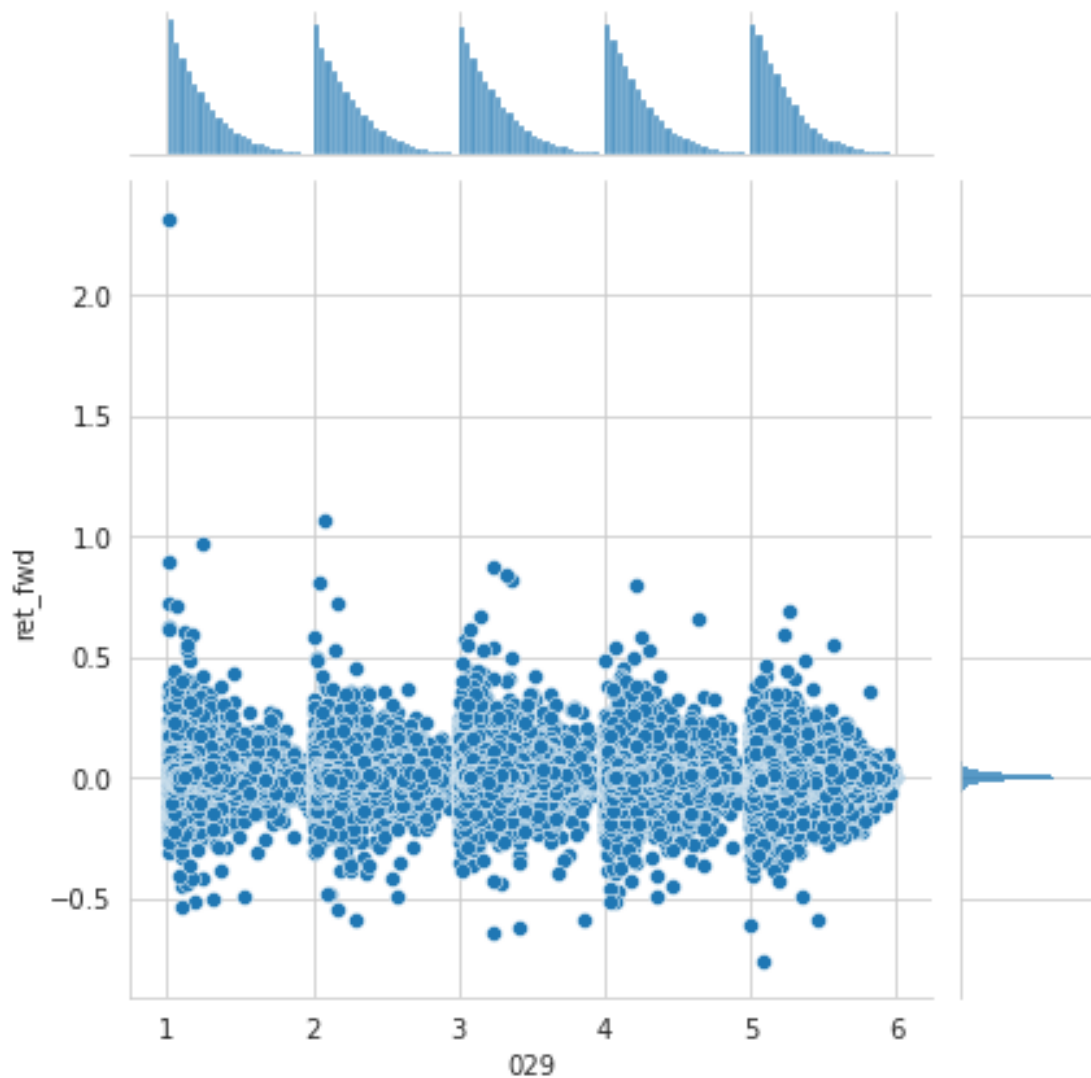
CPU times: user 3min 3s, sys: 188 ms, total: 3min 4s
Wall time: 3min 4s

```
[263]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[264]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[265]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[266]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[267]: mi[alpha]
```

```
[267]: 0.005814612612969228
```

1.34 Alpha 030

```
-rank(open - ts_lag(high, 1)) *  
rank(open - ts_lag(close, 1)) *  
rank(open - ts_lag(low, 1))
```

```
[268]: def alpha030(c, v):
        """(((1.0 - rank(((sign((close - ts_lag(close, 1))) +
            sign((ts_lag(close, 1) - ts_lag(close, 2)))) +
            sign((ts_lag(close, 2) - ts_lag(close, 3))))) *
            ts_sum(volume, 5)) / ts_sum(volume, 20))"""
        close_diff = ts_delta(c, 1)
        return (rank(sign(close_diff)
            .add(sign(ts_lag(close_diff, 1)))
            .add(sign(ts_lag(close_diff, 2))))
            .mul(-1).add(1)
            .mul(ts_sum(v, 5))
            .div(ts_sum(v, 20))
            .stack('ticker')
            .swaplevel())
```

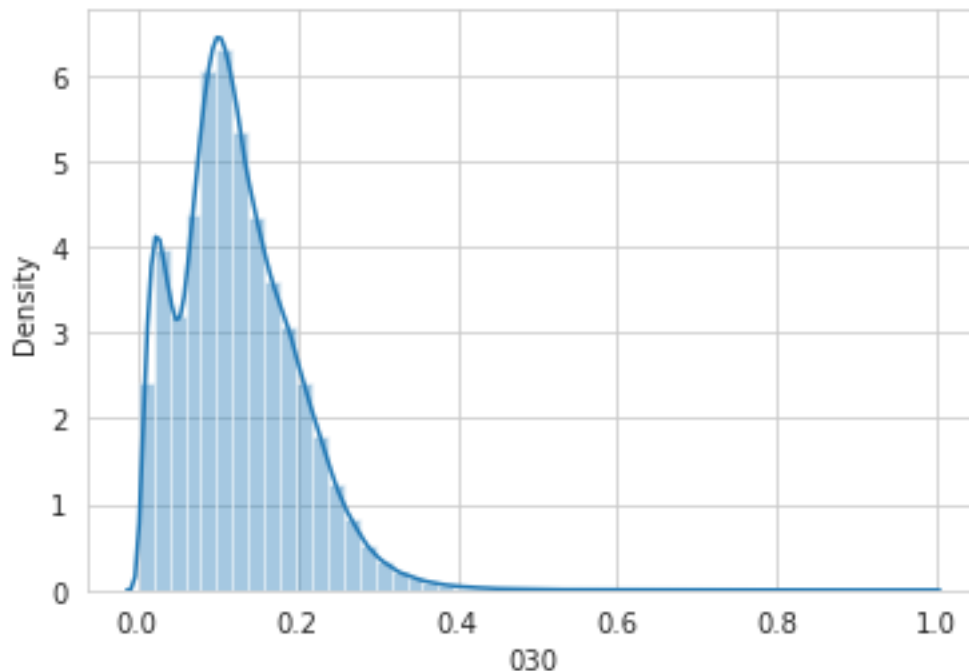
```
[269]: alpha = 30
```

```
[270]: %%time
        alphas[f'{alpha:03}'] = alpha030(c, v)
```

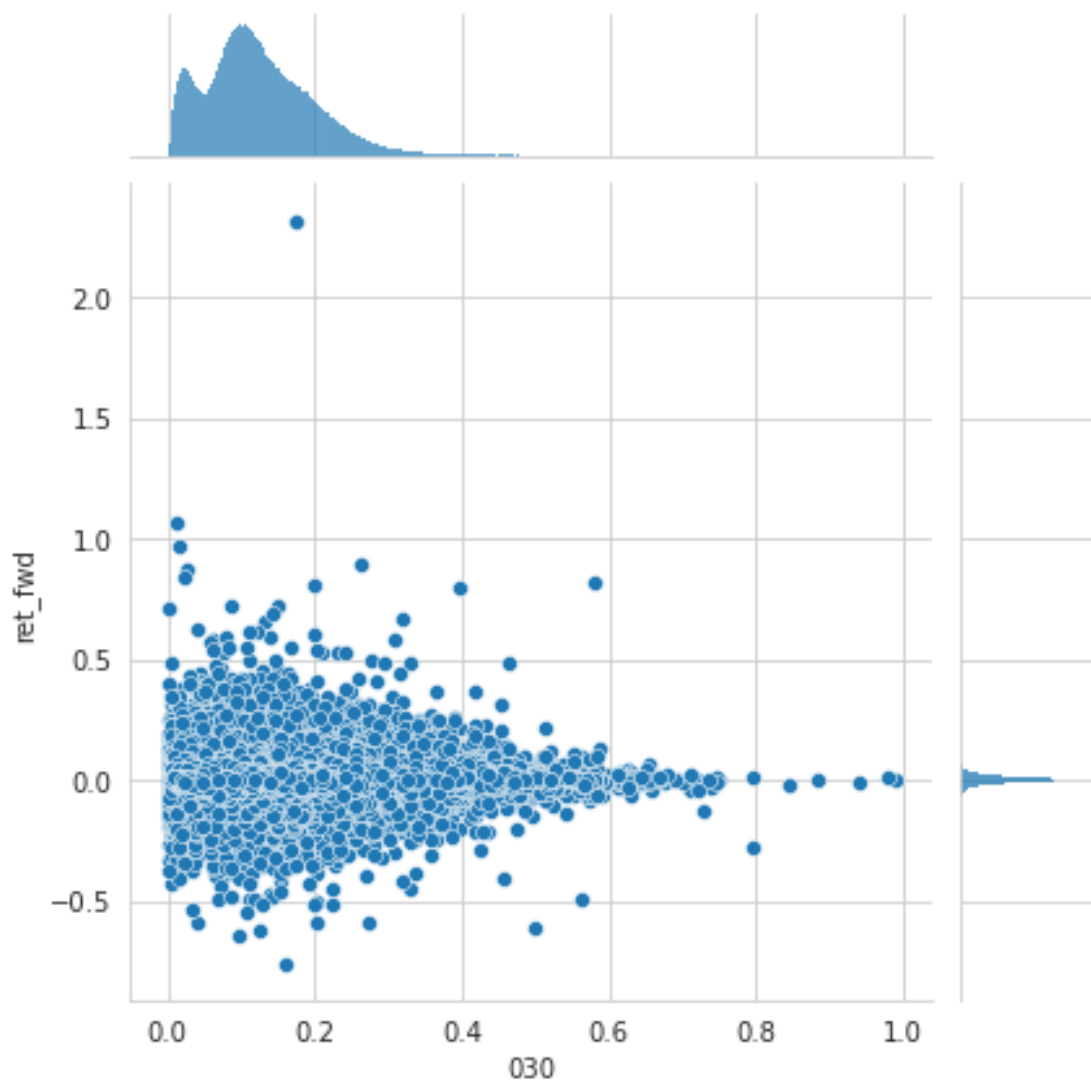
CPU times: user 2.39 s, sys: 48 ms, total: 2.44 s
Wall time: 2.38 s

```
[271]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[272]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[273]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[274]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[275]: mi[alpha]
```

```
[275]: 0
```

1.35 Alpha 031

```
ts_mean(close, 8) + ts_std(close, 8) < ts_mean(close, 2)
? -1
```

```

: (ts_mean(close,2) < ts_mean(close, 8) - ts_std(close, 8)
  ? 1
  : (volume / adv20 < 1
    ? -1
    : 1))

```

```

[276]: def alpha031(l, c, adv20):
        """((rank(rank(rank(ts_weighted_mean((-1 * rank(rank(ts_delta(close, 10))), 10))), 10))) +
        rank((-1 * ts_delta(close, 3)))) + sign(scale(ts_corr(adv20, low, 12))))
        """
        return (rank(rank(rank(ts_weighted_mean(rank(rank(ts_delta(c, 10))).
        mul(-1), 10))))
        .add(rank(ts_delta(c, 3).mul(-1)))
        .add(sign(scale(ts_corr(adv20, 1, 12)
        .replace([-np.inf, np.inf],
        np.nan)))))
        .stack('ticker')
        .swaplevel()

```

```

[277]: alpha = 31

```

```

[278]: %%time
alphas[f'{alpha:03}'] = alpha031(l, c, adv20)

```

CPU times: user 3.66 s, sys: 15.9 ms, total: 3.68 s
Wall time: 3.61 s

```

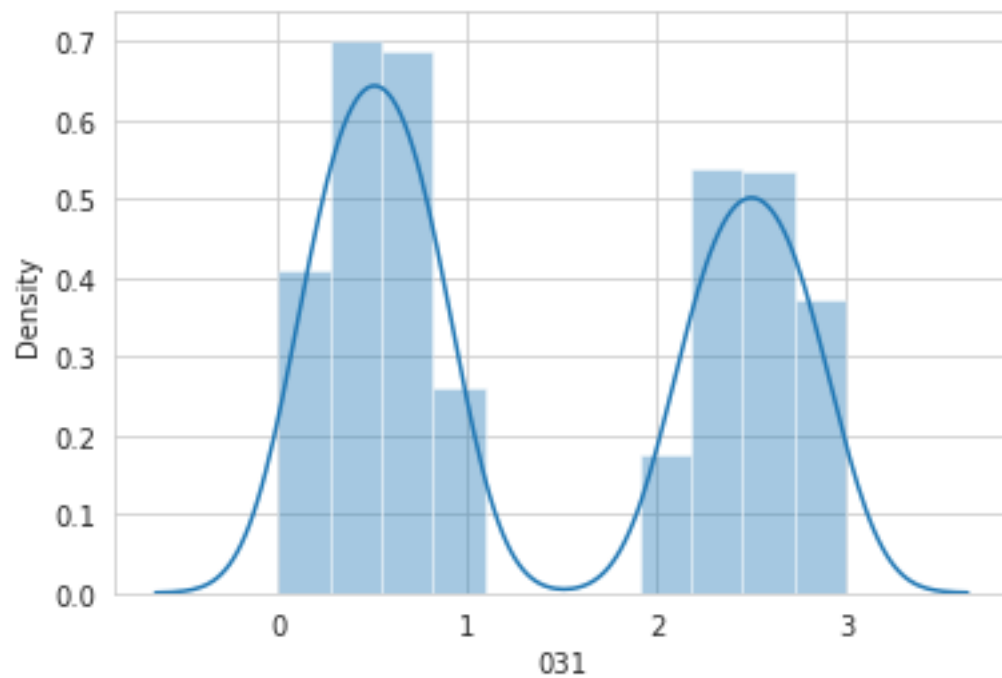
[279]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')

```

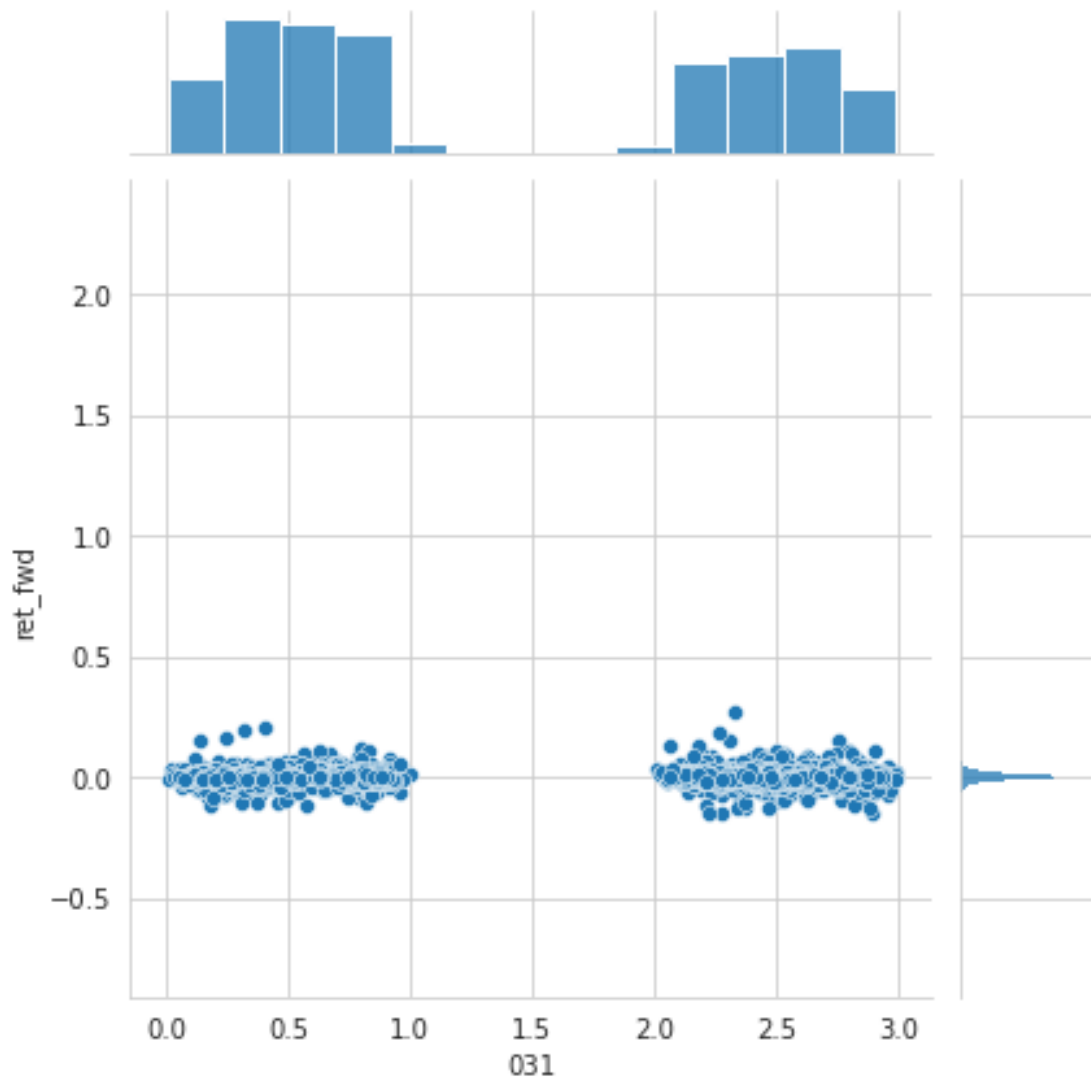
```

[280]: sns.distplot(alphas[f'{alpha:03}']);

```



```
[281]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



1.36 Alpha 032

```
scale(ts_mean(close, 7) - close) + (20 * scale(ts_corr(vwap,
ts_lag(close, 5),230)))
```

```
[282]: def alpha032(c, vwap):
        """scale(ts_mean(close, 7) - close) +
           (20 * scale(ts_corr(vwap, ts_lag(close, 5),230)))"""
        return (scale(ts_mean(c, 7).sub(c))
                .add(20 * scale(ts_corr(vwap,
                                       ts_lag(c, 5), 230)))
                .stack('ticker')
                .swaplevel())
```



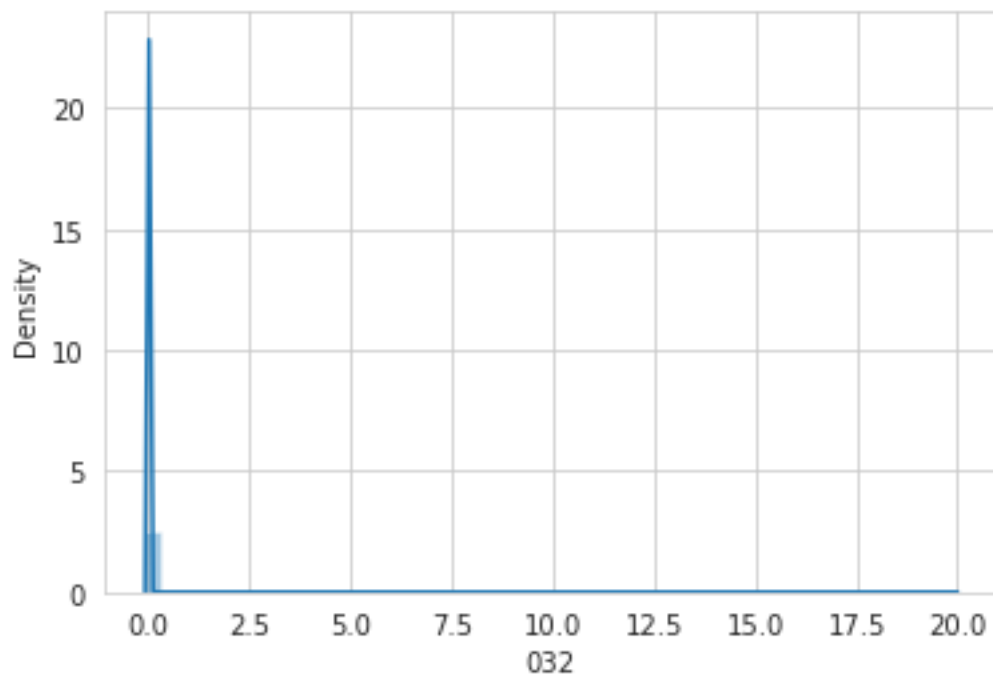
```
[283]: alpha = 32
```

```
[284]: %%time  
alphas[f'{alpha:03}'] = alpha032(c, vwap)
```

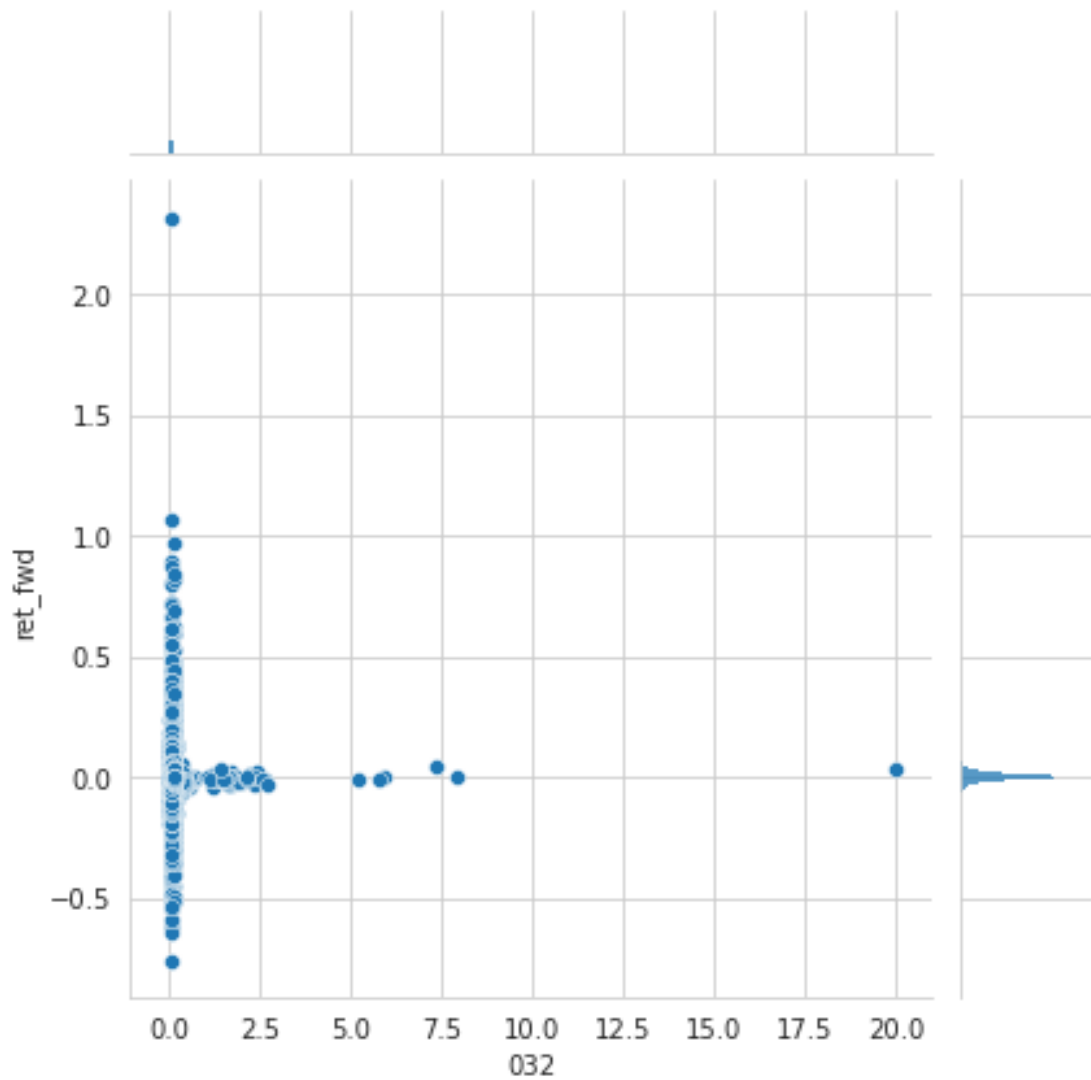
CPU times: user 3.68 s, sys: 15.9 ms, total: 3.69 s
Wall time: 3.63 s

```
[285]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[286]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[287]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[288]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd,
                                         alphas[f'{alpha:03}'])
```

```
[289]: mi[alpha]
```

```
[289]: 0.015078606589863597
```

1.37 Alpha 033

```
((ts_sum(high, 20) / 20) < high)
    ? (-1 * ts_delta(high, 2))
    : 0
```

```
[290]: def alpha033(o, c):
        """rank(-(1 - (open / close)))"""
        return (rank(o.div(c).mul(-1).add(1).mul(-1))
                .stack('ticker')
                .swaplevel())
```

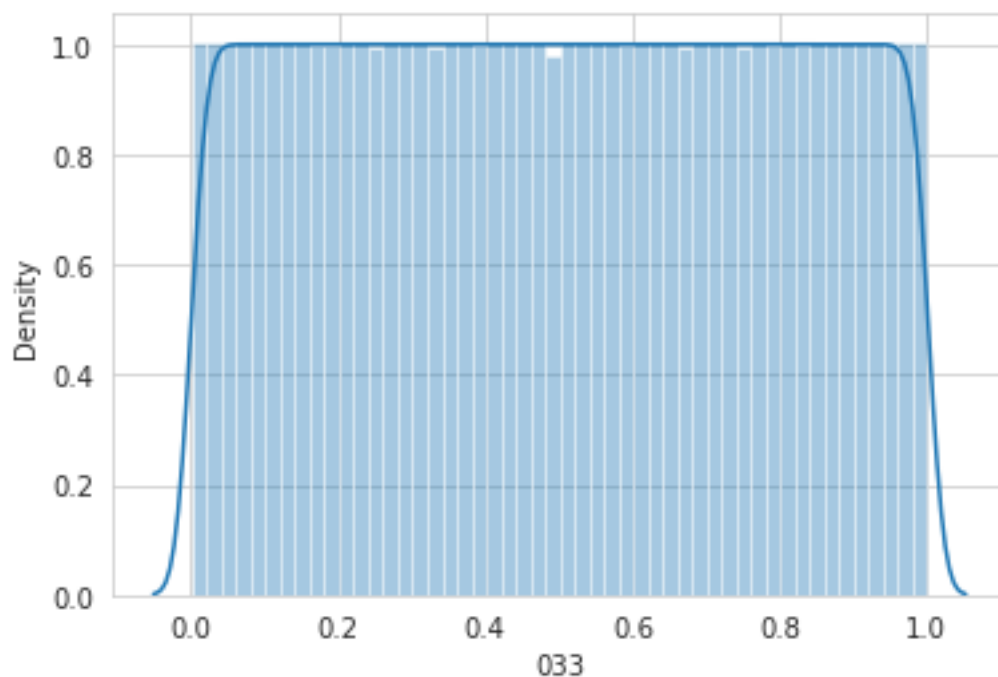
```
[291]: alpha = 33
```

```
[292]: %%time
        alphas[f'{alpha:03}'] = alpha033(o, c)
```

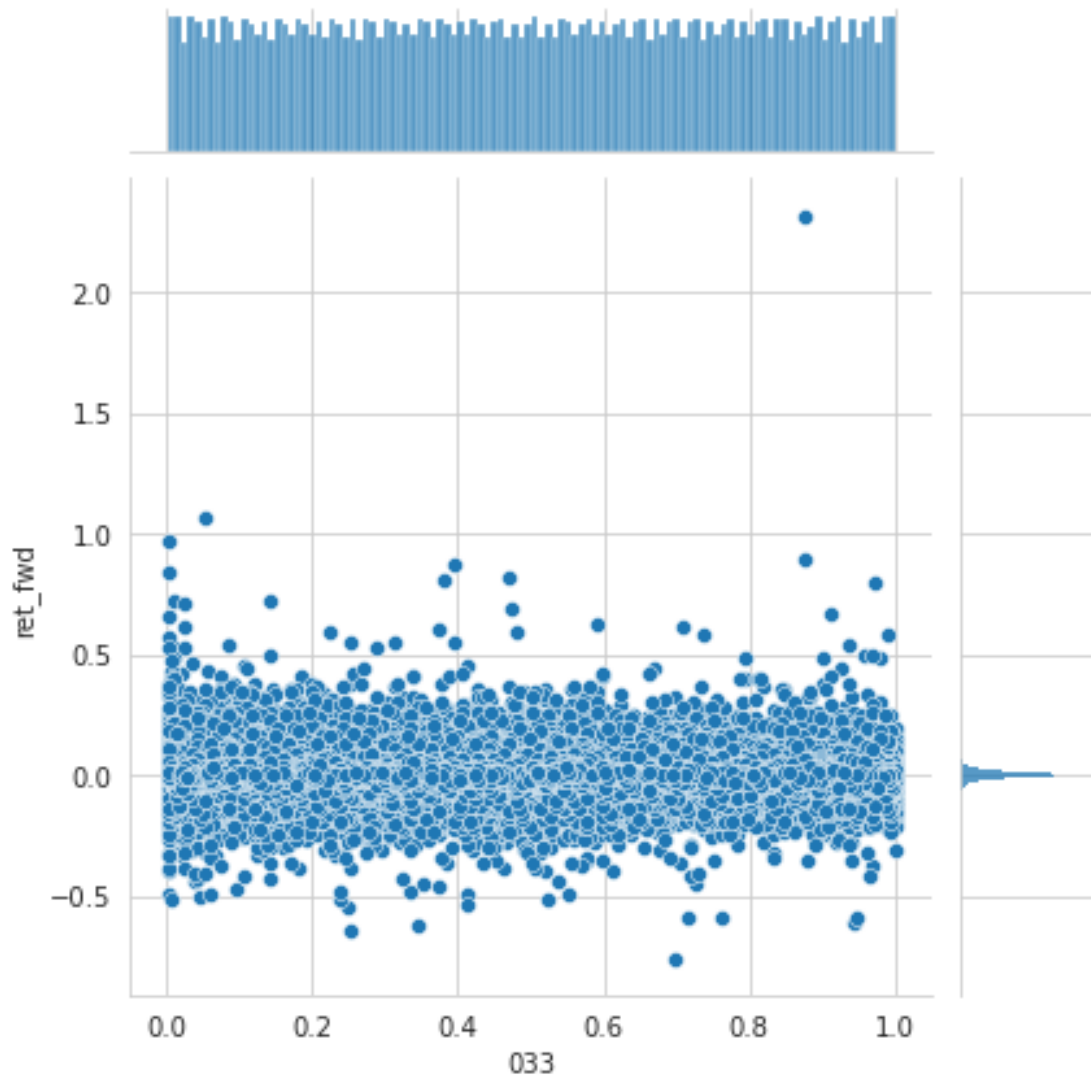
CPU times: user 2.66 s, sys: 12 ms, total: 2.67 s
Wall time: 2.64 s

```
[293]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[294]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[295]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[296]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[297]: mi[alpha]
```

```
[297]: 0.003290483361809038
```

1.38 Alpha 034

```
((((ts_delta((ts_mean(close, 100)), 100) / ts_lag(close, 100)) <= 0.05)
  ? (-1 * (close - ts_min(close, 100)))
  : (-1 * ts_delta(close, 3)))
```

```
[298]: def alpha034(c, r):
        """rank(((1 - rank((ts_std(returns, 2) / ts_std(returns, 5)))) + (1 -
        ↪rank(ts_delta(close, 1))))))"""

        return (rank(rank(ts_std(r, 2).div(ts_std(r, 5))
                        .replace([-np.inf, np.inf],
                                np.nan))
                    .mul(-1)
                    .sub(rank(ts_delta(c, 1)))
                    .add(2))
                .stack('ticker')
                .swaplevel())
```

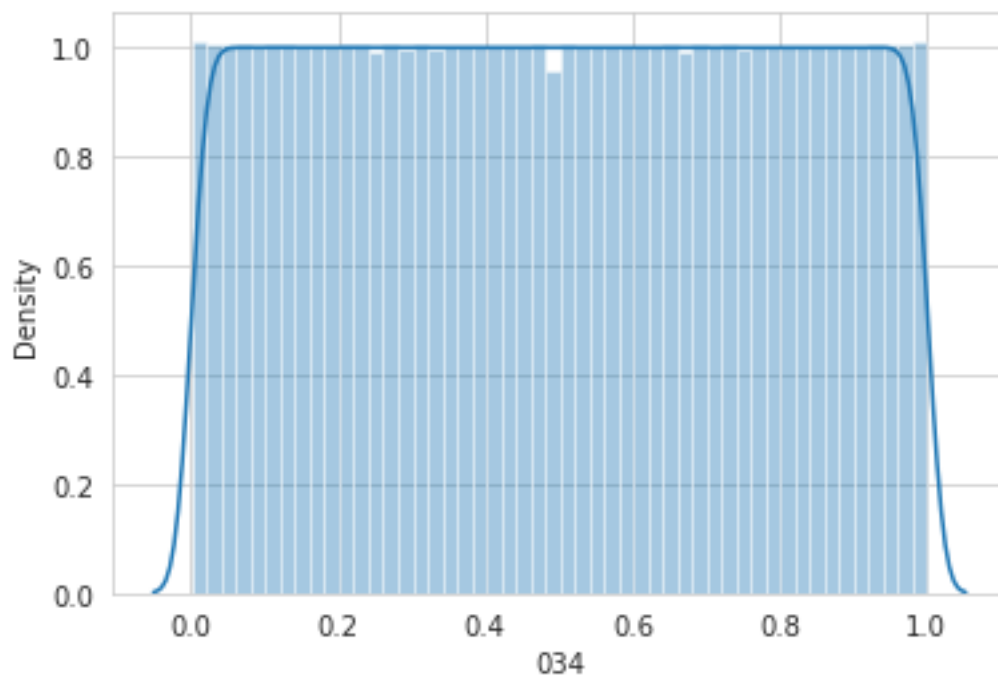
```
[299]: alpha = 34
```

```
[300]: %%time
        alphas[f'{alpha:03}'] = alpha034(c, r)
```

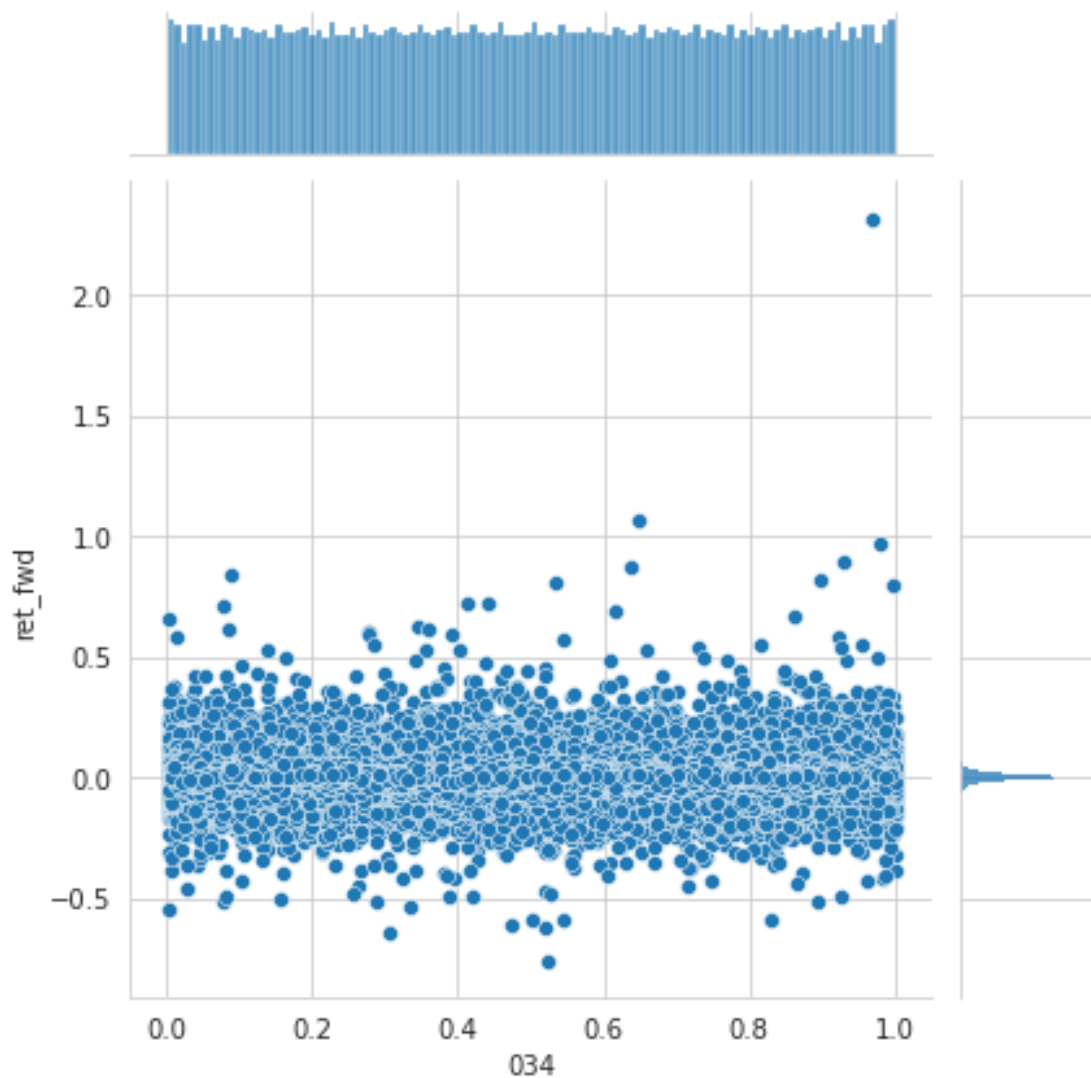
CPU times: user 1.75 s, sys: 8 ms, total: 1.76 s
Wall time: 1.73 s

```
[301]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[302]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[303]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[304]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[305]: mi[alpha]
```

```
[305]: 0
```

```
[306]: pd.Series(mi).to_csv('mi.csv')
```

1.39 Alpha 035

```
rank((-1 * returns) * adv20 * vwap * (high - close))
```

```
[307]: def alpha035(h, l, c, v, r):
        """((ts_Rank(volume, 32) *
            (1 - ts_Rank(((close + high) - low), 16))) *
            (1 - ts_Rank(returns, 32)))
        """
        return (ts_rank(v, 32)
                .mul(1 - ts_rank(c.add(h).sub(l), 16))
                .mul(1 - ts_rank(r, 32))
                .stack('ticker')
                .swaplevel())
```

```
[308]: alpha = 35
```

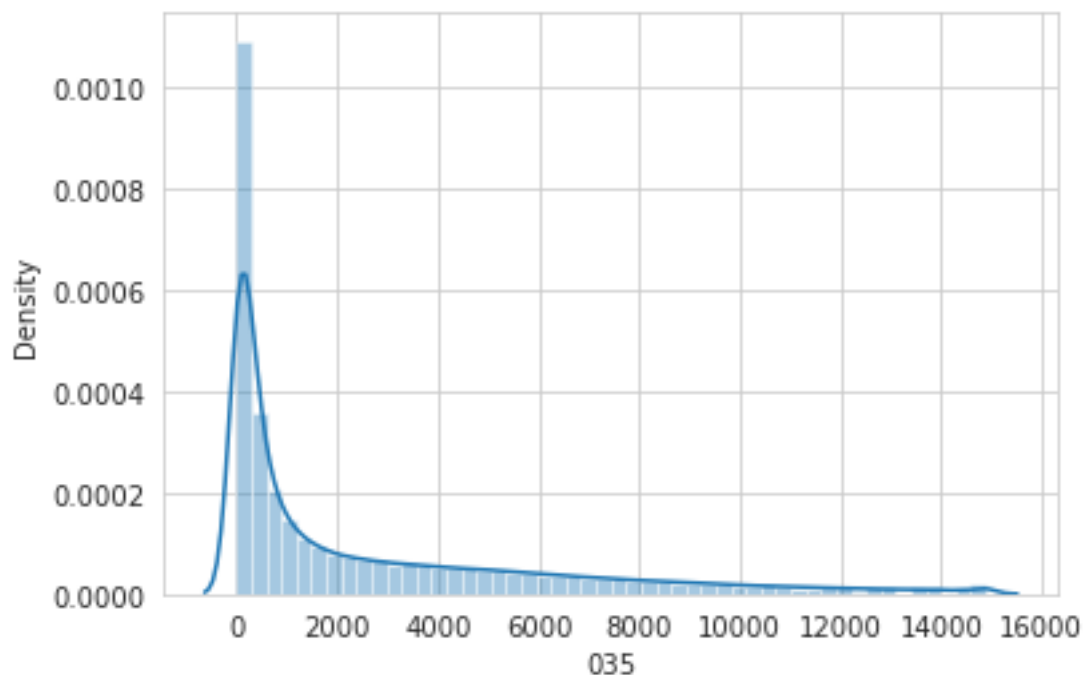
```
[309]: %%time
        alphas[f'{alpha:03}'] = alpha035(h, l, c, v, r)
```

CPU times: user 9min 1s, sys: 95.8 ms, total: 9min 1s

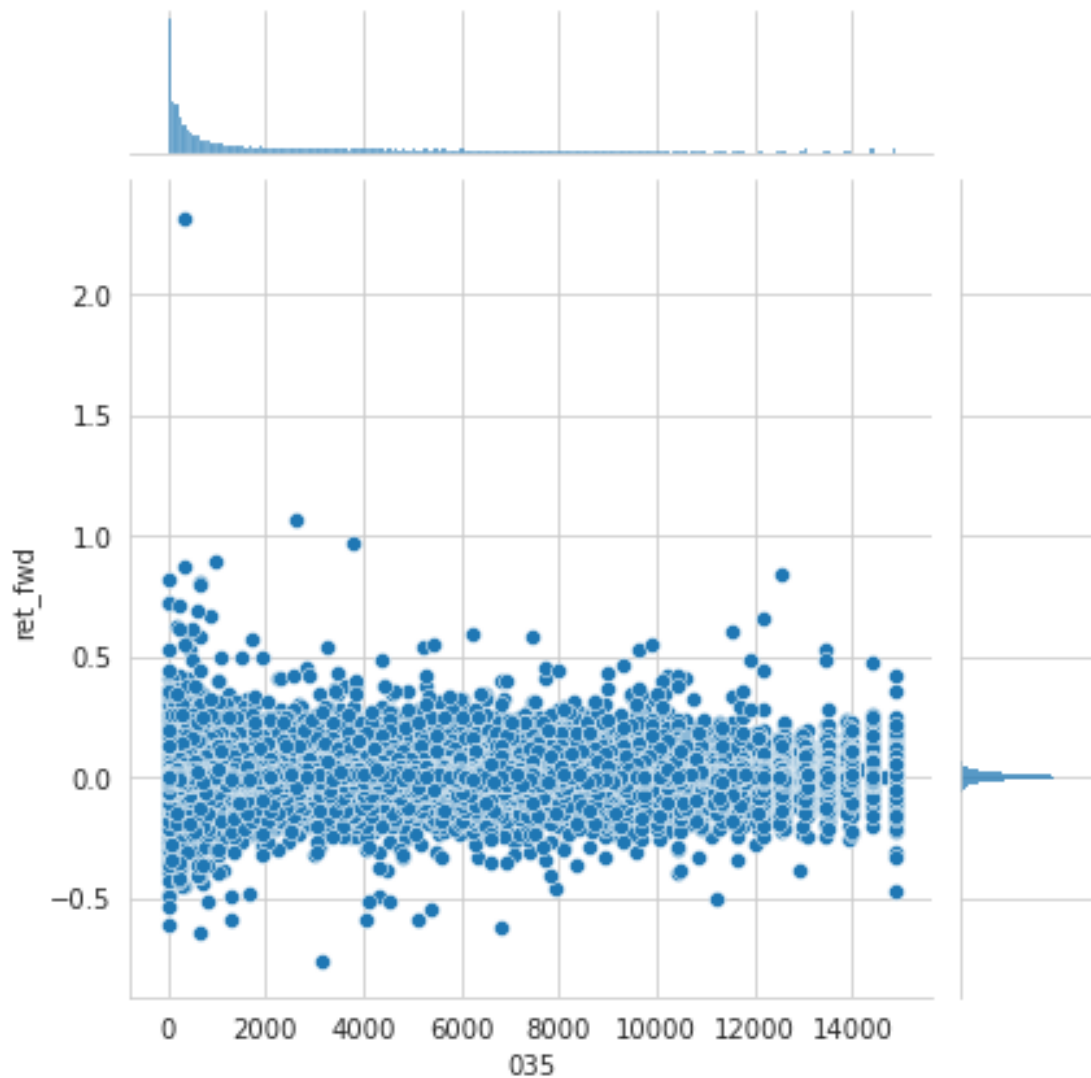
Wall time: 9min 1s

```
[310]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[311]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[312]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[313]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[314]: mi[alpha]
```

```
[314]: 0
```

1.40 Alpha 036

```
2.21 * rank(ts_corr((close - open), ts_lag(volume, 1), 15)) +
0.7 * rank((open - close)) +
0.73 * rank(ts_Rank(ts_lag(-1 * returns, 6), 5)) +
rank(abs(ts_corr(vwap, adv20, 6))) +
0.6 * rank(((ts_mean(close, 200) - open) * (close - open)))
```



```
[315]: def alpha036(o, c, v, r, adv20):
        """2.21 * rank(ts_corr((close - open), ts_lag(volume, 1), 15)) +
        0.7 * rank((open- close)) +
        0.73 * rank(ts_Rank(ts_lag(-1 * returns, 6), 5)) +
        rank(abs(ts_corr(vwap,adv20, 6))) +
        0.6 * rank(((ts_mean(close, 200) - open) * (close - open)))
        """

        return (rank(ts_corr(c.sub(o), ts_lag(v, 1), 15)).mul(2.21)
                .add(rank(o.sub(c)).mul(.7))
                .add(rank(ts_rank(ts_lag(-r, 6), 5)).mul(0.73))
                .add(rank(abs(ts_corr(vwap, adv20, 6))))
                .add(rank(ts_mean(c, 200).sub(o).mul(c.sub(o))).mul(0.6))
                .stack('ticker')
                .swaplevel())
```

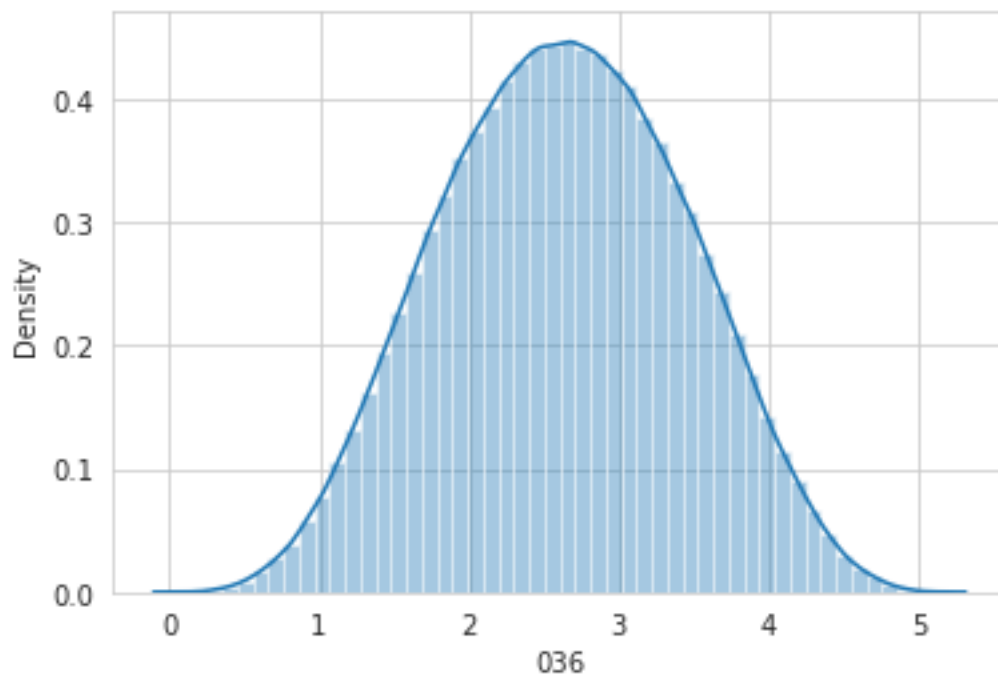
```
[316]: alpha = 36
```

```
[317]: %%time
        alphas[f'{alpha:03}'] = alpha036(o, c, v, r, adv20)
```

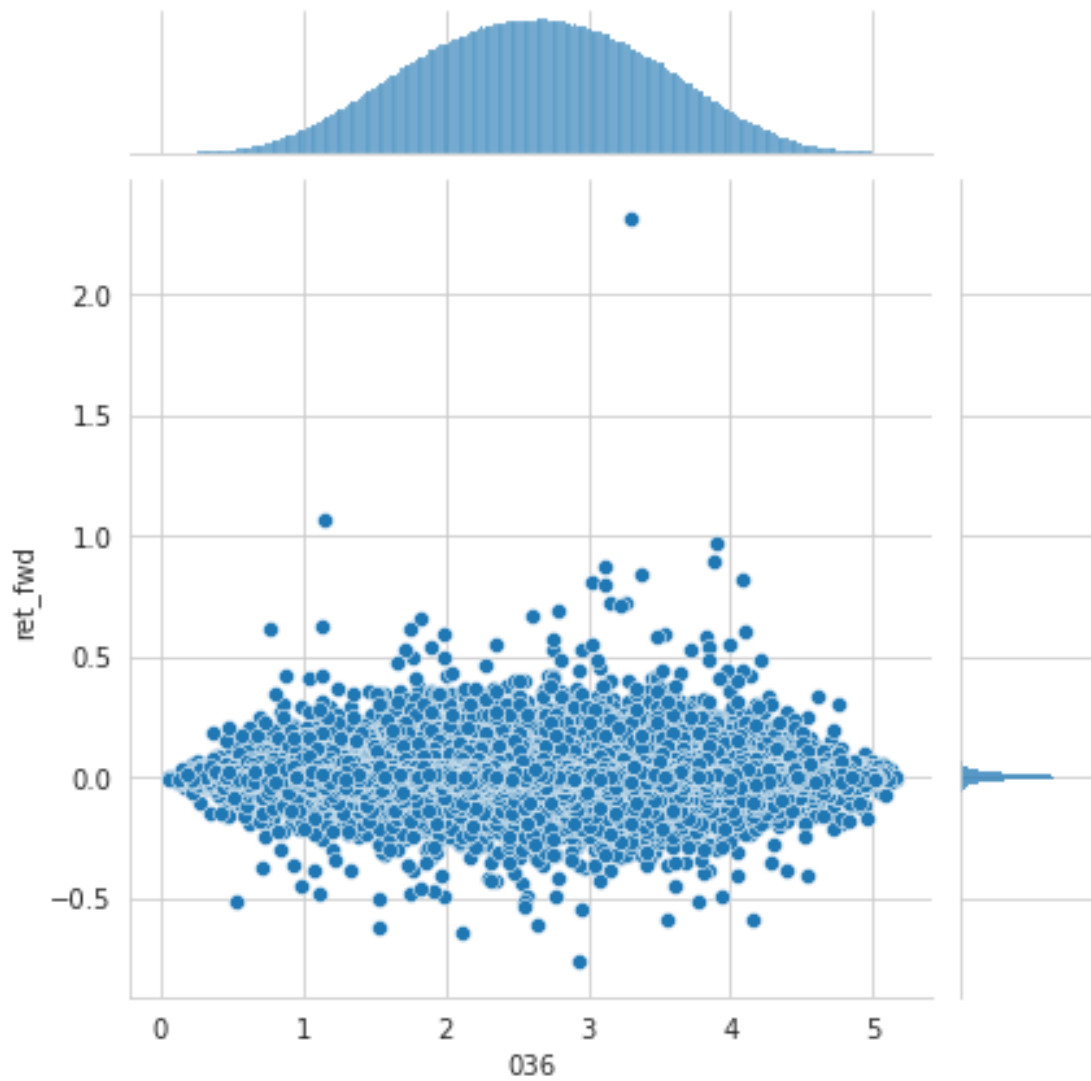
CPU times: user 3min 5s, sys: 51.9 ms, total: 3min 5s
 Wall time: 3min 5s

```
[318]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[319]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[320]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[321]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[322]: mi[alpha]
```

```
[322]: 0.0017091501300177114
```

1.41 Alpha 037

```
rank(ts_corr(ts_lag(open - close, 1), close, 200)) +
```

```
rank(open - close)
```

```
[323]: def alpha037(o, c):  
        """(rank(ts_corr(ts_lag((open - close), 1), close, 200)) + rank((open -  
        ↪close)))"""  
        return (rank(ts_corr(ts_lag(o.sub(c), 1), c, 200))  
                .add(rank(o.sub(c)))  
                .stack('ticker')  
                .swaplevel())
```

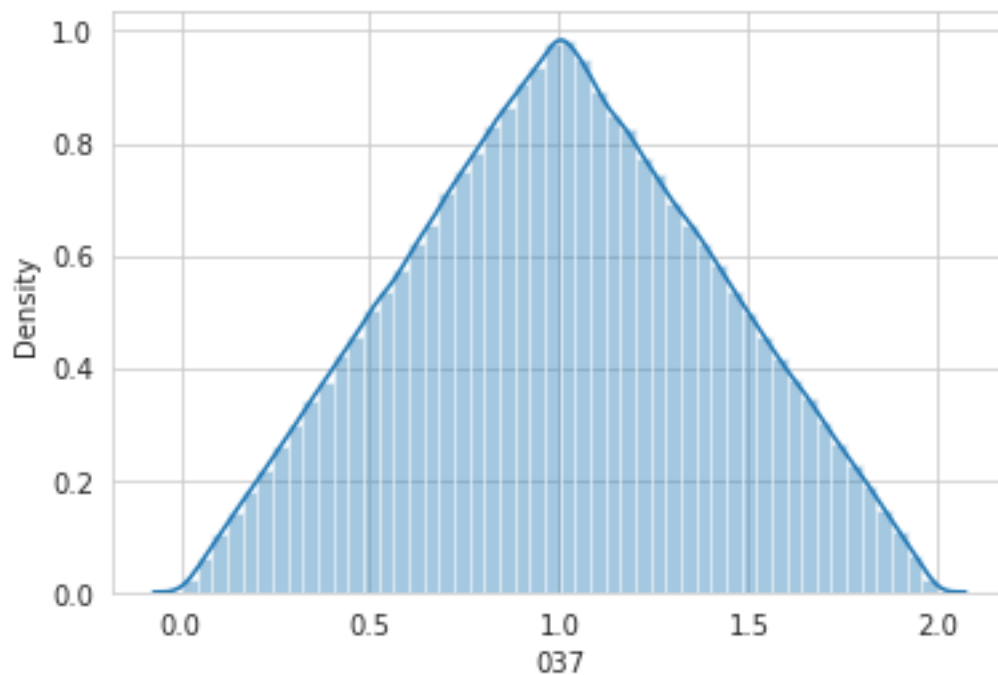
```
[324]: alpha = 37
```

```
[325]: %%time  
alphas[f'{alpha:03}'] = alpha037(o, c)
```

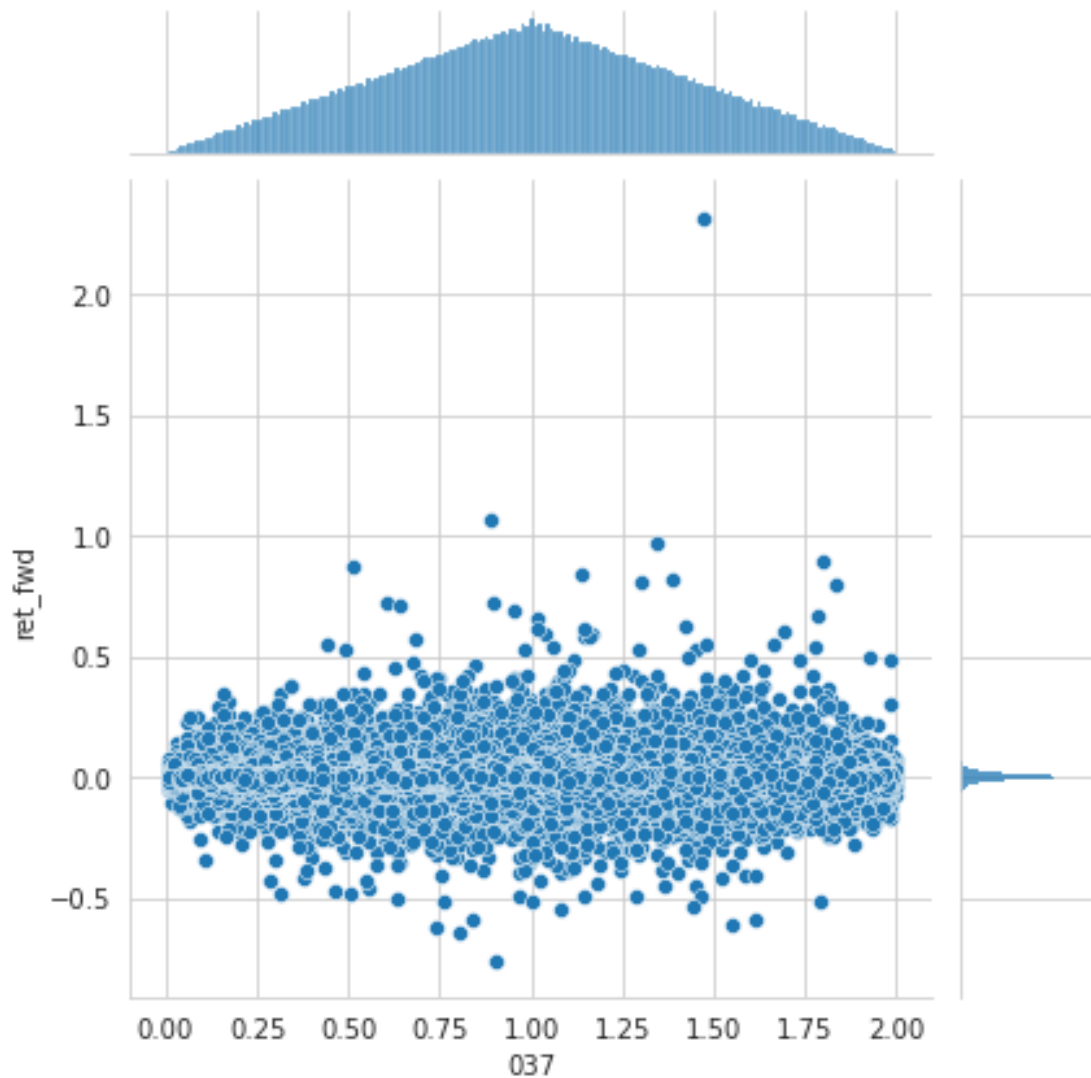
CPU times: user 3.51 s, sys: 32 ms, total: 3.54 s
Wall time: 3.49 s

```
[326]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[327]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[328]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[329]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[330]: mi[alpha]
```

```
[330]: 0.0011419189372663396
```

```
[331]: pd.Series(mi).to_csv('mi.csv')
```

1.42 Alpha 038

```
1 * rank(ts_rank(close, 10)) * rank(close / open)
```

```
[332]: def alpha038(o, c):
        """-1 * rank(ts_rank(close, 10)) * rank(close / open)"""
        return (rank(ts_rank(o, 10))
                .mul(rank(c.div(o).replace([-np.inf, np.inf], np.nan)))
                .mul(-1)
                .stack('ticker')
                .swaplevel())
```

```
[333]: alpha = 38
```

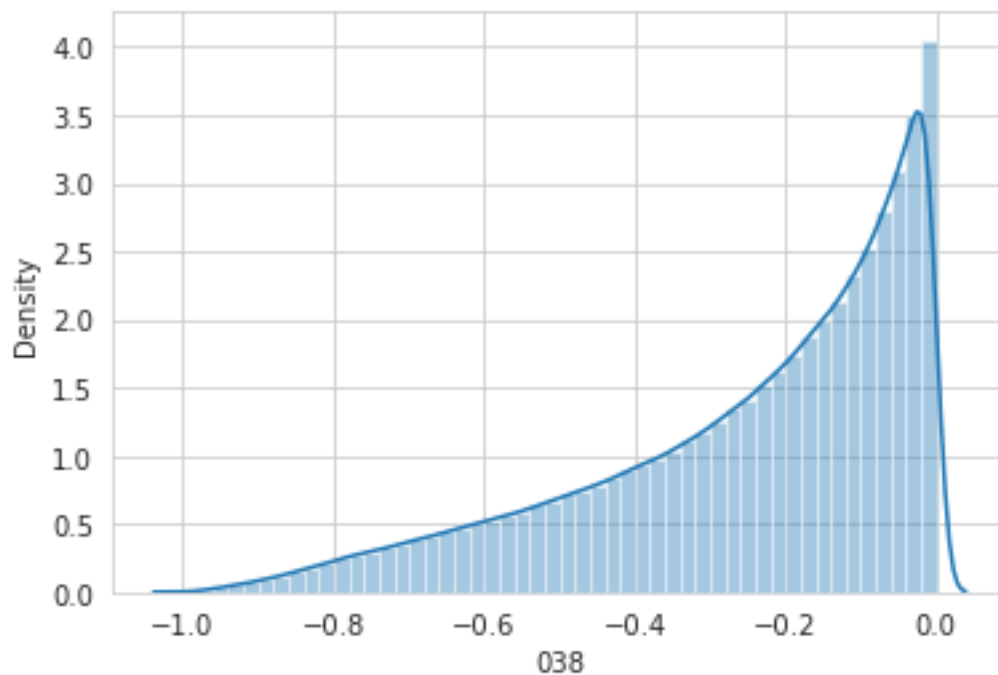
```
[334]: %%time
        alphas[f'{alpha:03}'] = alpha038(o, c)
```

CPU times: user 3min 5s, sys: 15.9 ms, total: 3min 5s

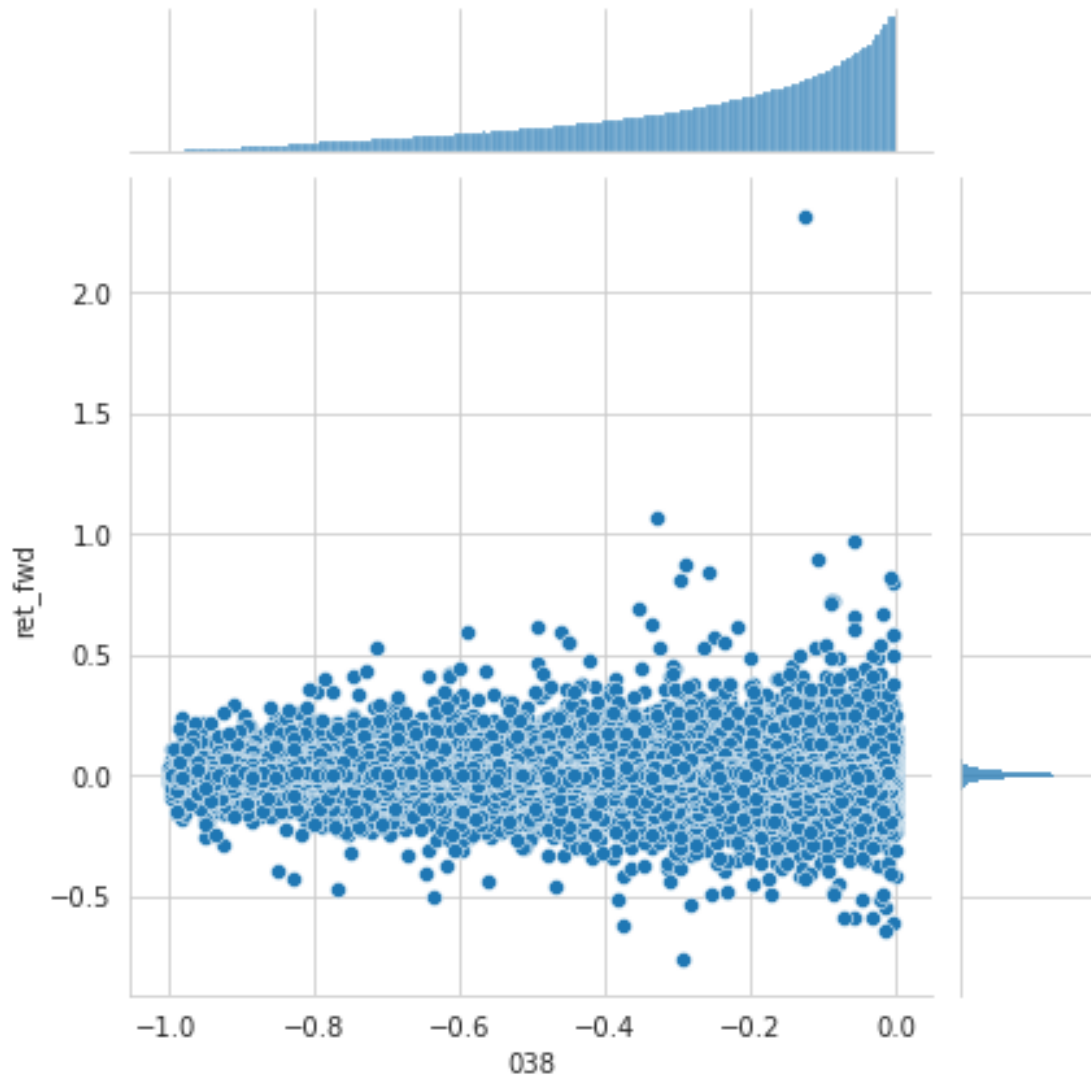
Wall time: 3min 5s

```
[335]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[336]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[337]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[338]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[339]: mi[alpha]
```

```
[339]: 0.003271738846891026
```

1.43 Alpha 039

```
-rank(ts_delta(close, 7) * (1 - rank(ts_weighted_mean(volume / adv20, 9)))) *  
    (1 + rank(ts_sum(returns, 250)))
```

```
[340]: def alpha039(c, v, r, adv20):  
        """-rank(ts_delta(close, 7) * (1 - rank(ts_weighted_mean(volume / adv20, 9)))) *  
            ↪ 9)))) *
```

```

        (1 + rank(ts_sum(returns, 250)))"""
    return (rank(ts_delta(c, 7).mul(rank(ts_weighted_mean(v.div(adv20), 9)).
→mul(-1).add(1))).mul(-1)
        .mul(rank(ts_mean(r, 250).add(1)))
        .stack('ticker')
        .swaplevel())

```

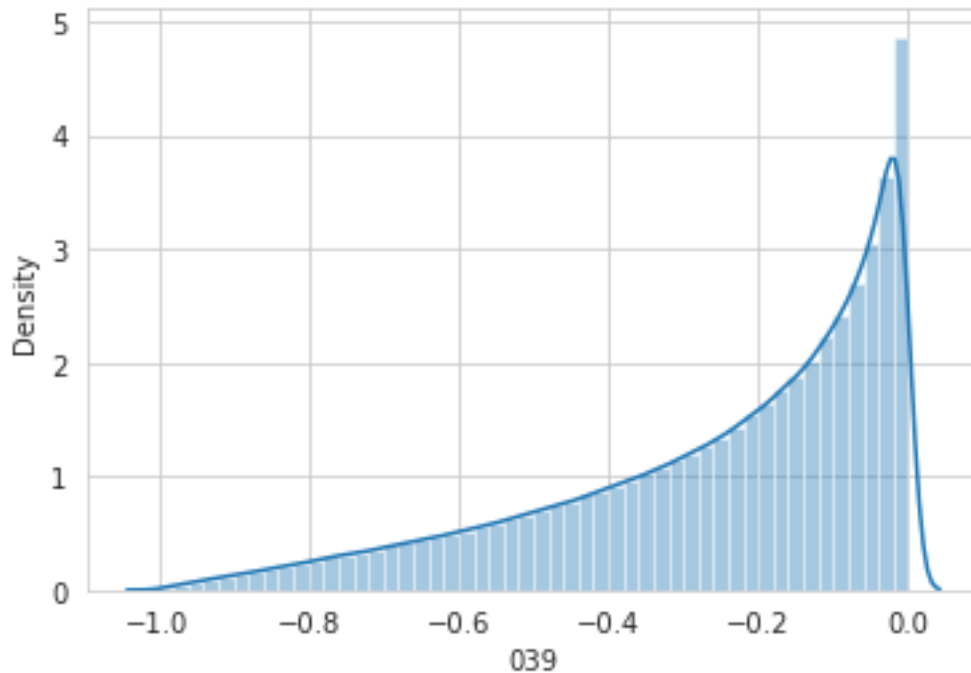
```
[341]: alpha = 39
```

```
[342]: %%time
alphas[f'{alpha:03}'] = alpha039(c, v, r, adv20)
```

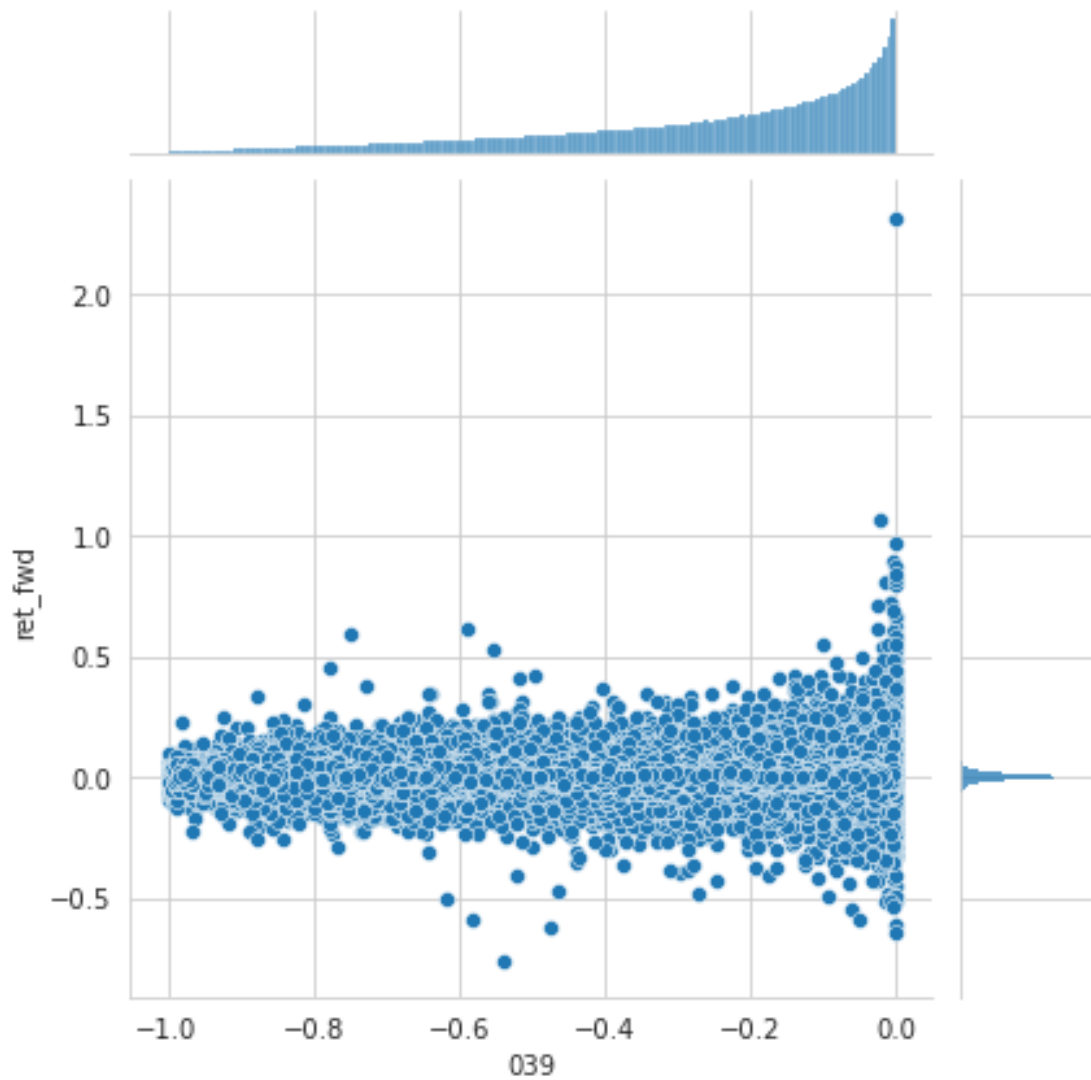
CPU times: user 2.55 s, sys: 16 ms, total: 2.56 s
Wall time: 2.5 s

```
[343]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[344]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[345]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[346]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
```

```
[347]: mi[alpha]
```

```
[347]: 0.0021417818275919487
```

1.44 Alpha 040

```
-rank(open - ts_lag(high, 1)) *  
rank(open - ts_lag(close, 1)) *  
rank(open - ts_lag(low, 1))
```



```
[348]: def alpha040(h, v):
        """((-1 * rank(ts_std(high, 10))) * ts_corr(high, volume, 10))
        """
        return (rank(ts_std(h, 10))
                .mul(ts_corr(h, v, 10))
                .mul(-1)
                .stack('ticker')
                .swaplevel())
```

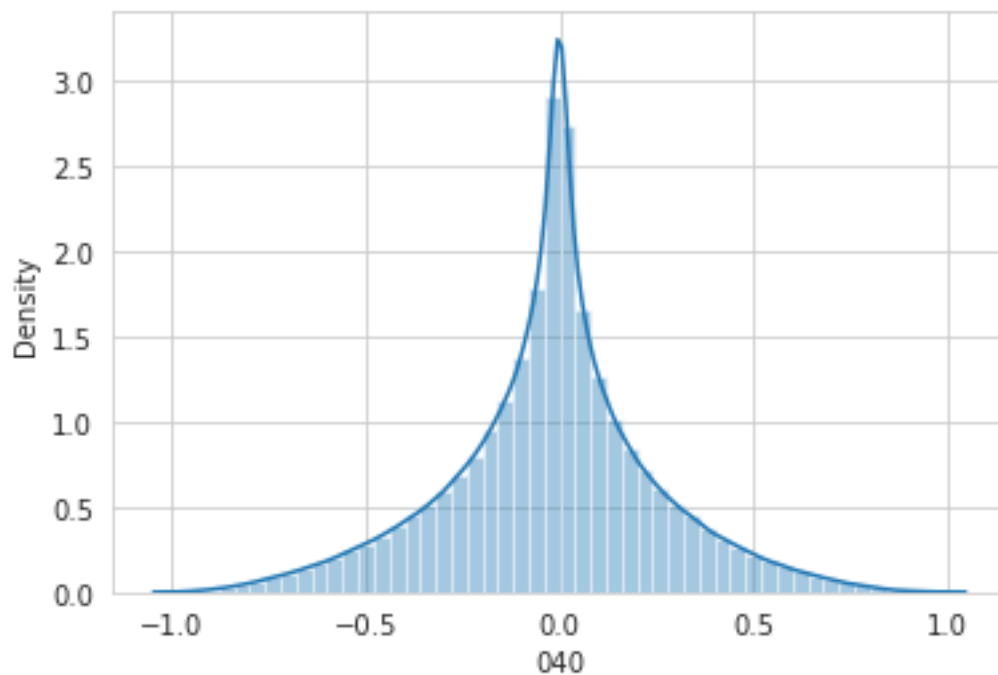
```
[349]: alpha = 40
```

```
[350]: %%time
        alphas[f'{alpha:03}'] = alpha040(h, v)
```

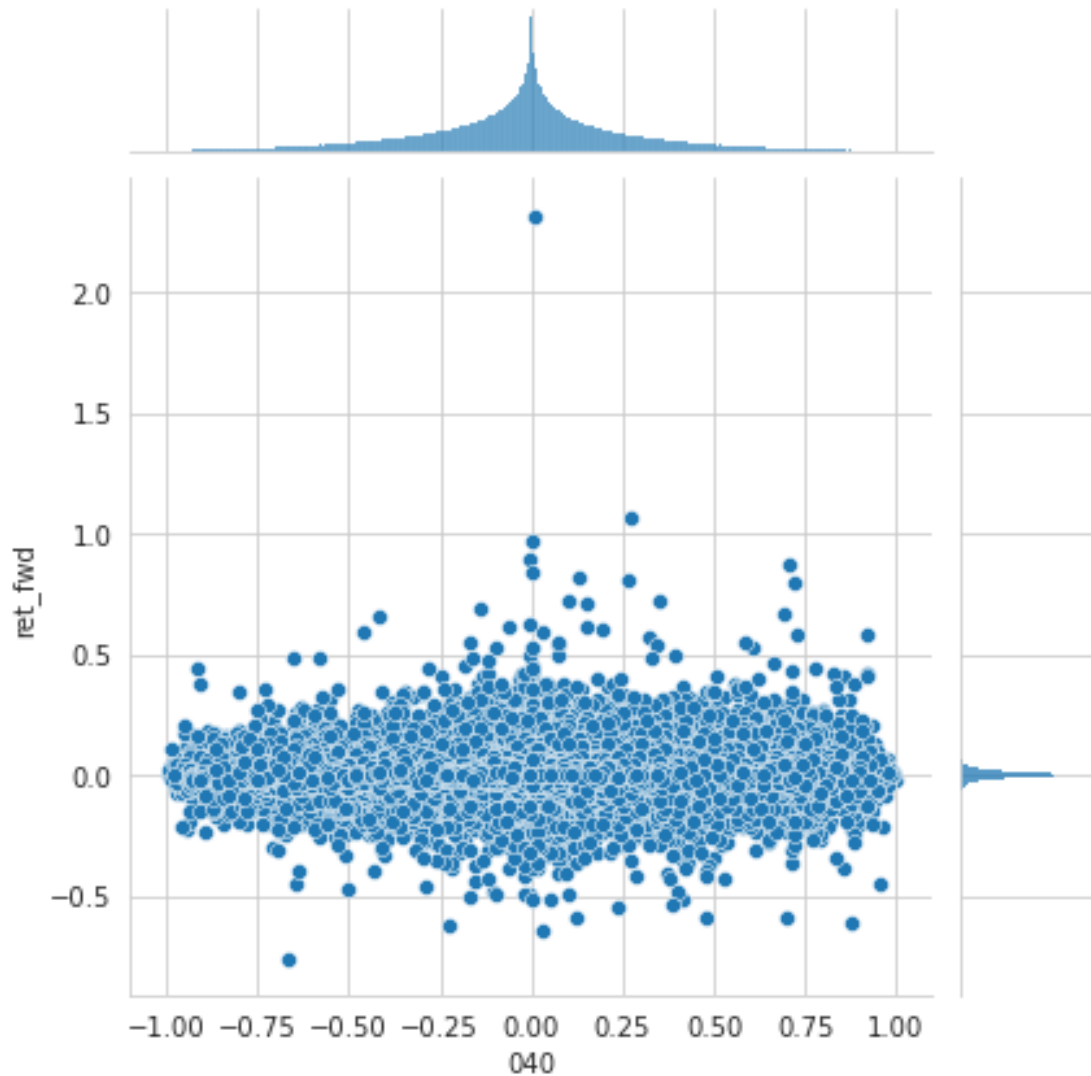
CPU times: user 3.71 s, sys: 15.9 ms, total: 3.72 s
Wall time: 3.68 s

```
[351]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[352]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[353]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[354]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[354]: 0.008389463848225809
```

1.45 Alpha 041

power(high * low, 0.5) - vwap

```
[355]: def alpha041(h, l, vwap):
        """power(high * low, 0.5 - vwap)"""
        return (power(h.mul(l), 0.5)
                .sub(vwap)
                .stack('ticker'))
```

```
.swaplevel()
```

```
[356]: alpha = 41
```

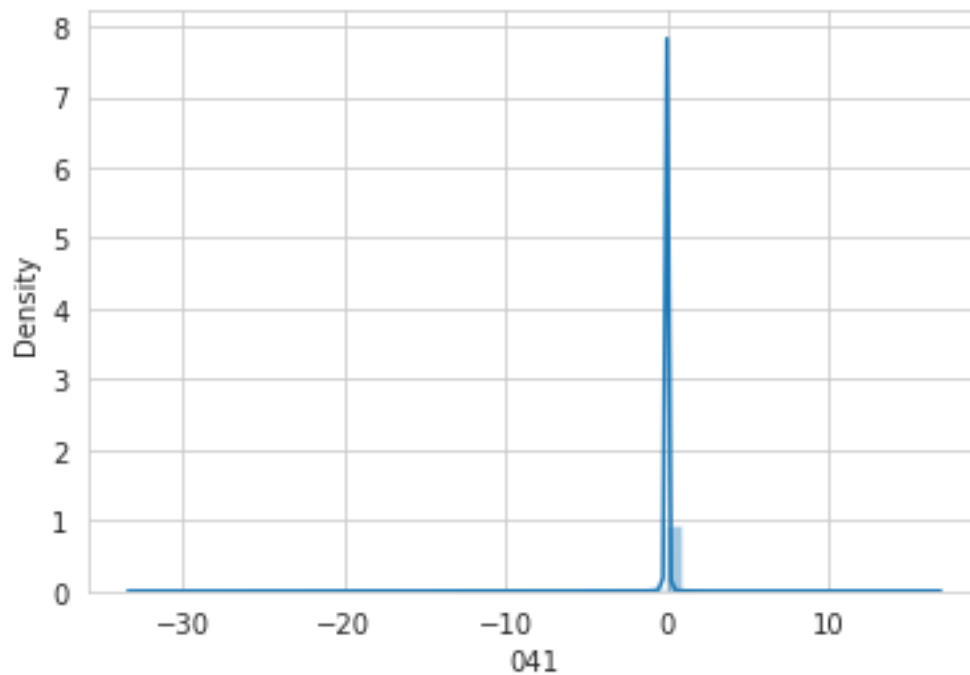
```
[357]: %%time  
alphas[f'{alpha:03}'] = alpha041(h, l, vwap)
```

CPU times: user 2.32 s, sys: 16 ms, total: 2.34 s

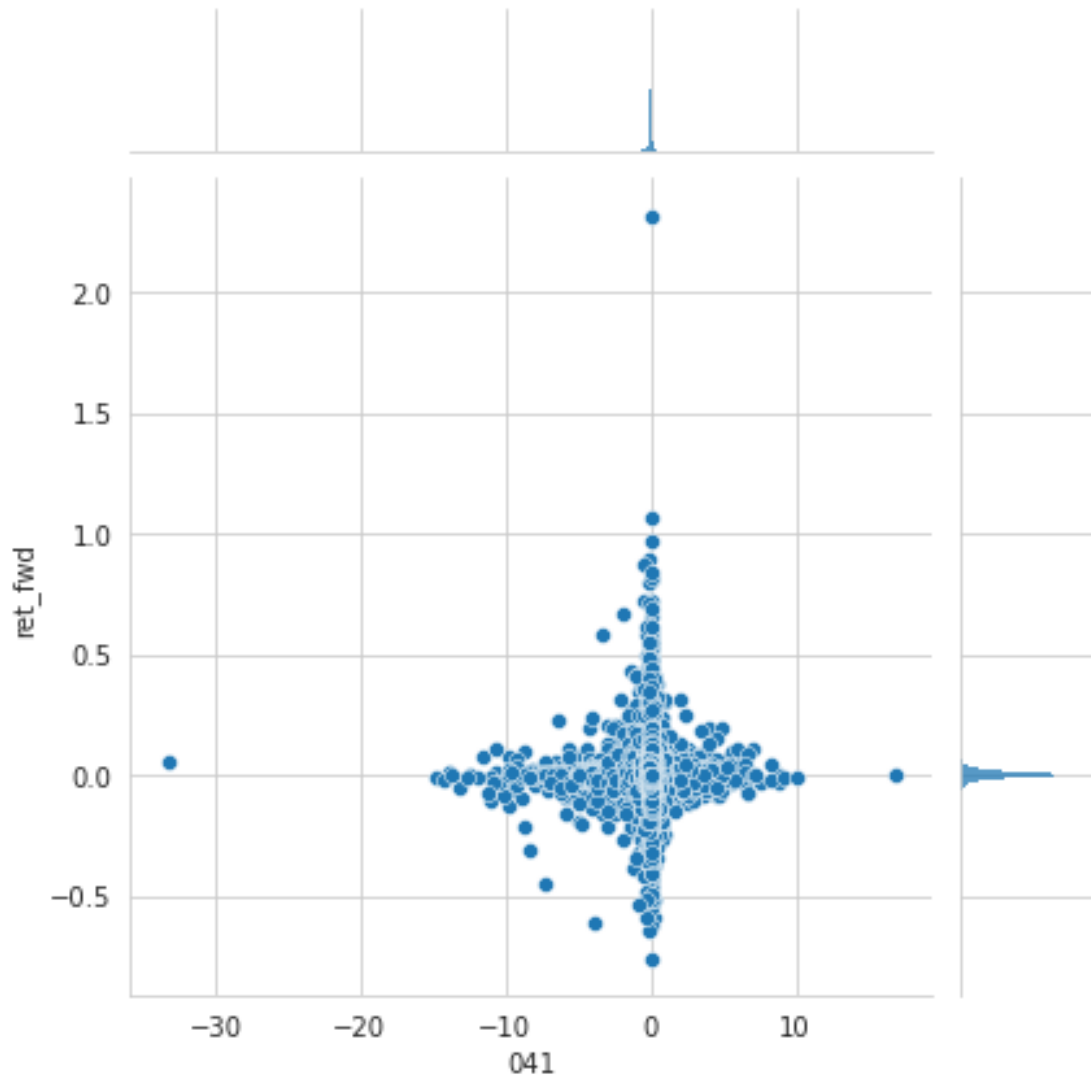
Wall time: 2.3 s

```
[358]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[359]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[360]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[361]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[361]: 0.004631922179492598
```

1.46 Alpha 042

$\text{rank}(\text{vwap} - \text{close}) / \text{rank}(\text{vwap} + \text{close})$

```
[362]: def alpha042(c, vwap):
        """rank(vwap - close) / rank(vwap + close)"""
        return (rank(vwap.sub(c))
                .div(rank(vwap.add(c)))
                .stack('ticker'))
```

```
.swaplevel()
```

```
[363]: alpha = 42
```

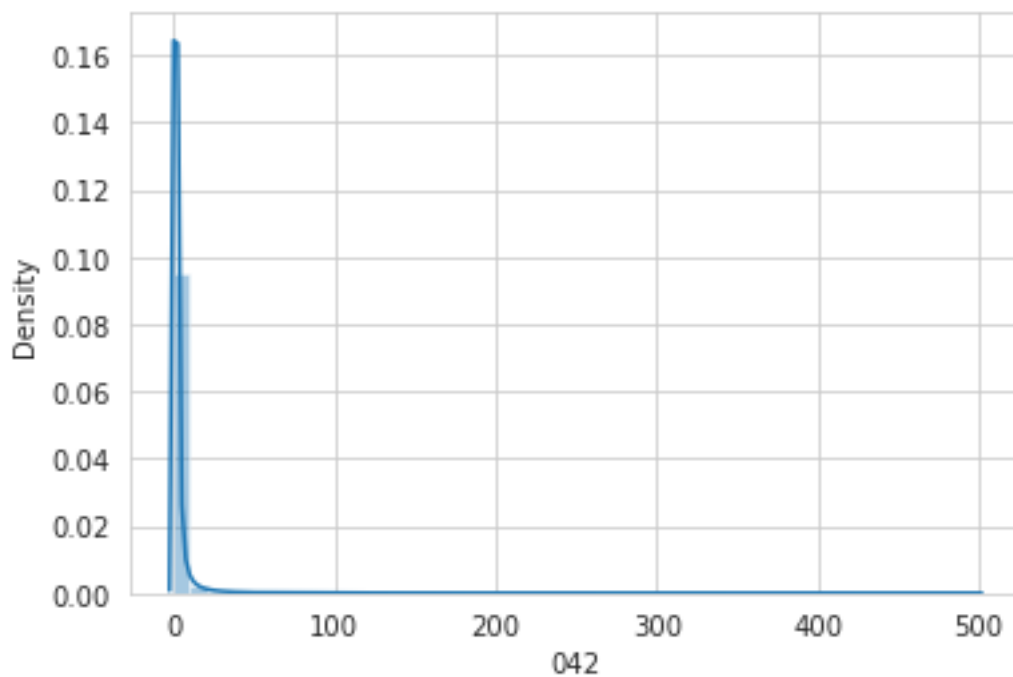
```
[364]: %%time  
alphas[f'{alpha:03}'] = alpha042(c, vwap)
```

CPU times: user 2.43 s, sys: 36 ms, total: 2.46 s

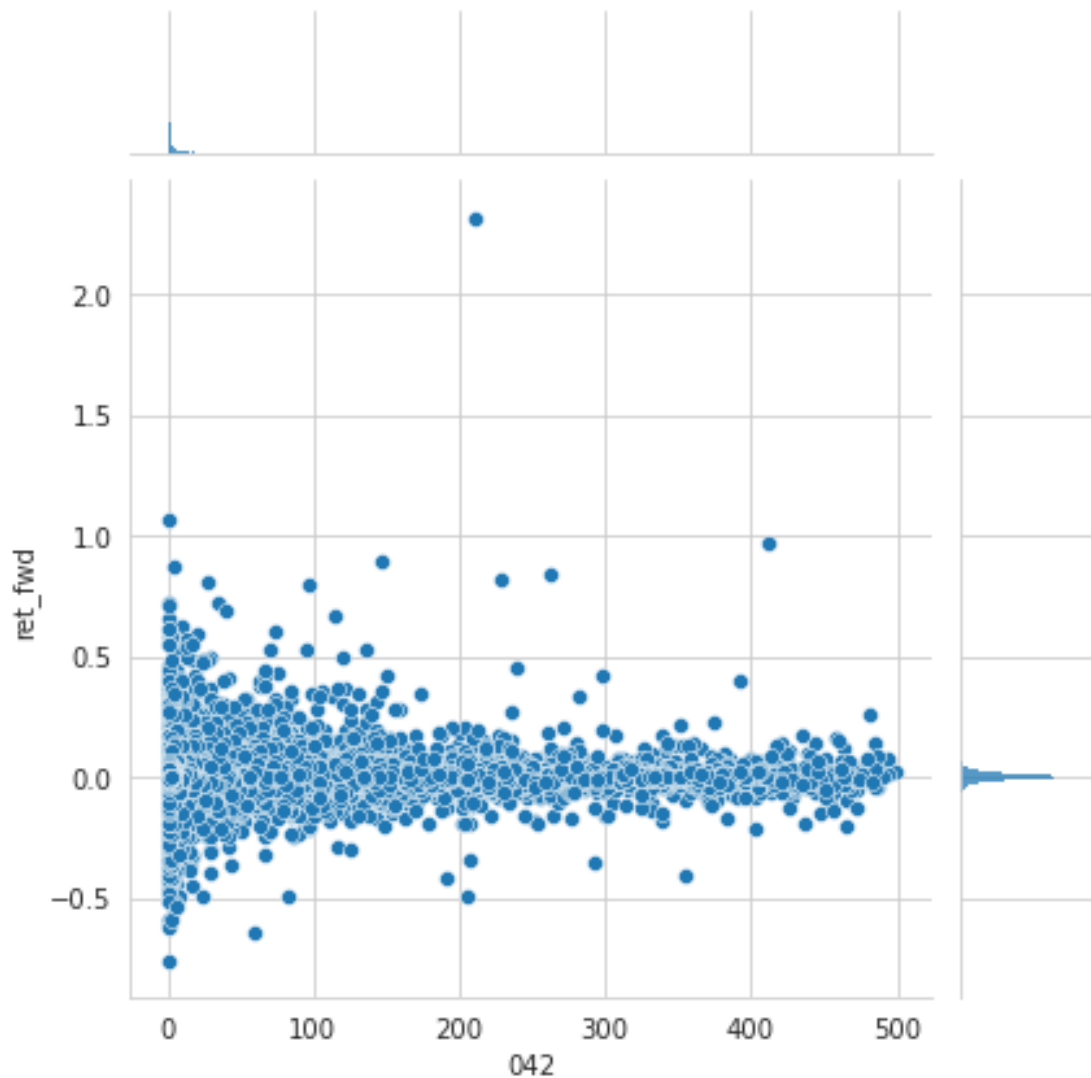
Wall time: 2.43 s

```
[365]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[366]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[367]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[368]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

[368]: 0.0003211794350050923

1.47 Alpha 043

```
((ts_sum(high, 20) / 20) < high)
    ? (-1 * ts_delta(high, 2))
    : 0
```

```
[369]: def alpha043(c, adv20):
        """(ts_rank((volume / adv20), 20) * ts_rank((-1 * ts_delta(close, 7)), 8))"""
```

```

return (ts_rank(v.div(adv20), 20)
        .mul(ts_rank(ts_delta(c, 7).mul(-1), 8))
        .stack('ticker')
        .swaplevel())

```

```
[370]: alpha = 43
```

```

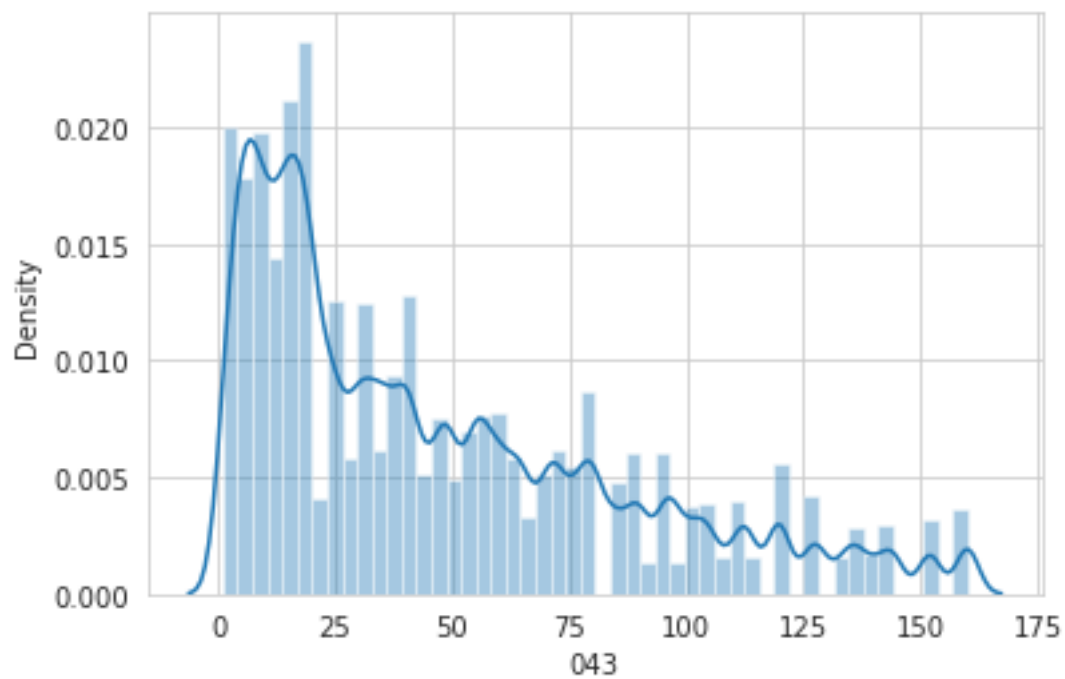
[371]: %%time
alphas[f'{alpha:03}'] = alpha043(c, adv20)

```

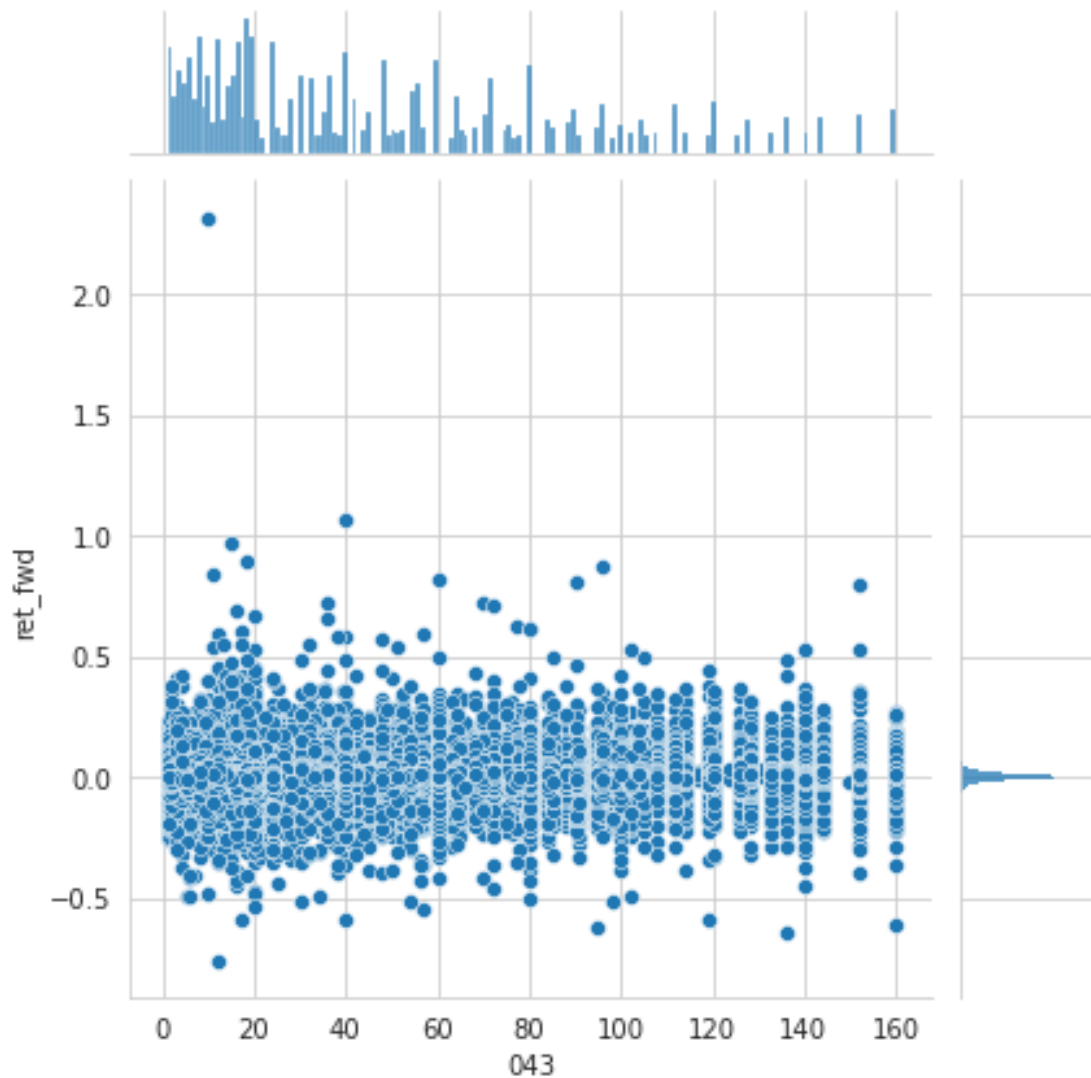
CPU times: user 6min 2s, sys: 72 ms, total: 6min 2s
Wall time: 6min 2s

```
[372]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[373]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[374]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[375]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[375]: 3.31646414482023e-05
```

1.48 Alpha 044

```
-ts_corr(high, rank(volume), 5)
```

```
[376]: def alpha044(h, v):
        """-ts_corr(high, rank(volume), 5)"""

        return (ts_corr(h, rank(v), 5)
                .replace([-np.inf, np.inf], np.nan))
```



```
.mul(-1)
.stack('ticker')
.swaplevel()
```

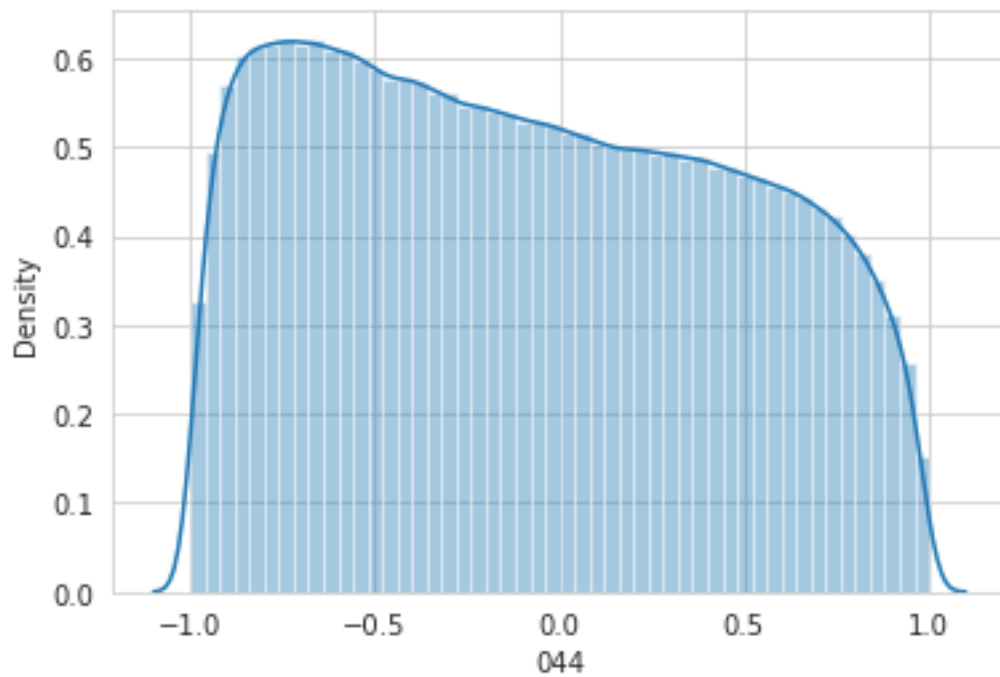
```
[377]: alpha = 44
```

```
[378]: %%time
alphas[f'{alpha:03}'] = alpha044(h, v)
```

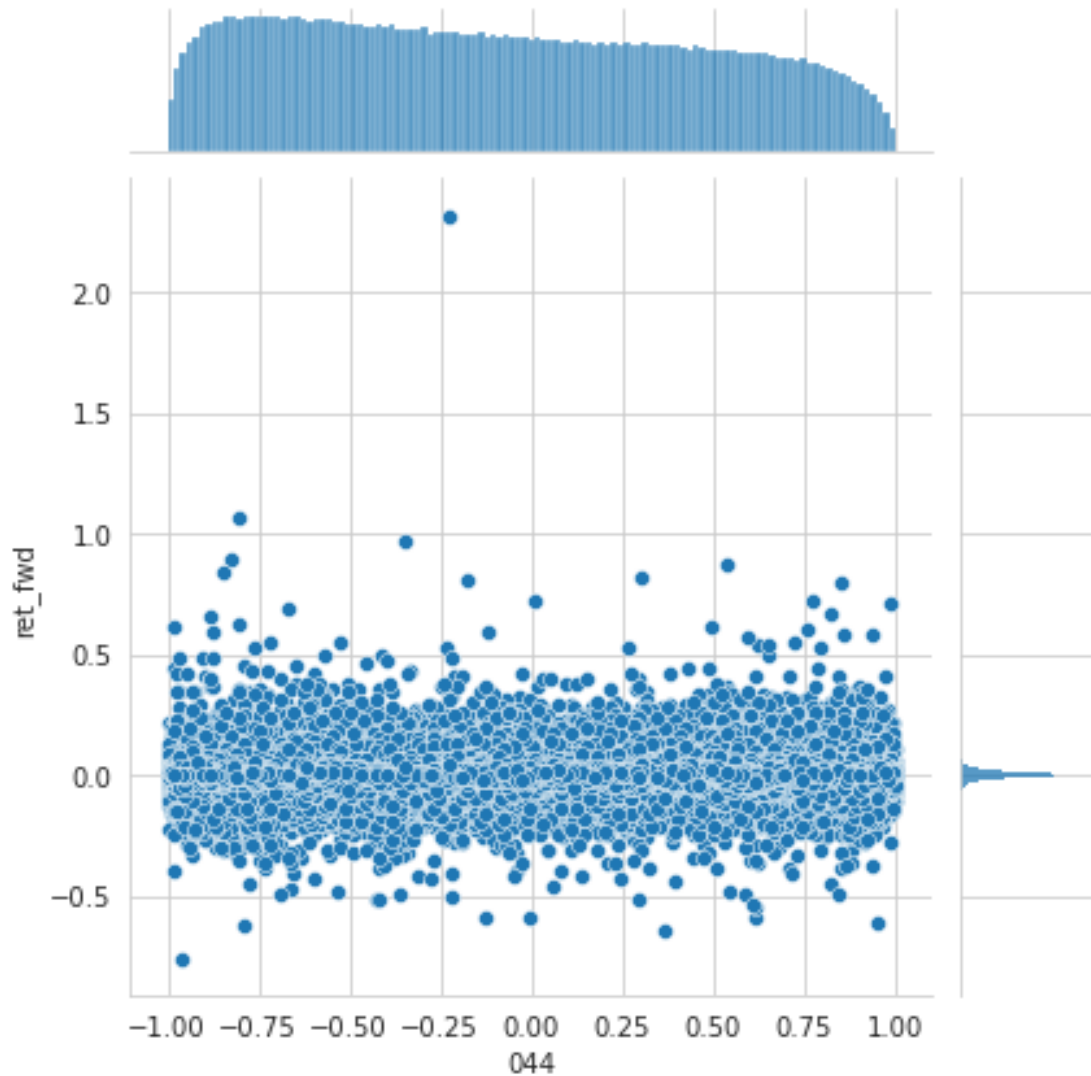
CPU times: user 3.69 s, sys: 28 ms, total: 3.72 s
Wall time: 3.69 s

```
[379]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[380]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[381]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[382]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

[382]: 0

1.49 Alpha 045

```
-(rank((ts_mean(ts_lag(close, 5), 20)) *
      ts_corr(close, volume, 2)) *
  rank(ts_corr(ts_sum(close, 5), ts_sum(close, 20), 2)))
```

```
[383]: def alpha045(c, v):
        """-(rank((ts_mean(ts_lag(close, 5), 20)) *
                  ts_corr(close, volume, 2)) *
            rank(ts_corr(ts_sum(close, 5), ts_sum(close, 20), 2)))
```

```

rank(ts_corr(ts_sum(close, 5), ts_sum(close, 20), 2)))"""

return (rank(ts_mean(ts_lag(c, 5), 20))
        .mul(ts_corr(c, v, 2)
              .replace([-np.inf, np.inf], np.nan))
        .mul(rank(ts_corr(ts_sum(c, 5),
                          ts_sum(c, 20), 2)))
        .mul(-1)
        .stack('ticker')
        .swaplevel())

```

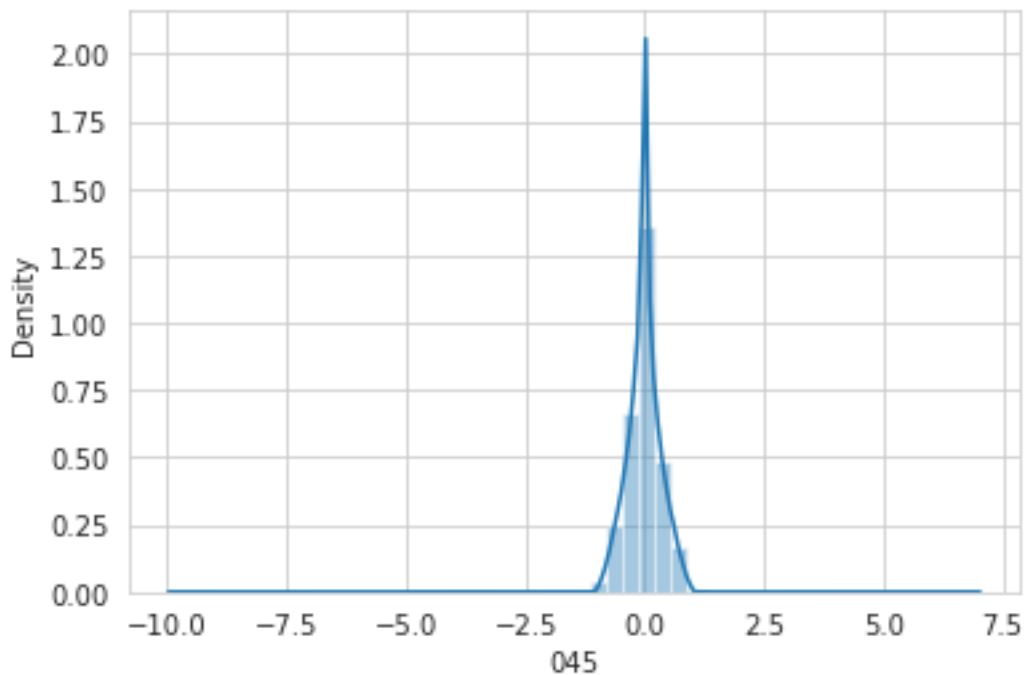
```
[384]: alpha = 45
```

```
[385]: %%time
alphas[f'{alpha:03}'] = alpha045(c, v)
```

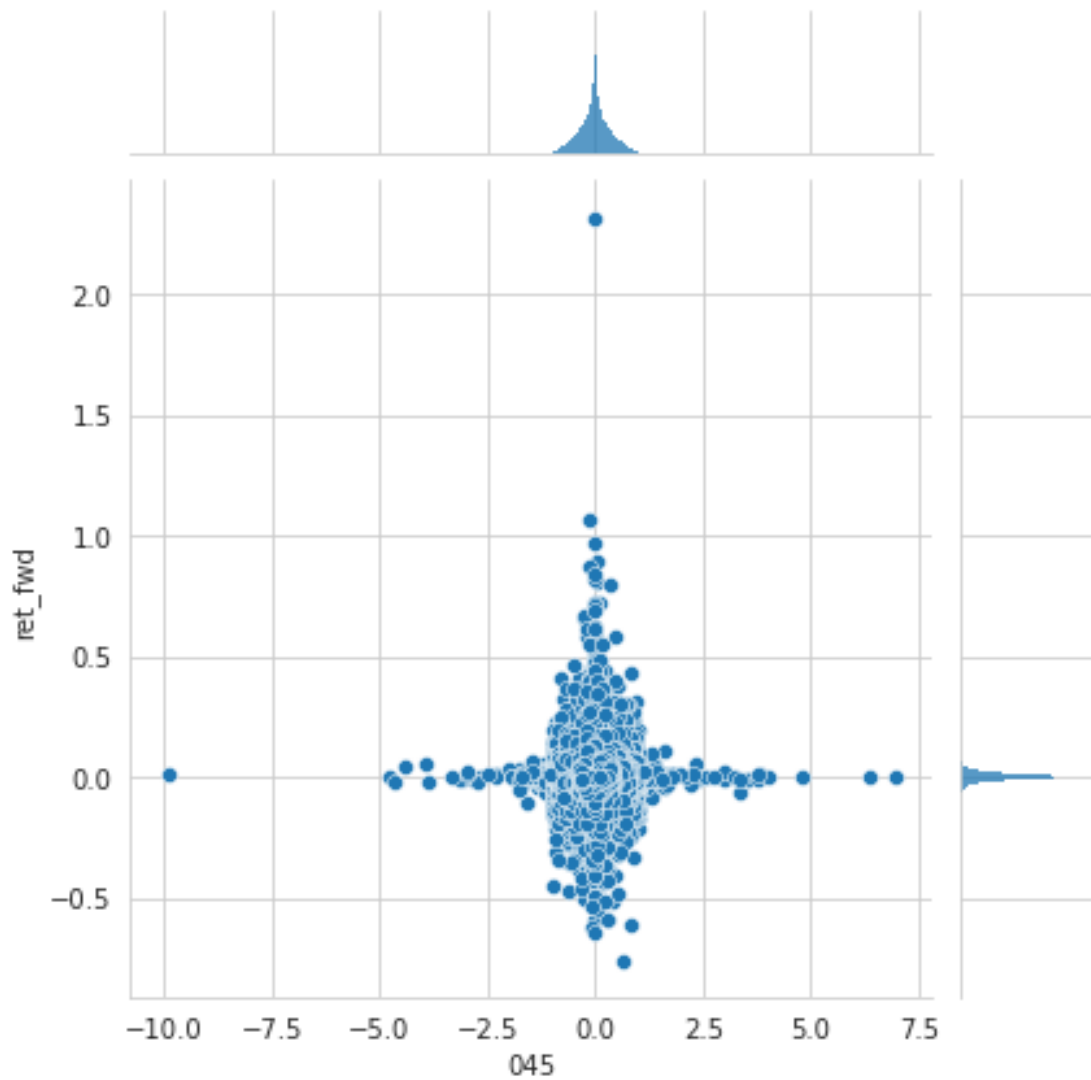
CPU times: user 5.26 s, sys: 76 ms, total: 5.33 s
Wall time: 5.18 s

```
[386]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[387]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[388]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[389]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[389]: 0.008224194668740914
```

1.50 Alpha 046

```
0.25 < ts_lag(ts_delta(close, 10), 10) / 10 - ts_delta(close, 10) / 10
? -1
: ((ts_lag(ts_delta(close, 10), 10) / 10 - ts_delta(close, 10) / 10 < 0)
? 1
: -ts_delta(close, 1))
```

```
[390]: def alpha046(c):
        """0.25 < ts_lag(ts_delta(close, 10), 10) / 10 - ts_delta(close, 10) / 10
           ? -1
           : ((ts_lag(ts_delta(close, 10), 10) / 10 - ts_delta(close, 10) / 10)
           ↪ < 0)
           ? 1
           : -ts_delta(close, 1))
        """
        cond = ts_lag(ts_delta(c, 10), 10).div(10).sub(ts_delta(c, 10).div(10))
        alpha = pd.DataFrame(-np.ones_like(cond),
                              index=c.index,
                              columns=c.columns)
        alpha[cond.isnull()] = np.nan
        return (cond.where(cond > 0.25,
                           -alpha.where(cond < 0,
                                         -ts_delta(c, 1)))
                .stack('ticker')
                .swaplevel())
```

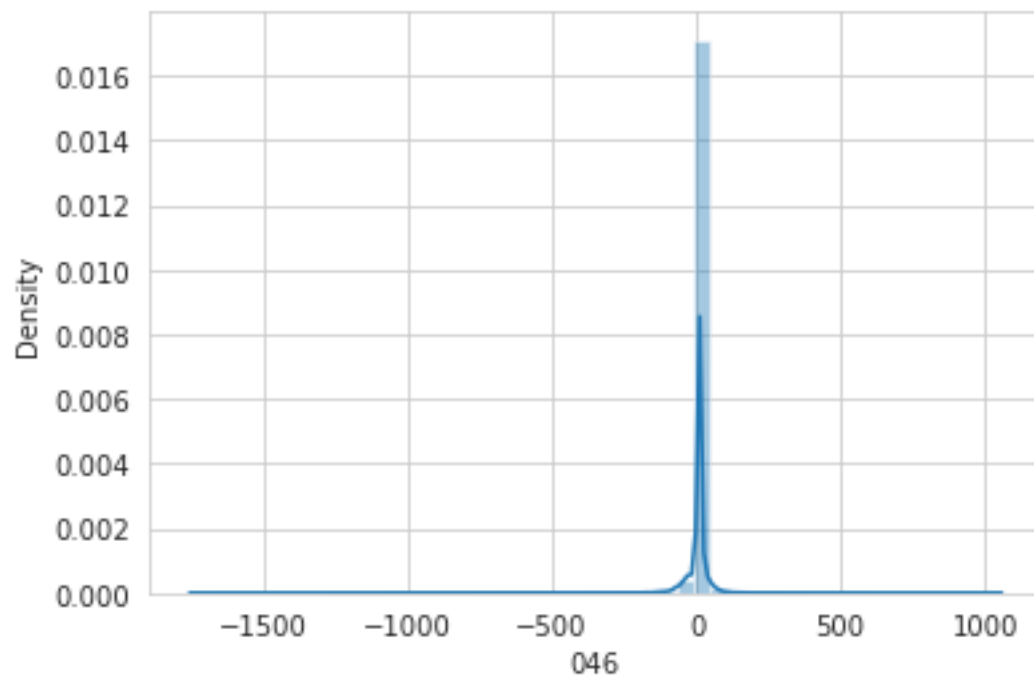
```
[391]: alpha = 46
```

```
[392]: %%time
        alphas[f'{alpha:03}'] = alpha046(c)
```

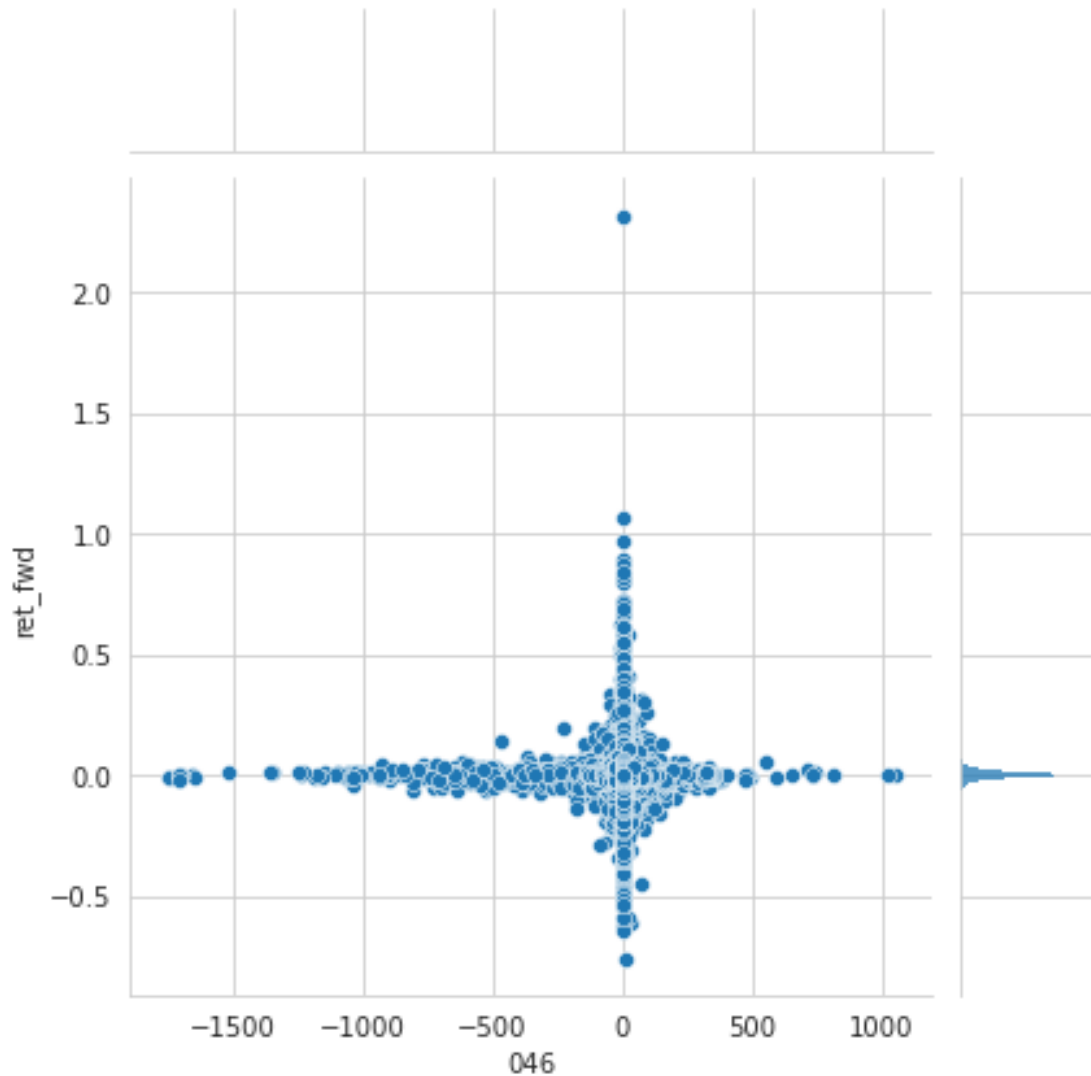
```
CPU times: user 2.37 s, sys: 12 ms, total: 2.38 s
Wall time: 2.34 s
```

```
[393]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[394]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[395]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[396]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[396]: 0.00870732283747122
```

1.51 Alpha 047

```
rank(ts_corr(ts_lag(open - close, 1), close, 200)) +
rank(open - close)
```

```
[397]: def alpha047(h, c, v, vwap, adv20):
        """((((rank((1 / close)) * volume) / adv20) * ((high * rank((high -
        ↪close))) /
        (ts_sum(high, 5) / 5))) - rank((vwap - ts_lag(vwap, 5))))"""
```

```

return (rank(c.pow(-1)).mul(v).div(adv20)
        .mul(h.mul(rank(h.sub(c))
                    .div(ts_mean(h, 5)))
        .sub(rank(ts_delta(vwap, 5))))
        .stack('ticker')
        .swaplevel())

```

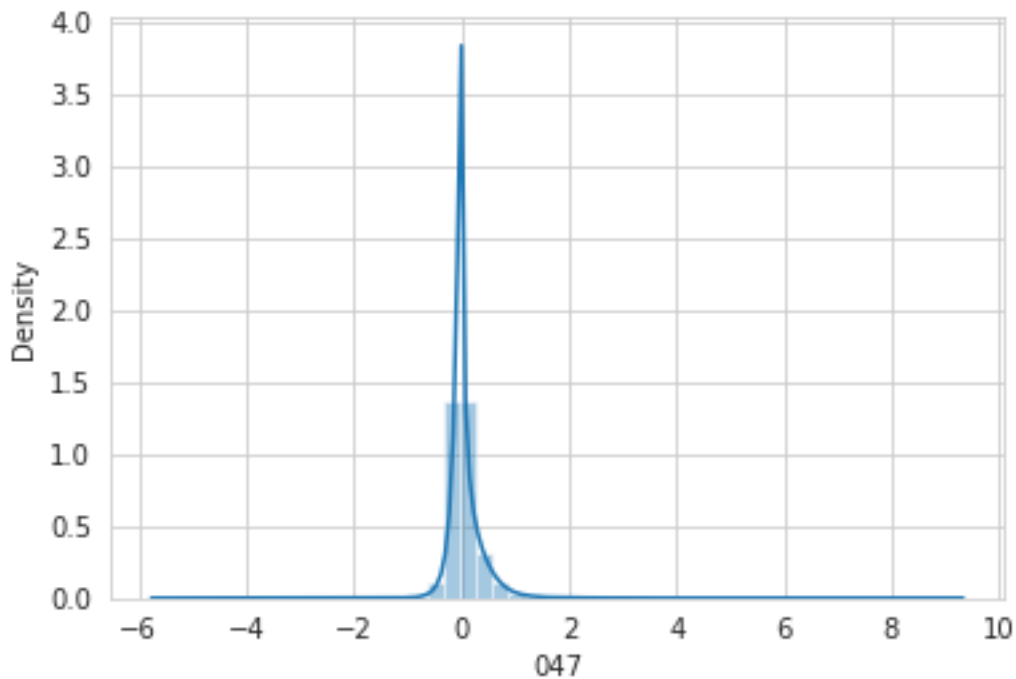
```
[398]: alpha = 47
```

```
[399]: %%time
alphas[f'{alpha:03}'] = alpha047(h, c, v, vwap, adv20)
```

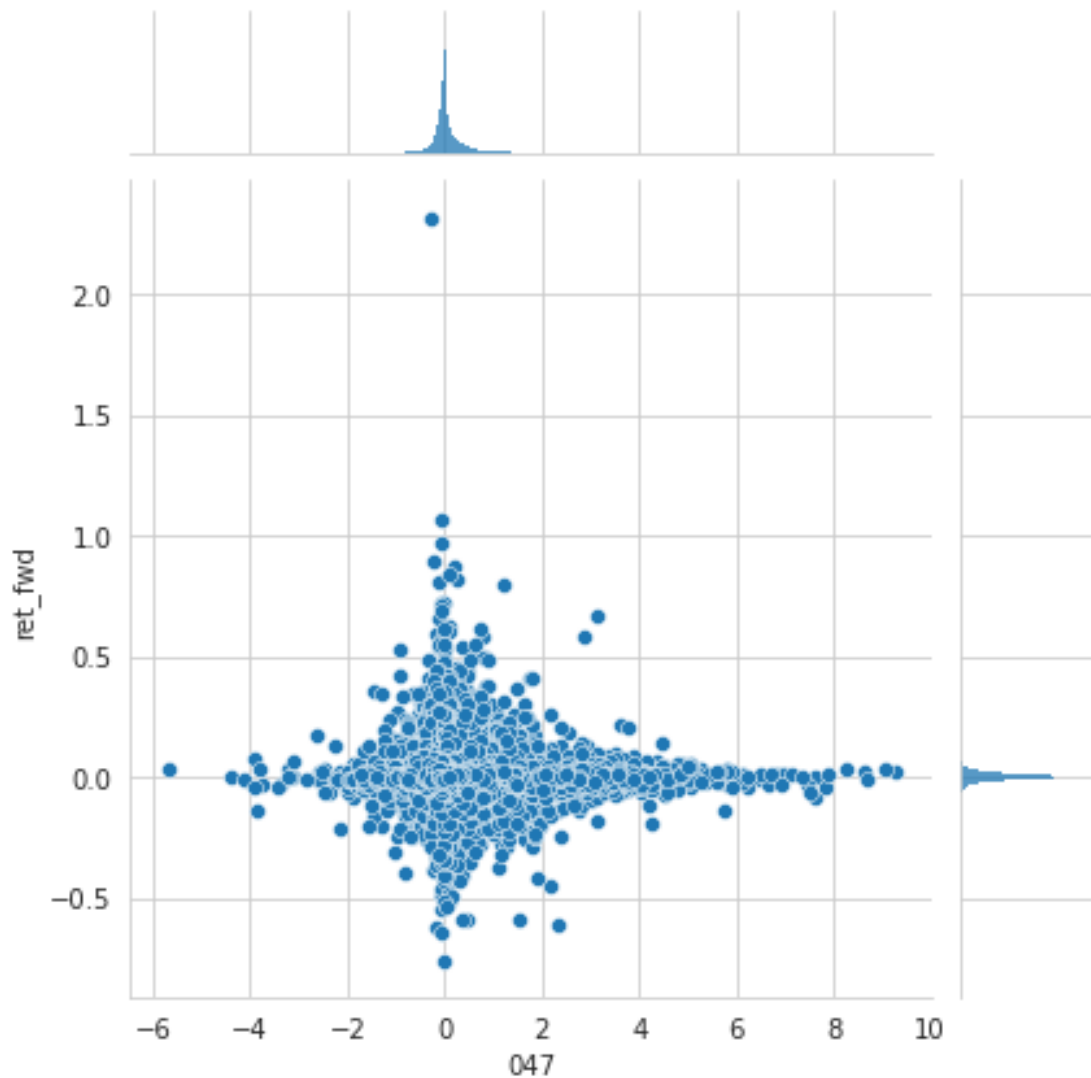
CPU times: user 2.87 s, sys: 64.1 ms, total: 2.93 s
Wall time: 2.81 s

```
[400]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[401]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[402]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```

```
[403]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[403]: 0.0009995443357944112
```

1.52 Alpha 048

```
(indneutralize(((ts_corr(ts_delta(close, 1), ts_delta(ts_lag(close, 1), 1), 250) *ts_delta(close,
```

```
[404]: def alpha48(c, industry):
    """(indneutralize(((ts_corr(ts_delta(close, 1), ts_delta(ts_lag(close, 1), 1), 250) *
    ts_delta(close, 1)) / close), IndClass.subindustry) /
    ts_sum(((ts_delta(close, 1) / ts_lag(close, 1))2), 250))"""
```

```
pass
```

```
[405]: alpha = 48
```

```
[406]: # %%time  
# alphas[f'{alpha:03}'] = alpha48(o, c)
```

```
[407]: # alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[408]: # sns.distplot(alphas[f'{alpha:03}']);
```

```
[409]: # g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);  
#
```

```
[410]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])  
# mi[alpha]
```

1.53 Alpha 049

```
ts_delta(ts_lag(close, 10), 10).div(10).sub(ts_delta(close, 10).div(10)) < -0.1 * c  
    ? 1  
    : -ts_delta(close, 1)
```

```
[411]: def alpha049(c):  
        """ts_delta(ts_lag(close, 10), 10).div(10).sub(ts_delta(close, 10).div(10))  
        ↪ < -0.1 * c  
        ? 1  
        : -ts_delta(close, 1)"""  
        cond = (ts_delta(ts_lag(c, 10), 10).div(10)  
                .sub(ts_delta(c, 10).div(10)) >= -0.1 * c)  
        return (-ts_delta(c, 1)  
                .where(cond, 1)  
                .stack('ticker')  
                .swaplevel())
```

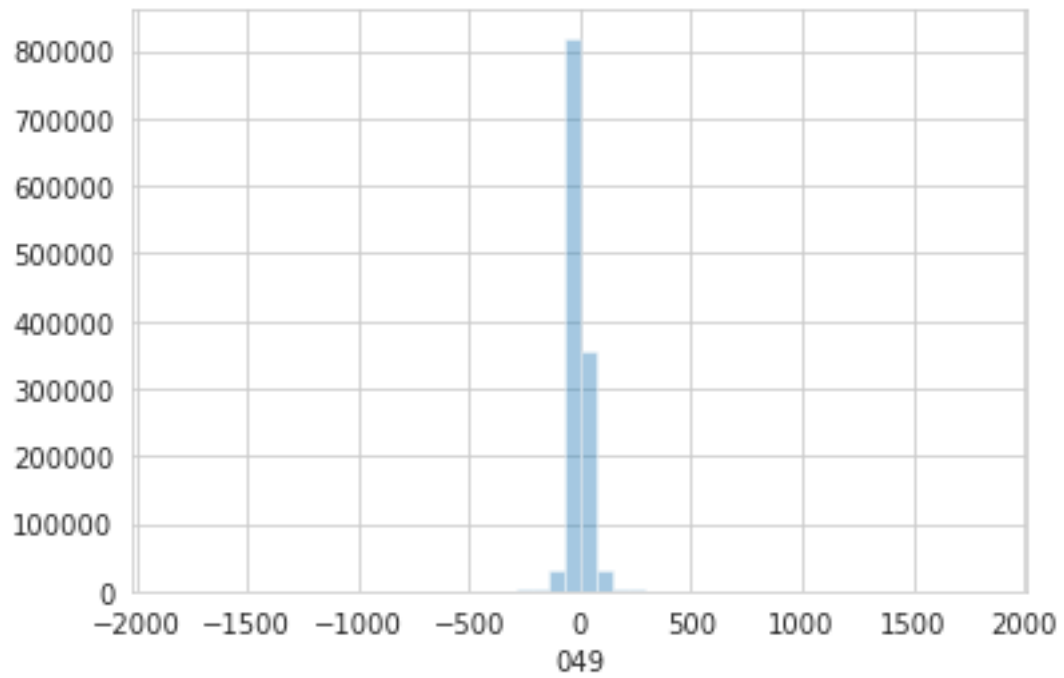
```
[412]: alpha = 49
```

```
[413]: %%time  
alphas[f'{alpha:03}'] = alpha049(c)
```

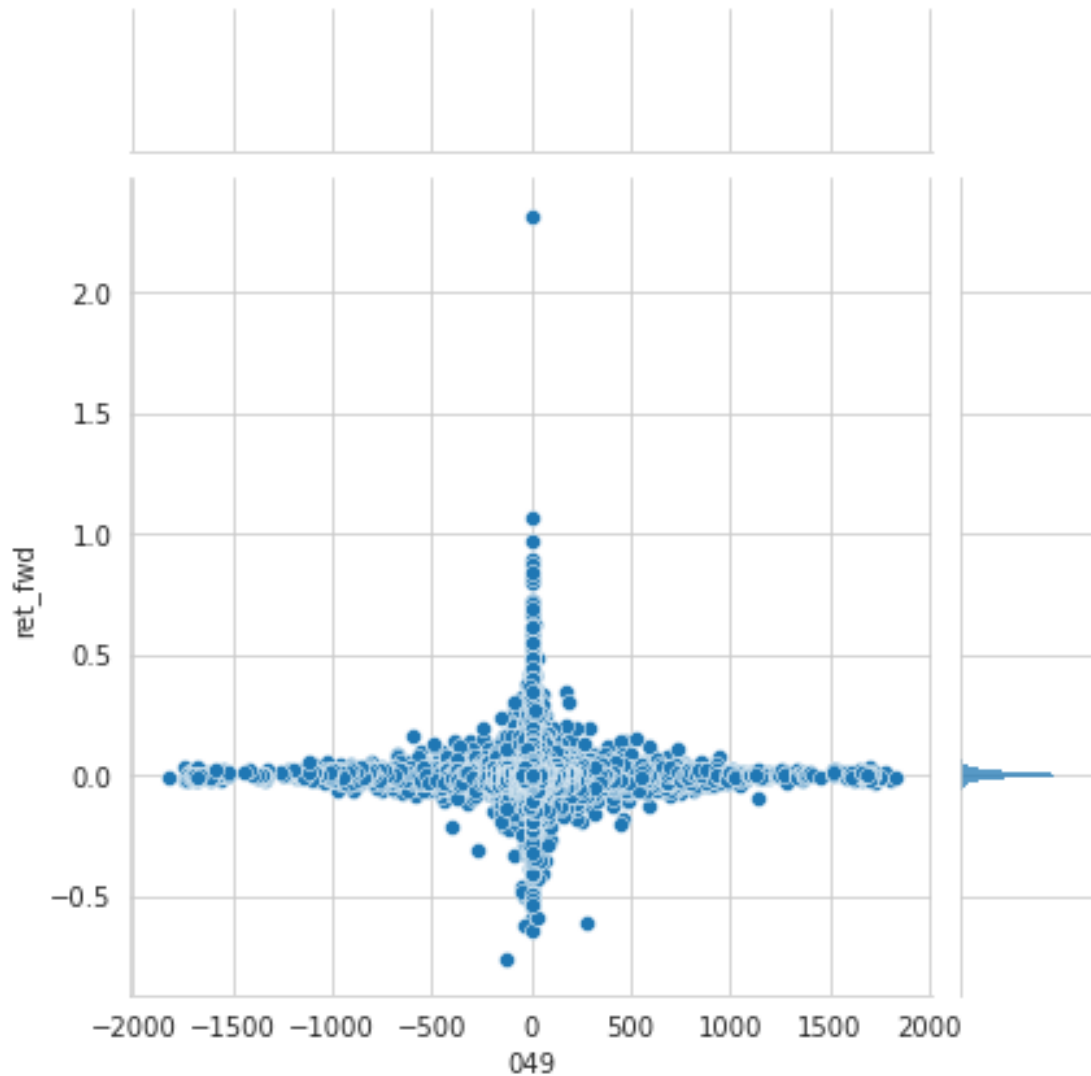
```
CPU times: user 3.13 s, sys: 32 ms, total: 3.16 s  
Wall time: 3.12 s
```

```
[414]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[415]: sns.distplot(alphas[f'{alpha:03}'], kde=False);
```



```
[416]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



1.54 Alpha 050

```
-ts_max(rank(ts_corr(rank(volume), rank(vwap), 5)), 5)
```

```
[417]: def alpha050(v, vwap):
        """-ts_max(rank(ts_corr(rank(volume), rank(vwap), 5)), 5)"""
        return (ts_max(rank(ts_corr(rank(v),
                                     rank(vwap), 5)), 5)
                .mul(-1)
                .stack('ticker')
                .swaplevel())
```

```
[418]: alpha = 50
```

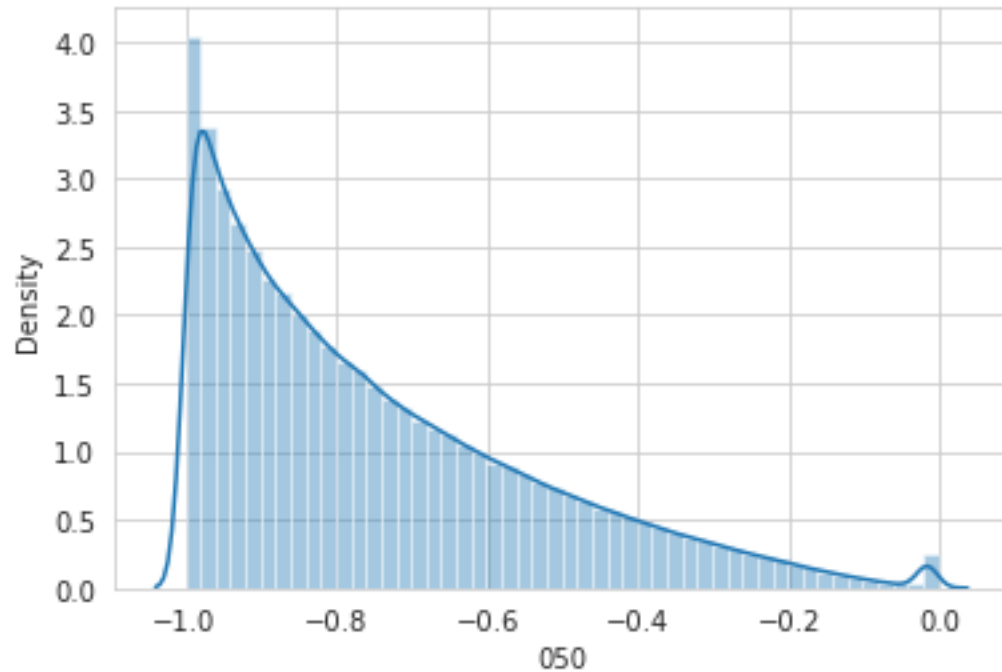
```
[419]: %%time
alphas[f'{alpha:03}'] = alpha050(v, vwap)
```

CPU times: user 2.98 s, sys: 88 ms, total: 3.07 s

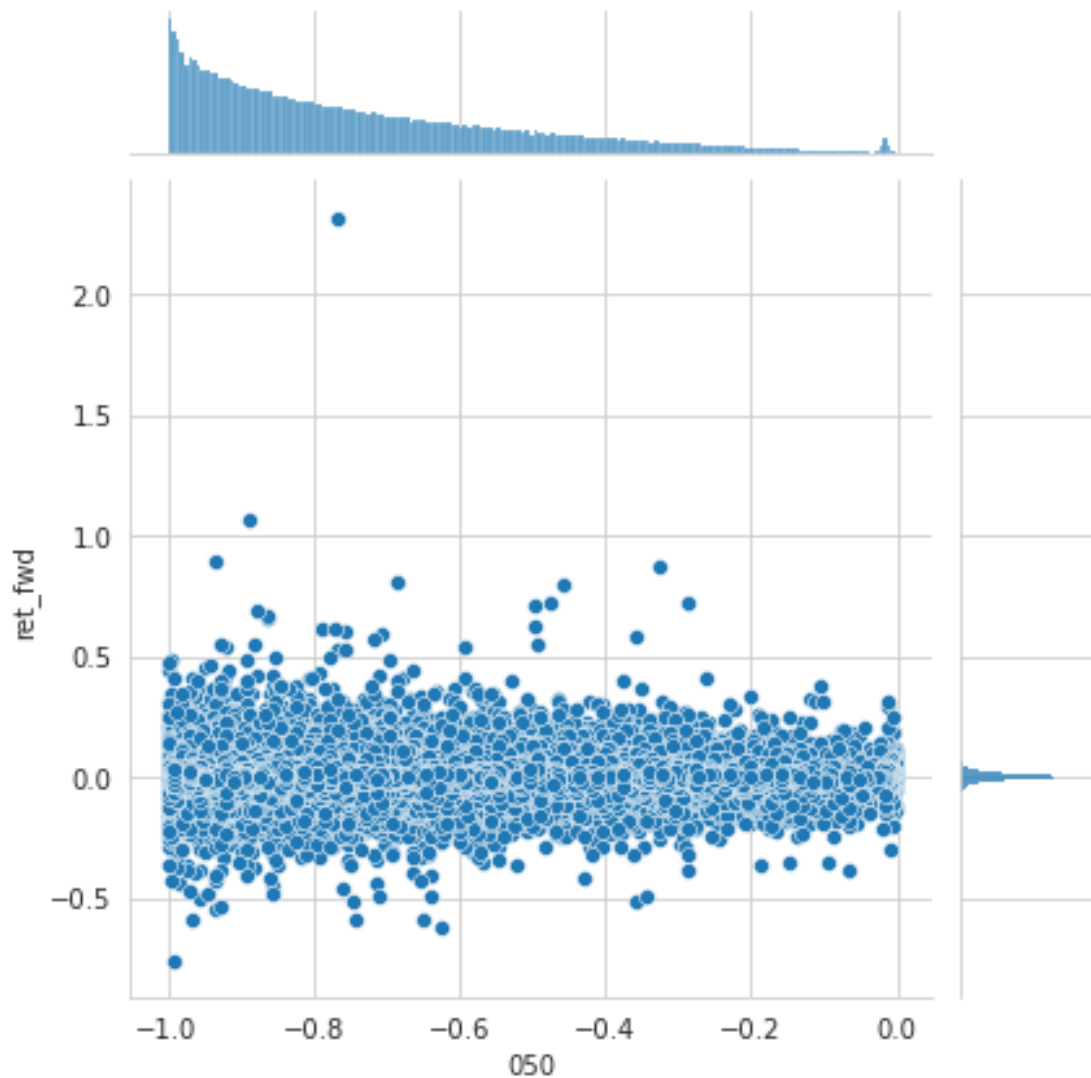
Wall time: 3.01 s

```
[420]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[421]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[422]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[423]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

[423]: 0

1.55 Alpha 051

```
ts_delta(ts_lag(close, 10), 10).div(10).sub(ts_delta(close, 10).div(10)) < -0.05 * c
? 1
: -ts_delta(close, 1)
```

```
[424]: def alpha051(c):
        """ts_delta(ts_lag(close, 10), 10).div(10).sub(ts_delta(close, 10).div(10)) <
        ↪ -0.05 * c
```

```

? 1
: -ts_delta(close, 1)"""
cond = (ts_delta(ts_lag(c, 10), 10).div(10)
        .sub(ts_delta(c, 10).div(10)) >= -0.05 * c)
return (-ts_delta(c, 1)
        .where(cond, 1)
        .stack('ticker')
        .swaplevel())

```

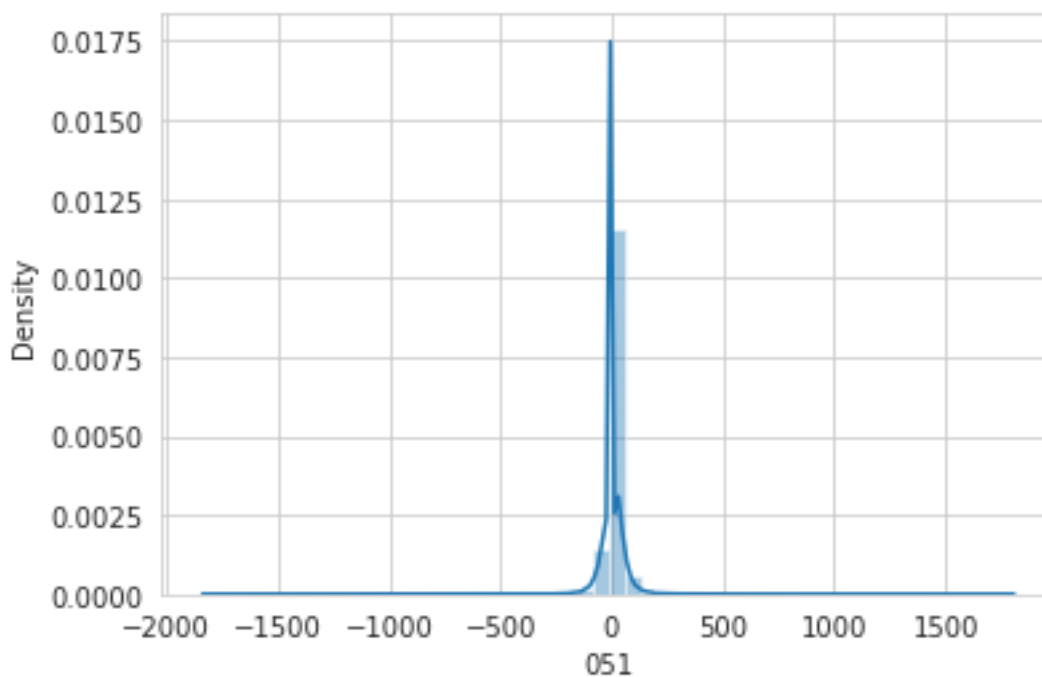
```
[425]: alpha = 51
```

```
[426]: %%time
alphas[f'{alpha:03}'] = alpha051(c)
```

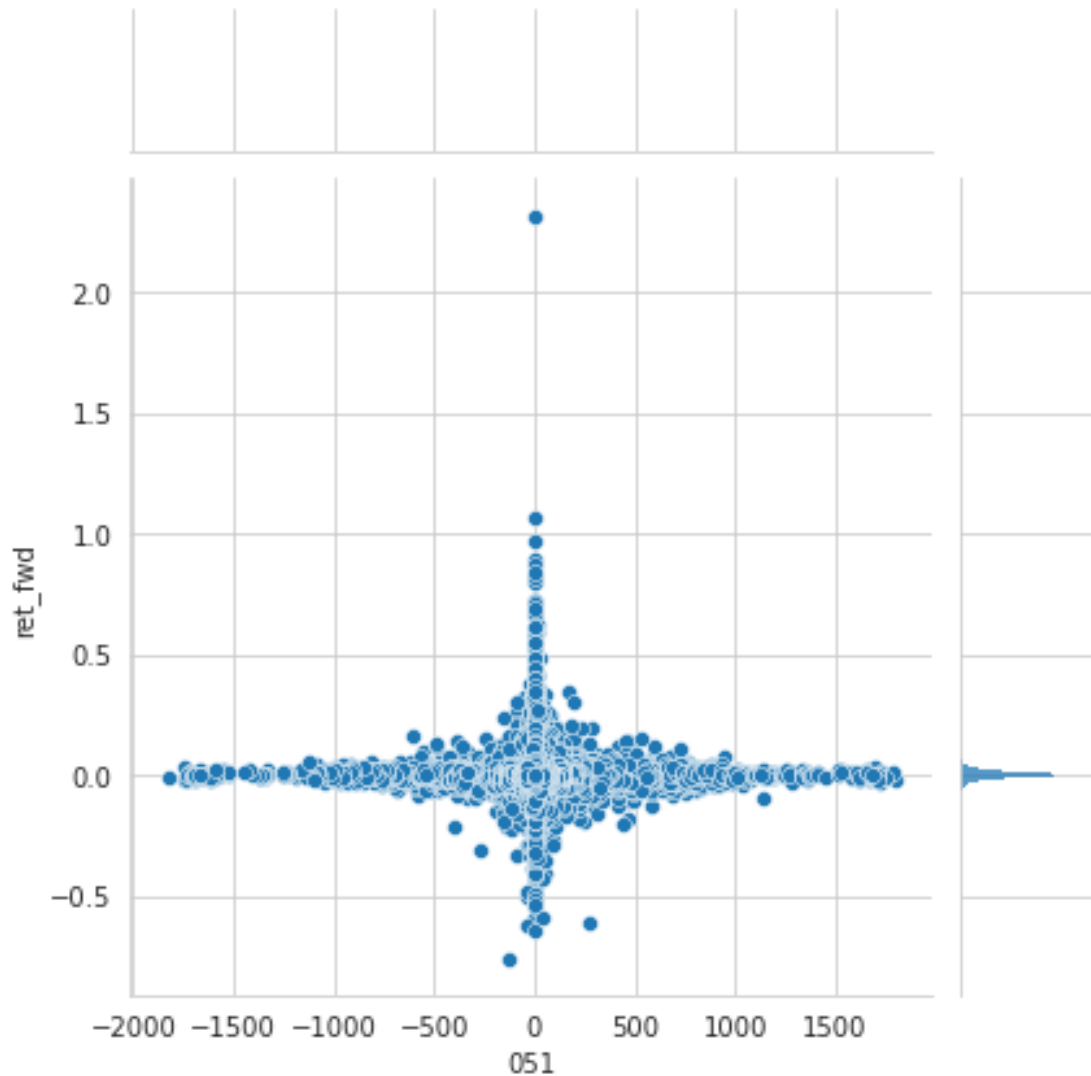
CPU times: user 5.92 s, sys: 128 ms, total: 6.05 s
Wall time: 6 s

```
[427]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[428]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[429]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[430]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

[430]: 0.010624389695808034

1.56 Alpha 052

```
(ts_lag(ts_min(low, 5), 5) - ts_min(low, 5)) *
    rank((ts_sum(returns, 240) - ts_sum(returns, 20)) / 220) *
    ts_rank(volume, 5)
```

```
[431]: def alpha052(l, v, r):
        """(ts_lag(ts_min(low, 5), 5) - ts_min(low, 5)) *
            rank((ts_sum(returns, 240) - ts_sum(returns, 20)) / 220) *
```



```

        ts_rank(volume, 5)
    """
    return (ts_delta(ts_min(l, 5), 5)
            .mul(rank(ts_sum(r, 240)
                      .sub(ts_sum(r, 20))
                      .div(220)))
            .mul(ts_rank(v, 5))
            .stack('ticker')
            .swaplevel())

```

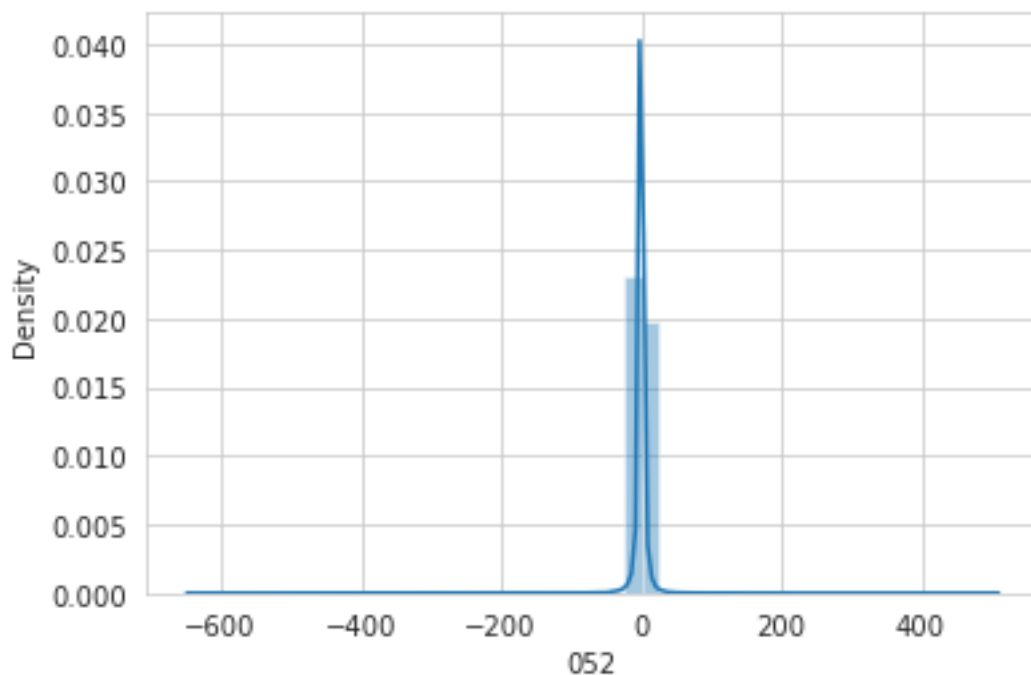
```
[432]: alpha = 52
```

```
[433]: %%time
alphas[f'{alpha:03}'] = alpha052(l, v, r)
```

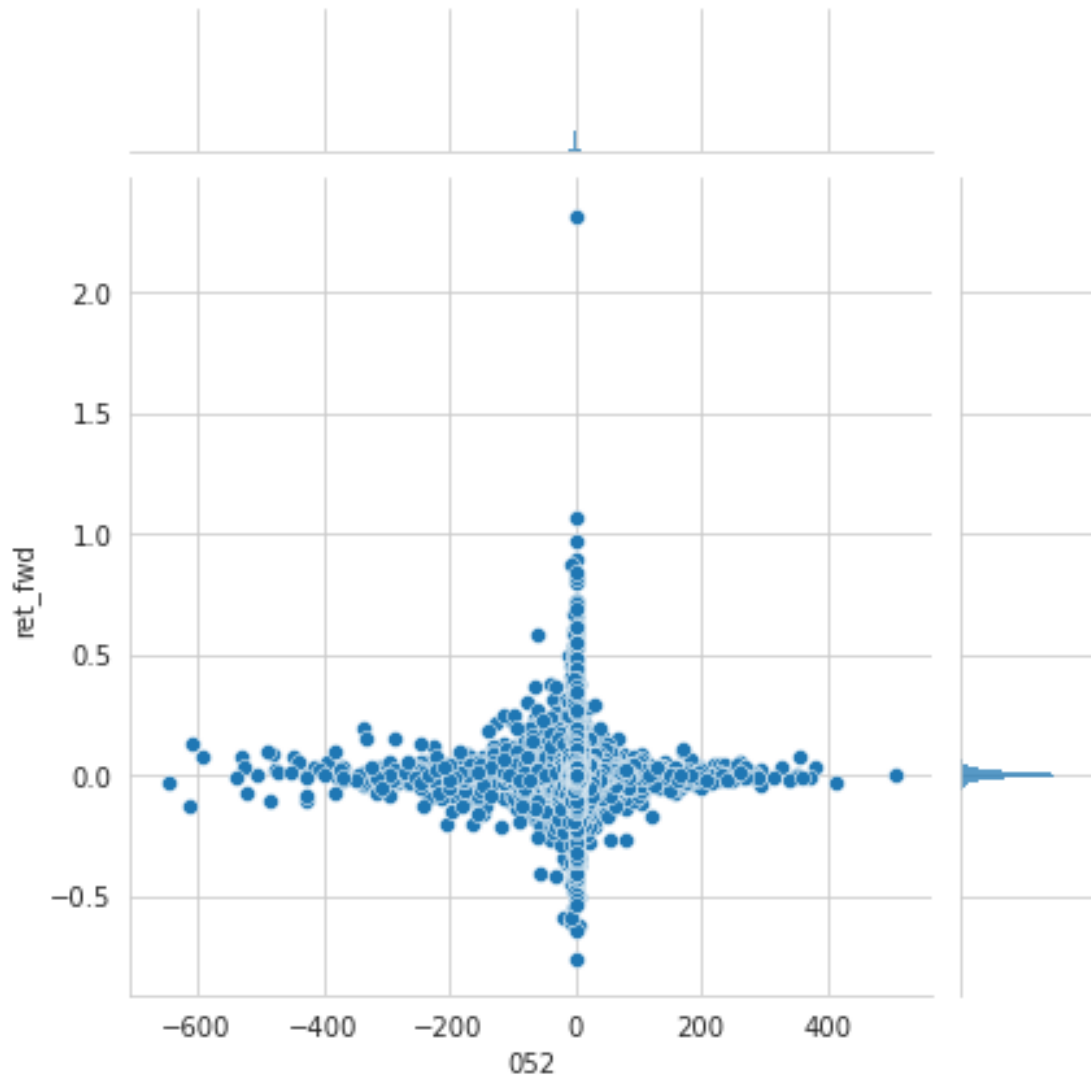
CPU times: user 3min 4s, sys: 52 ms, total: 3min 4s
Wall time: 3min 4s

```
[434]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[435]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[436]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[437]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[437]: 0.007025458925477679
```

1.57 Alpha 053

```
((ts_sum(high, 20) / 20) < high)
? (-1 * ts_delta(high, 2))
: 0
```

```
[438]: def alpha053(h, l, c):
        """-1 * ts_delta(1 - (high - close) / (close - low), 9)"""
        inner = (c.sub(1)).add(1e-6)
```

```

return (ts_delta(h.sub(c)
                .mul(-1).add(1)
                .div(c.sub(1)
                    .add(1e-6)), 9)
        .mul(-1)
        .stack('ticker')
        .swaplevel())

```

```
[439]: alpha = 53
```

```

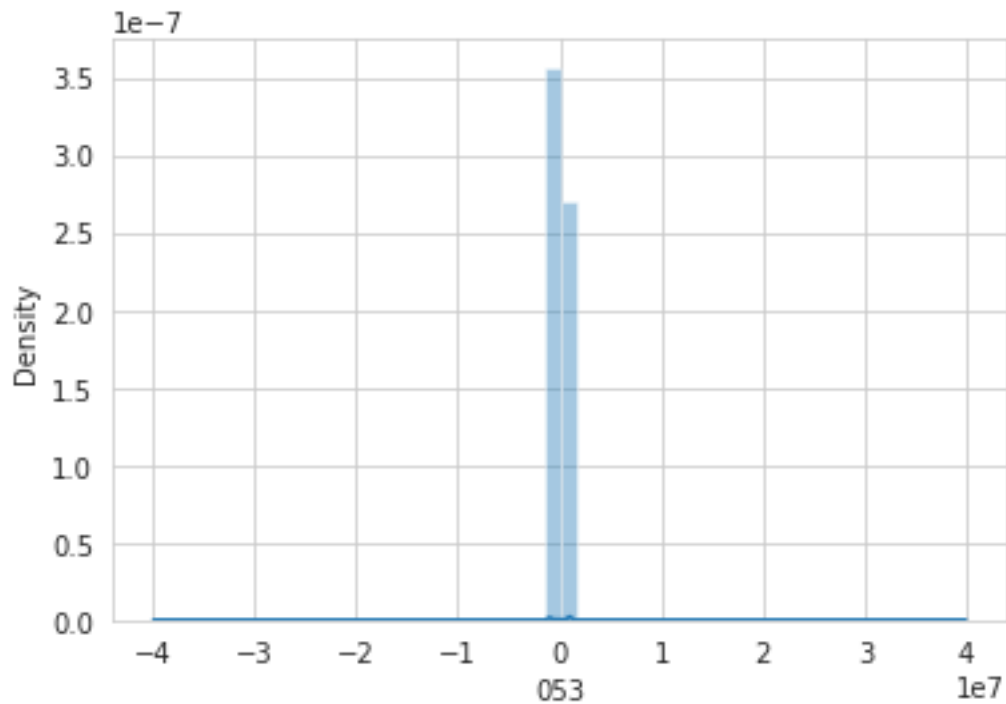
[440]: %%time
alphas[f'{alpha:03}'] = alpha053(h, 1, c)

```

CPU times: user 1.51 s, sys: 0 ns, total: 1.51 s
Wall time: 1.45 s

```
[441]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[442]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[ ]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```

```

[ ]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
     # mi[alpha]

```

1.58 Alpha 054

```
-(low - close) * power(open, 5) / ((low - high) * power(close, 5))
```

```
[35]: def alpha054(o, h, l, c):  
      """-(low - close) * power(open, 5) / ((low - high) * power(close, 5))"""  
      return (l.sub(c).mul(o.pow(5)).mul(-1)  
              .div(l.sub(h).replace(0, -0.0001).mul(c ** 5))  
              .stack('ticker')  
              .swaplevel())
```

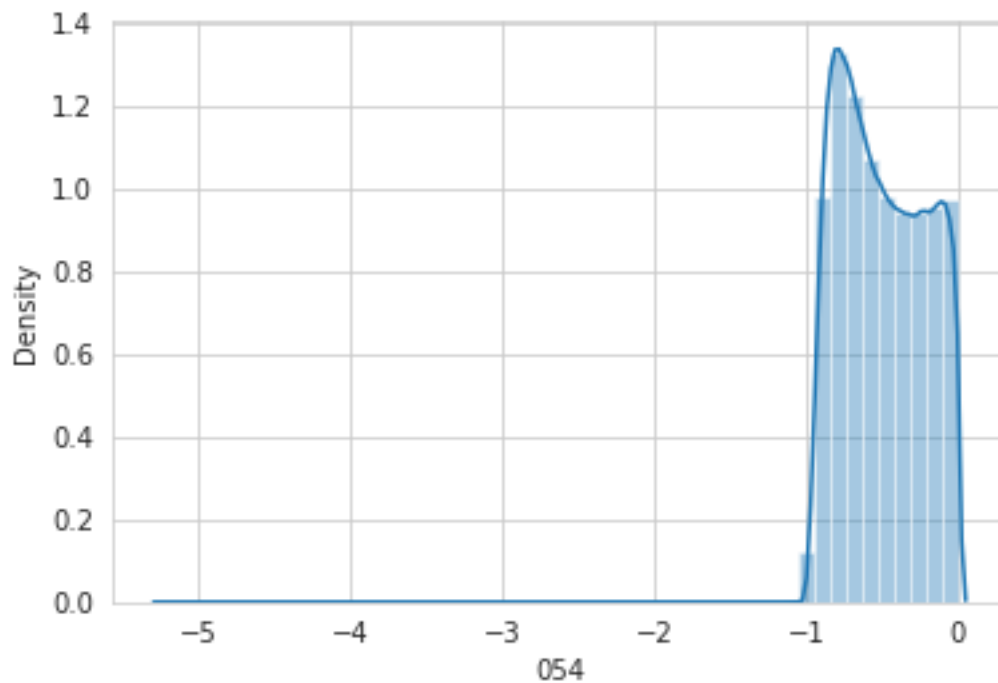
```
[36]: alpha = 54
```

```
[37]: %%time  
      alphas[f'{alpha:03}'] = alpha054(o, h, l, c)
```

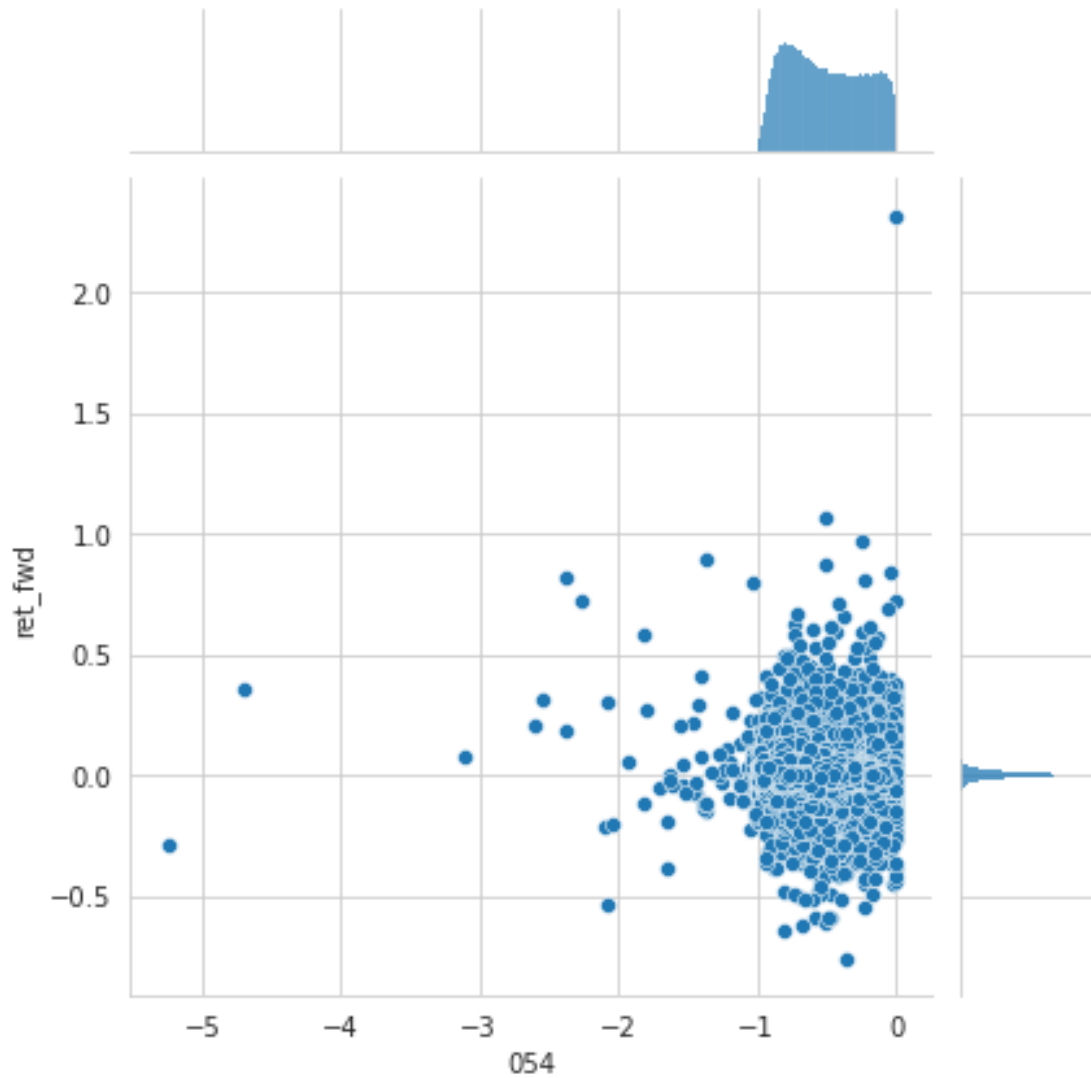
CPU times: user 1.99 s, sys: 42.8 ms, total: 2.03 s
Wall time: 1.91 s

```
[38]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[39]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[40]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[41]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[41]: 0.0007175455210228776
```

```
[42]: pd.Series(mi).tail()
```

```
[42]: 54    0.000718
dtype: float64
```

1.59 Alpha 055

```
(-1 * ts_corr(rank(((close - ts_min(low, 12)) /
                    (ts_max(high, 12) - ts_min(low, 12)))),
```

```
rank(volume), 6))
```

```
[43]: def alpha055(h, l, c):  
      """(-1 * ts_corr(rank(((close - ts_min(low, 12)) /  
                          (ts_max(high, 12) - ts_min(low, 12))))),  
          rank(volume), 6))"""  
  
      return (ts_corr(rank(c.sub(ts_min(l, 12))  
                              .div(ts_max(h, 12).sub(ts_min(l, 12))  
                              .replace(0, 1e-6))),  
              rank(v), 6)  
              .replace([-np.inf, np.inf], np.nan)  
              .mul(-1)  
              .stack('ticker')  
              .swaplevel())
```

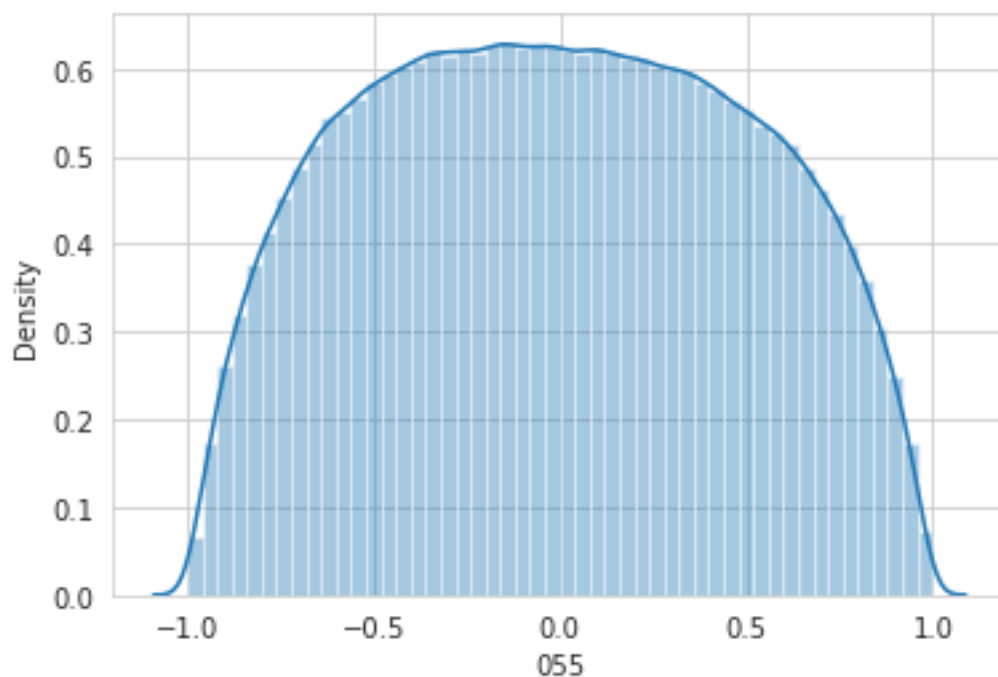
```
[44]: alpha = 55
```

```
[45]: %%time  
      alphas[f'{alpha:03}'] = alpha055(h, l, c)
```

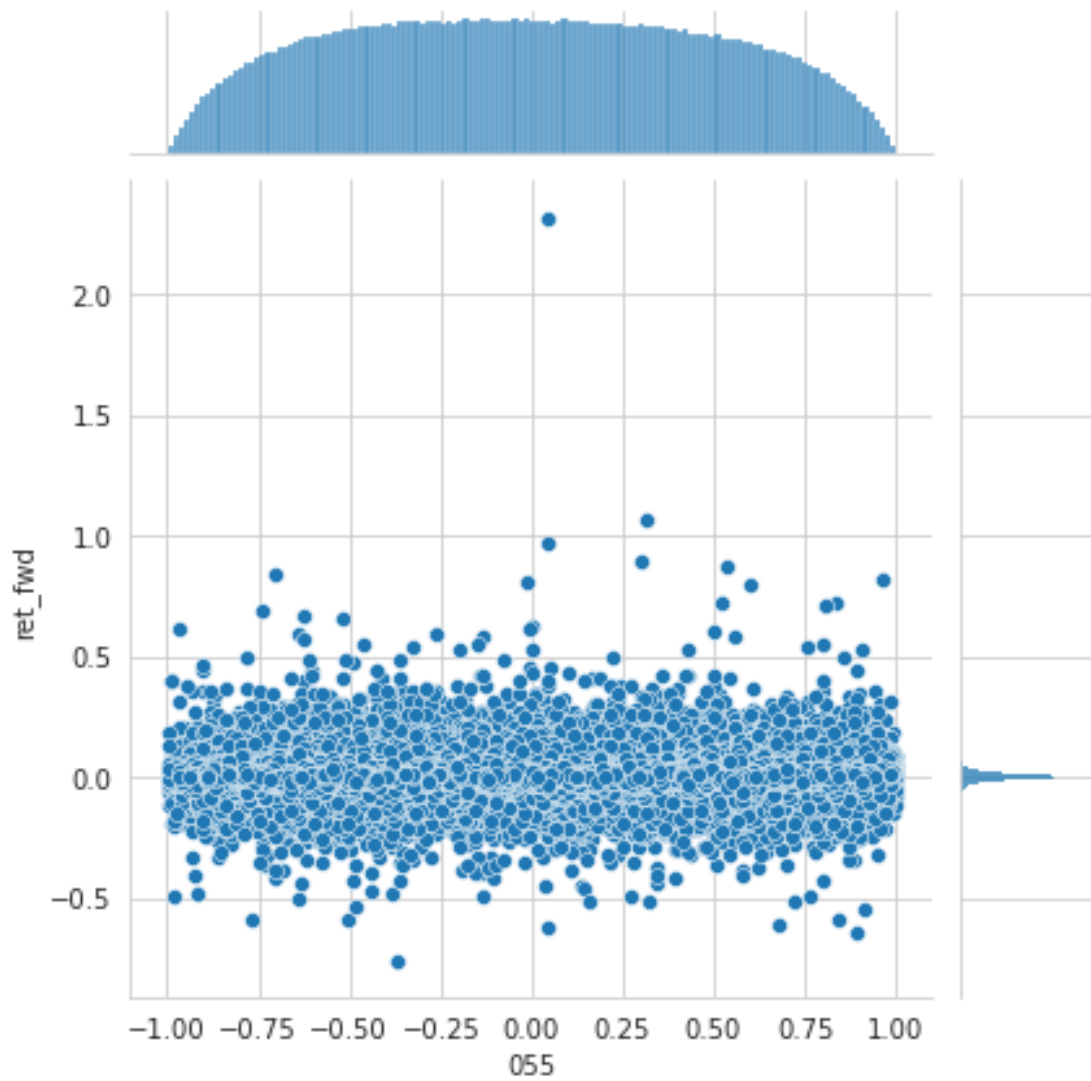
```
CPU times: user 3.83 s, sys: 65.4 ms, total: 3.89 s  
Wall time: 3.81 s
```

```
[46]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[47]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[48]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[49]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[49]: 0
```

1.60 Alpha 056

```
-rank(ts_sum(returns, 10) / ts_sum(ts_sum(returns, 2), 3)) *
rank((returns * cap))
```

```
[50]: def alpha056(r, cap):
        """-rank(ts_sum(returns, 10) / ts_sum(ts_sum(returns, 2), 3)) *
            rank((returns * cap))
        """
        pass
```

1.61 Alpha 057

```
rank(ts_corr(ts_lag(open - close, 1), close, 200)) +
    rank(open - close)
```

```
[51]: def alpha057(c, vwap):
        """-(close - vwap) / ts_weighted_mean(rank(ts_argmax(close, 30)), 2)"""
        return (c.sub(vwap.add(1e-5))
                .div(ts_weighted_mean(rank(ts_argmax(c, 30))))
                .mul(-1)
                .stack('ticker')
                .swaplevel())
```

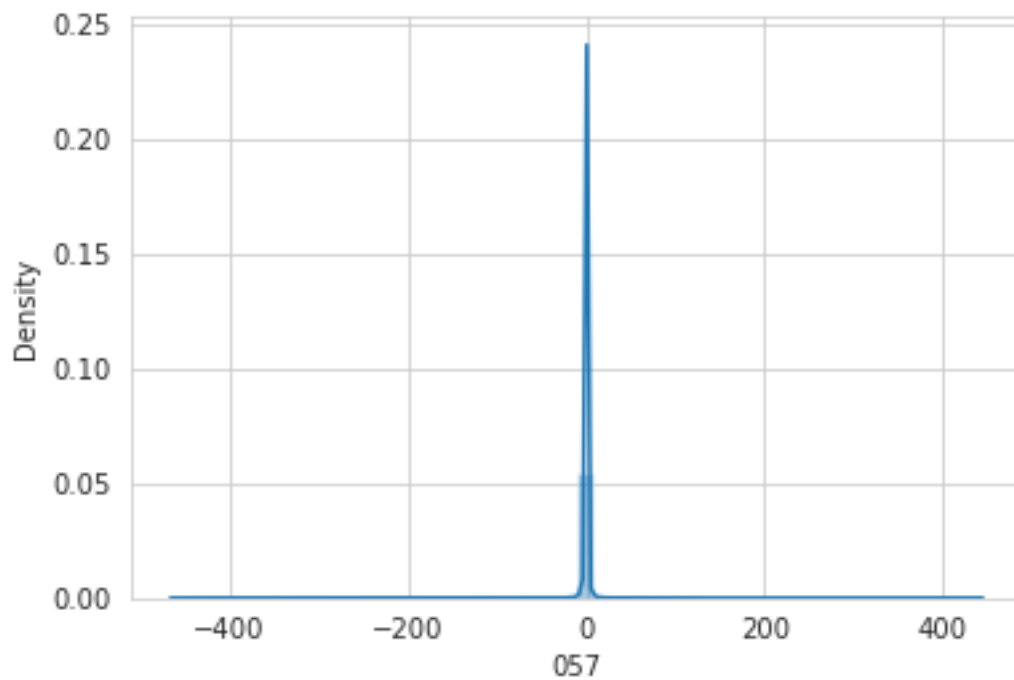
```
[52]: alpha = 57
```

```
[53]: %%time
        alphas[f'{alpha:03}'] = alpha057(c, vwap)
```

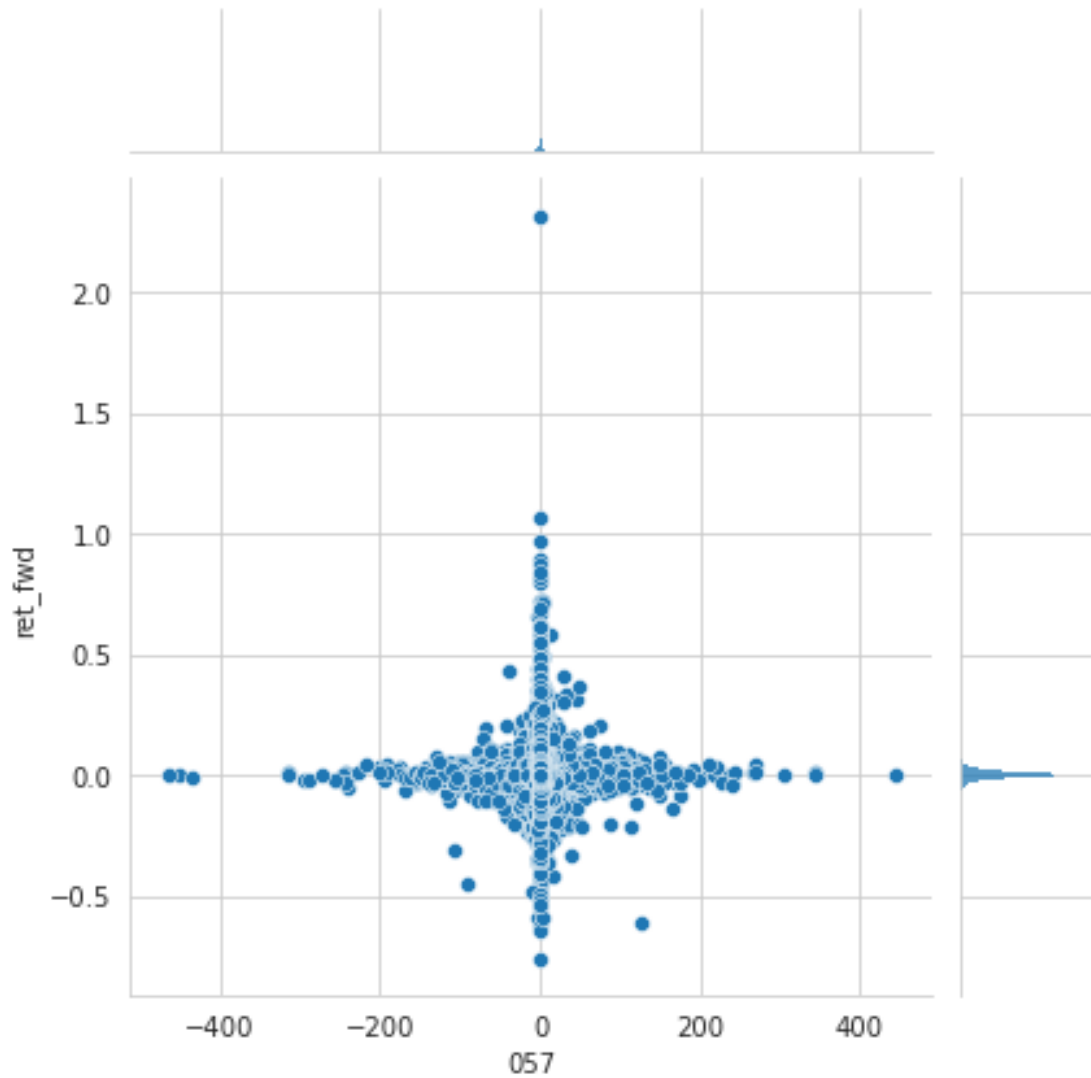
```
CPU times: user 1min 54s, sys: 280 ms, total: 1min 54s
Wall time: 1min 53s
```

```
[54]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[55]: sns.distplot(alphas[f'{alpha:03}']);
```

```
[56]: g = sns.jointplot(x=f'alpha:03', y='ret_fwd', data=alphas);
```



```
[57]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
      mi[alpha]
```

```
[57]: 0.006408315924969266
```

1.62 Alpha 058

```
(indneutralize(((ts_corr(ts_delta(close, 1), ts_delta(ts_lag(close, 1), 1), 250) *ts_delta(close,
```

```
[58]: def alpha58(v, wvap, sector):
      """(-1 * ts_rank(ts_weighted_mean(ts_corr(IndNeutralize(vwap, IndClass.
      ↪sector), volume, 3), 7), 5))"""
      pass
```

1.63 Alpha 059

```
(indneutralize(((ts_corr(ts_delta(close, 1), ts_delta(ts_lag(close, 1), 1), 250) *ts_delta(clos
```

```
[59]: def alpha59(v, wvwap, industry):  
        """-ts_rank(ts_weighted_mean(ts_corr(IndNeutralize(vwap, IndClass.  
        ↪industry), volume, 4), 16), 8)"""  
        pass
```

1.64 Alpha 060

```
-ts_max(rank(ts_corr(rank(volume), rank(vwap), 5)), 5)
```

```
[60]: def alpha060(l, h, c, v):  
        """-((2 * scale(rank((((close - low) - (high - close)) / (high - low)) *  
        ↪volume)))) -scale(rank(ts_argmax(close, 10))))"""  
        return (scale(rank(c.mul(2).sub(1).sub(h)  
                        .div(h.sub(1).replace(0, 1e-5))  
                        .mul(v))).mul(2)  
                .sub(scale(rank(ts_argmax(c, 10)))).mul(-1)  
                .stack('ticker')  
                .swaplevel())
```

```
[61]: alpha = 60
```

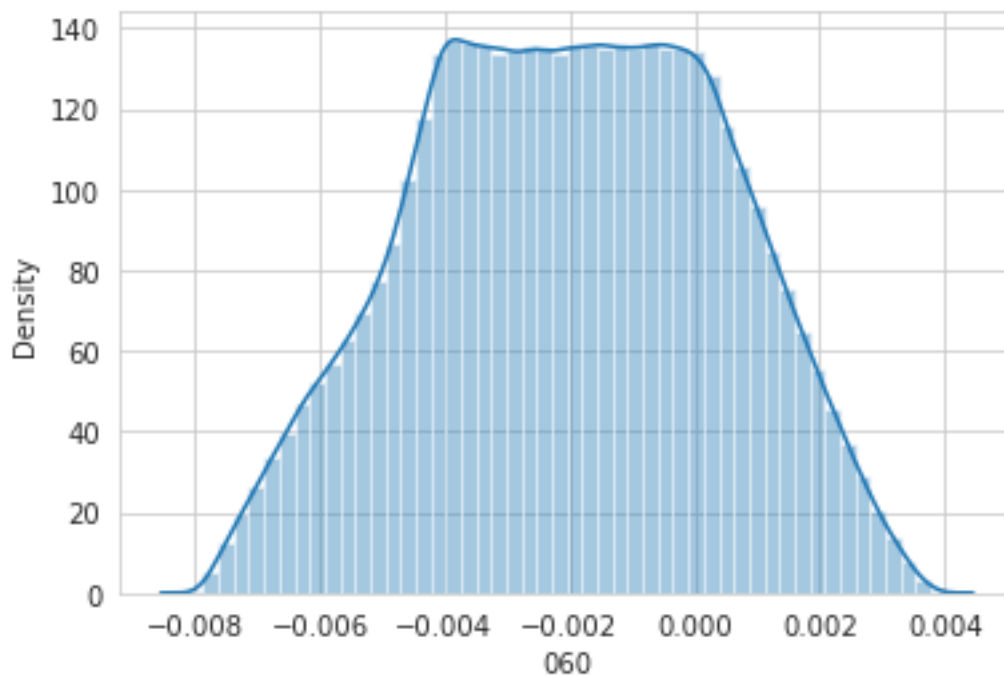
```
[62]: %%time  
      alphas[f'{alpha:03}'] = alpha060(l, h, c, v)
```

CPU times: user 1min 57s, sys: 372 ms, total: 1min 57s

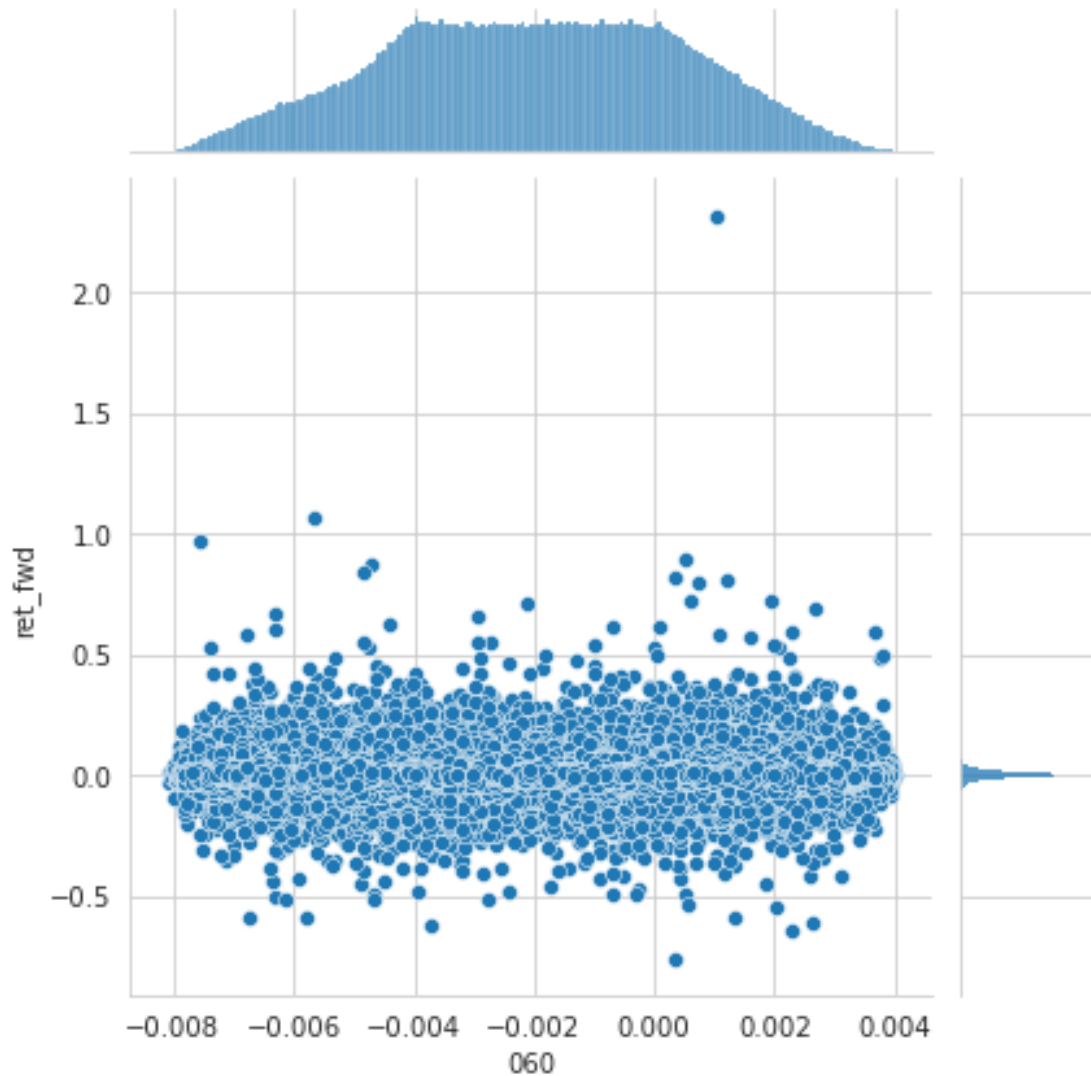
Wall time: 1min 56s

```
[63]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[64]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[65]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[66]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
      mi[alpha]
```

```
[66]: 0
```

1.65 Alpha 061

```
(rank((vwap - ts_min(vwap, 16.1219))) < rank(ts_corr(vwap, adv180, 17.9282)))
```

```
[67]: def alpha061(v, vwap):
      """rank((vwap - ts_min(vwap, 16))) < rank(ts_corr(vwap, adv180, 17))"""

      return (rank(vwap.sub(ts_min(vwap, 16)))
              .lt(rank(ts_corr(vwap, ts_mean(v, 180), 18)))
```

```
.astype(int)
.stack('ticker')
.swaplevel()
```

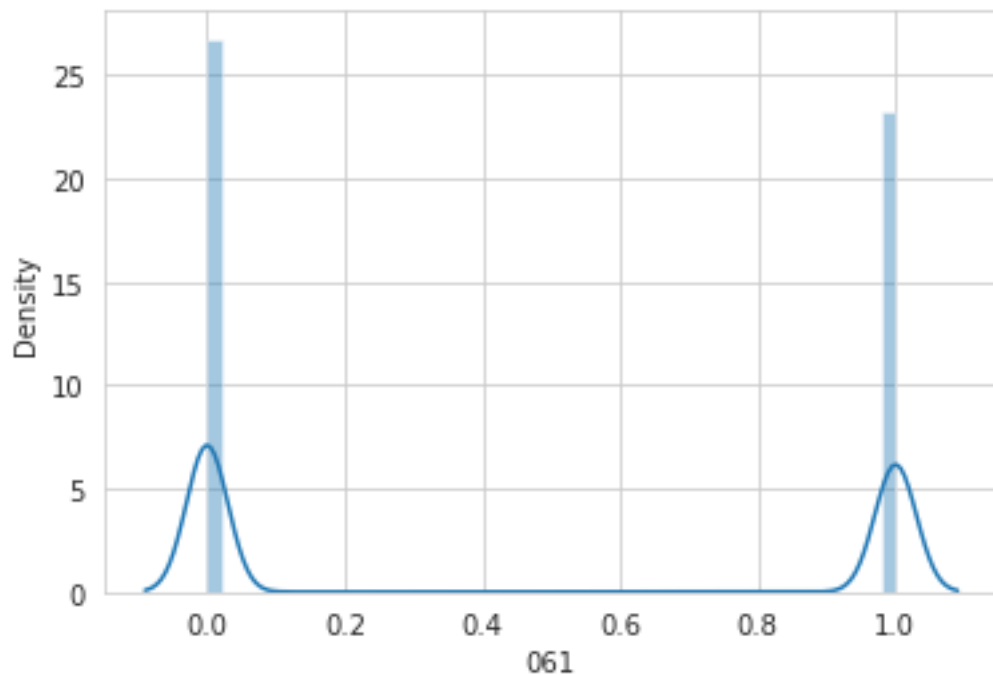
```
[68]: alpha = 61
```

```
[69]: %%time
alphas[f'{alpha:03}'] = alpha061(v, vwap)
```

CPU times: user 4.58 s, sys: 112 ms, total: 4.69 s
Wall time: 4.64 s

```
[70]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[71]: sns.distplot(alphas[f'{alpha:03}']);
```



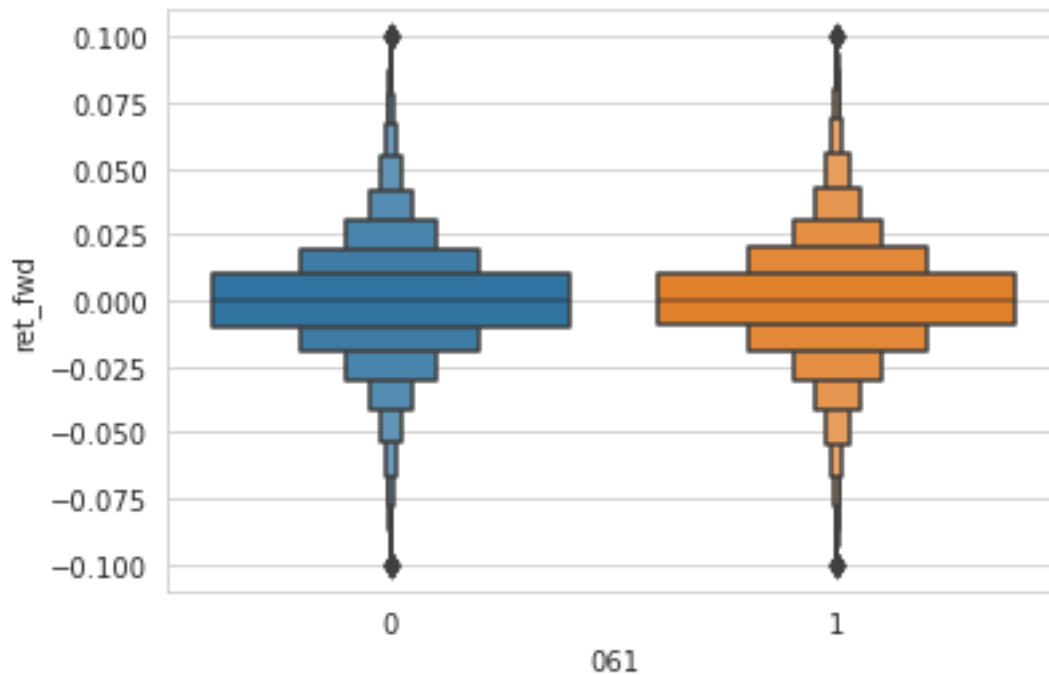
```
[72]: alphas.groupby(alphas[f'{alpha:03}']).ret_fwd.describe()
```

```
[72]:
```

	count	mean	std	min	25%	50%	75%	\
061								
0	671704.0	0.000458	0.025757	-0.757755	-0.009681	0.000413	0.010529	
1	583389.0	0.000724	0.025759	-0.643066	-0.009615	0.000599	0.010819	
		max						
061								

```
0    2.317073
1    0.972222
```

```
[73]: g = sns.boxenplot(x=f'{alpha:03}', y='ret_fwd', data=alphas[alphas.ret_fwd.
↪between(-.1, .1)]);
```



1.66 Alpha 062

```
((rank(ts_corr(vwap, ts_sum(adv20, 22.4101), 9.91009)) < rank(((rank(open) + rank(open)) < (rank
```

```
[74]: def alpha062(o, h, l, vwap, adv20):
    """((rank(ts_corr(vwap, ts_sum(adv20, 22.4101), 9.91009)) <
    rank(((rank(open) + rank(open)) < (rank((high + low) / 2)) +
    ↪rank(high)))) * -1)"""
    return (rank(ts_corr(vwap, ts_sum(adv20, 22), 9))
            .lt(rank(
                rank(o).mul(2)
                .lt(rank(h.add(1).div(2))
                    .add(rank(h))))
            .mul(-1)
            .stack('ticker')
            .swaplevel())
```

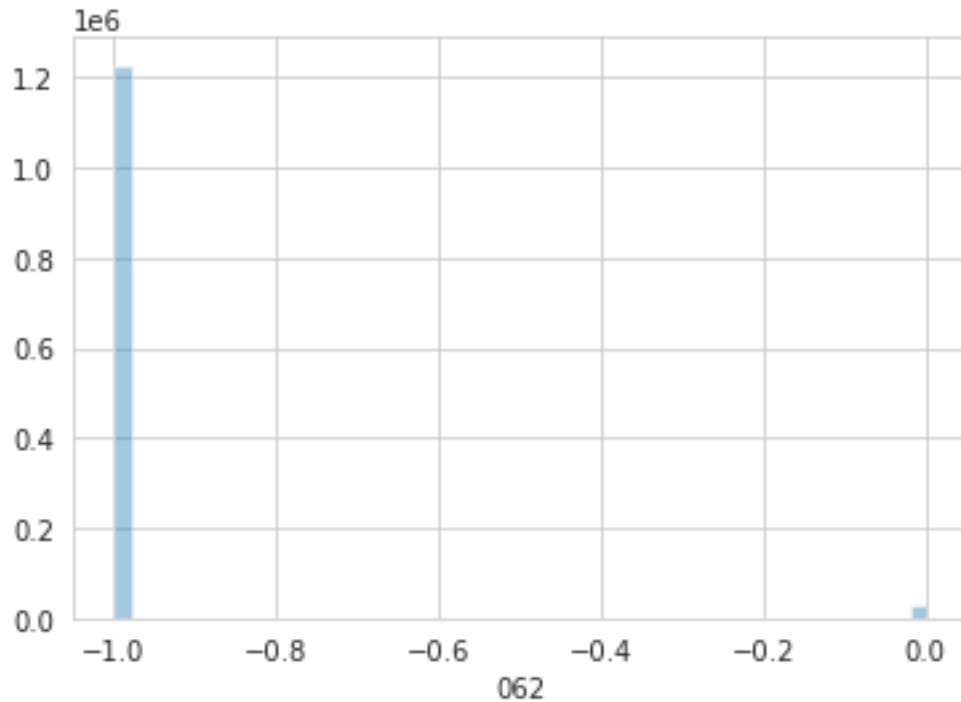
```
[75]: alpha = 62
```

```
[76]: %%time
alphas[f'{alpha:03}'] = alpha062(o, h, l, vwap, adv20)
```

CPU times: user 3.67 s, sys: 36.3 ms, total: 3.71 s
Wall time: 3.64 s

```
[77]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[78]: sns.distplot(alphas[f'{alpha:03}'], kde=False);
```

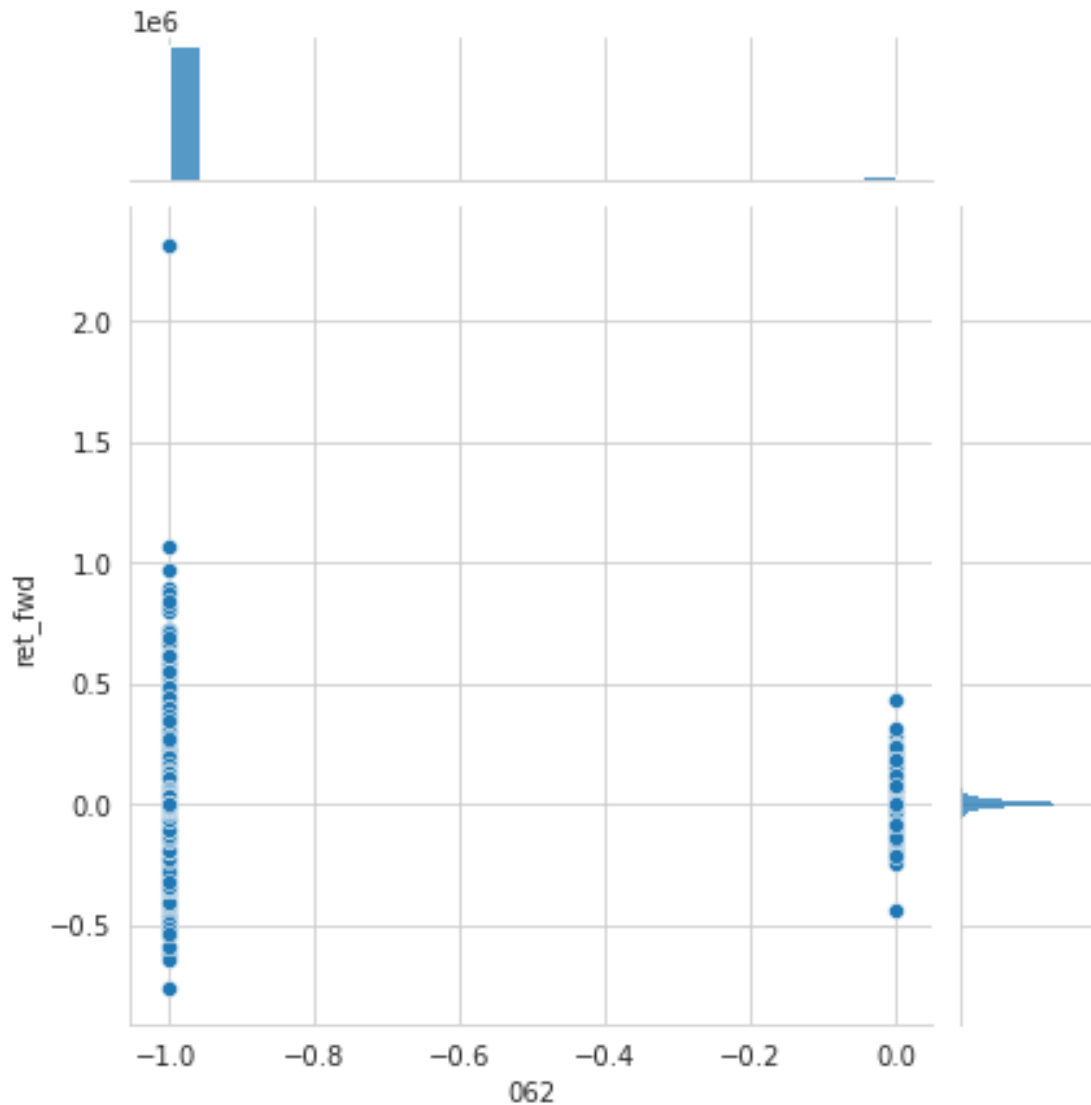


```
[79]: alphas.groupby(alphas[f'{alpha:03}']).ret_fwd.describe()
```

```
[79]:
```

	count	mean	std	min	25%	50%	75%	\
062								
-1	1227397.0	0.000582	0.025877	-0.757755	-0.009697	0.000496	0.010711	
0	27696.0	0.000545	0.019795	-0.441048	-0.007788	0.000648	0.008919	
	max							
062								
-1	2.317073							
0	0.436170							

```
[80]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```

```
[81]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
# mi[alpha]
```

1.67 Alpha 063

```
((rank(ts_weighted_mean(ts_delta(IndNeutralize(close, IndClass.industry), 2.25164), 8.22237))-
```

```
[82]: def alpha63(v, wv, industry):
    """((rank(ts_weighted_mean(ts_delta(IndNeutralize(close, IndClass.
    ↪ industry), 2), 8)) -
    rank(ts_weighted_mean(ts_corr(((vwap * 0.318108) + (open * (1 - 0.
    ↪ 318108)))),
    ts_sum(adv180, 37), 13), 12))) * -1)
```

```
"""
pass
```

```
[83]: alpha = 63
```

```
[84]: # %%time
# alphas[f'{alpha:03}'] = alpha48(o, c)
```

```
[85]: # alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[86]: # sns.distplot(alphas[f'{alpha:03}']);
```

```
[87]: # g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
#
```

```
[88]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
# mi[alpha]
```

1.68 Alpha 064

```
-ts_max(rank(ts_corr(rank(volume), rank(vwap), 5)), 5)
```

```
[89]: def alpha064(o, h, l, v, vwap):
    """((rank(ts_corr(ts_sum(((open * 0.178404) + (low * (1 - 0.178404))), 12.
    →7054), ts_sum(adv120, 12.7054), 16.6208)) <
        rank(ts_delta((((high + low) / 2) * 0.178404) + (vwap * (1 - 0.
    →178404))), 3.69741))) * -1)"""
    w = 0.178404
    return (rank(ts_corr(ts_sum(o.mul(w).add(l.mul(1 - w)), 12),
        ts_sum(ts_mean(v, 120), 12), 16))
        .lt(rank(ts_delta(h.add(l).div(2).mul(w)
        .add(vwap.mul(1 - w)), 3)))
        .mul(-1)
        .stack('ticker')
        .swaplevel())
```

```
[90]: alpha = 64
```

```
[91]: %%time
alphas[f'{alpha:03}'] = alpha064(o, h, l, v, vwap)
```

```
CPU times: user 3.74 s, sys: 40.3 ms, total: 3.79 s
Wall time: 3.69 s
```

```
[92]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[93]: alphas.groupby(alphas[f'{alpha:03}']).ret_fwd.describe()
```

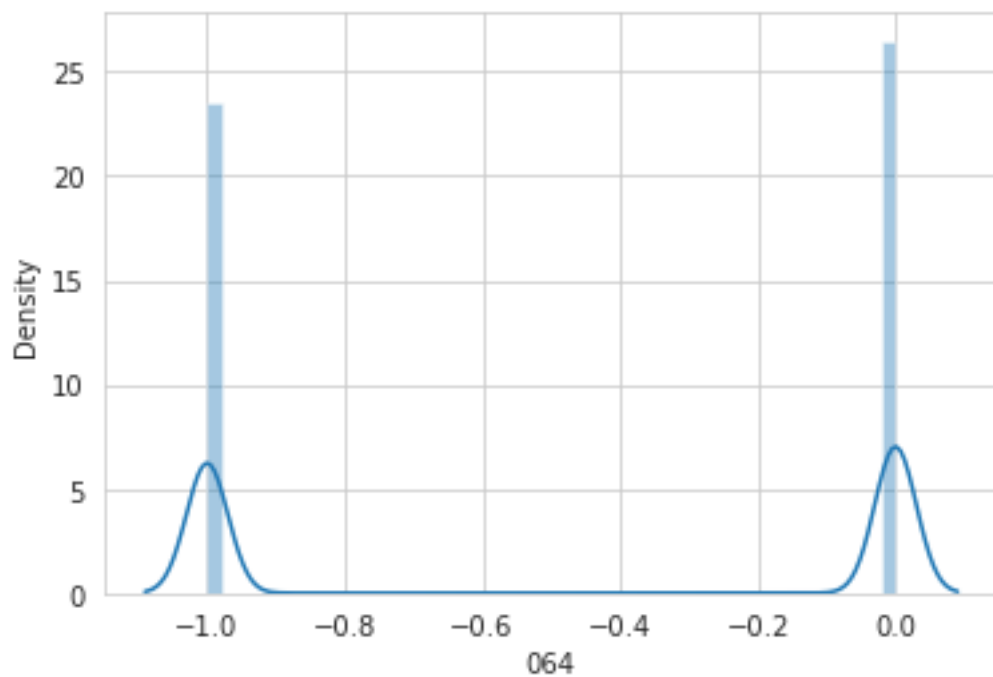
```
[93]:
```

	count	mean	std	min	25%	50%	75%	\
064								
-1	590330.0	0.000463	0.025842	-0.757755	-0.009714	0.000410	0.010584	
0	664763.0	0.000687	0.025683	-0.619752	-0.009595	0.000576	0.010741	

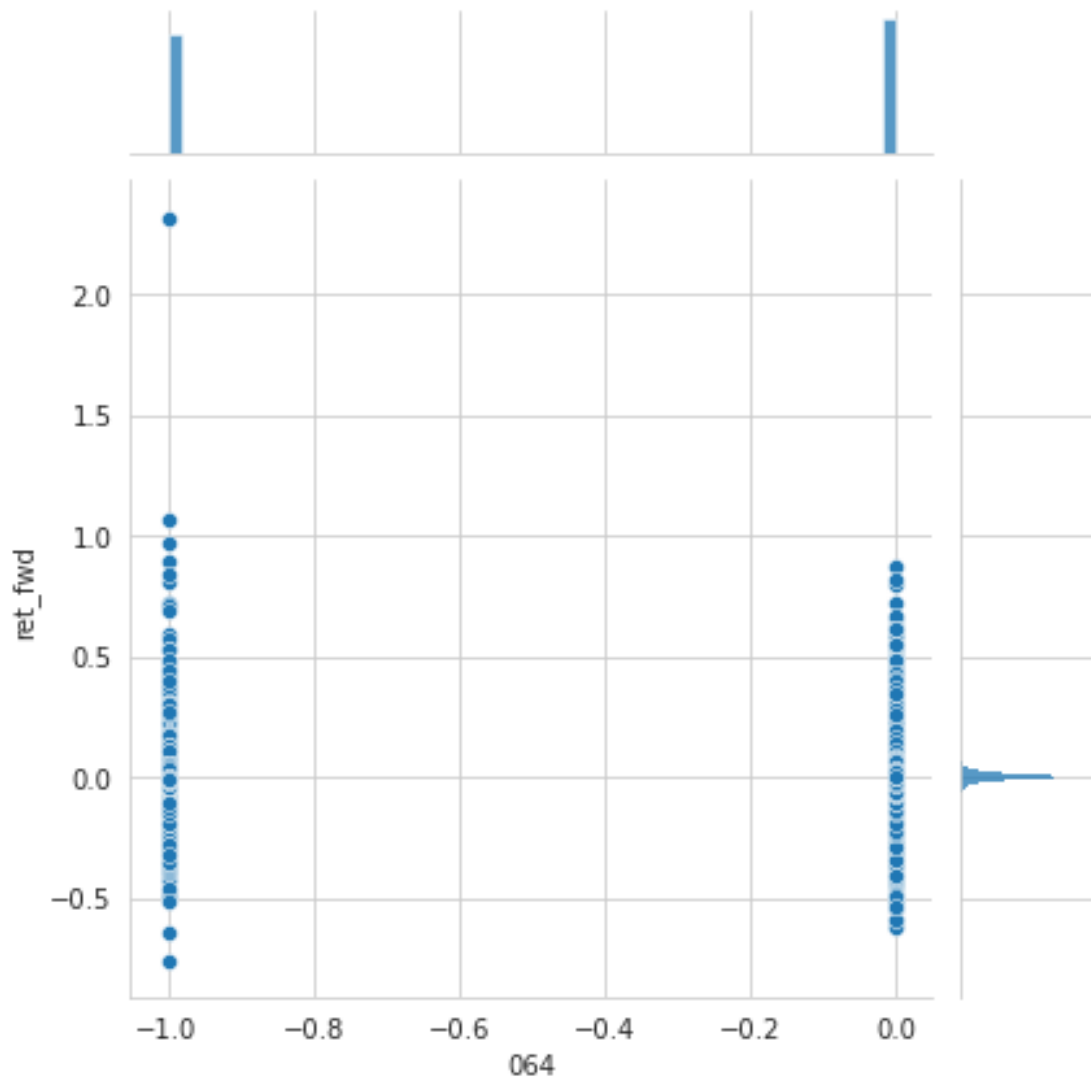

```
max
```

064	
-1	2.317073
0	0.869835

```
[94]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[95]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



1.69 Alpha 065

```
((rank(ts_corr(((open * 0.00817205) + (vwap * (1 - 0.00817205))),
                ts_sum(adv60,8.6911), 6.40374)) <
 rank((open - ts_min(open, 13.635)))) * -1)
```

```
[96]: def alpha065(o, v, vwap):
    """((rank(ts_corr(((open * 0.00817205) + (vwap * (1 - 0.00817205))),
                        ts_sum(adv60,8.6911), 6.40374)) <
        rank((open - ts_min(open, 13.635)))) * -1)
    """
    w = 0.00817205
    return (rank(ts_corr(o.mul(w).add(vwap.mul(1 - w)),
                        ts_mean(ts_mean(v, 60), 9), 6))
```

```
.lt(rank(o.sub(ts_min(o, 13))))
.mul(-1)
.stack('ticker')
.swaplevel()
```

```
[97]: alpha = 65
```

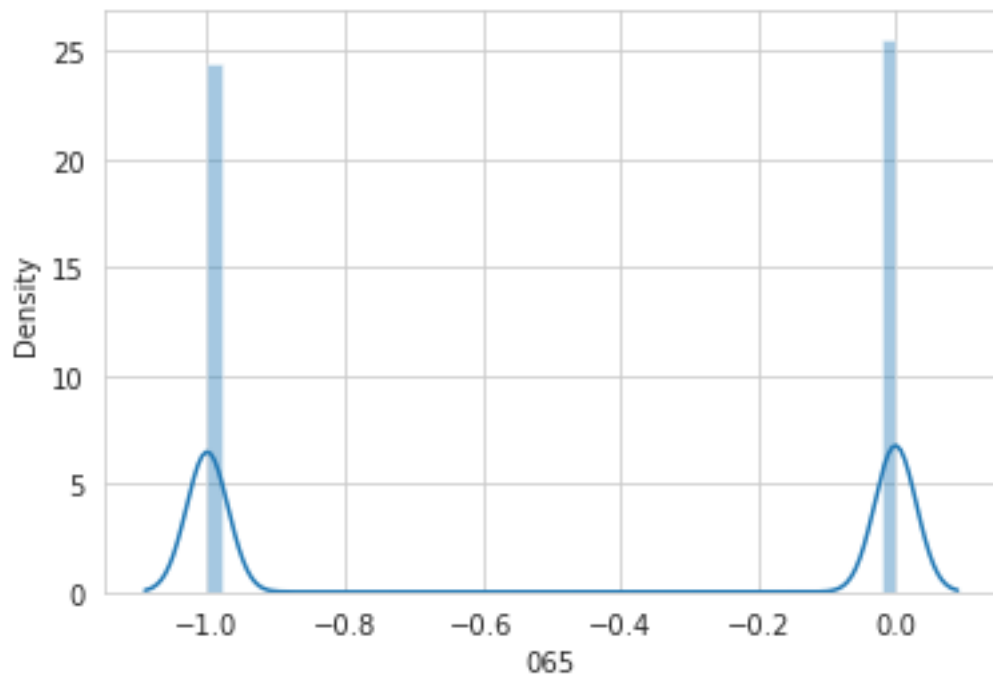
```
[98]: %%time
alphas[f'{alpha:03}'] = alpha065(o, v, vwap)
```

CPU times: user 3.83 s, sys: 32 ms, total: 3.86 s

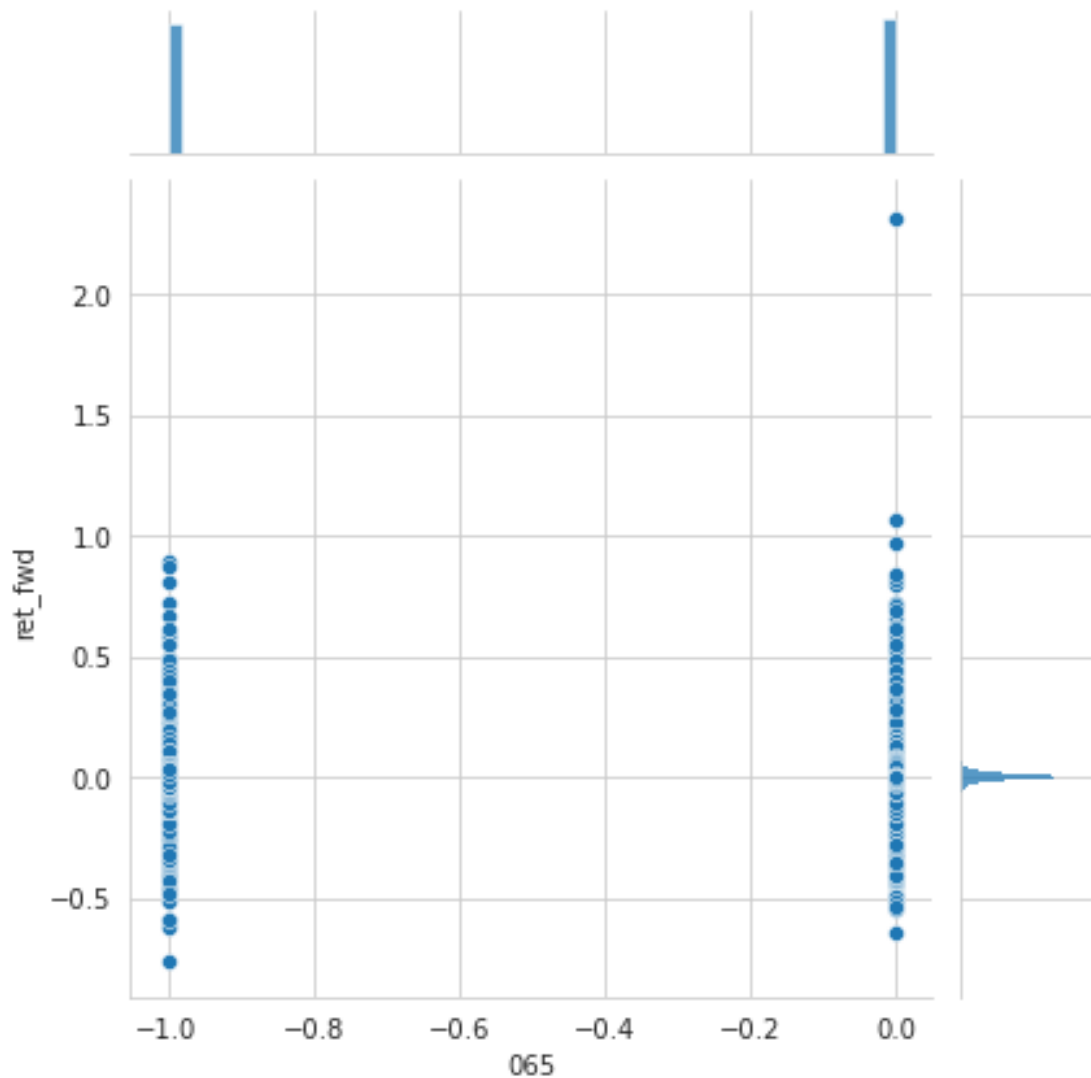
Wall time: 3.79 s

```
[99]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[100]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[101]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



1.70 Alpha 066

```
((rank(ts_weighted_mean(ts_delta(vwap, 3.51013), 7.23052)) +
    ts_rank(ts_weighted_mean((((low* 0.96633) + (low *
        (1 - 0.96633)))) - vwap) /
        (open - ((high + low) / 2))), 11.4157), 6.72611)) * -1)
```

```
[102]: def alpha066(l, h, vwap):
    """((rank(ts_weighted_mean(ts_delta(vwap, 3.51013), 7.23052)) +
        ts_rank(ts_weighted_mean((((low* 0.96633) + (low *
            (1 - 0.96633)))) - vwap) /
            (open - ((high + low) / 2))), 11.4157), 6.
        ↪ 72611)) * -1)
    """
```

```

w = 0.96633
return (rank(ts_weighted_mean(ts_delta(vwap, 4), 7))
        .add(ts_rank(ts_weighted_mean(l.mul(w).add(1.mul(1 - w))
        .sub(vwap)
        .div(o.sub(h.add(1).div(2))).
↪add(1e-3)), 11), 7))
        .mul(-1)
        .stack('ticker')
        .swaplevel())

```

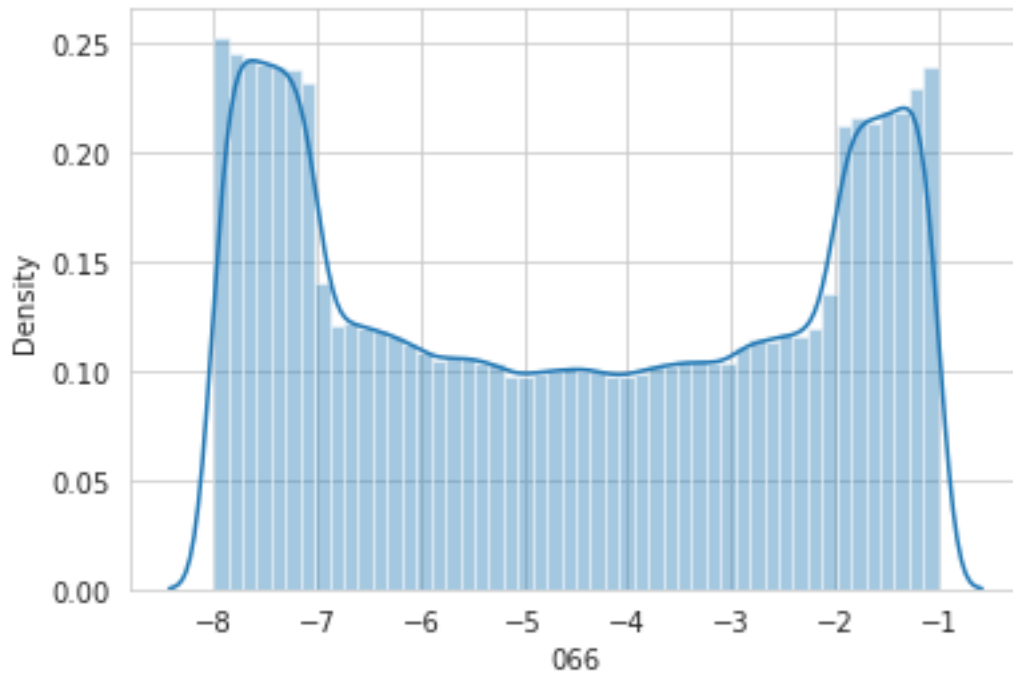
```
[103]: alpha = 66
```

```
[104]: %%time
alphas[f'{alpha:03}'] = alpha066(l, h, vwap)
```

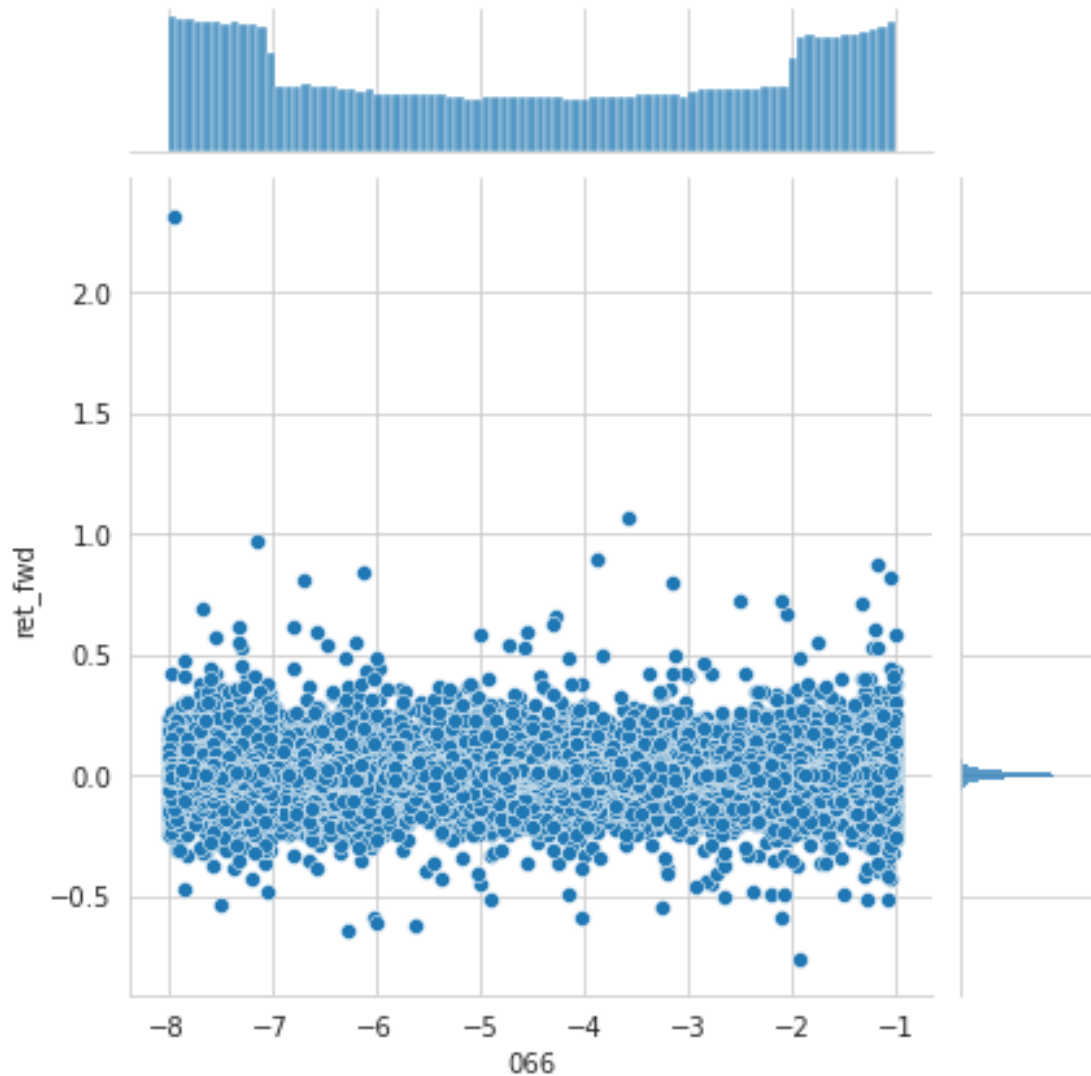
CPU times: user 3min, sys: 257 ms, total: 3min
Wall time: 3min

```
[105]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[106]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[107]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[108]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[108]: 0.009140875589292108
```

1.71 Alpha 067

```
(rank(ts_delta(IndNeutralize(((close * 0.60733) + (open * (1 - 0.60733)))),IndClass.sector), 1.2)
rank(ts_corr(Ts_Rank(vwap, 3.60973), Ts_Rank(adv150,9.18637), 14.6644)))
```

```
[109]: def alpha067(h, v, sector, subindustry):
        """(power(rank((high - ts_min(high, 2.14593))),
            rank(ts_corr(IndNeutralize(vwap,IndClass.sector),
                IndNeutralize(adv20, IndClass.subindustry), 6.02936)))) * -1)
```



```
"""
pass
```

```
[110]: alpha = 67
```

```
[111]: # %%time
# alphas[f'{alpha:03}'] = alpha056(r, cap)
```

```
[112]: # alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[113]: # sns.distplot(alphas[f'{alpha:03}']);
```

```
[114]: # g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
#
```

```
[115]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
# mi[alpha]
```

1.72 Alpha 068

```
((ts_rank(ts_corr(rank(high), rank(adv15), 8.91644), 13.9333) <
rank(ts_delta(((close * 0.518371) + (low * (1 - 0.518371))), 1.06157))) * -1)
```

```
[116]: def alpha068(h, c, v):
        """((ts_rank(ts_corr(rank(high), rank(adv15), 8.91644), 13.9333) <
            rank(ts_delta(((close * 0.518371) + (low * (1 - 0.518371))), 1.06157)))
        ↪ * -1)
        """
        w = 0.518371
        return (ts_rank(ts_corr(rank(h), rank(ts_mean(v, 15))), 9), 14)
                .lt(rank(ts_delta(c.mul(w).add(l.mul(1 - w)), 1)))
                .mul(-1)
                .stack('ticker')
                .swaplevel())
```

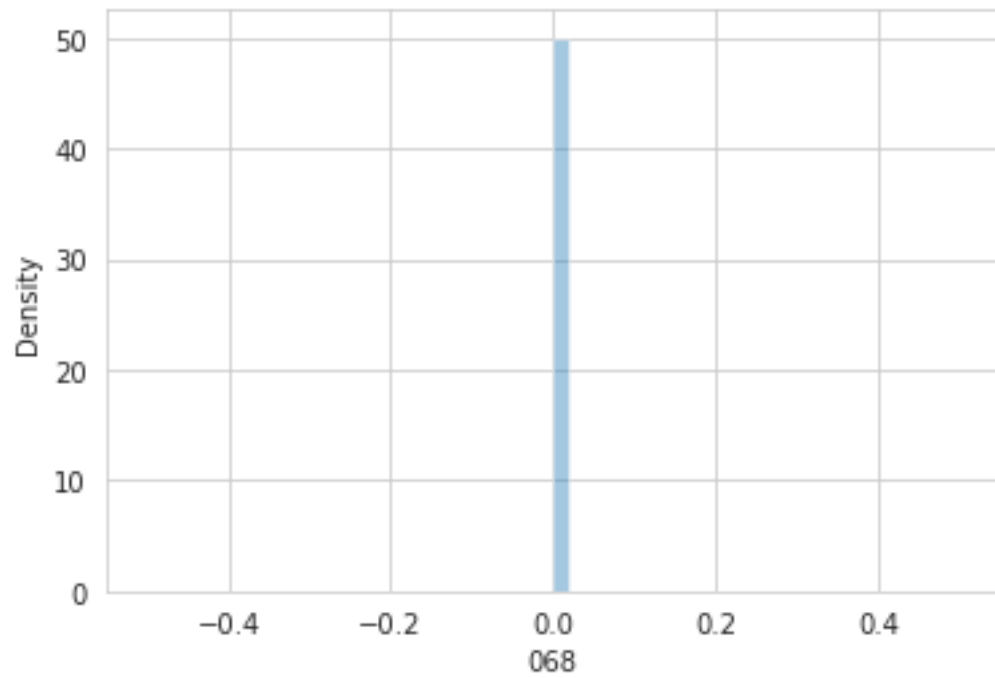
```
[117]: alpha = 68
```

```
[118]: %%time
alphas[f'{alpha:03}'] = alpha068(h, c, v)
```

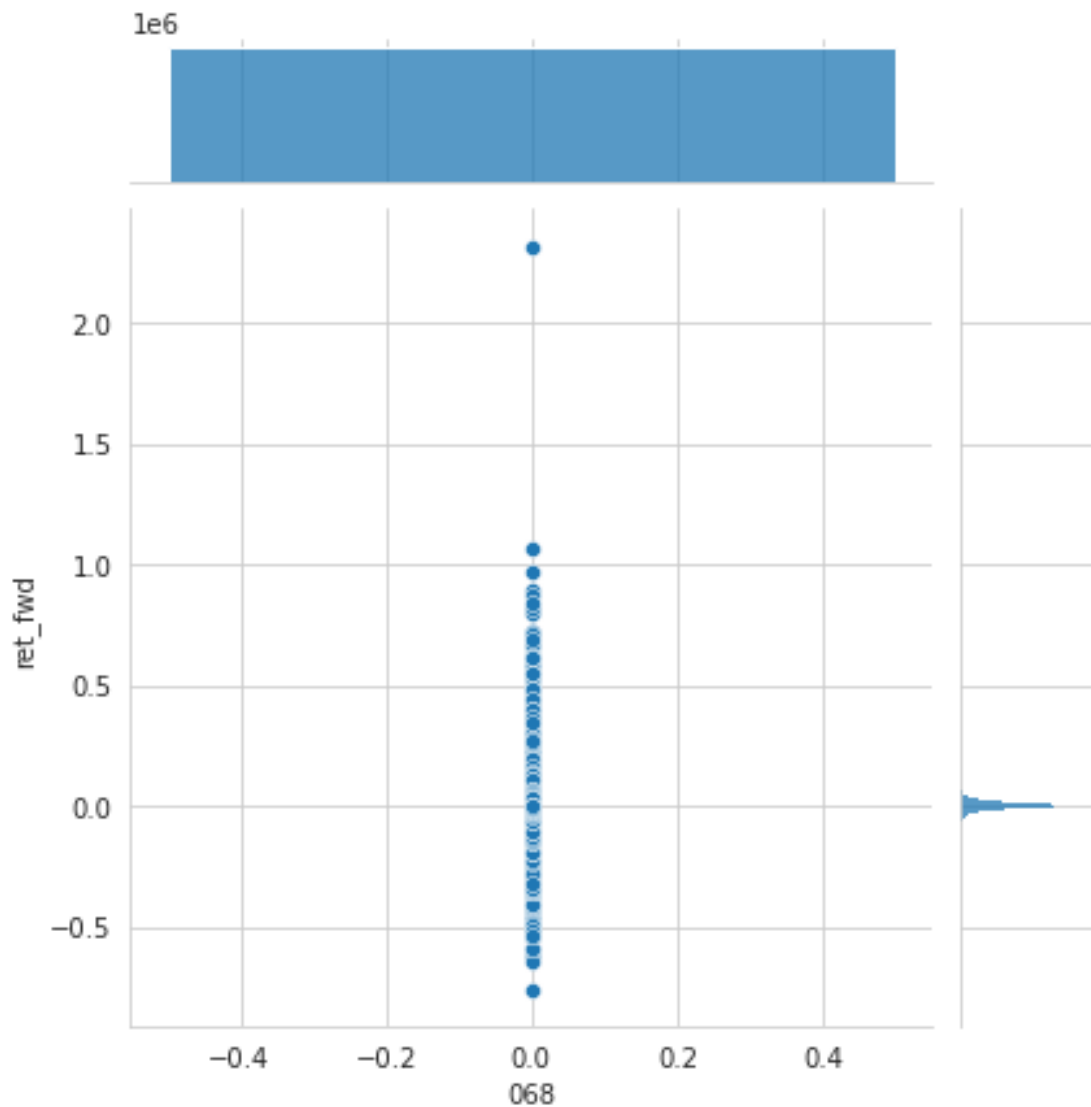
CPU times: user 2min 54s, sys: 66.2 ms, total: 2min 54s
Wall time: 2min 54s

```
[119]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[120]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[121]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



1.73 Alpha 069

```
((power(rank(ts_max(ts_delta(IndNeutralize(vwap, IndClass.industry), 2.72412), 4.79344)),
      Ts_Rank(ts_corr(((close * 0.490655) + (vwap * (1 - 0.490655))), adv20, 4.92416), 9.0615),
```

```
[122]: def alpha069(c, vwap, industry):
        """((power(rank(ts_max(ts_delta(IndNeutralize(vwap, IndClass.industry), 2.
        ↪ 72412), 4.79344)),
        Ts_Rank(ts_corr(((close * 0.490655) + (vwap * (1 - 0.490655))), adv20, 4.
        ↪ 92416), 9.0615))) * -1)
        """
        pass
```

1.74 Alpha 070

```
((power(rank(ts_delta(vwap, 1.29456)),
        ts_rank(ts_corr(IndNeutralize(close, IndClass.industry), adv50, 17.8256), 17.9171))) * -1)
```

```
[123]: def alpha076(c, v, vwap, industry):
        """((power(rank(ts_delta(vwap, 1.29456)),
            ts_rank(ts_corr(IndNeutralize(close, IndClass.industry), adv50, 17.
            ↪8256), 17.9171))) * -1)
        """
        pass
```

```
[124]: alpha = 70
```

1.75 Alpha 071

```
-ts_max(rank(ts_corr(rank(volume), rank(vwap), 5)), 5)
```

```
[125]: def alpha071(o, c, v, vwap):
        """max(ts_rank(ts_weighted_mean(ts_corr(ts_rank(close, 3.43976),
        ↪ts_rank(adv180,12.0647), 18.0175), 4.20501), 15.6948),
            ts_rank(ts_weighted_mean((rank(((low + open) - (vwap +vwap))))^2),
        ↪16.4662), 4.4388))"""

        s1 = (ts_rank(ts_weighted_mean(ts_corr(ts_rank(c, 3),
            ts_rank(ts_mean(v, 180), 12), 18),
        ↪4), 16))
        s2 = (ts_rank(ts_weighted_mean(rank(1.add(o).
            sub(vwap.mul(2)))
            .pow(2), 16), 4))

        return (s1.where(s1 > s2, s2)
            .stack('ticker')
            .swaplevel())
```

```
[126]: alpha = 71
```

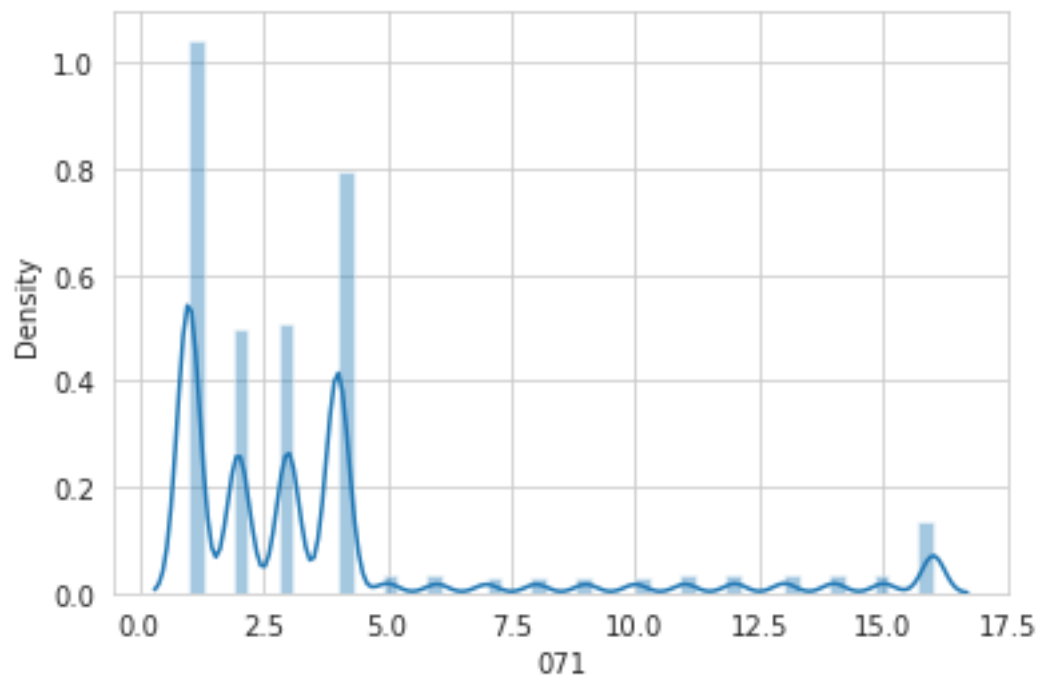
```
[127]: %%time
        alphas[f'{alpha:03}'] = alpha071(o, c, v, vwap)
```

CPU times: user 9min 28s, sys: 81.3 ms, total: 9min 29s

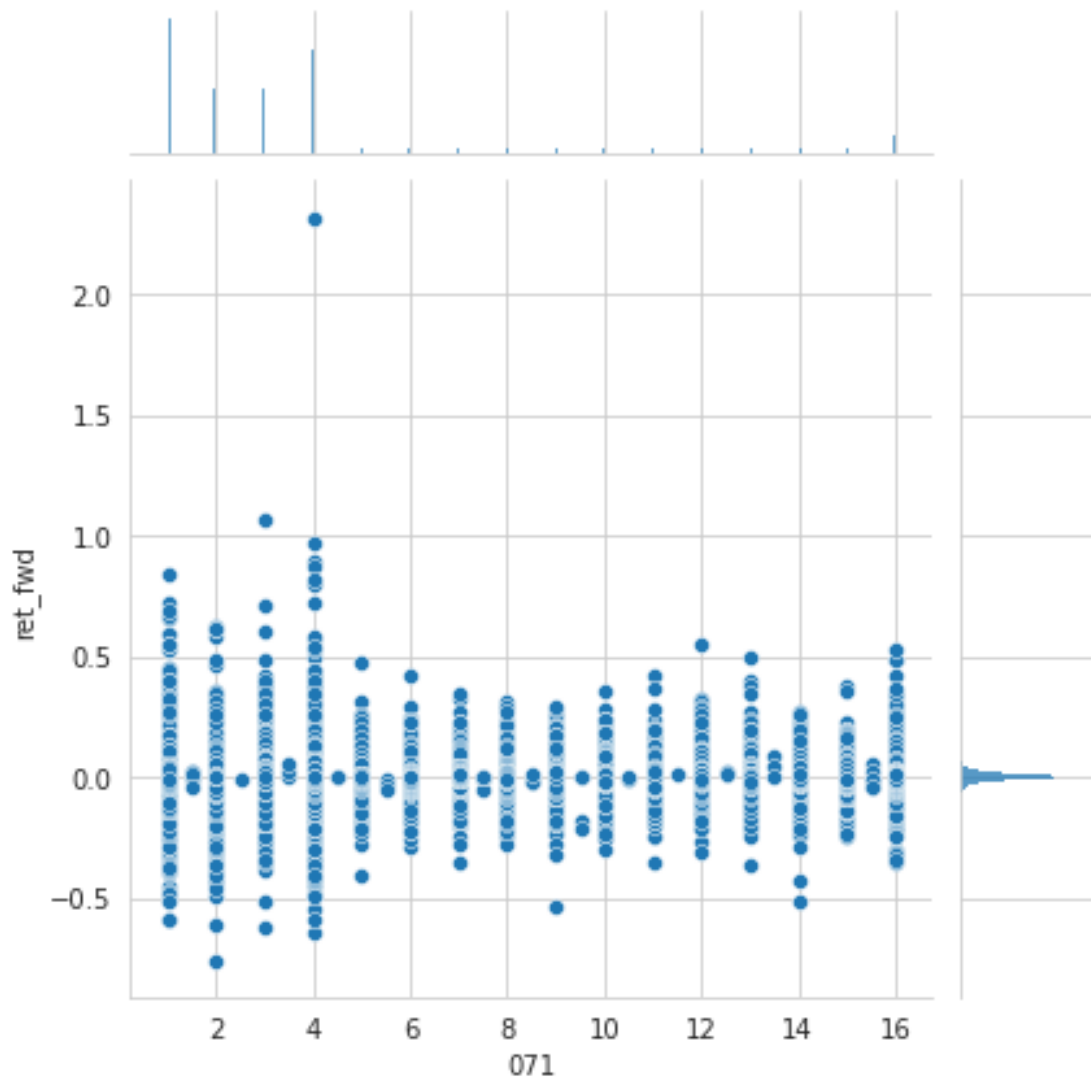
Wall time: 9min 28s

```
[128]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[129]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[130]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[131]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[131]: 0.002330970321091197
```

1.76 Alpha 072

```
(rank(ts_weighted_mean(ts_corr(((high + low) / 2), adv40, 8.93345), 10.1519)) /
 rank(ts_weighted_mean(ts_corr(ts_rank(vwap, 3.72469), ts_rank(volume, 18.5188), 6.8667
```

```
[132]: def alpha072(h, l, v, vwap):
        """(rank(ts_weighted_mean(ts_corr(((high + low) / 2), adv40, 8.93345), 10.
        ↪1519)) /
```

```

        rank(ts_weighted_mean(ts_corr(ts_rank(vwap, 3.72469), ts_rank(volume, 18.5188), 6.86671), 2.95011)))
        """
        return (rank(ts_weighted_mean(ts_corr(h.add(1).div(2), ts_mean(v, 40), 9), 10))
                    .div(rank(ts_weighted_mean(ts_corr(ts_rank(vwap, 3), ts_rank(v, 18), 6), 2)))
                    .stack('ticker')
                    .swaplevel())

```

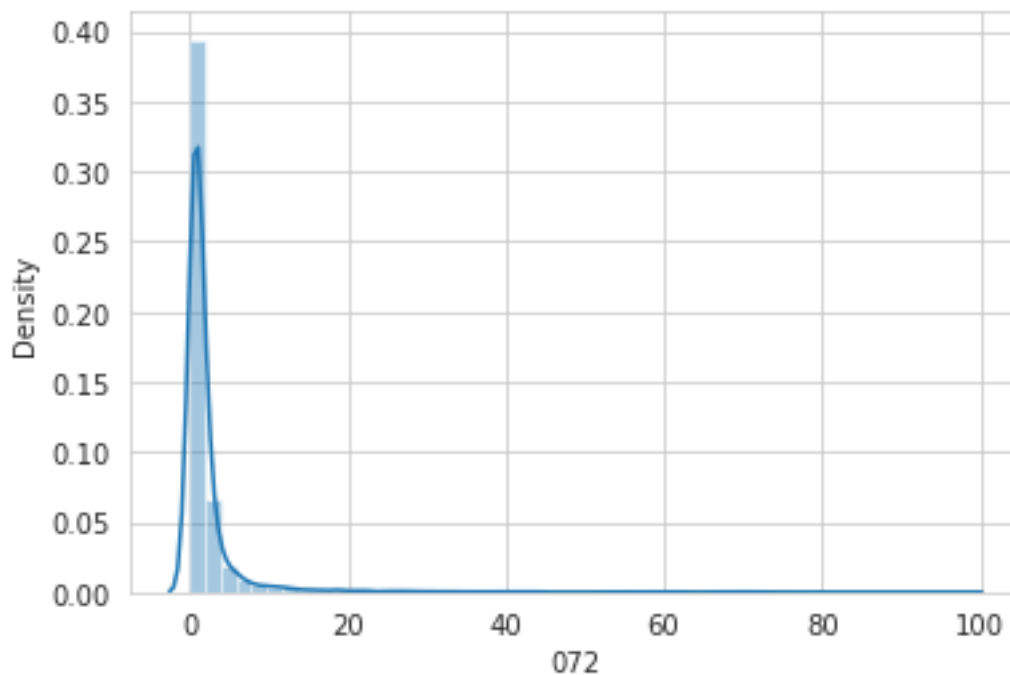
```
[133]: alpha = 72
```

```
[134]: %%time
alphas[f'{alpha:03}'] = alpha072(h, l, v, vwap)
```

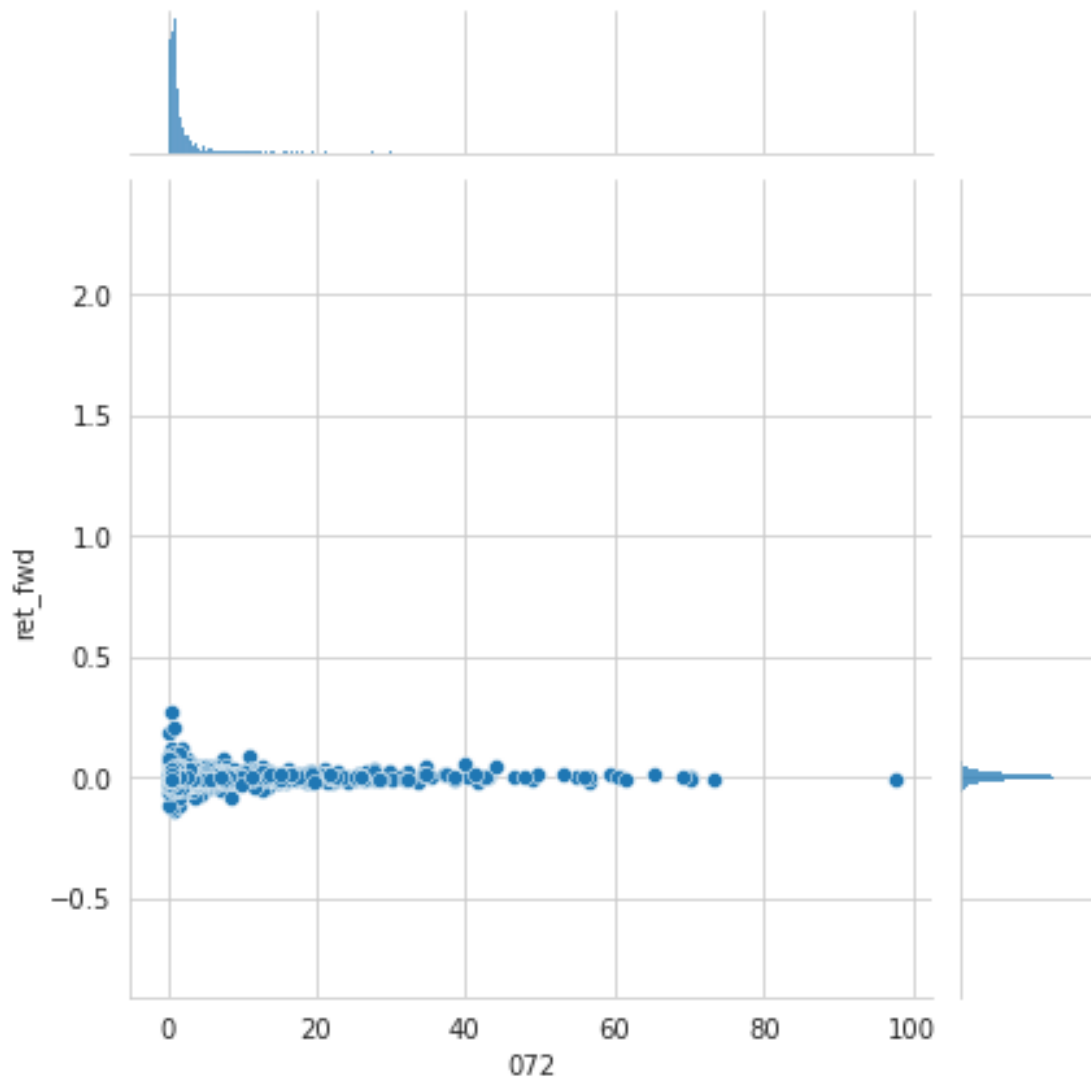
CPU times: user 6min 5s, sys: 95.2 ms, total: 6min 5s
Wall time: 6min 4s

```
[135]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[136]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[137]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



1.77 Alpha 073

```
(max(rank(ts_weighted_mean(ts_delta(vwap, 4.72775), 2.91864)),
    ts_rank(ts_weighted_mean(((ts_delta(((open * 0.147155) +
        (low * (1 - 0.147155)))), 2.03608) /
        ((open * 0.147155) + (low * (1 - 0.147155)))) * -1), 3.33829), 16.7411)) * -1)
```

```
[138]: def alpha073(1, vwap):
    """(max(rank(ts_weighted_mean(ts_delta(vwap, 4.72775), 2.91864)),
        ts_rank(ts_weighted_mean(((ts_delta(((open * 0.147155) +
            (low * (1 - 0.147155)))), 2.03608) /
            ((open * 0.147155) + (low * (1 - 0.147155)))) * -1), 3.33829), 16.
        ↪ 7411)) * -1)
    """
```



```

w = 0.147155
s1 = rank(ts_weighted_mean(ts_delta(vwap, 5), 3))
s2 = (ts_rank(ts_weighted_mean(ts_delta(o.mul(w).add(1.mul(1 - w))), 2)
        .div(o.mul(w).add(1.mul(1 - w)).mul(-1)),
↪3), 16))

print(s2)
return (s1.where(s1 > s2, s2)
        .mul(-1)
        .stack('ticker')
        .swaplevel())

```

```
[139]: alpha = 73
```

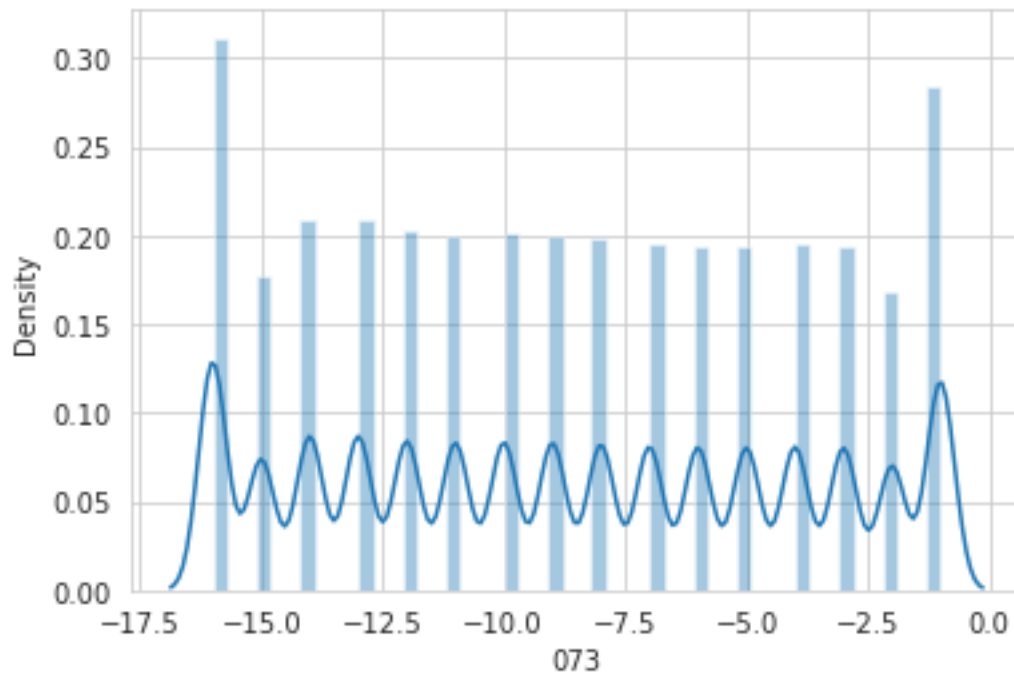
```
[140]: # %%time
alphas[f'{alpha:03}'] = alpha073(1, vwap)
```

ticker	A	AAL	AAP	AAPL	ABC	ABT	ACN	ADBE	ADI	ADM	...	\
date												
2007-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2007-01-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2007-01-08	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2007-01-09	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2007-01-10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
...	
2016-12-22	8.0	5.0	11.0	12.0	9.0	7.0	16.0	10.0	6.0	14.0	...	
2016-12-23	10.0	14.0	15.0	14.0	12.0	6.0	15.0	12.0	8.0	15.0	...	
2016-12-27	7.0	16.0	13.0	13.0	10.0	5.0	12.0	11.0	7.0	13.0	...	
2016-12-28	11.0	16.0	10.0	11.0	7.0	6.0	12.0	14.0	7.0	11.0	...	
2016-12-29	14.0	16.0	10.0	12.0	6.0	9.0	12.0	14.0	13.0	9.0	...	
ticker	XEC	XEL	XL	XLNX	XOM	XRAY	XRX	YUM	ZBH	ZION		
date												
2007-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		
2007-01-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		
2007-01-08	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		
2007-01-09	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		
2007-01-10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		
...		
2016-12-22	13.0	10.0	3.0	5.0	15.0	13.0	10.0	11.0	11.0	9.0		
2016-12-23	12.0	11.0	4.0	7.0	9.0	14.0	14.0	12.0	9.0	10.0		
2016-12-27	8.0	11.0	4.0	9.0	6.0	11.0	13.0	12.0	7.0	12.0		
2016-12-28	7.0	15.0	7.0	11.0	8.0	14.0	15.0	11.0	5.0	12.0		
2016-12-29	8.0	16.0	13.0	16.0	15.0	16.0	15.0	13.0	5.0	15.0		

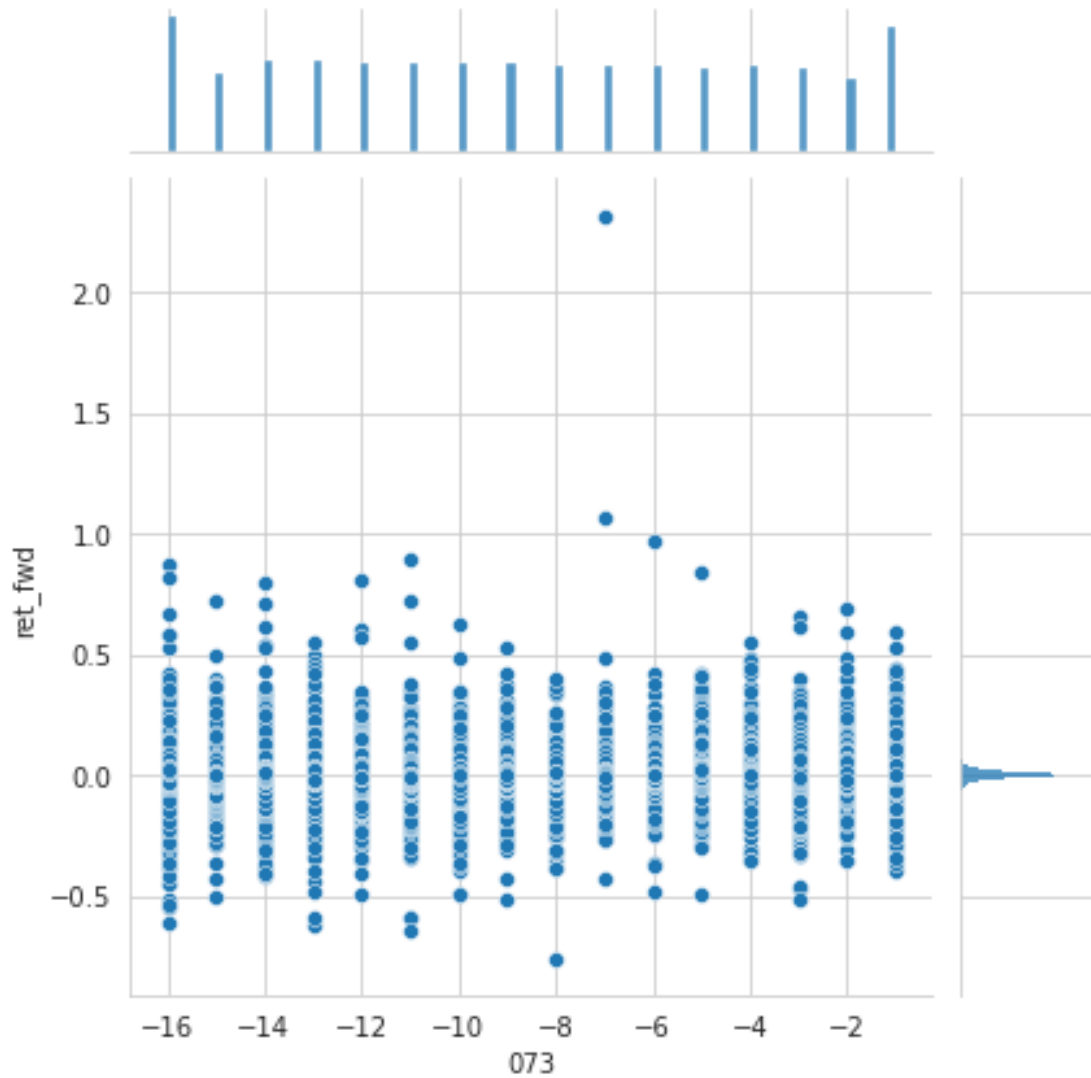
[2516 rows x 500 columns]

```
[141]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[142]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[143]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[144]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[144]: 0.0019621236363676076
```

1.78 Alpha 074

```
((rank(ts_corr(close, ts_sum(adv30, 37.4843), 15.1365)) <
rank(ts_corr(rank(((high * 0.0261661) + (vwap * (1 - 0.0261661)))), rank(volume), 11.4791))) * -1)"""
```

```
[145]: def alpha074(v, vwap):
    """((rank(ts_corr(close, ts_sum(adv30, 37.4843), 15.1365)) <
rank(ts_corr(rank(((high * 0.0261661) + (vwap * (1 - 0.0261661)))),
rank(volume), 11.4791))) * -1)"""
```

```

w = 0.0261661
return (rank(ts_corr(c, ts_mean(ts_mean(v, 30), 37), 15))
        .lt(rank(ts_corr(rank(h.mul(w).add(vwap.mul(1 - w))), rank(v), 11)))
        .mul(-1)
        .stack('ticker')
        .swaplevel())

```

```
[146]: alpha = 74
```

```

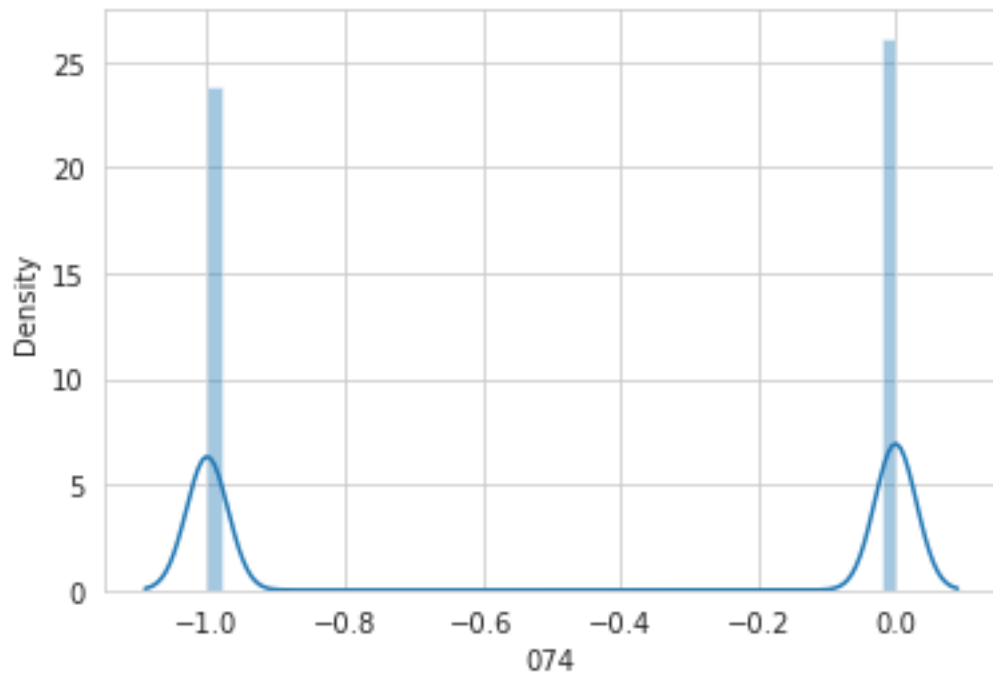
[147]: %%time
alphas[f'{alpha:03}'] = alpha074(v, vwap)

```

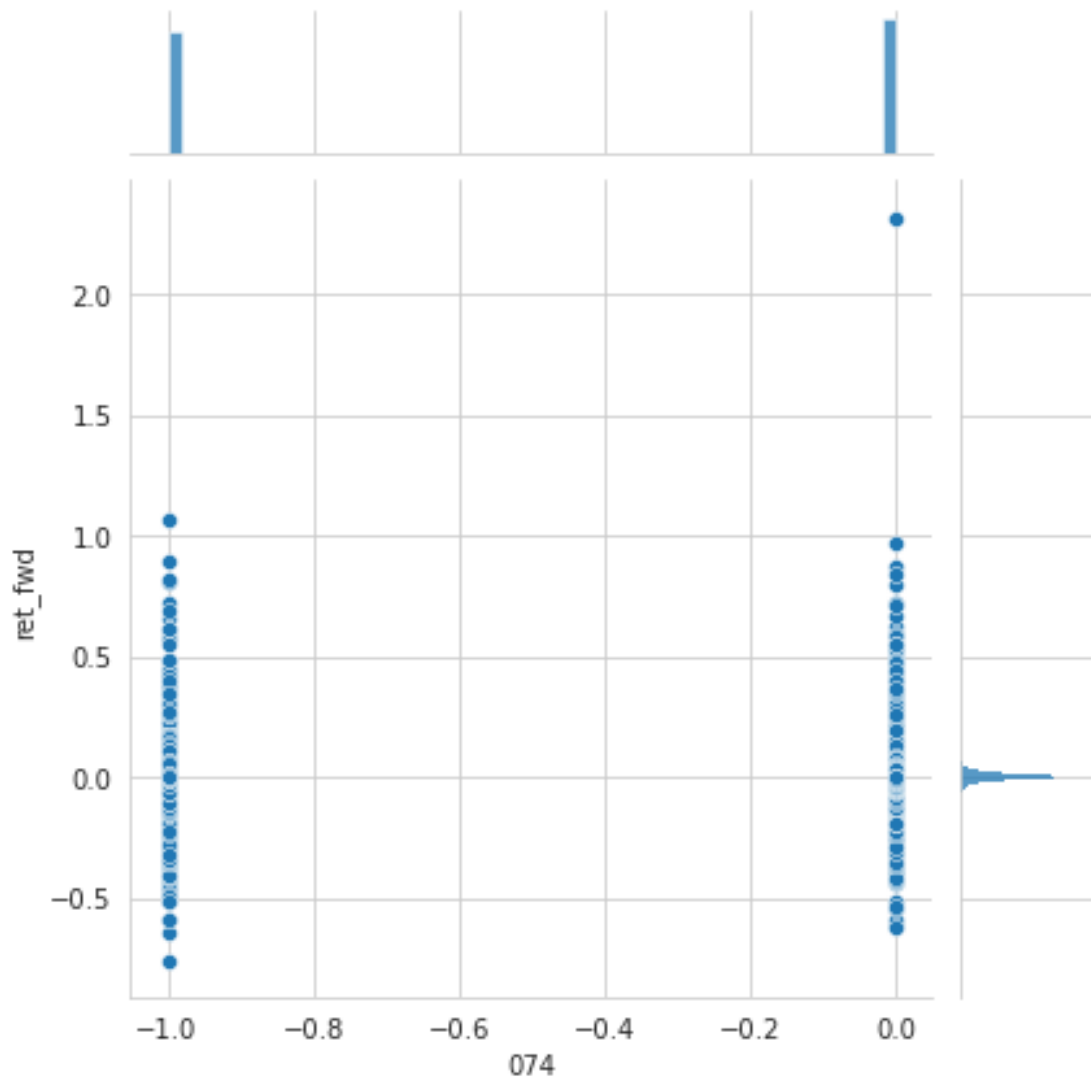
CPU times: user 4.76 s, sys: 64 ms, total: 4.82 s
Wall time: 4.73 s

```
[148]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[149]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[150]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[151]: alphas.groupby(alphas[f'{alpha:03}']).ret_fwd.describe()
```

```
[151]:
```

	count	mean	std	min	25%	50%	75%	\
074								
-1	598745.0	0.000563	0.026009	-0.757755	-0.009807	0.000456	0.010752	
0	656348.0	0.000599	0.025527	-0.619752	-0.009509	0.000538	0.010582	
		max						
074								
-1	1.061026							
0	2.317073							

1.79 Alpha 075

```
(rank(ts_corr(vwap, volume, 4.24304)) <
 rank(ts_corr(rank(low), rank(adv50), 12.4413)))
```

```
[152]: def alpha075(l, v, vwap):
        """(rank(ts_corr(vwap, volume, 4.24304)) <
            rank(ts_corr(rank(low), rank(adv50), 12.4413)))
        """

        return (rank(ts_corr(vwap, v, 4))
                .lt(rank(ts_corr(rank(l), rank(ts_mean(v, 50)), 12)))
                .astype(int)
                .stack('ticker')
                .swaplevel())
```

```
[153]: alpha = 75
```

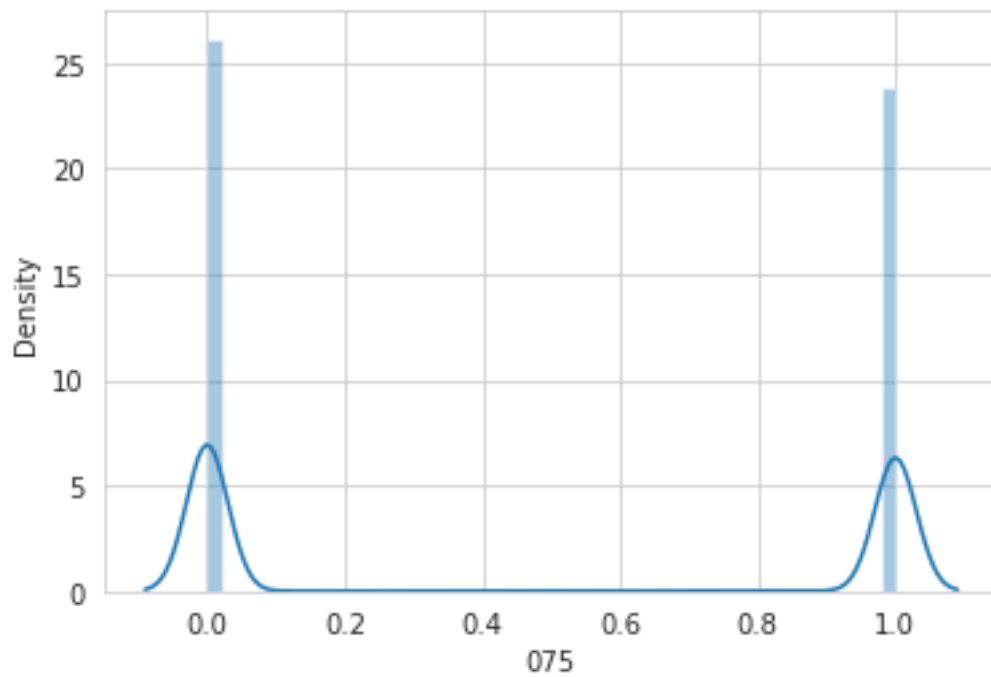
```
[154]: %%time
        alphas[f'{alpha:03}'] = alpha075(l, v, vwap)
```

CPU times: user 4.79 s, sys: 36 ms, total: 4.83 s

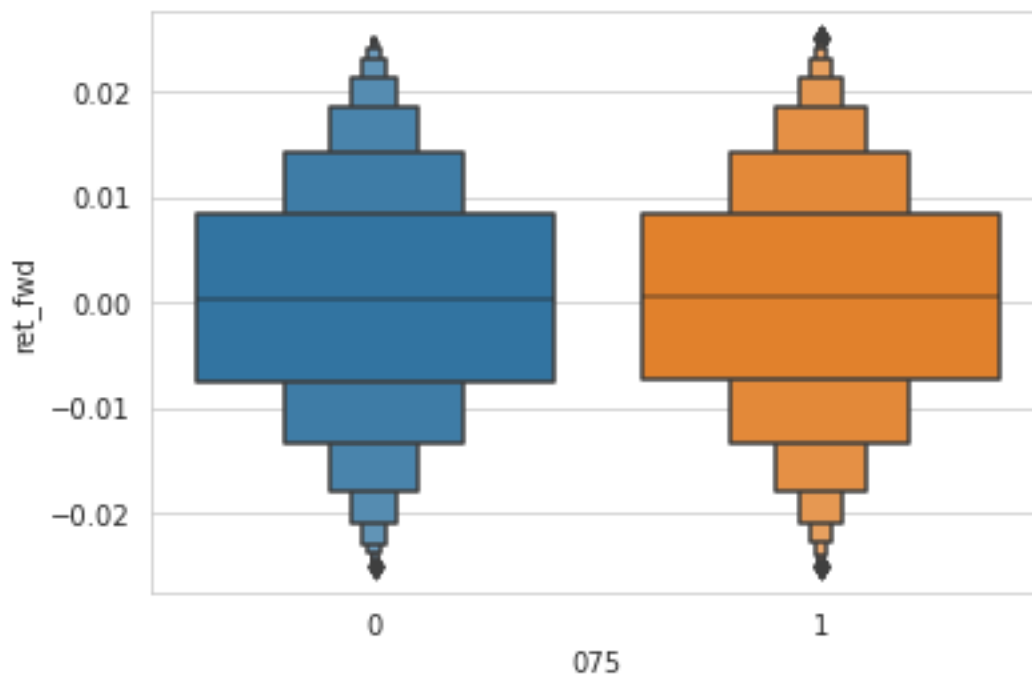
Wall time: 4.76 s

```
[155]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[156]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[157]: g = sns.boxenplot(x=f'{alpha:03}', y='ret_fwd', data=alphas[alphas.ret_fwd.
↪between(-.025, .025)]);
```



```
[158]: alphas.groupby(alphas[f'{alpha:03}']).ret_fwd.describe()
```

```
[158]:
```

	count	mean	std	min	25%	50%	75%	\
075								
0	656508.0	0.000545	0.026144	-0.757755	-0.009674	0.000472	0.010631	
1	598585.0	0.000621	0.025328	-0.607908	-0.009625	0.000529	0.010706	
	max							
075								
0	2.317073							
1	0.814423							

1.80 Alpha 076

```
(rank(ts_delta(IndNeutralize(((close * 0.60733) + (open * (1 - 0.60733))),IndClass.sector), 1.2
rank(ts_corr(Ts_Rank(vwap, 3.60973), Ts_Rank(adv150,9.18637), 14.6644)))
```

```
[159]: def alpha076(l, vwap, sector):
        """(max(rank(ts_weighted_mean(ts_delta(vwap, 1.24383), 11.8259))),
```

```

        ts_rank(ts_weighted_mean(ts_rank(ts_corr(IndNeutralize(low,
↪IndClass.sector), adv81,8.14941), 19.569), 17.1543), 19.383)) * -1)
        """
        pass

```

```
[160]: alpha = 76
```

1.81 Alpha 077

```

min(rank(ts_weighted_mean((((high + low) / 2) + high) - (vwap + high)), 20.0451)),
    rank(ts_weighted_mean(ts_corr(((high + low) / 2), adv40, 3.1614), 5.64125)))

```

```

[161]: def alpha077(l, h, vwap):
        """min(rank(ts_weighted_mean((((high + low) / 2) + high) - (vwap + high)),
↪20.0451)),
        rank(ts_weighted_mean(ts_corr(((high + low) / 2), adv40, 3.1614), 5.
↪64125)))
        """

        s1 = rank(ts_weighted_mean(h.add(1).div(2).sub(vwap), 20))
        s2 = rank(ts_weighted_mean(ts_corr(h.add(1).div(2), ts_mean(v, 40), 3), 5))
        return (s1.where(s1 < s2, s2)
                .stack('ticker')
                .swaplevel())

```

```
[162]: alpha = 77
```

```

[163]: %%time
        alphas[f'{alpha:03}'] = alpha077(l, h, vwap)

```

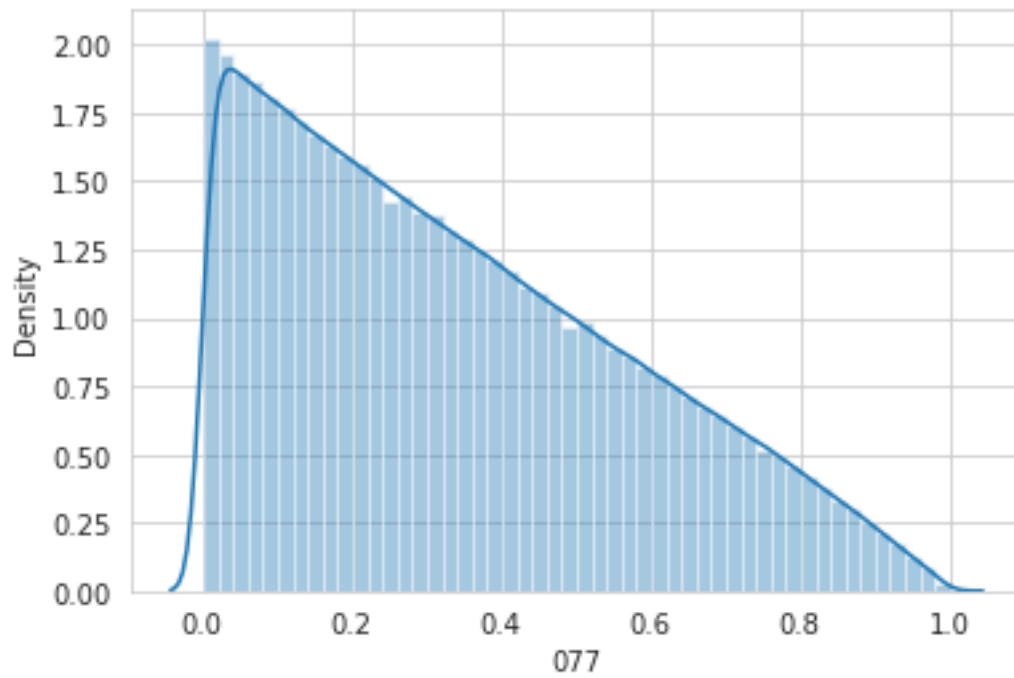
```

CPU times: user 3.73 s, sys: 16 ms, total: 3.75 s
Wall time: 3.66 s

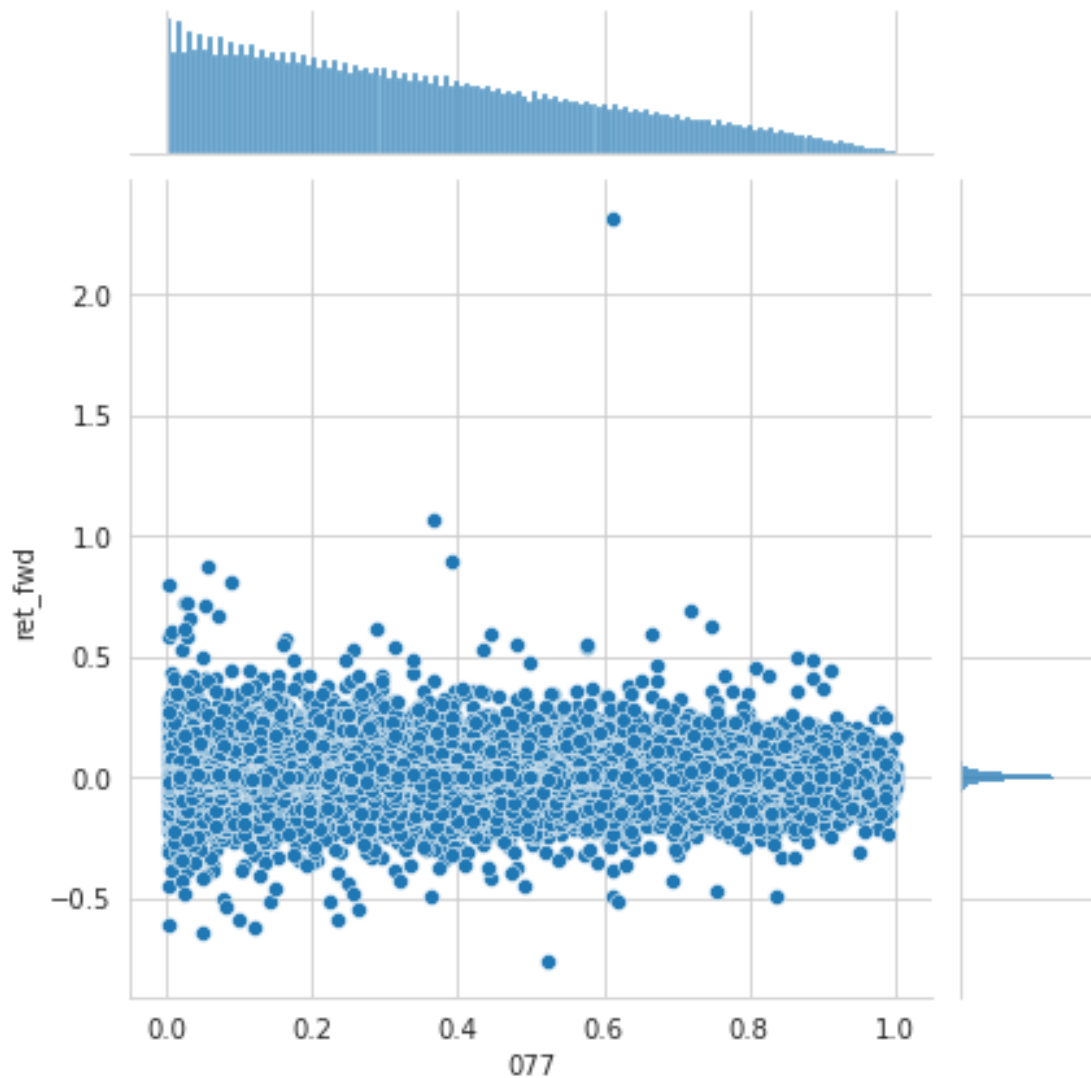
```

```
[164]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[165]: sns.distplot(alphas[f'{alpha:03}']);
```

```
[166]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[167]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[167]: 0
```

1.82 Alpha 078

```
(rank(ts_corr(ts_sum(((low * 0.352233) + (vwap * (1 - 0.352233)))), 19.7428),
ts_sum(adv40, 19.7428), 6.83313))^rank(ts_corr(rank(vwap), rank(volume), 5.77492)))
```

```
[168]: def alpha078(l, v, vwap):
        """(rank(ts_corr(ts_sum(((low * 0.352233) + (vwap * (1 - 0.352233)))), 19.
        ↪ 7428),
```

```

        ts_sum(adv40, 19.7428), 6.83313)) ~rank(ts_corr(rank(vwap),
↪rank(volume), 5.77492)))"""

w = 0.352233
return (rank(ts_corr(ts_sum((l.mul(w).add(vwap.mul(1 - w))), 19),
        ts_sum(ts_mean(v, 40), 19), 6))
        .pow(rank(ts_corr(rank(vwap), rank(v), 5)))
        .stack('ticker')
        .swaplevel())

```

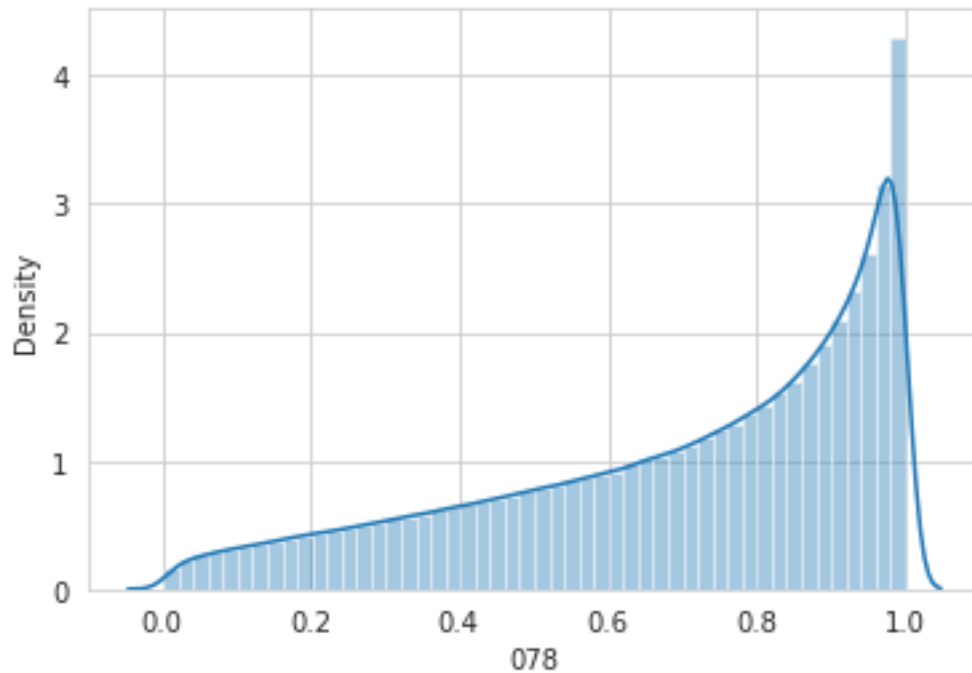
```
[169]: alpha = 78
```

```
[170]: %%time
alphas[f'{alpha:03}'] = alpha078(l, v, vwap)
```

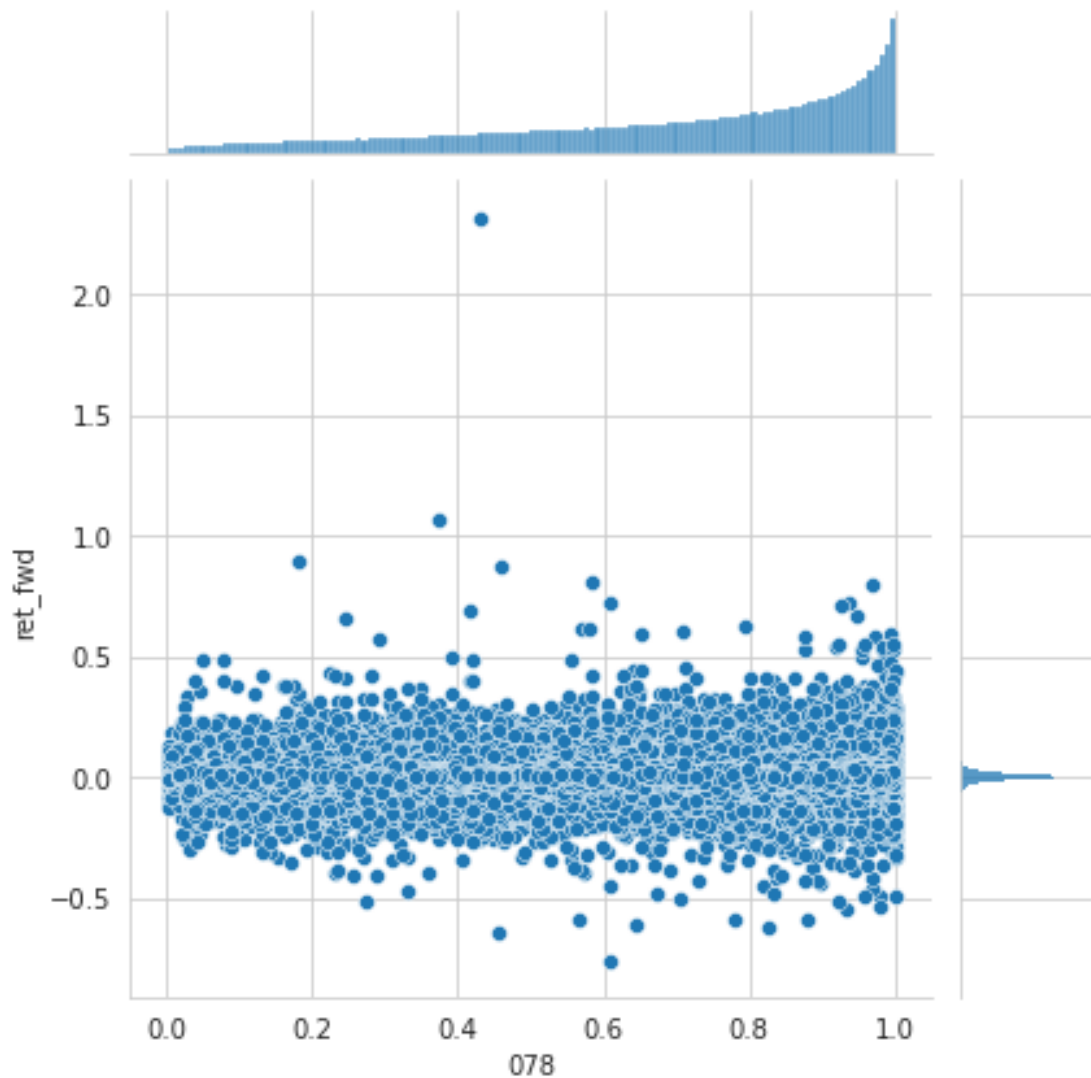
CPU times: user 4.67 s, sys: 20 ms, total: 4.69 s
Wall time: 4.59 s

```
[171]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[172]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[173]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[174]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[174]: 0.0002397436931129704
```

1.83 Alpha 079

```
(rank(ts_delta(IndNeutralize(((close * 0.60733) + (open * (1 - 0.60733))), IndClass.sector), 1.23438)) <
rank(ts_corr(Ts_Rank(vwap, 3.60973), Ts_Rank(adv150, 9.18637), 14.6644)))
```

```
[175]: def alpha079(o, v, sector):
        """(rank(ts_delta(IndNeutralize(((close * 0.60733) + (open * (1 - 0.
        ↪60733))), IndClass.sector), 1.23438)) <
        rank(ts_corr(Ts_Rank(vwap, 3.60973), Ts_Rank(adv150, 9.18637), 14.6644)))
```

```
"""
pass
```

1.84 Alpha 080

```
((power(rank(sign(ts_delta(IndNeutralize(((open * 0.868128) + (high * (1 - 0.868128)))), IndClass.
ts_rank(ts_corr(high, adv10, 5.11456), 5.53756)) * -1)
```

```
[176]: def alpha080(h, industry):
        """((power(rank(sign(ts_delta(IndNeutralize(((open * 0.868128) + (high * (1 -
        ↪ 0.868128)))), IndClass.industry), 4.04545))),
            ts_rank(ts_corr(high, adv10, 5.11456), 5.53756)) * -1)
        """
        pass
```

1.85 Alpha 081

```
-(rank(log(ts_product(rank((rank(ts_corr(vwap, ts_sum(adv10, 49.6054), 8.47743))^4)), 14.9655))),
rank(ts_corr(rank(vwap), rank(volume), 5.07914)))
```

```
[177]: def alpha081(v, vwap):
        """-(rank(log(ts_product(rank((rank(ts_corr(vwap, ts_sum(adv10, 49.6054), 8.
        ↪ 4.7743))^4)), 14.9655))) <
            rank(ts_corr(rank(vwap), rank(volume), 5.07914)))"""

        return (rank(log(ts_product(rank(rank(ts_corr(vwap,
                                                    ts_sum(ts_mean(v, 10), 50),
        ↪ 8))
                                                    .pow(4)), 15)))
                .lt(rank(ts_corr(rank(vwap), rank(v), 5)))
                .mul(-1)
                .stack('ticker')
                .swaplevel())
```

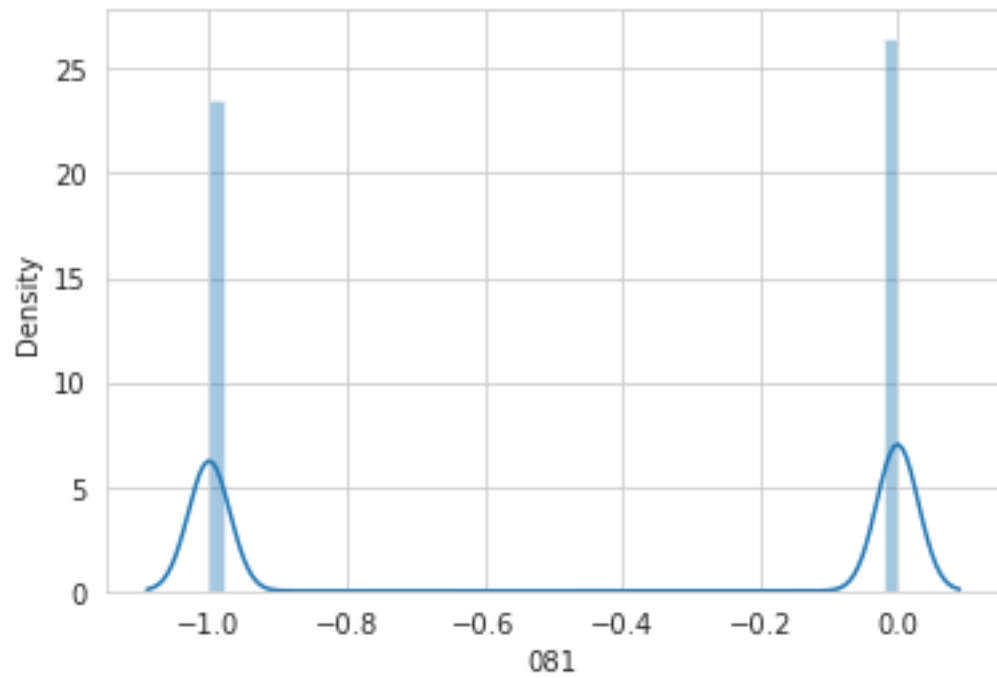
```
[178]: alpha = 81
```

```
[179]: %%time
        alphas[f'{alpha:03}'] = alpha081(v, vwap)
```

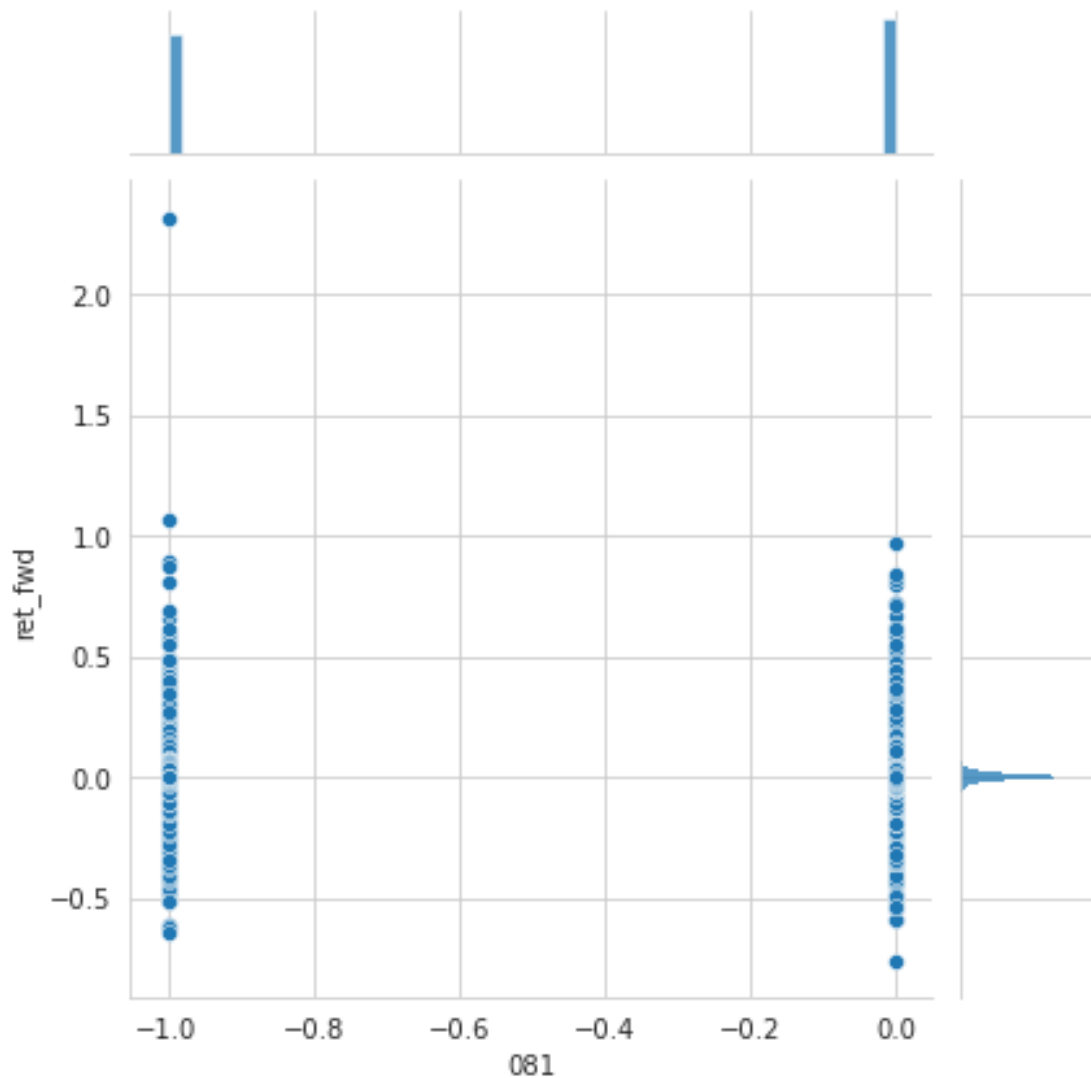
```
CPU times: user 1min 57s, sys: 593 ms, total: 1min 57s
Wall time: 1min 57s
```

```
[180]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[181]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[182]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[183]: # mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
# mi[alpha]
```

1.86 Alpha 082

```
-rank(ts_sum(returns, 10) / ts_sum(ts_sum(returns, 2), 3)) *
rank((returns * cap))
```

```
[184]: def alpha082(o, v, sector):
    """(min(rank(ts_weighted_mean(ts_delta(open, 1.46063), 14.8717)),
    ts_rank(ts_weighted_mean(ts_corr(IndNeutralize(volume, IndClass.
    ↪sector),
    ((open * 0.634196) +(open * (1 - 0.634196))), 17.4842), 6.92131), 13.
    ↪4283)) * -1)
```

```
"""
pass
```

1.87 Alpha 083

```
(rank(ts_lag((high - low) / ts_mean(close, 5), 2)) * rank(rank(volume)) /
  (((high - low) / ts_mean(close, 5) / (vwap - close)))
```

```
[185]: def alpha083(h, l, c):
        """(rank(ts_lag((high - low) / ts_mean(close, 5), 2)) * rank(rank(volume)) /
        ↪
        (((high - low) / ts_mean(close, 5) / (vwap - close)))
        """
        s = h.sub(l).div(ts_mean(c, 5))

        return (rank(rank(ts_lag(s, 2))
                        .mul(rank(rank(v)))
                        .div(s).div(vwap.sub(c).add(1e-3)))
                .stack('ticker')
                .swaplevel()
                .replace((np.inf, -np.inf), np.nan))
```

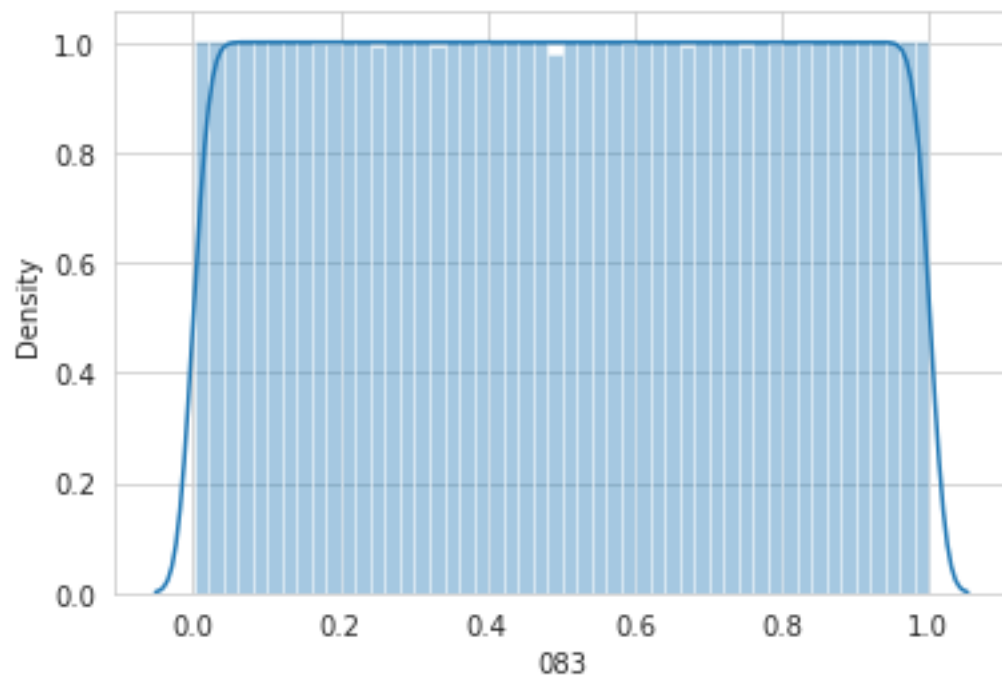
```
[186]: alpha = 83
```

```
[187]: %%time
        alphas[f'{alpha:03}'] = alpha083(h, l, c)
```

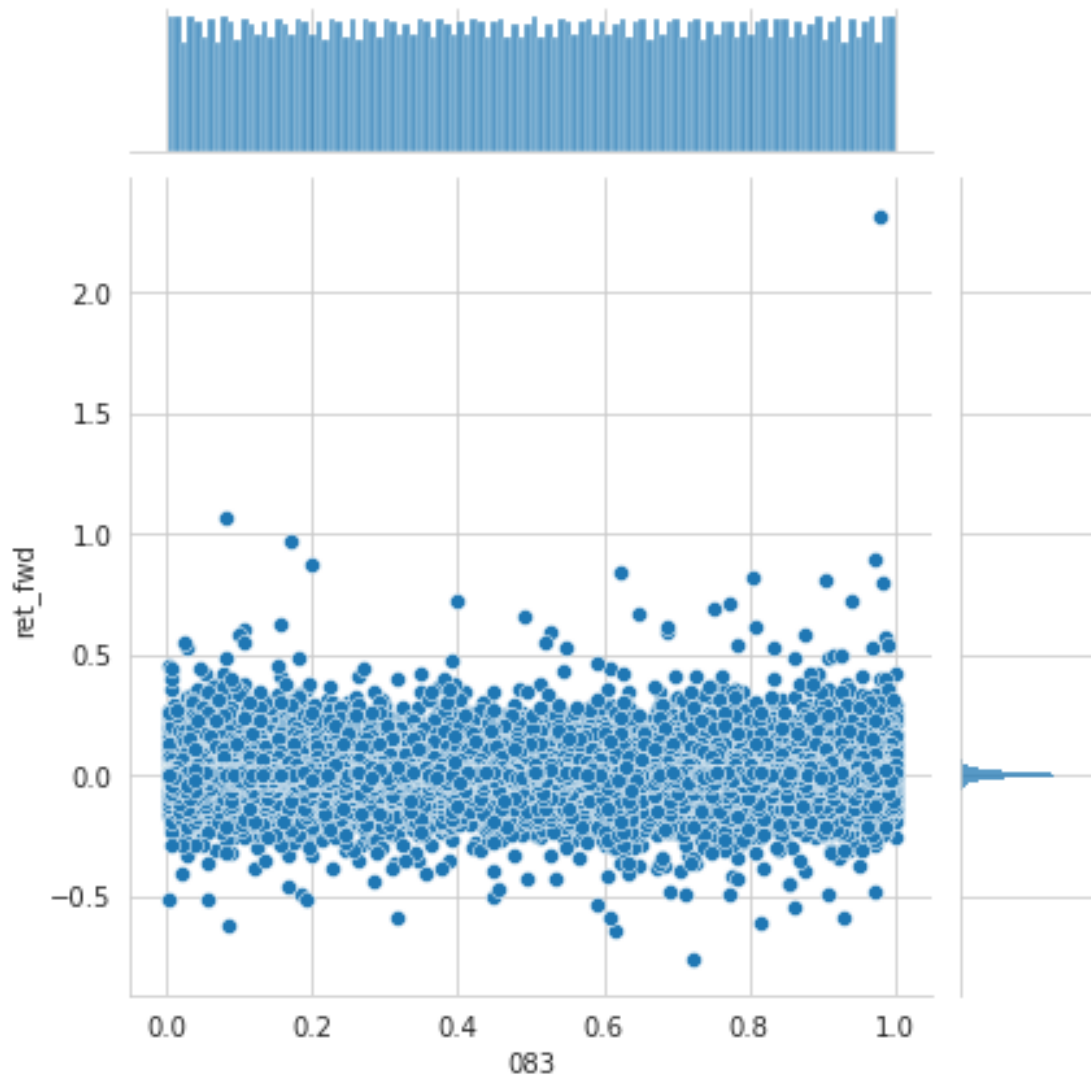
```
CPU times: user 2.8 s, sys: 16 ms, total: 2.81 s
Wall time: 2.75 s
```

```
[188]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[189]: sns.distplot(alphas[f'{alpha:03}']);
```

```
[190]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[191]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[191]: 0.0005100117425032025
```

1.88 Alpha 084

```
power(ts_rank((vwap - ts_max(vwap, 15.3217)), 20.7127),
      ts_delta(close,4.96796))
```

```
[192]: def alpha084(c, vwap):
        """power(ts_rank((vwap - ts_max(vwap, 15.3217)), 20.7127),
                  ts_delta(close,4.96796))"""
        return (rank(power(ts_rank(vwap.sub(ts_max(vwap, 15)), 20),
```

```

        ts_delta(c, 6)))
    .stack('ticker')
    .swaplevel()

```

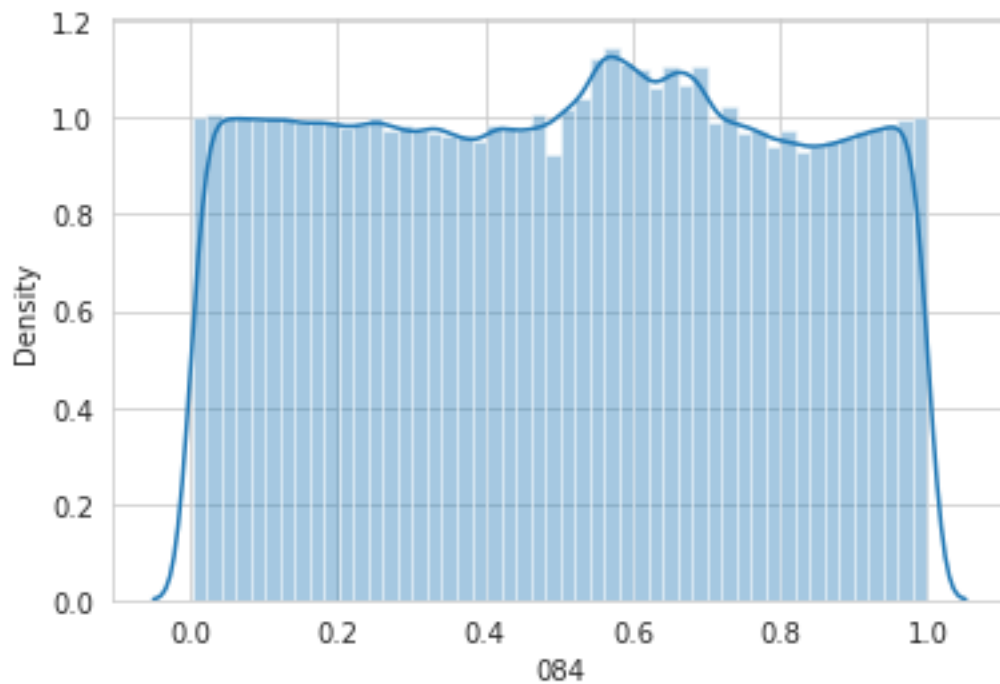
```
[193]: alpha = 84
```

```
[194]: %%time
alphas[f'{alpha:03}'] = alpha084(c, vwap)
```

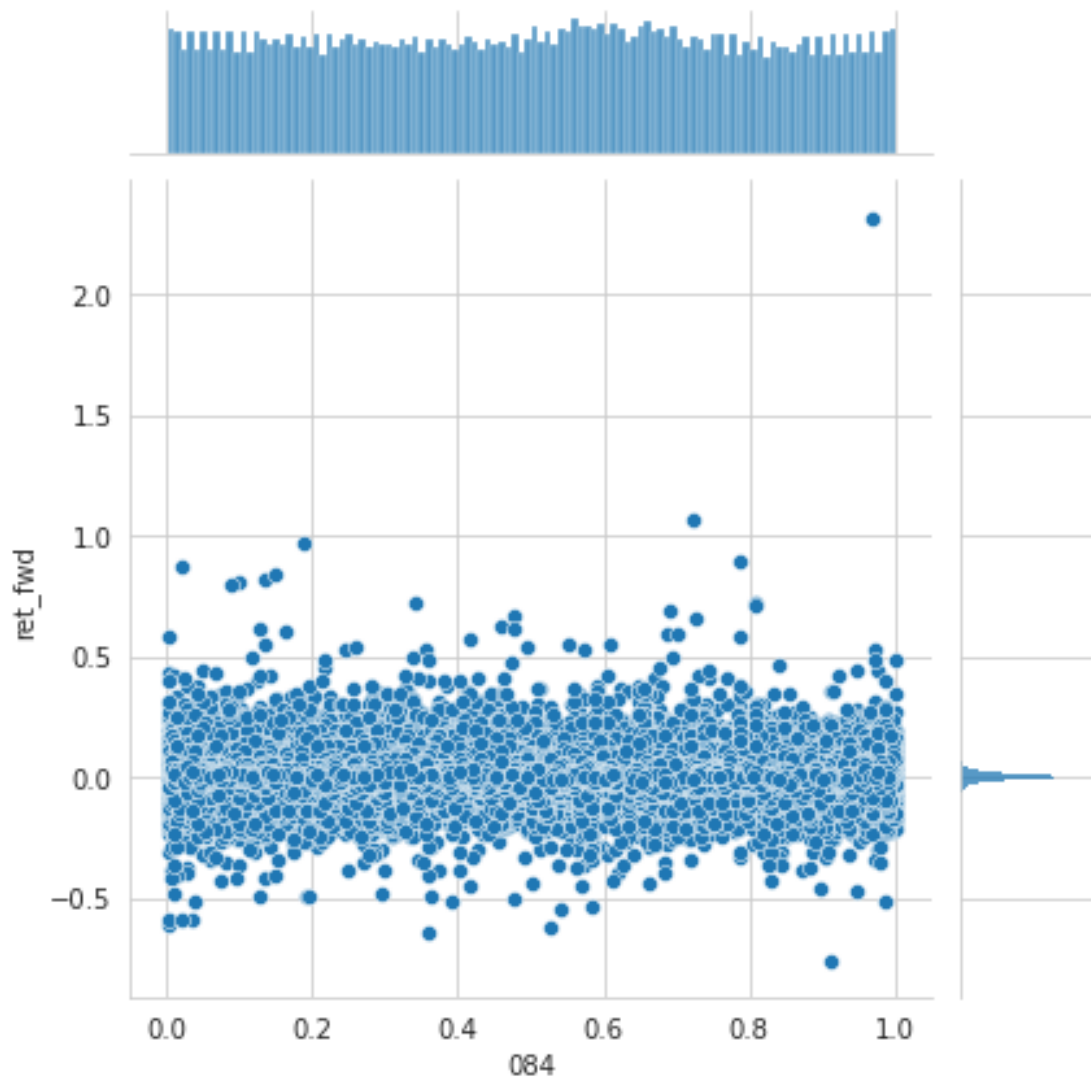
CPU times: user 3min 2s, sys: 47.7 ms, total: 3min 2s
 Wall time: 3min 2s

```
[195]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[196]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[197]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[198]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[198]: 0.008688359695486092
```

1.89 Alpha 085

```
power(rank(ts_corr(((high * 0.876703) + (close * (1 - 0.876703)))), adv30,9.61331)),
      rank(ts_corr(ts_rank(((high + low) / 2), 3.70596),
                    ts_rank(volume, 10.1595),7.11408)))
```

```
[199]: def alpha085(l, v):
        """power(rank(ts_corr(((high * 0.876703) + (close * (1 - 0.876703)))),
        ↪adv30,9.61331)),
```

```

        rank(ts_corr(ts_rank(((high + low) / 2), 3.70596),
                        ts_rank(volume, 10.1595), 7.11408)))
        """
w = 0.876703
return (rank(ts_corr(h.mul(w).add(c.mul(1 - w)), ts_mean(v, 30), 10))
        .pow(rank(ts_corr(ts_rank(h.add(1).div(2), 4),
                            ts_rank(v, 10), 7)))
        .stack('ticker')
        .swaplevel())

```

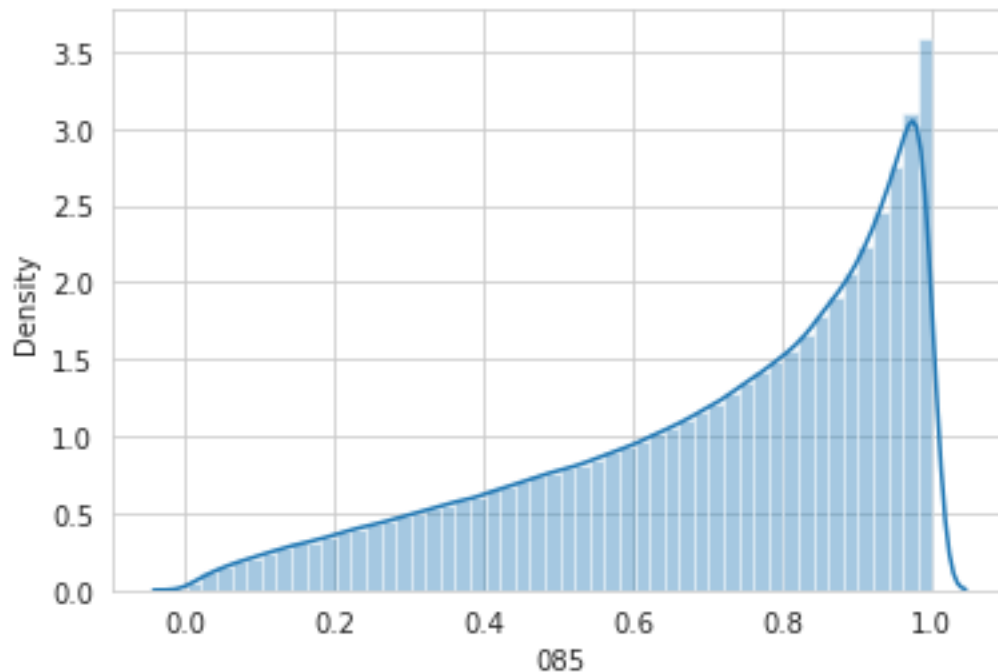
```
[200]: alpha = 85
```

```
[201]: %%time
alphas[f'{alpha:03}'] = alpha085(1, v)
```

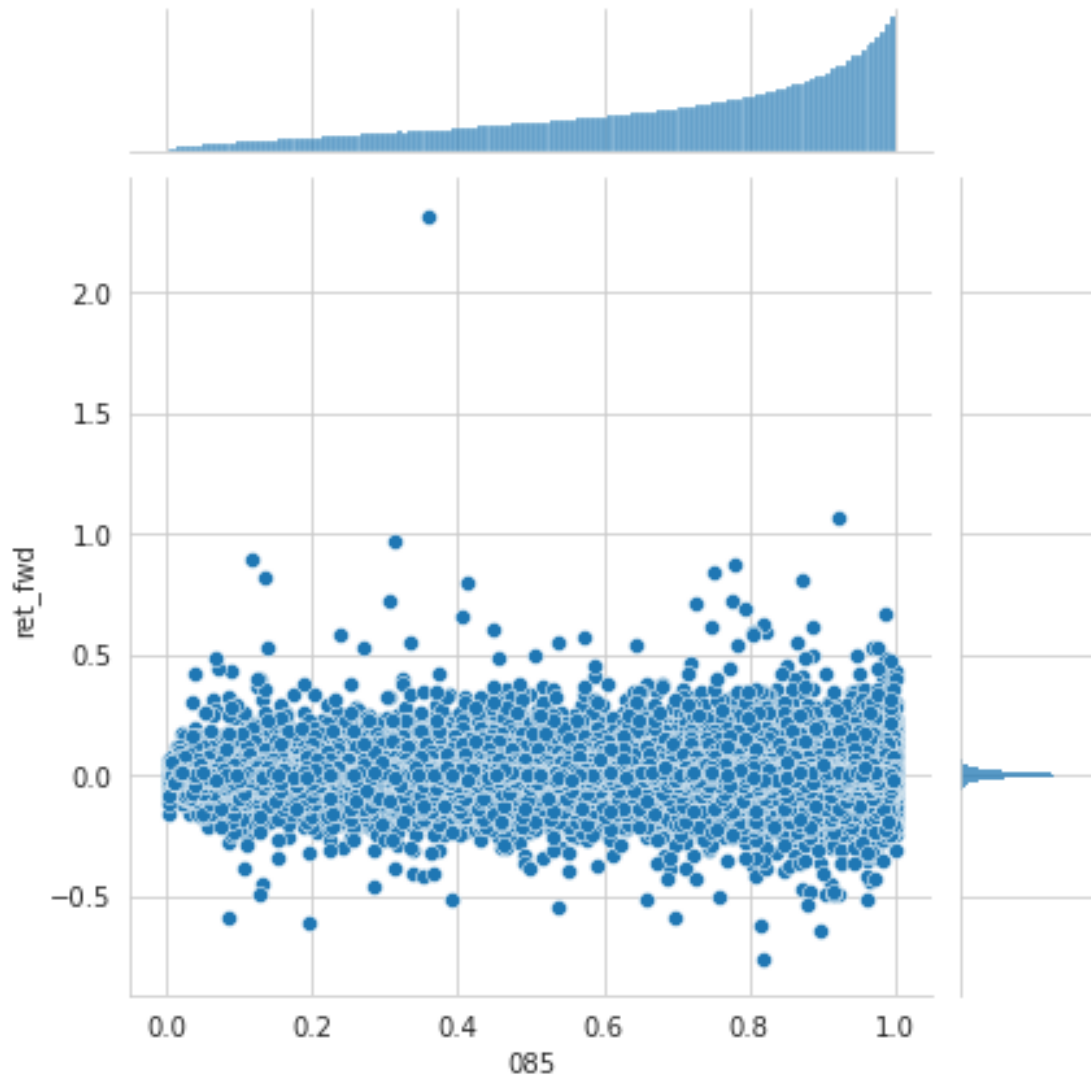
CPU times: user 6min 14s, sys: 123 ms, total: 6min 14s
Wall time: 6min 14s

```
[202]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[203]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[204]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[205]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[205]: 0.001101338196749957
```

1.90 Alpha 086

```
((ts_rank(ts_corr(close, ts_sum(adv20, 14.7444), 6.00049), 20.4195) <
rank(((open + close) - (vwap + open)))) * -1)
```

```
[206]: def alpha086(c, v, vwap):
        """((ts_rank(ts_corr(close, ts_sum(adv20, 14.7444), 6.00049), 20.4195) <
            rank(((open + close) - (vwap + open)))) * -1)
        """
```

```

return (ts_rank(ts_corr(c, ts_mean(ts_mean(v, 20), 15), 6), 20)
        .lt(rank(c.sub(vwap)))
        .mul(-1)
        .stack('ticker')
        .swaplevel())

```

```
[207]: alpha = 86
```

```

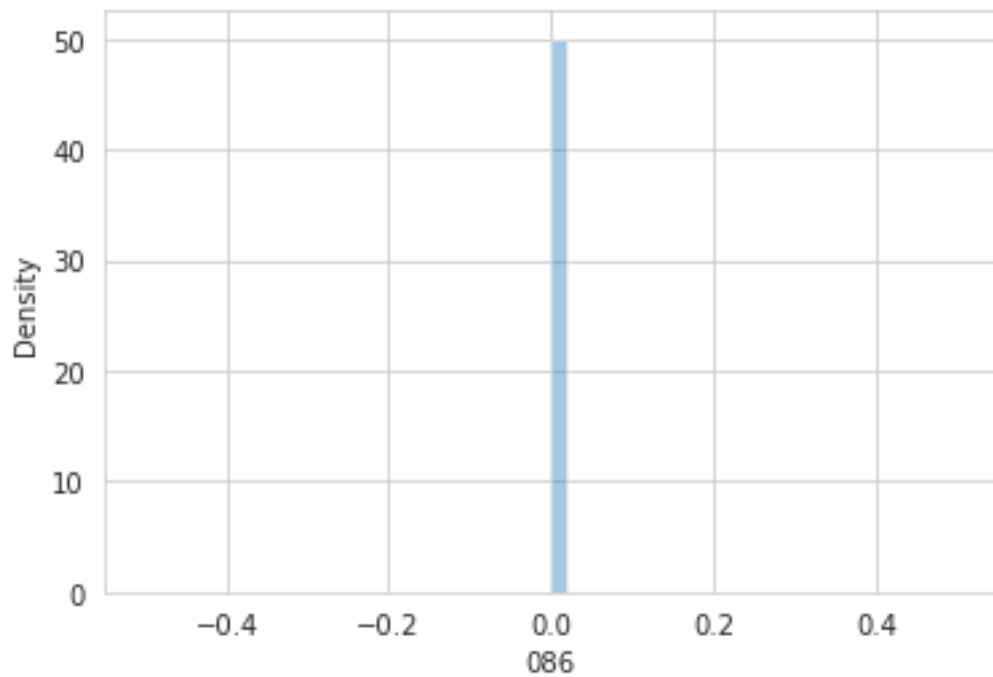
[208]: %%time
alphas[f'{alpha:03}'] = alpha086(c, v, vwap)

```

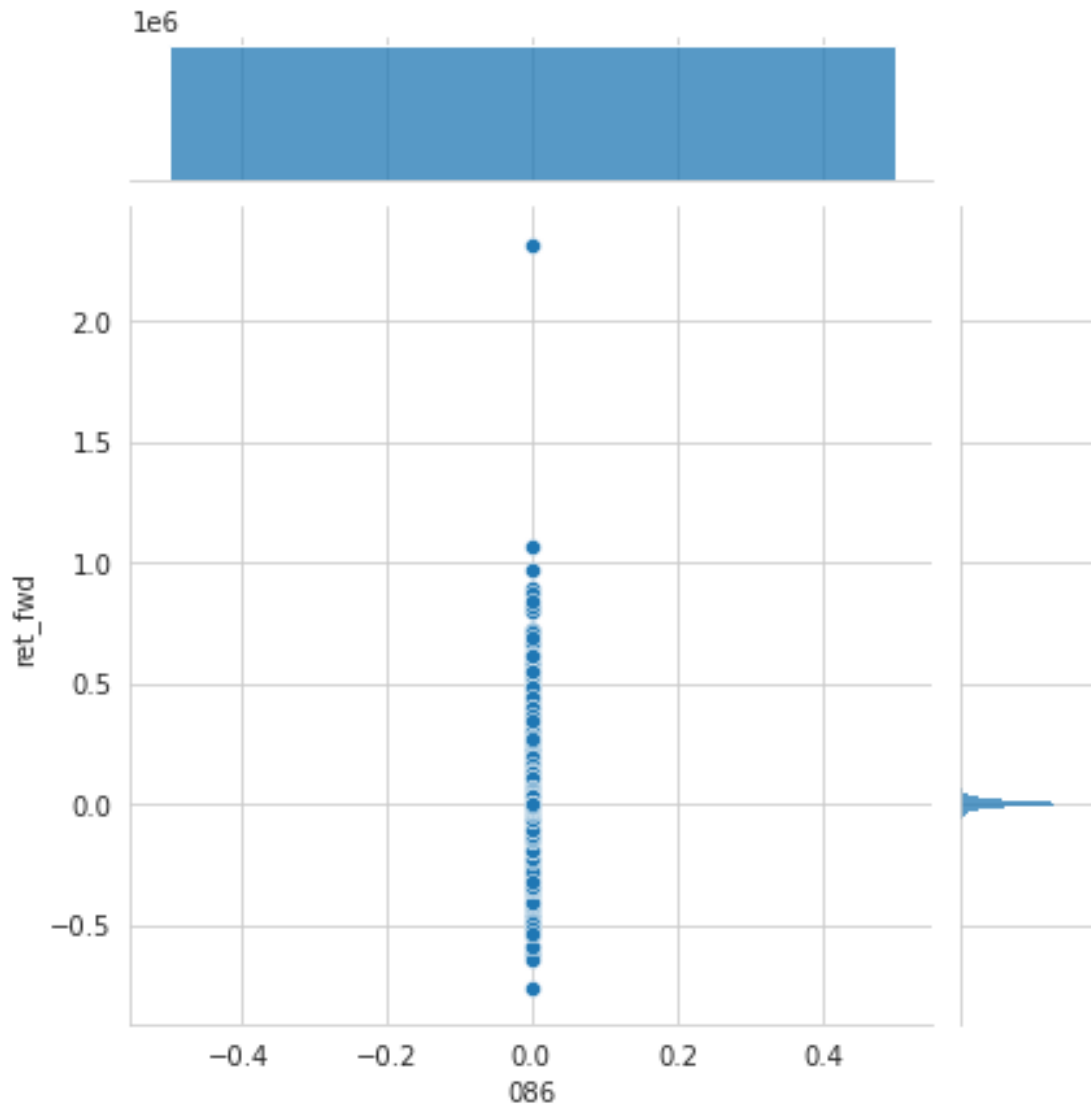
CPU times: user 3min 2s, sys: 152 ms, total: 3min 2s
Wall time: 3min 2s

```
[209]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[210]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[211]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[212]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[212]: 0.00020642450841457105
```

1.91 Alpha 087

```
(max(rank(ts_weighted_mean(ts_delta(((close * 0.369701) + (vwap * (1 -
0.369701)))),1.91233), 2.65461)),
close, 13.4132)), 4.89768), 14.4535)) * -1)
ts_rank(ts_weighted_mean(abs(ts_corr(IndNeutraliz
```

```
[213]:
```



```
def alpha087(c, vwap, industry):
    """(max(rank(ts_weighted_mean(ts_delta(((close * 0.369701) + (vwap * (1 - 0.
    ↪369701))),1.91233), 2.65461))),
        ts_rank(ts_weighted_mean(abs(ts_corr(IndNeutralize(adv81,IndClass.
    ↪industry), close, 13.4132))), 4.89768), 14.4535)) * -1)
    """
    pass
```

1.92 Alpha 088

```
min(rank(ts_weighted_mean(((rank(open) + rank(low)) - (rank(high) + rank(close))),8.06882)),
    ts_rank(ts_weighted_mean(ts_corr(ts_rank(close, 8.44728),
    ts_rank(adv60,20.6966), 8.01266), 6.65053), 2.61957))
```

```
[214]: def alpha088(o, h, l, c, v):
    """min(rank(ts_weighted_mean(((rank(open) + rank(low)) - (rank(high) +
    ↪rank(close))),8.06882)),
        ts_rank(ts_weighted_mean(ts_corr(ts_rank(close, 8.44728),
        ts_rank(adv60,20.6966), 8.01266), 6.65053), 2.61957))"""

    s1 = (rank(ts_weighted_mean(rank(o)
                                .add(rank(l))
                                .sub(rank(h))
                                .add(rank(c)), 8)))
    s2 = ts_rank(ts_weighted_mean(ts_corr(ts_rank(c, 8),
    ts_rank(ts_mean(v, 60), 20), 8), 6),
    ↪2)

    return (s1.where(s1 < s2, s2)
            .stack('ticker')
            .swaplevel())
```

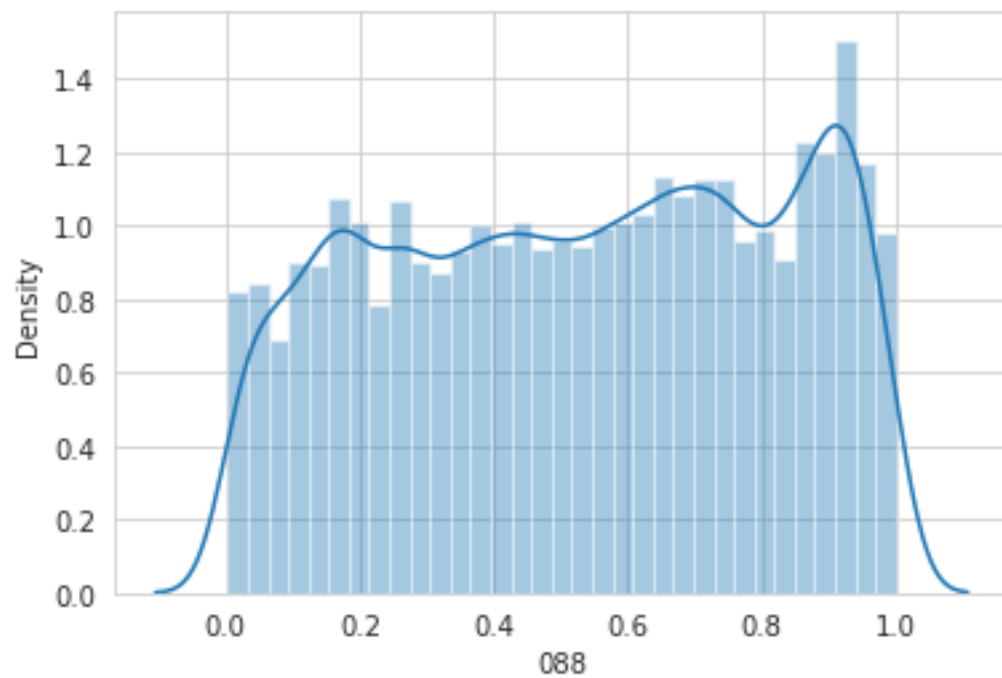
```
[215]: alpha = 88
```

```
[216]: %%time
alphas[f'{alpha:03}'] = alpha088(o, h, l, c, v)
```

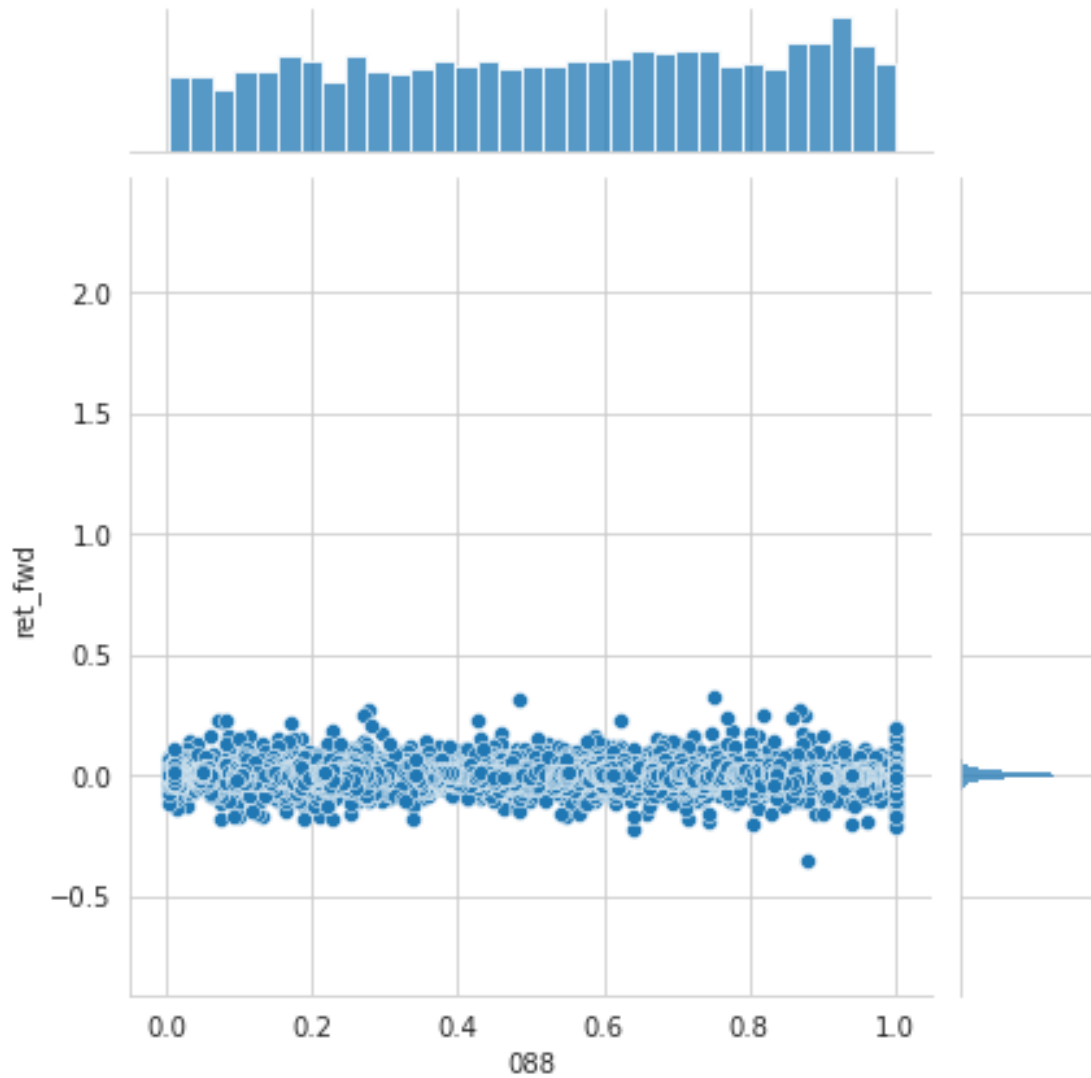
```
CPU times: user 6min 7s, sys: 79.6 ms, total: 6min 7s
Wall time: 6min 7s
```

```
[217]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[218]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[219]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[220]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'], n=30000)
mi[alpha]
```

[220]: 0.019146991360432075

1.93 Alpha 089

```
-rank(ts_sum(returns, 10) / ts_sum(ts_sum(returns, 2), 3)) *
rank((returns * cap))
```

```
[221]: def alpha089(l, v, vwap, industry):
        """(ts_rank(ts_weighted_mean(ts_corr(((low * 0.967285) +
        (low * (1 - 0.967285)))), adv10, 6.94279), 5.51607), 3.79744) -
```

```

        ts_rank(ts_weighted_mean(ts_delta(IndNeutralize(vwap, IndClass.
↪industry), 3.48158), 10.1466), 15.3012))
        """
    pass

```

1.94 Alpha 090

```

-rank(ts_sum(returns, 10) / ts_sum(ts_sum(returns, 2), 3)) *
    rank((returns * cap))

```

```

[222]: def alpha090(c, l, subindustry):
        """((rank((close - ts_max(close, 4.66719)))
            ^ts_rank(ts_corr(IndNeutralize(adv40, IndClass.subindustry), low, 5.
↪38375), 3.21856)) * -1)
        """
    pass

```

1.95 Alpha 091

```

((ts_rank(ts_weighted_mean(ts_weighted_mean(ts_corr(IndNeutralize(close, IndClass.industry), vo
    rank(ts_weighted_mean(ts_corr(vwap, adv30, 4.01303), 2.6809))) * -1)

```

```

[223]: def alpha091(v, vwap, industry):
        □
        ↪"""((ts_rank(ts_weighted_mean(ts_weighted_mean(ts_corr(IndNeutralize(close, IndClass.
↪industry), volume, 9.74928), 16.398), 3.83219), 4.8667) -
            rank(ts_weighted_mean(ts_corr(vwap, adv30, 4.01303), 2.6809))) * -1)
        """
    pass

```

1.96 Alpha 092

```

min(ts_rank(ts_weighted_mean((((high + low) / 2) + close) < (low + open)), 14.7221), 18.8683),
    ts_rank(ts_weighted_mean(ts_corr(rank(low), rank(adv30), 7.58555), 6.94024), 6.80584))

```

```

[224]: def alpha092(o, l, c, v):
        """min(ts_rank(ts_weighted_mean((((high + low) / 2) + close) < (low + □
↪open)), 14.7221), 18.8683),
            ts_rank(ts_weighted_mean(ts_corr(rank(low), rank(adv30), 7.58555), □
↪6.94024), 6.80584))
        """
        p1 = ts_rank(ts_weighted_mean(h.add(l).div(2).add(c).lt(l.add(o)), 15), 18)
        p2 = ts_rank(ts_weighted_mean(ts_corr(rank(l), rank(ts_mean(v, 30))), 7), □
↪6), 6)

        return (p1.where(p1<p2, p2)
                .stack('ticker'))

```

```
.swaplevel()
```

```
[225]: alpha = 92
```

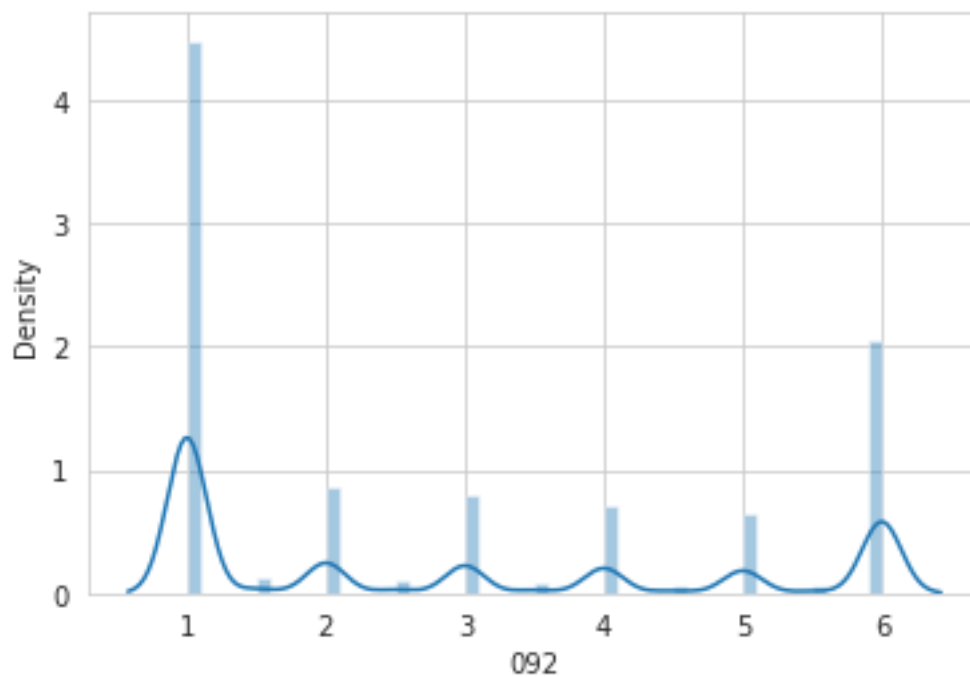
```
[226]: %%time  
alphas[f'{alpha:03}'] = alpha092(o, l, c, v)
```

CPU times: user 4min 33s, sys: 39.8 ms, total: 4min 34s

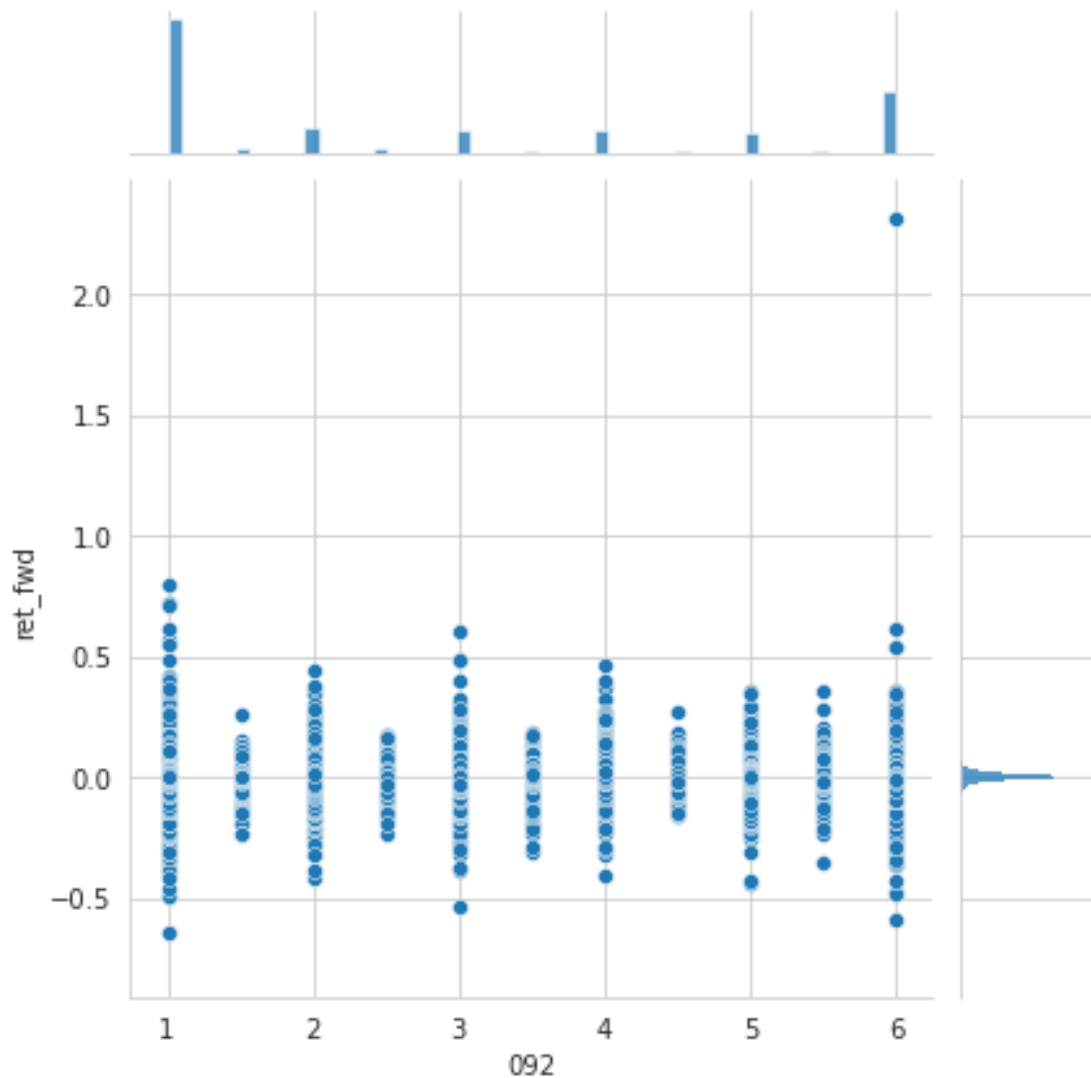
Wall time: 4min 33s

```
[227]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[228]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[229]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[230]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[230]: 0.0034230030791526644
```

1.97 Alpha 093

```
(ts_rank(ts_weighted_mean(ts_corr(IndNeutralize(vwap, IndClass.industry), adv81,17.4193), 19.848),
rank(ts_weighted_mean(ts_delta(((close * 0.524434) + (vwap * (1 -0.524434)))), 2.77377)
```

```
[231]: def alpha093(c, v, vwap, industry):
        """(ts_rank(ts_weighted_mean(ts_corr(IndNeutralize(vwap, IndClass.
        ↪industry), adv81,17.4193), 19.848), 7.54455) /
```

```

rank(ts_weighted_mean(ts_delta(((close * 0.524434) + (vwap * (1 - 0.
→524434))), 2.77377), 16.2664)))
"""
pass

```

1.98 Alpha 094

```

((rank((vwap - ts_min(vwap, 11.5783)))^ts_rank(ts_corr(ts_rank(vwap,19.6462),
ts_rank(adv60, 4.02992), 18.0926), 2.70756)) * -1)

```

```

[232]: def alpha094(v, vwap):
        """((rank((vwap - ts_min(vwap, 11.5783)))^ts_rank(ts_corr(ts_rank(vwap,19.
→6462),
        ts_rank(adv60, 4.02992), 18.0926), 2.70756)) * -1)
        """

        return (rank(vwap.sub(ts_min(vwap, 11)))
                .pow(ts_rank(ts_corr(ts_rank(vwap, 20),
                                    ts_rank(ts_mean(v, 60), 4), 18), 2))
                .mul(-1)
                .stack('ticker')
                .swaplevel())

```

```

[233]: alpha = 94

```

```

[234]: %%time
alphas[f'{alpha:03}'] = alpha094(v, vwap)

```

CPU times: user 8min 59s, sys: 164 ms, total: 8min 59s
Wall time: 9min

```

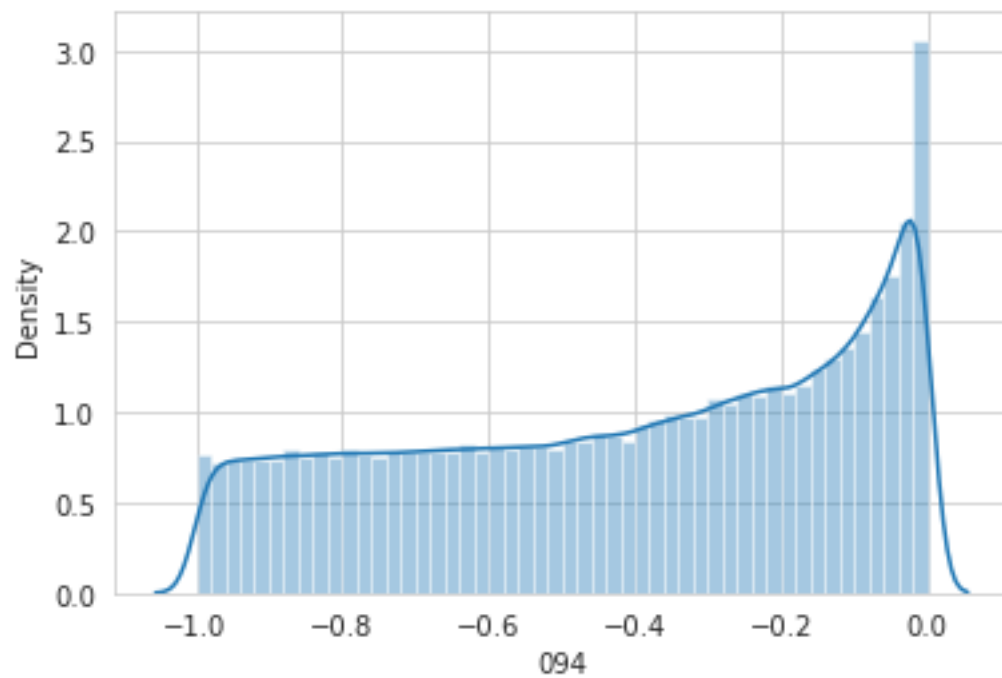
[235]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')

```

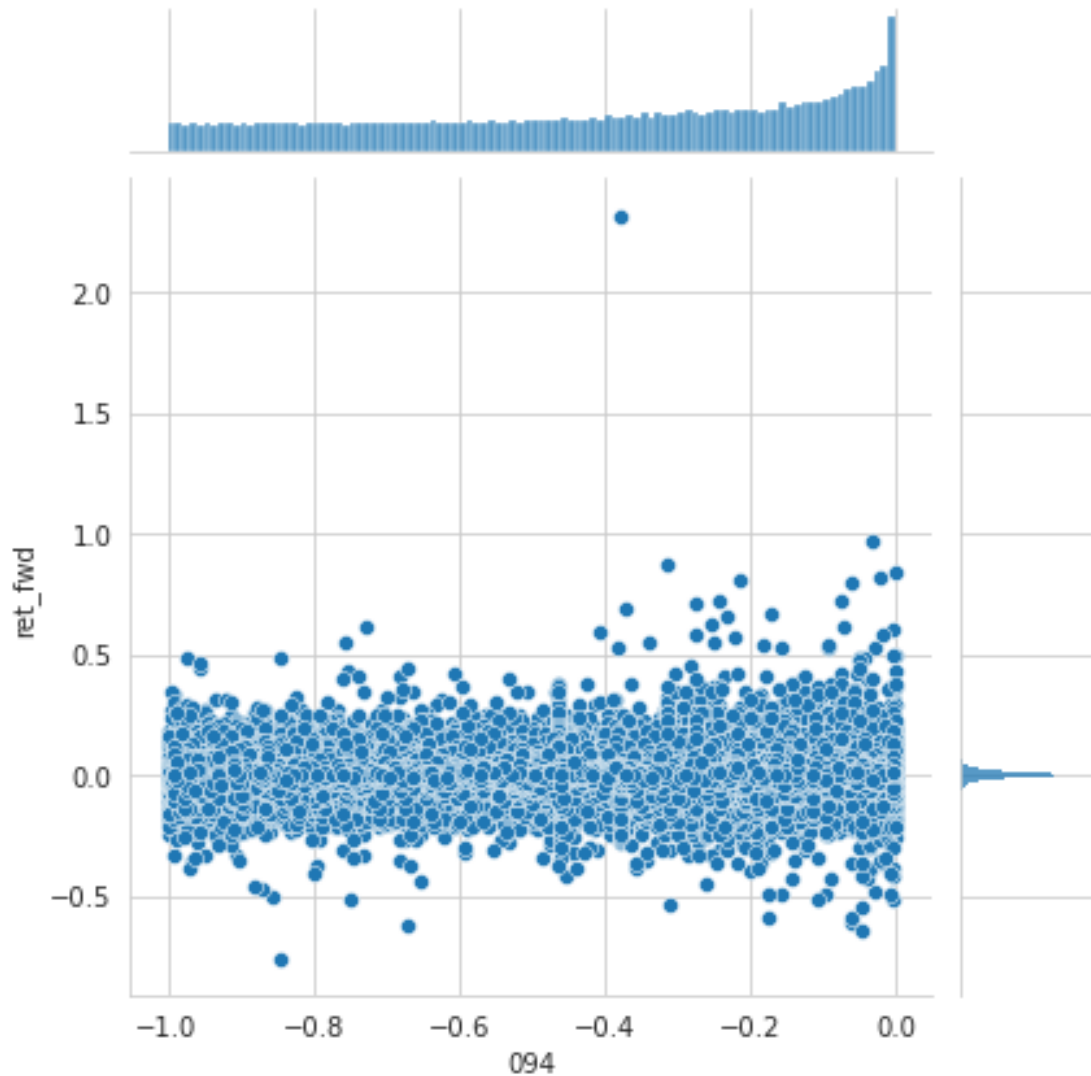
```

[236]: sns.distplot(alphas[f'{alpha:03}']);

```



```
[237]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```

```
[238]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[238]: 0.005173119205998944
```

1.99 Alpha 095

```
(rank((open - ts_min(open, 12.4105))) <
 ts_rank((rank(ts_corr(ts_sum(((high + low)/ 2), 19.1351),
 ts_sum(adv40, 19.1351), 12.8742))^5), 11.7584))
```

```
[239]: def alpha095(o, l, v):
        """(rank((open - ts_min(open, 12.4105))) <
```

```

        ts_rank((rank(ts_corr(ts_sum(((high + low)/ 2), 19.1351), ts_sum(adv40,
→19.1351), 12.8742)) ^5), 11.7584))
        """

    return (rank(o.sub(ts_min(o, 12)))
            .lt(ts_rank(rank(ts_corr(ts_mean(h.add(1).div(2), 19),
                                ts_sum(ts_mean(v, 40), 19), 13).pow(5)),
→12))
            .astype(int)
            .stack('ticker')
            .swaplevel())

```

```
[240]: alpha = 95
```

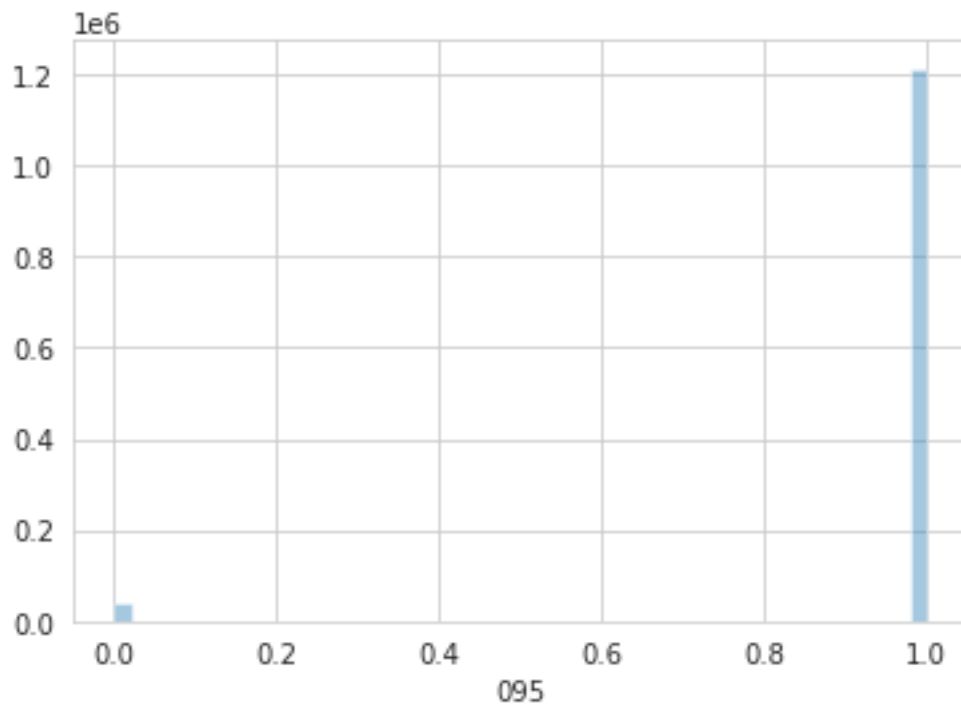
```
[241]: %%time
alphas[f'{alpha:03}'] = alpha095(o, l, v)
```

CPU times: user 3min 3s, sys: 43.9 ms, total: 3min 3s

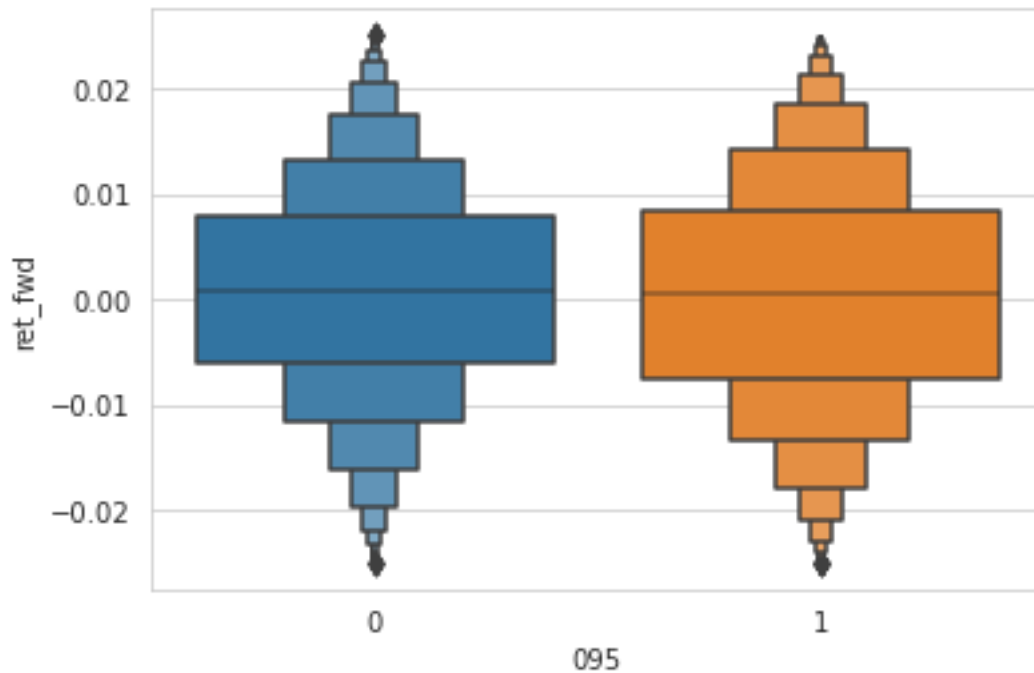
Wall time: 3min 3s

```
[242]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[243]: sns.distplot(alphas[f'{alpha:03}'], kde=False);
```



```
[244]: g = sns.boxenplot(x=f'{alpha:03}', y='ret_fwd', data=alphas[alphas.ret_fwd.
→between(-.025, .025)]);
```



```
[245]: alphas.groupby(alphas[f'{alpha:03}']).ret_fwd.describe()
```

```
[245]:
```

	count	mean	std	min	25%	50%	75%	\
095								
0	42212.0	0.001107	0.018945	-0.441048	-0.006959	0.000900	0.008912	
1	1212881.0	0.000563	0.025963	-0.757755	-0.009764	0.000481	0.010736	
	max							
095								
0	0.500000							
1	2.317073							

1.100 Alpha 096

```
(max(ts_rank(ts_weighted_mean(ts_corr(rank(vwap), rank(volume), 5.83878),4.16783), 8.38151),
    ts_rank(ts_weighted_mean(ts_argmax(ts_corr(ts_rank(close, 7.45404), ts_rank(adv60, 4.1
```

```
[246]: def alpha096(c, v, vwap):
    """(max(ts_rank(ts_weighted_mean(ts_corr(rank(vwap), rank(volume), 5.
→83878),4.16783), 8.38151),
        ts_rank(ts_weighted_mean(ts_argmax(ts_corr(ts_rank(close, 7.45404),
→ts_rank(adv60, 4.13242), 3.65459), 12.6556), 14.0365), 13.4143)) * -1)"""
```

```

s1 = ts_rank(ts_weighted_mean(ts_corr(rank(vwap), rank(v), 10), 4), 8)
s2 = ts_rank(ts_weighted_mean(ts_argmax(ts_corr(ts_rank(c, 7),
                                                ts_rank(ts_mean(v, 60),
→10), 10), 12), 14), 13)
    return (s1.where(s1 > s2, s2)
            .mul(-1)
            .stack('ticker')
            .swaplevel())

```

```
[247]: alpha = 96
```

```

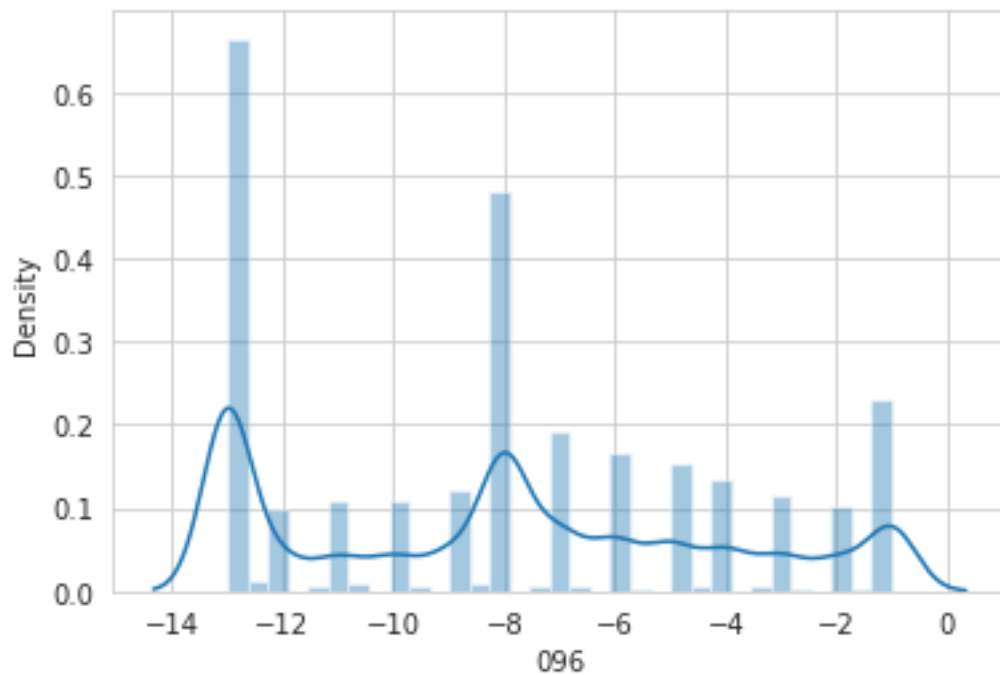
[248]: %%time
alphas[f'{alpha:03}'] = alpha096(c, v, vwap)

```

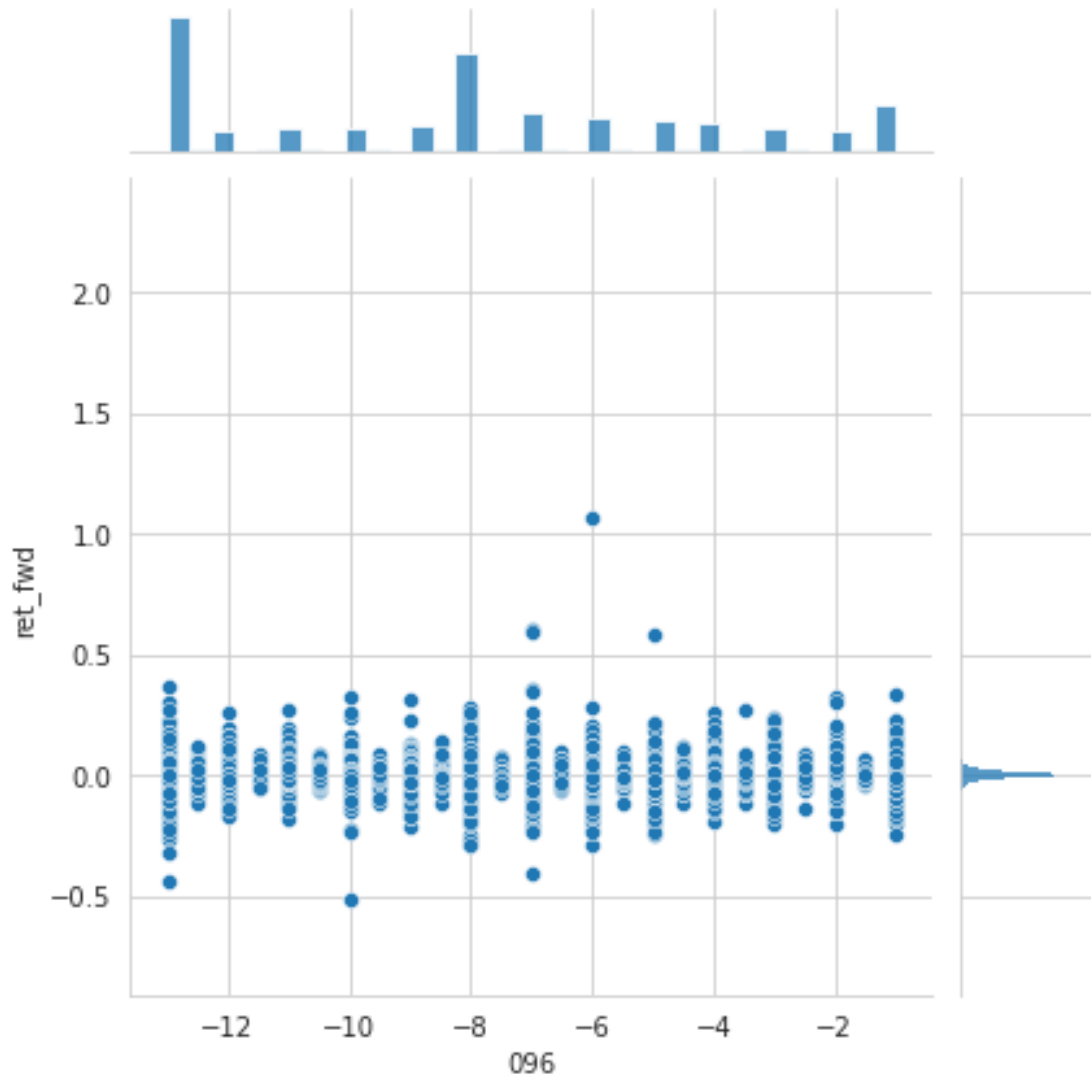
CPU times: user 10min 4s, sys: 432 ms, total: 10min 4s
 Wall time: 10min 6s

```
[249]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[250]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[251]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas)
```



1.101 Alpha 097

```
-rank(ts_sum(returns, 10) / ts_sum(ts_sum(returns, 2), 3)) *
rank((returns * cap))
```

```
[253]: def alpha097(1):
        """((rank(ts_weighted_mean(ts_delta(IndNeutralize(((low * 0.721001) +
        (vwap * (1 - 0.721001))), IndClass.industry), 3.3705), 20.4523)) -
        ts_rank(ts_weighted_mean(ts_rank(ts_corr(Ts_Rank(low, 7.87871),
        ts_rank(adv60, 17.255), 4.97547), 18.5925), 15.7152), 6.71659)) * -1)
        """
        pass
```

1.102 Alpha 098

```
(rank(ts_weighted_mean(ts_corr(vwap, ts_sum(adv5, 26.4719), 4.58418), 7.18088)) -  
    rank(ts_weighted_mean(ts_tank(ts_argmin(ts_corr(rank(open),  
    rank(adv15), 20.8187), 8.62571), 6.95668), 8.07206)))
```

```
[254]: def alpha098(o, v, vwap):  
        """(rank(ts_weighted_mean(ts_corr(vwap, ts_sum(adv5, 26.4719), 4.58418), 7.  
        ↪18088)) -  
            rank(ts_weighted_mean(ts_tank(ts_argmin(ts_corr(rank(open),  
            rank(adv15), 20.8187), 8.62571), 6.95668), 8.07206)))  
        """  
        adv5 = ts_mean(v, 5)  
        adv15 = ts_mean(v, 15)  
        return (rank(ts_weighted_mean(ts_corr(vwap, ts_mean(adv5, 26), 4), 7))  
                .sub(rank(ts_weighted_mean(ts_rank(ts_argmin(ts_corr(rank(o),  
                rank(adv15),  
        ↪20), 8), 6))))  
                .stack('ticker')  
                .swaplevel())
```

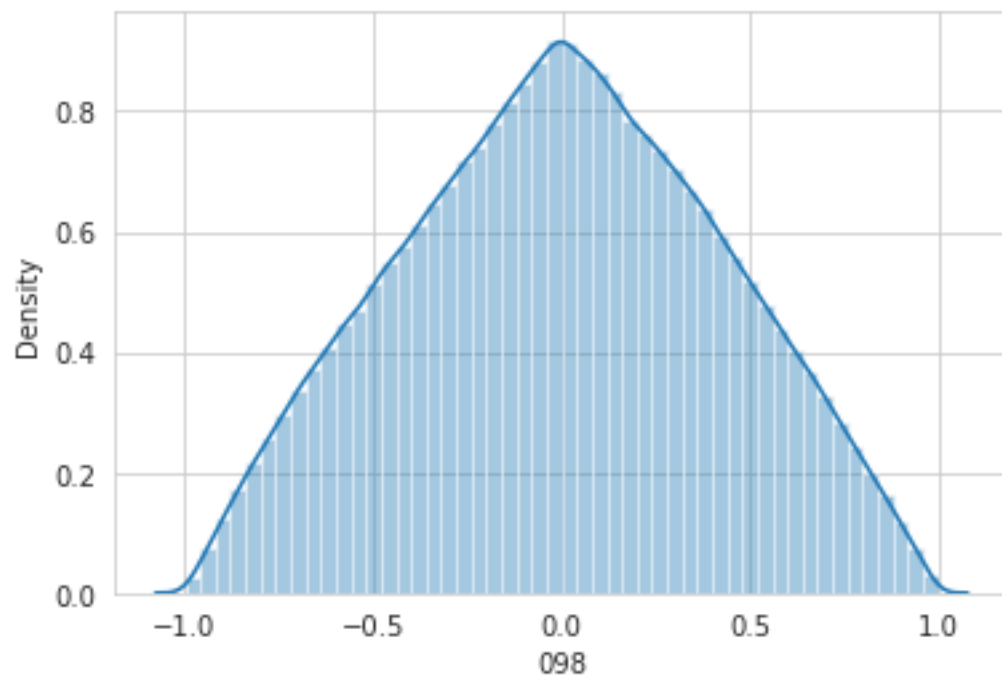
```
[255]: alpha = 98
```

```
[256]: %%time  
alphas[f'{alpha:03}'] = alpha098(o, v, vwap)
```

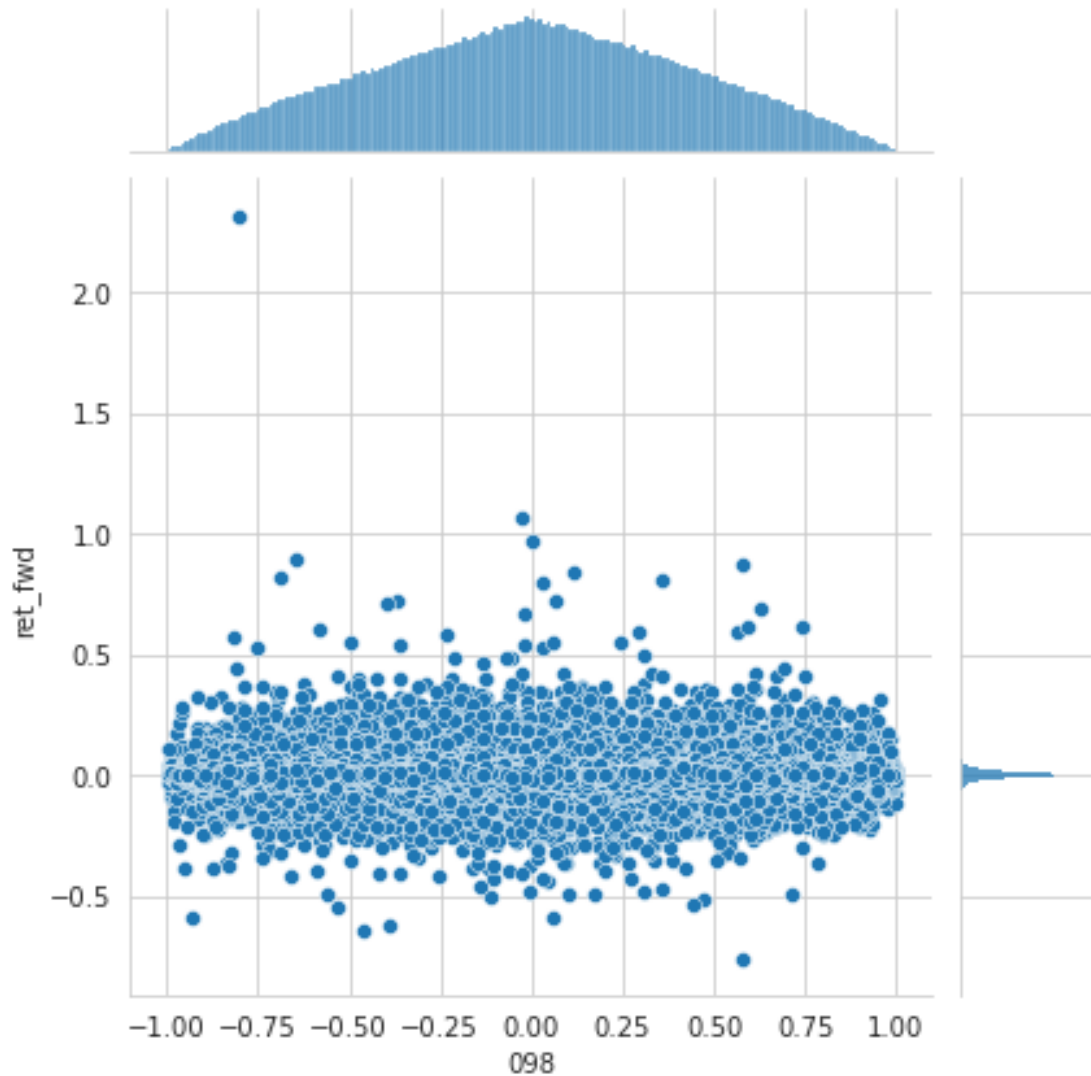
CPU times: user 4min 54s, sys: 389 ms, total: 4min 54s
Wall time: 4min 54s

```
[257]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[258]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[259]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[260]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

[260]: 0

1.103 Alpha 099

```
((rank(ts_corr(ts_sum(((high + low) / 2), 19.8975),
                    ts_sum(adv60, 19.8975), 8.8136)) <
rank(ts_corr(low, volume, 6.28259))) * -1)
```

```
[261]: def alpha099(l, v):
        """((rank(ts_corr(ts_sum(((high + low) / 2), 19.8975),
                    ts_sum(adv60, 19.8975), 8.8136)) <
```



```

        rank(ts_corr(low, volume, 6.28259))) * -1)"""

    return ((rank(ts_corr(ts_sum((h.add(1).div(2)), 19),
                               ts_sum(ts_mean(v, 60), 19), 8))
                .lt(rank(ts_corr(l, v, 6)))
                .mul(-1))
            .stack('ticker')
            .swaplevel())

```

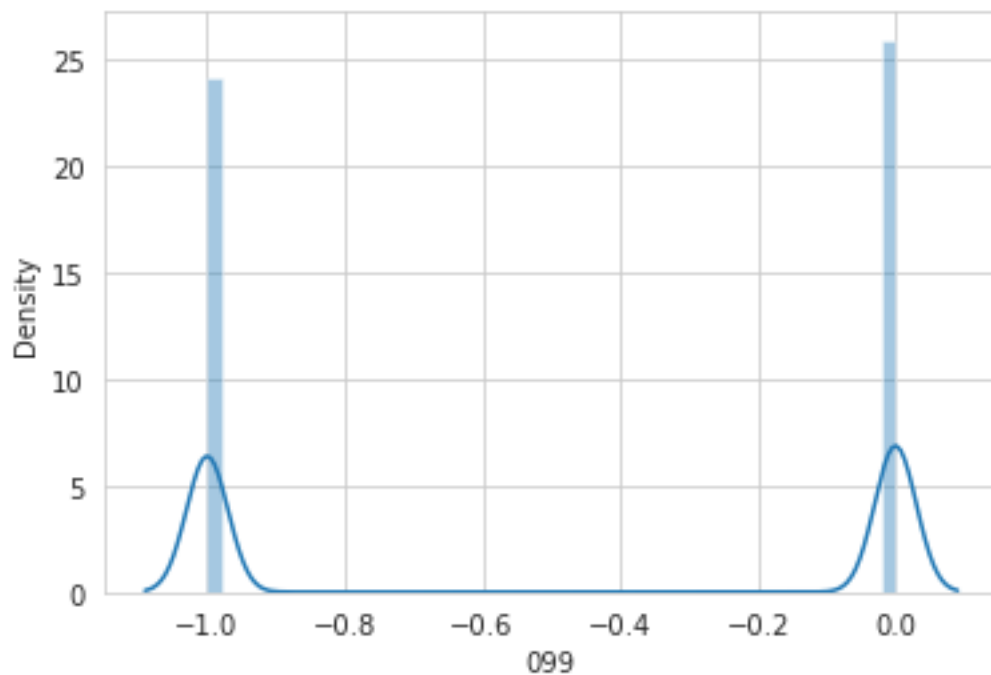
```
[262]: alpha = 99
```

```
[263]: %%time
alphas[f'{alpha:03}'] = alpha099(l, v)
```

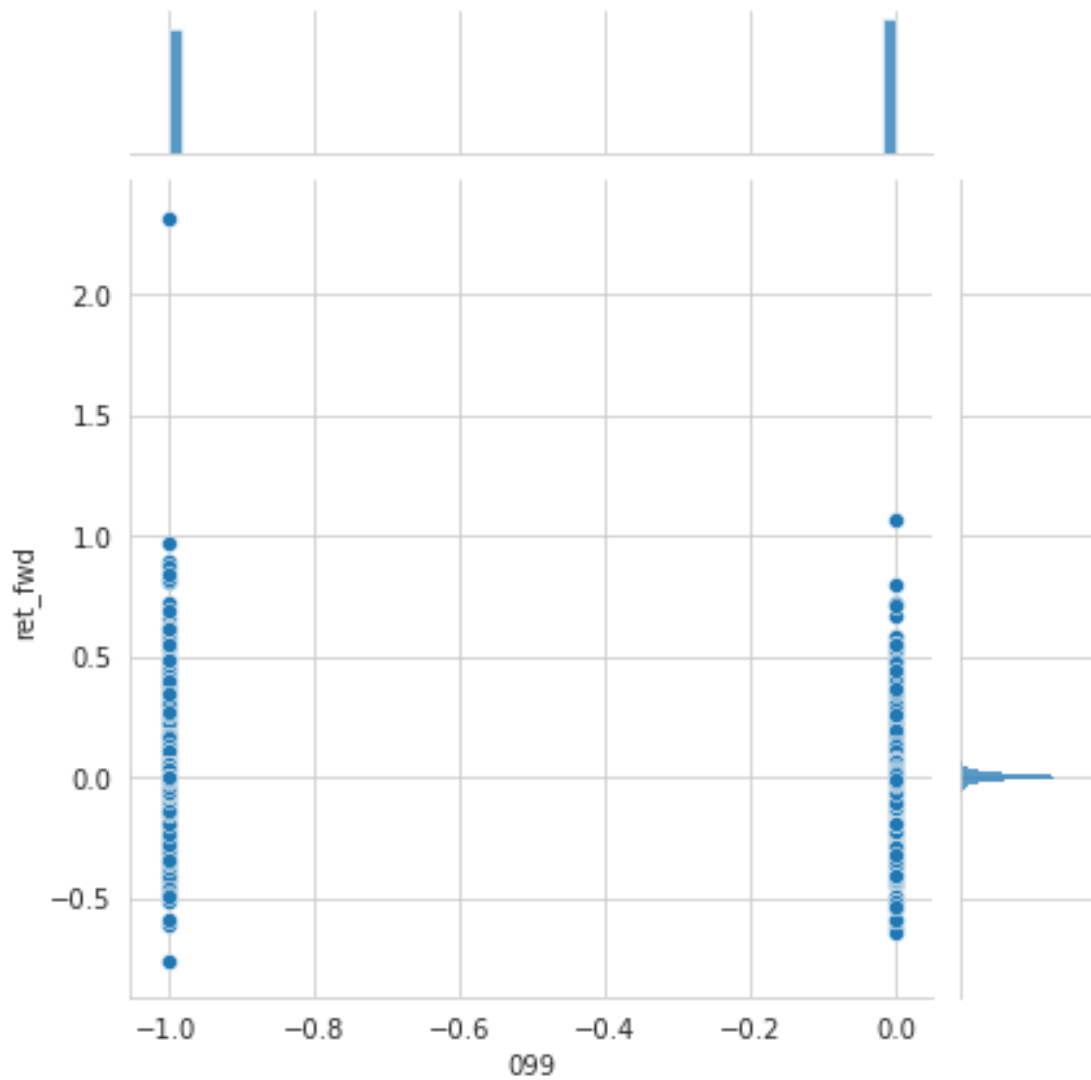
CPU times: user 4.53 s, sys: 21 μ s, total: 4.53 s
 Wall time: 4.44 s

```
[264]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[265]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[266]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[267]: alphas.groupby(alphas['{alpha:03}']).ret_fwd.describe()
```

```
[267]:
```

	count	mean	std	min	25%	50%	75%	\
099								
-1	604583.0	0.000537	0.026239	-0.757755	-0.009783	0.000448	0.010714	
0	650510.0	0.000622	0.025303	-0.643066	-0.009524	0.000547	0.010625	

	max
099	
-1	2.317073
0	1.061026

1.104 Alpha 100

```
-rank(ts_sum(returns, 10) / ts_sum(ts_sum(returns, 2), 3)) *  
    rank((returns * cap))
```

```
[268]: def alpha100(r, cap):  
        """(0 - (1 * (((1.5 * scale(indneutralize(  
            indneutralize(rank((((close - low) - (high - close)) / (high -  
            ↪ low)) * volume))),  
                                IndClass.subindustry), IndClass.subindustry)))  
        ↪ -  
        scale(indneutralize((ts_corr(close, rank(adv20), 5) - rank(ts_argmin(close,  
            ↪ 30))), IndClass.subindustry))) * (volume / adv20))))  
        """  
        pass
```

1.105 Alpha 101

```
-ts_max(rank(ts_corr(rank(volume), rank(vwap), 5)), 5)
```

```
[269]: def alpha101(o, h, l, c):  
        """((close - open) / ((high - low) + .001))"""  
        return (c.sub(o).div(h.sub(l).add(1e-3))  
                .stack('ticker')  
                .swaplevel())
```

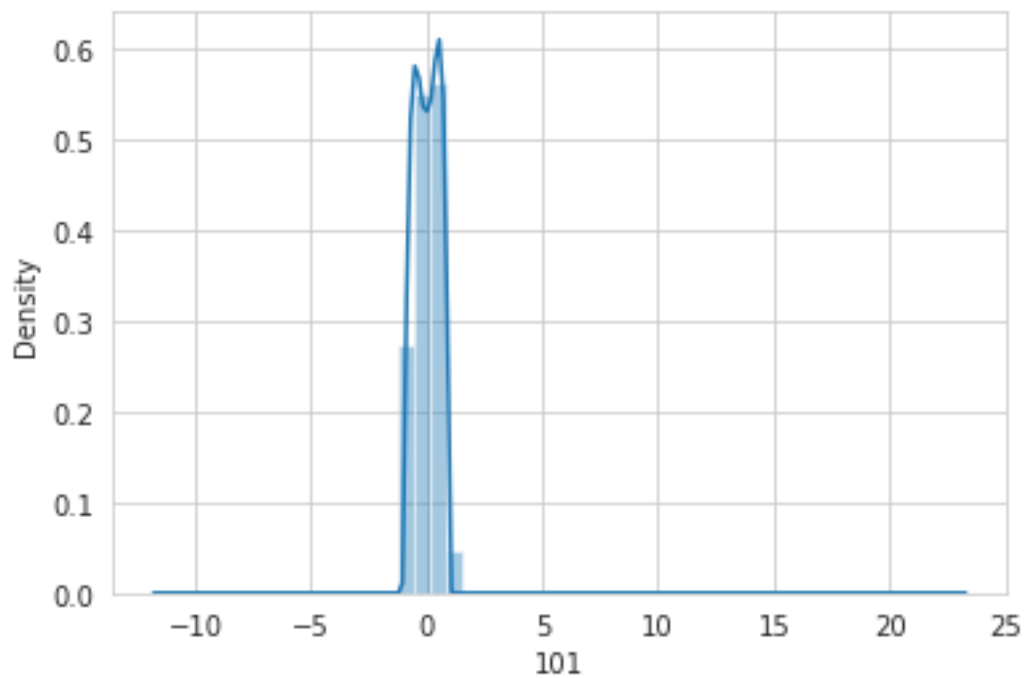
```
[270]: alpha = 101
```

```
[271]: %%time  
alphas[f'{alpha:03}'] = alpha101(o, h, l, c)
```

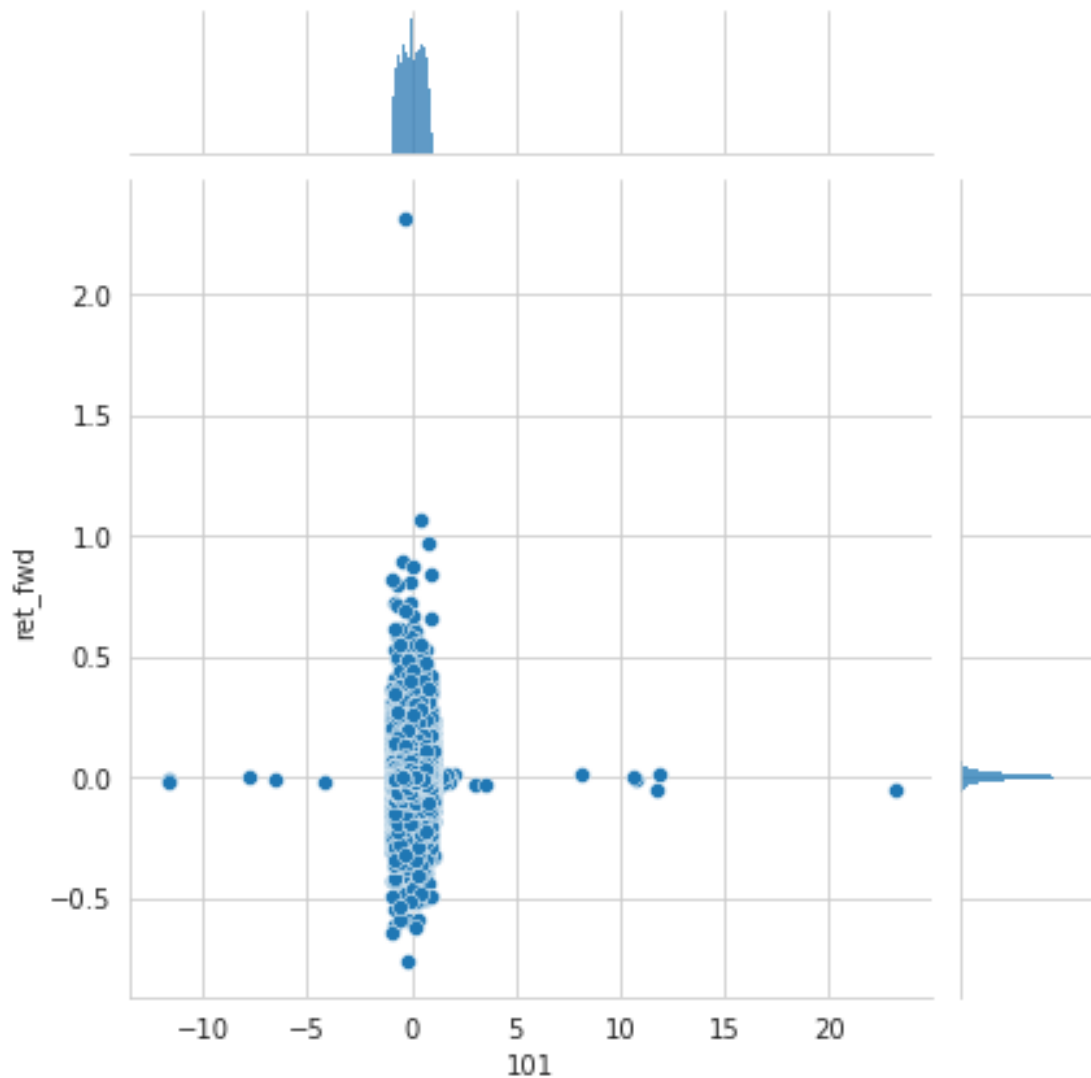
```
CPU times: user 1.89 s, sys: 12 ms, total: 1.9 s  
Wall time: 1.87 s
```

```
[272]: alphas[f'{alpha:03}'].to_hdf('alphas.h5', f'alphas/{alpha:03}')
```

```
[273]: sns.distplot(alphas[f'{alpha:03}']);
```



```
[274]: g = sns.jointplot(x=f'{alpha:03}', y='ret_fwd', data=alphas);
```



```
[275]: mi[alpha] = get_mutual_info_score(alphas.ret_fwd, alphas[f'{alpha:03}'])
mi[alpha]
```

```
[275]: 0.0008757897861757513
```

1.106 Store results

```
[276]: alphas = []
with pd.HDFStore('alphas.h5') as store:
    keys = [k[1:] for k in store.keys()]
    for key in keys:
        i = int(key.split('/')[-1])
        alphas.append(store[key].to_frame(i))
```

```
alphas = pd.concat(alphas, axis=1)
```

```
[277]: alphas.info(null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>  
MultiIndex: 1255093 entries, ('A', Timestamp('2007-01-04 00:00:00')) to ('ZION',  
Timestamp('2016-12-29 00:00:00'))  
Data columns (total 82 columns):
```

#	Column	Non-Null Count	Dtype
0	1	1243849 non-null	float64
1	2	1243080 non-null	float64
2	3	1227126 non-null	float64
3	4	1250757 non-null	float64
4	5	1247733 non-null	float64
5	6	1250204 non-null	float64
6	7	1241633 non-null	float64
7	8	1247172 non-null	float64
8	9	1247548 non-null	float64
9	10	1247548 non-null	float64
10	11	1245756 non-null	float64
11	12	1247548 non-null	float64
12	13	1243849 non-null	float64
13	14	1250204 non-null	float64
14	15	1048657 non-null	float64
15	16	1252899 non-null	float64
16	17	1240819 non-null	float64
17	18	1240862 non-null	float64
18	19	1127248 non-null	float64
19	20	1250162 non-null	float64
20	21	1255093 non-null	int64
21	22	1235676 non-null	float64
22	23	1255093 non-null	float64
23	24	1247006 non-null	float64
24	25	1244833 non-null	float64
25	26	1163949 non-null	float64
26	27	1255093 non-null	float64
27	28	1242843 non-null	float64
28	29	1240358 non-null	float64
29	30	1243877 non-null	float64
30	31	2368 non-null	float64
31	32	1126346 non-null	float64
32	33	1250702 non-null	float64
33	34	1246900 non-null	float64
34	35	1237517 non-null	float64
35	36	1144017 non-null	float64
36	37	1143512 non-null	float64

37	38	1247733	non-null	float64
38	39	1123375	non-null	float64
39	40	1250204	non-null	float64
40	41	1255093	non-null	float64
41	42	1250702	non-null	float64
42	43	1235049	non-null	float64
43	44	1249557	non-null	float64
44	45	1226678	non-null	float64
45	46	1247676	non-null	float64
46	47	1244823	non-null	float64
47	49	1254955	non-null	float64
48	50	1181356	non-null	float64
49	51	1254967	non-null	float64
50	52	1132298	non-null	float64
51	53	1244431	non-null	float64
52	54	1255093	non-null	float64
53	55	1243526	non-null	float64
54	57	1229293	non-null	float64
55	60	1250224	non-null	float64
56	61	1255093	non-null	int64
57	62	1255093	non-null	int64
58	64	1255093	non-null	int64
59	65	1255093	non-null	int64
60	66	1240293	non-null	float64
61	68	1255093	non-null	int64
62	71	1239293	non-null	float64
63	72	6612	non-null	float64
64	73	1238793	non-null	float64
65	74	1255093	non-null	int64
66	75	1255093	non-null	int64
67	77	1206268	non-null	float64
68	78	1191505	non-null	float64
69	81	1255093	non-null	int64
70	83	1251765	non-null	float64
71	84	1237712	non-null	float64
72	85	1219026	non-null	float64
73	86	1255093	non-null	int64
74	88	33449	non-null	float64
75	92	609037	non-null	float64
76	94	1182264	non-null	float64
77	95	1255093	non-null	int64
78	96	53769	non-null	float64
79	98	1068846	non-null	float64
80	99	1255093	non-null	int64
81	101	1255093	non-null	float64

dtypes: float64(70), int64(12)

memory usage: 790.8+ MB

```
[278]: alphas.to_hdf('data.h5', 'factors/formulaic')
```