Operator Submission inquire

`Yunbo39_SaurabhAgarwal_Wenshengli_Chenyuwu_pick_out_the_exact_EA_jumpnum(data_of_origin, days)`

```
def pick_out_the_exact_EA_jumpnum(data_of_origin, days):
    """
    Args:
    data_of_origin: the data you want to extract.
    [list of arrays shaped [shape(n,d)]]
    days the number of days after the earnings announcement day
     when you want to extract data.
     [int]
    Returns:
    Earning_jump_fill: the data assigns each non-
     earnings announcement day as the data at the
    time of the previous earnings announcement day.
    """
    delay = 0
    valid_mask = trading_days_til_next_ann==1
    valid_effect_mask = np.zeros_like(valid_mask, dtype=np.float32)
    for si, di in zip(*np.where((valid_mask == 1) & (close != np.nan))):

        # skip if we are unable to get the target
        if di + delay >= numdates:
            continue

        time = fs_next_ann_time[si, di]

        if time <= 1500:
            target_di = di
        elif time > 1500 and di + 1 < numdates:
            target_di = di+1
        if valid_effect_mask[si, target_di] != 1:
            valid_effect_mask[si, target_di] = 1

    bet_days_mask = valid_effect_mask==1
    bet_days_mask = op.at_nan2zero(bet_days_mask*1.0)
    flatten_mask = (bet_days_mask==1.0)

    earning_jump = np.where(ts_delay(flatten_mask*1.0, days+1) == 1, data_of_origin, np.nan)
    earning_jump_fill = ts_fill(earning_jump)
    return earning_jump_fill
```

This operator correctly extracts data around the earnings announcement day, and assigns the data of each non-earnings announcement day as the data at the time of the previous earnings announcement day. It uses the data around the previous earnings announcement day as a highlight for daily trading.

Args:

data_of_origin: the data you want to extract.
`[list of arrays shaped [shape(n,d)]]`
days the number of days after the earnings announcement day when you want to extract data.
`[int]`

Returns:

Earning_jump_fill: the data assigns each non-earnings announcement day as the data at the time of the previous earnings announcement day.

Notes:

This operator performs excellently in multiple scenarios. For example, if you want to extract the previous Earning announcement jump (ret1_excess/ret1), you can write `pick_out_the_exact_EA_jumpnum(ret1_excess, 0).` The correct earning_jump itself is a very good factor, and when it is combined with many other operators, it improves the ir performance in most scenarios.

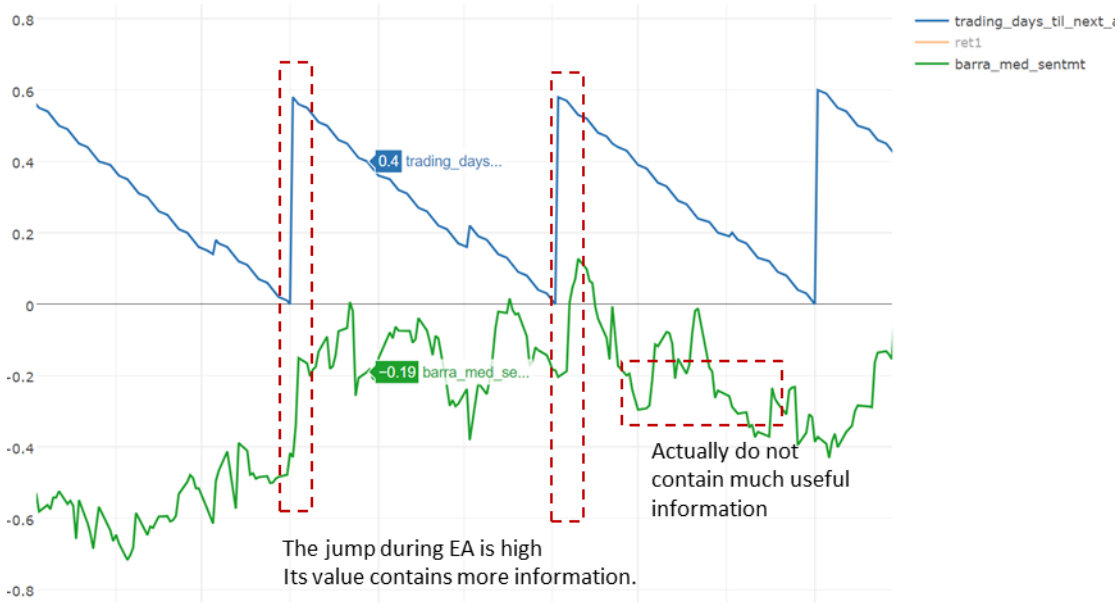
Example:

If you want to extract the sentiment score of the current earning_call transcript, you can write `pick_out_the_exact_EA_jumpnum(ts_zscore(barra_med_sentmt, 11), 3)`.

This operator meet the three criterion:

1.there is a use case for other researchers using this operator.

Often times, data around the release of financial reports tends to have more information (such as sentiment data, price and volume data before and after earnings announcements, options data, liquidity data, intraday data, etc.), while daily data often does not have a high signal-to-noise ratio. Therefore, in many scenarios, we need to use the data from a certain day after the previous earnings announcement day as the daily data for non-trading days.



In addition, the data `ret_0day_before_ann_date` and the data `ret_1day_after_earn_ann_date` are not exactly correct because they did not take into account that different companies may choose to release financial reports before the close and after the close. For companies that release financial reports after the close, `ret_0day_before_ann_date` and the data `ret_1day_after_earn_ann_date` incorrectly chose the previous day as the EA jump. For a detailed analysis, see [GAR-Earnings-Team-2 (Wensheng Li / Chenyu Wu) on Earnings Announcement Return Extrapolation.]

2.there are no existing global operators that perform the same function.

As far as I know, after doing search in the operator library, I did not find anything as same as this operator.

3.the operator does not produce undefined behavior when presented with unexpected input.

I've done so many test of my own alphas, it did not give out unexpected results. For example:

https://www.trexsim.com/trexsim/dj/alpha/3924558_USA

https://www.trexsim.com/trexsim/dj/alpha/3927458_USA

https://www.trexsim.com/trexsim/dj/alpha/3927332_USA

https://www.trexsim.com/trexsim/dj/alpha/3927455_USA

https://www.trexsim.com/trexsim/pysim/alpha/10987650/

https://www.trexsim.com/trexsim/dj/alpha/3927332_USA

For https://www.trexsim.com/trexsim/dj/alpha/3924558_USA

Benchmark test

- according to the data analysis of ***GAR-Earnings-Team-2 (Wensheng Li / Chenyu Wu) on Earnings Announcement Return Extrapolation***, neither *ret_0day_before_earn_ann_date* nor *ret_1day_after_earn_ann_date* provide accurate EA jump information.
- Therefore, when constructing factors, fs_next_ann_time is first used to distinguish companies that release financial reports at different times. This leads to the refinement of earning_jump_fill as follows:

After benchmark testing, I also found that using incorrect returns significantly increased the noise.

```
delay = 0
valid_mask = trading_days_til_next_ann==1
valid_effect_mask = np.zeros_like(valid_mask, dtype=np.float32)
for si, di in zip(*np.where((valid_mask == 1) & (close != np.nan))):

    # skip if we are unable to get the target
    if di + delay >= numdates:
        continue

    time = fs_next_ann_time[si, di]

    if time <= 1500:
        target_di = di
    elif time > 1500 and di + 1 < numdates:
        target_di = di+1
    if valid_effect_mask[si, target_di] != 1:
        valid_effect_mask[si, target_di] = 1

bet_days_mask = valid_effect_mask==1
bet_days_mask = op.at_nan2zero(bet_days_mask*1.0)
flatten_mask = (bet_days_mask==1.0)

earning_jump = np.where(flatten_mask[:,:-1] == 1, ret1[:,1:], np.nan)
earning_jump = np.concatenate((np.full((ret1.shape[0], 1), np.nan, dtype=np.float32), earning_jump), axis=-1)
earning_jump_fill = ts_fill(earning_jump)
```

Incorrect EA returns

```
OS TEST RESULTS

Performance Test
ir since 2006:   0.041
ir since 2011:   0.028
ir since 2017:   0.014
numstk       :     900
```

trexsim.com/trexsim/pysim/alpha/10951513/

better

Submitted one

```
OS TEST RESULTS

Performance Test
ir since 2006:   0.087
ir since 2011:   0.072
ir since 2017:   0.137
numstk       :    2861
```

trexsim.com/trexsim/pysim/alpha/10944695/