Strategy description:

Trading strategies based on Directional Change (DC) intrinsic time series in FX markets can be effective, as DC events can capture significant price movements over time. For this project, I have considered DC Reversal Strategy.

Logic:

Enter trading positions by taking a contrary position to the last DC event's direction. In other words:

- if the last DC event signals an 'up' direction, go short (signal=”sell”)
- if the direction is 'down,' go long (signal=”buy”)
- Exit positions when the next DC event occurs with a different direction.

This strategy aims to profit from potential reversals after sharp price movements.

Data:

The trading strategy was created and tested on 2 datasets:

1. EURGBP data for one month of 06/2016
2. EURUSD data for one month of 12/2019

Code:

```
import pandas as pd
import mysql.connector
mydb=mysql.connector.connect(
host="localhost",
user="username",
password="password",
database="database")
cursor = mydb.cursor()
cursor.execute("select * from vw_DAT_ASCII__EURUSD_T_2")
#second dataset vw_DAT_ASCII_EURGBP_T_2
result=cursor.fetchall()
df=pd.DataFrame(result,columns=cursor.column_names)
#print(df)


def intrinsic_event_algo(df, column, dc_threshold):

#Set up the extreme references, starting with first value and having 2
extreme references (a max and a min)until the first DC event
reference_extreme = df.at[0, column]
last_dc_direction = None
dc_event = None
dc_os_events = []
init_min= df.at[0, column]
init_max= df.at[0, column]
```

```
for index, row in df.iterrows():
curr_price = row[column]
# Calculate price_change and see if it's greater than or equal to the DC
threshold
if last_dc_direction==None:
price_change_min=(curr_price-init_min)/init_min
price_change_max=(curr_price-init_max)/init_max
is_dc = max(abs(price_change_max),abs(price_change_min))>= dc_threshold
if abs(price_change_max)>=dc_threshold:
price_change=price_change_max
elif abs(price_change_min)>=dc_threshold:
price_change=price_change_min
else:
price_change = (curr_price - reference_extreme) / reference_extreme
is_dc = abs(price_change) >= dc_threshold

if is_dc:
direction = 'up' if price_change > 0 else 'down'

# Check if the current direction is different from the last_dc_direction
if direction != last_dc_direction:
reference_extreme = curr_price
dc_event  =  {"timestamp":  row["timestamp"],  "mid":  curr_price,"bid":
row["bid"],"ask": row["ask"], "type": "DC", "direction": direction}
if dc_event is not None:
dc_os_events.append(dc_event)
last_dc_direction = direction
dc_event = None
else:
if last_dc_direction==None and curr_price >init_max:
init_max=curr_price
elif last_dc_direction==None and curr_price <init_min:
init_min=curr_price
if    (last_dc_direction=='up'and    curr_price>reference_extreme)    or
(last_dc_direction=='down' and curr_price<reference_extreme):
reference_extreme=curr_price
return pd.DataFrame(dc_os_events)

# Create the intrinsic Time series for a
dc_threshold = 0.01
event_sequence = intrinsic_event_algo(df, 'mid', dc_threshold)
print(event_sequence)


########################################################################
########### TRADING STRATEGY DC REVERSAL ###############################
########################################################################
##


def dc_reversal_strategy(event_sequence, df):
initial_equity = 1.0 # Initial capital
```

```
position = None
pnl = []
equity_curve= initial_equity

for index, event in event_sequence.iterrows():
if event['type'] != 'DC':
continue

event_time = event['timestamp']
price_data = df[df['timestamp'] == event_time].iloc[0]

# Follow the reverse of directional change
if event['direction'] == 'up':
signal = 'sell'
elif event['direction'] == 'down':
signal = 'buy'

if position:
# Calculate the PnL
if position['type'] == 'buy':
pnl.append(price_data['bid'] - position['price'])
elif position['type'] == 'sell':
pnl.append(position['price'] - price_data['ask'])

position = None

# Open a position
if signal == 'buy':
position = {'price': price_data['ask'], 'type': 'buy'}
elif signal == 'sell':
position = {'price': price_data['bid'], 'type': 'sell'}

# Calculate the final PnL and the equity curve
total_pnl = sum(pnl)
equity_curve = [initial_equity + sum(pnl[:i+1]) for i in range(len(pnl))]
equity_curve.insert(0,initial_equity)
return {'pnl': pnl, 'total_pnl': total_pnl, 'equity_curve': equity_curve}

# Execute the strategy
results = dc_reversal_strategy(event_sequence, df)

print("PnL per Trade:", results['pnl'])
print("Total PnL:", results['total_pnl'])
print("Equity Curve:", results['equity_curve'])

########################################################################
########## TRY TO OPTIMISE the choice of DC threshold #################

import numpy as np

def optimal(df,min_DC,max_DC,step):
best= None
```

```
max_pnl=float("-inf")

for threshold in np.arange(min_DC,max_DC,step):
event_sequence =intrinsic_event_algo(df, 'mid', threshold)
results=dc_reversal_strategy(event_sequence,df)

if results['total_pnl']> max_pnl:
max_pnl=results['total_pnl']
best_threshold = threshold
return best_threshold, max_pnl
min_DC=0.01
max_DC=0.05
step=0.01
optimal_DC,max_pnl=optimal(df,min_DC,max_DC,step)
print("OPTIMAL DC THRESHOLD:",optimal_DC)
print("max pnl",max_pnl)
```

Code description:

The ***intrinsic_event_algo*** function will identify and record all DC events into a data frame called "event_sequence". It will output the following details: timestamp, mid, ask, bid, type, direction.

The ***dc_reversal_strategy*** function will be responsible for execution of the strategy. This will check the event_sequence dataframe and will open a long position at the event's timestamp if the direction if "down" and a short position at the event's timestamp if the direction if "up". Subsequently, once the next DC is confirmed this would close the current position. Finally, this will calculate the total pnl and return the total pnl value, the pnl per trade and the equity curve, so we have more tools to assess the performance of the strategy across time.

The ***find_optimal_dc_threshold*** function is aimed to find the optimal DC threshold. It takes the data frame (df), some minimum and maximum threshold values provided based on the judgement on price volatility, and a step size. The function iterates for each value of the threshold, calculates directional change events, and calls the dc_reversal_strategy function. The threshold value resulting in the highest pnl will be considered the optimal one.

Execution and back testing:

The strategy will be back tested on 2 FX time series from KEATS and the performance will be recorded and analysed. We are assuming no fees and commissions for these trades.

Note, as the data contains bid and ask prices it is difficult to find the Directional changes based on both. Thus, "mid" price, ((bid+ask)/2), will be used to identify the Directional changes, and then the bid and ask prices will be used to record the prices of the trading activity.

List of steps for back testing

1. **Download the data**
2. **Import it in the SQL database.**
3. **Create views containing: timestamp, mid price, bid and ask.**

   It is challenging to identify the Directional changes based on 2 prices at the same time, thus I decided to create a new column called "mid price" that would average the bid and ask.

```sql
#Convert to timestamp
update DAT_ASCII_EURGBP_T
set timestamp = str_to_date(timestamp,'%Y%m%d %H%i%s%f')

update DAT_ASCII_EURUSD_T
set timestamp = str_to_date(timestamp,'%Y%m%d %H%i%s%f')

create view vw_DAT_ASCII_EURGBP_T_2 as
select `timestamp`,(ask+bid)/2 as `mid`, bid, ask from DAT_ASCII_EURGBP_T

create view vw_DAT_ASCII__EURUSD_T_2 as
select `timestamp`,(ask+bid)/2 as `mid`, bid, ask from DAT_ASCII_EURUSD_T
```

4. **Understand how volatile is the data.**

   I decided to keep it simple here and just looked at the extremes and maximum change. This should help choose appropriate DC thresholds. The results are below with EURUSD data being first.

```sql
select min(mid), max(mid), abs(min(mid)- max(mid))/min(mid) as maximum
```

	min(mid)	max(mid)	maximum change
1	1.1002999544	1.1239199638	0.0214668821

```sql
select min(mid), max(mid), abs(min(mid)- max(mid))/min(mid) as maximum
```

	min(mid)	max(mid)	maximum change
1	0.7601000071	0.8382200003	0.1027759406

   From here, we can observe that the EURUSD data is less volatile than the EURGBP. Thus, the DC threshold would need to be adjusted accordingly.

5. **Import the data in a Python data frame (df)**

   The data was imported in a python data frame called df. In order to do that, mysql.connector was used.

6. **Run the Code for an example DC threshold and output the results.**

   Based on information from step 4, it seemed reasonable to try to run for dc_threshold= 0.02 for the first time. The output is below. It can be observed that for EURGBP dataset it identifies 10 DC events at this threshold and for EURUSD it identifies only one. Thus 2% can be considered as a bad threshold for EURUSD dataset. Based on the output of the dc_reversal_strategy() function the recorded PnL for EURGBP data was -1.012%.

a. EURGBP

```
✓ dc_threshold = 0.02 ···
...                  timestamp       mid      bid      ask type direction
0   2016-06-03 16:09:27.663000  0.782520  0.78249  0.78255   DC        up
1   2016-06-19 17:00:57.307000  0.783280  0.78283  0.78373   DC      down
2   2016-06-23 19:17:42.417000  0.775325  0.77495  0.77570   DC        up
3   2016-06-23 19:28:02.667000  0.776000  0.77574  0.77626   DC      down
4   2016-06-23 21:06:48.887000  0.785865  0.78548  0.78625   DC        up
5   2016-06-23 21:23:41.887000  0.782330  0.78164  0.78302   DC      down
6   2016-06-23 22:15:54.120000  0.790210  0.78986  0.79056   DC        up
7   2016-06-24 01:44:44.340000  0.814860  0.81478  0.81494   DC      down
8   2016-06-24 10:58:15.503000  0.814085  0.81407  0.81410   DC        up
9   2016-06-29 10:45:54.317000  0.821180  0.82113  0.82123   DC      down
10  2016-06-30 11:12:22.150000  0.837110  0.83703  0.83719   DC        up
PnL per Trade: [-0.0012400000000000189, -0.00878000000000001, -0.0013099999999999223, 0.009220000000000006, 0.0024599999999999067, 0.006839999999999957, -0.02
Total PnL: -0.010120000000000129
Equity Curve: [1.0, 0.99876, 0.98998, 0.98867, 0.99789, 1.00035, 1.00719, 0.9821099999999999, 0.9812399999999999, 0.9740799999999998, 0.9898799999999999]
```

b. EURUSD

```
✓ import pandas as pd ···
...                  timestamp       mid      bid      ask type direction
0  2019-12-31 06:33:55.169000  1.122315  1.12231  1.12232   DC        up

✓ def dc_reversal_strategy(event_sequence, df): ···
... PnL per Trade: []
    Total PnL: 0
    Equity Curve: [1.0]
```

7. **Use the *find_optimal_dc_threshold* function to find the optimal dc_threshold for this specific dataset.**
   a. EURGBP:
   The max_DC and min_DC for the function were chosen as 5% and 1% respectively. Because of the processing performance limitations, the step was chosen to be 1%. In other words, the strategy ran for DC thresholds of 1%, 2%, 3%, 4% and 5% , outputting the optimal DC threshold as 4%. The recorded pnl at the optimal DC threshold is 7.86%.

```
OPTIMAL DC THRESHOLD: 0.04
max pnl 0.07862999999999987

✓ dc_threshold = 0.04 ···
..                   timestamp       mid      bid      ask type direction
0  2016-06-13 05:10:20.417000  0.797835  0.79781  0.79786   DC        up
1  2016-06-21 05:31:04.520000  0.767460  0.76744  0.76748   DC      down
2  2016-06-23 19:17:53.917000  0.791195  0.79114  0.79125   DC        up
3  2016-06-24 04:50:44.060000  0.798215  0.79817  0.79826   DC      down
4  2016-06-27 05:20:10.983000  0.830055  0.83002  0.83009   DC        up
PnL per Trade: [0.030329999999999968, 0.023659999999999903, -0.007120000000000015, 0.03176000000000001]
Total PnL: 0.07862999999999987
Equity Curve: [1.0, 1.03033, 1.0539899999999998, 1.0468699999999997, 1.07863]
```

   b. EURUSD:
   The max_DC and min_DC for the function were chosen as 2% and 0.4% respectively, with a step of 0.4%. In other words, the strategy ran for DC thresholds of 0.4%, 0.8%, 1.2%,

1.6% and 2%, outputting the optimal DC threshold as 0.8%. The recorded pnl at the optimal DC threshold is 0.254%

```
OPTIMAL DC THRESHOLD: 0.008
max pnl 0.0025400000000002088

✓ def intrinsic_event_algo(df, column, dc_threshold): ···

                   timestamp       mid      bid      ask type direction
0  2019-12-03 11:29:40.794000  1.109115  1.10911  1.10912   DC        up
1  2019-12-19 08:17:59.489000  1.111005  1.11099  1.11102   DC      down
2  2019-12-27 07:22:54.324000  1.115485  1.11547  1.11550   DC        up
PnL per Trade: [-0.0019099999999998563, 0.004450000000000065]
Total PnL: 0.0025400000000002088
Equity Curve: [1.0, 0.9980900000000001, 1.0025400000000002]
```

Interpretation of empirical results

This strategy was tested using two datasets: EURGBP tick data from June 2016 and EURUSD tick data from December 2019. When running for the same DC threshold, we can plainly see the difference in the number of DC occurrences between these two datasets. As an example, if we run this with a 2% DC threshold, we obtain 10 DC occurrences in the EURGBP dataset but just 1 DC event in the EURUSD dataset. This is due to EURGBP data being significantly more volatile than EURUSD data for the relevant months. As a result, we may deduce that the strategy's parameters must be adjusted to the underlying data for it to be efficient.

The strategy was initially executed with a pre-set DC threshold of 2%. Step 5 contains the outcomes. As we can see, the strategy produced a negative PnL for the EURGBP pair and was thus unprofitable at this DC level. After experimenting with several DC threshold values and documenting the findings, it became evident that an optimisation function could be required to choose the DC threshold depending on the underlying dataset. This should be accomplished by maximising the overall PnL. As a consequence, the function find_optimal_dc_threshold was constructed. This function assisted in determining the best DC threshold for both datasets. However, because it takes a substantial amount of processing power, it was performed only with a significantly large Step, as specified in step 6.

As a general pattern, it can be observed that higher DC thresholds generate positive pnl. This is most likely explained by the fact that at low DC thresholds, DC events are difficult to distinguish from market noise.

Furthermore, the following hypothesis can be advanced: The DC Reversal Strategy is more effective in a volatile market. This hypothesis can be supported by the comparison of the simulation outcomes for the two datasets, the one with more volatility yielding a higher profit. However, in order to be able to confirm this empirically, more simulations are needed.

In conclusion, DC reversal strategy proved to be profitable on FX tick-data for the given currency pairs. The key for this to generate positive pnl is the optimisation of the DC threshold. The strategy also proved to be more profitable when run on volatile data with multiple Directional Change events.