TRADERCRAFT 2021

DUAL LISTING TRADING STRATEGY

1. INTRODUCTION

A "dual listing" is the listing of any security on two different exchanges. That means the same instrument is traded in the same currency, in two different places (e.g. Paris and Milan). In this exercise, Optibook will be configured with two instruments, "PHILIPS_A" and "PHILIPS_B", and it is your task to trade one against the other.

2. STOCKS

"PHILIPS_A" and "PHILIPS_B" represent such a dual listing - they are both the same instrument (Philips). "PHILIPS_A" is more liquid, and "PHILIPS_B" is less liquid. However, since both listings are the same instrument, they should technically have the same valuation. Tick sizes are 0.10 eur.

Part 1

In Part 1, come up with an arbitrage strategy that shoots for opportunities in the less liquid order book, and hedges the trades in the more liquid order book.

Part 2

In Part 2, try to improve the market by continuously quoting 2-sided in the less liquid order book, and hedging your trades in the more liquid order book.

For implementing your algorithm, take inspiration from the manual actions provided in the 'Manual.ipynb' notebook.

Some Python functions are pre-provided to aid in running this analysis.

NOTE: The stock names are fictious, their values are not in any way related to the real-world equivalents,

3. LIMITS

Your algorithm is not allowed to cross a few pre-predefined limits:

- Total position (long or short) per instrument cannot go over 500
- You are not allowed to have total orders outstanding for over 800 lots per instrument
- You are only allowed to send 25 updates (inserts/deletes/amends) to the market per second

If you breach any of these limits, your algorithm automatically gets disconnected and your outstanding orders will be pulled. You should then reconnect, but if you start your algorithm again, you might well breach again. Perhaps you want to manually reduce your position or change some settings. Can you alter your algorithm to account for all of these automatically?

4. HINTS

Do not implement your whole algorithm at once. Start with a single piece of functionality and run it manually. Only when you are convinced you know how to do each separate part, add it all together and run it on a loop to complete your algorithm.

Make extensive use of print()-statements to keep track of what your algorithm is doing. For added convenience you can use the clear_output(wait=True) command to clear a cell's output during runtime.