# Appendix

Part of Python Code:

1. # CVaR
2. mu\_h = BitcoinMean['A'][i+5]
3. sig = BitcoinSD['A'][i+5]
4. alpha = 0.01
5. sig\_h = sig \* np.sqrt(h / windowA)
6. lev = 100 \* (1 - alpha)
7. CVaR\_n\_A = alpha \*\* -1 \* norm.pdf(norm.ppf(alpha)) \* sig\_h - mu\_h
9. # Strategy
10. **def** Strategy(i, delta=0.01):
11. # delta = 0.0001
12. Sigma = np.mat([[GoldRisk[i], BitcoinRisk[i]], [BitcoinRisk[i], GoldRisk[i]]])
13. Omega = np.matmul((delta \* np.linalg.inv(Sigma)),
14. np.mat([[GoldMeanRes[i]], [BitcoinMeanRes[i]]]))
15. **if** Omega[0] \* Omega[1] > 0:
16. **if** Omega[0] < 0 **and** Omega[1] < 0:
17. Omega[0] = 0
18. Omega[1] = 0
19. **else**:
20. temp = (Omega[0] + Omega[1])
21. Omega[0] = Omega[0] / temp
22. Omega[1] = Omega[1] / temp
23. **else**:
24. **if** Omega[0] < 0:
25. Omega[0] = 0
26. Omega[1] = 1
27. **if** Omega[1] < 0:
28. Omega[1] = 0
29. Omega[0] = 1
31. **return** Omega
33. **for** i **in** range(len(GoldRisk)-1):
34. # print("i:", i, "\n")
35. Omega = Strategy(i, 0.0005)
36. **print**(Omega)
37. # 0 为gold， 1 为bitcoin
39. **if** Omega[0] == lastOmega[0] **and** Omega[1] == lastOmega[1]:
40. lastOmega = Omega
41. # print("Indication: 1\n")
43. **elif** Cash > 0:
44. trade = Omega
45. GoldAmount = float(Cash) \* trade[0] / GoldPrice[window + i - 1] \* (1 - alphaGold / 10000)
46. BitcoinAmount = float(Cash) \* trade[1] / BitcoinPrice[window + i - 1] \* (1 - alphaBitcoin / 10000)
47. lastOmega = Omega
48. Cash = 0
49. # print("Indication: 2\n")
51. **elif** Cash == 0:
52. **if** Omega[0] == 0 **and** Omega[1] == 0:
53. Cash = GoldAmount \* GoldPrice[window + i - 1] \* (1 - alphaGold / 10000) + BitcoinAmount \* BitcoinPrice[window + i - 1] \* (1 - alphaBitcoin / 10000)
54. GoldAmount = 0
55. BitcoinAmount = 0
56. **else**:
57. OmegaDiff = Omega - lastOmega
59. **if** float(OmegaDiff[0]) < 0: # gold
60. Cash = GoldAmount \* abs(float(OmegaDiff[0])) \* GoldPrice[window + i - 1] \*(1 - alphaGold / 10000)
61. GoldAmount = GoldAmount - GoldAmount \* abs(float(OmegaDiff[0]))
62. BitcoinAmount = BitcoinAmount + Cash / BitcoinPrice[window + i - 1] \* (1 - alphaBitcoin / 10000)
63. Cash = 0
64. # print("Indication: 3\n")
66. **elif** float(OmegaDiff[1]) < 0:
67. Cash = BitcoinAmount \* abs(float(OmegaDiff[1]))\* BitcoinPrice[window + i - 1] \* (1 - alphaBitcoin / 10000)
68. BitcoinAmount = BitcoinAmount - BitcoinAmount \* abs(float(OmegaDiff[1]))
69. GoldAmount = GoldAmount + Cash / GoldPrice[window + i - 1] \* (1 - alphaGold / 10000)
70. Cash = 0
71. # print("Indication: 4\n")
73. lastOmega = Omega
74. value = float(Cash) + GoldAmount \* GoldPrice[window+i] + BitcoinAmount \* BitcoinPrice[window+i]
76. Value.append(float(value))
77. **print**(i, value, "\n")

Part of Python Code:

1. **for** (i **in** BayesWindow:length(Gold)){
2. tempData= c(Gold[(i-BayesWindow):i])
3. tempData = zoo(tempData, index(tempData))
4. ss = AddSemilocalLinearTrend(list(), tempData)
5. model = bsts(tempData, state.specification = ss, niter = 500)
6. Res = predict(model)
7. Mean=Res$mean
8. sd = Res$mean-Res$interval[1]
10. GoldBMean = c(GoldBMean, Mean)
11. RelativeGoldBMean = c(RelativeGoldBMean, (Mean-tempData[BayesWindow])/tempData[BayesWindow])
12. GoldBSD = c(GoldBSD, sd)
14. cat(i, "\n")
15. cat("\n")
16. cat("\n")
17. }