Final

Problem A. Asian Option Pricing using Monte Carlo Control Variate:

(a) The price of a geometric Asian option in the Black-Scholes model is 15.17113

(b) The price of arithmetic Asian call option by Monte Carlo scheme is 17.52603,
The confidence interval is (17.32147,17.93058)
The time it takes 5920.5006 secs.

(c) The price of arithmetic Asian call option by Monte Carlo scheme is 15.06854, which is close to The price that calculated by BS model.

(d) The value of b∗ is 1.151145.

(e) The error of pricing for the geometric Asian is 0.218688.

(f) I try the M with 1e+03,1e+04 and 1e+05, the result as follow:

M=100	M=10000	M=10000
18.69038	17.95927	17.45871

We can see that the bigger M is, the result will be the more close option price by Monte carlo.

Bonus:

Comment: I download AMZN Asian option data with 5 different time to marturity and 5 different strike price correspondingly from the Bloomberg terminal. Repeat the parts (a) to (f) and here is the result table:

amzn.bs.geo	amzn.arith	amzn.geo	amzn.b	amzn.err	amzn.m
1.4357111	1.3606961	1.1708596	1.027425	-0.014330994	1.022576
1.8247676	2.2062225	1.8260466	1.029846	-0.088331275	1.739103
3.1692529	2.2767663	3.3659314	1.029600	0.009782826	4.319384
3.5140698	4.2597186	4.0048936	NaN	-0.429371956	5.239457
3.4702953	3.2056080	3.1940632	1.028823	0.228177605	3.276389
0.9801242	0.7800987	0.9080172	NaN	-0.115629811	NaN
1.0577416	0.7498137	1.1439178	NaN	0.136281076	NaN
1.8289716	2.1404024	1.9407384	NaN	0.279461512	NaN
2.0833350	2.5722506	1.7684439	NaN	-0.275866691	NaN
2.0460085	2.4676621	1.8042160	NaN	-0.023573762	NaN
1.0136742	1.3043167	0.9897642	NaN	0.137415174	NaN
0.8002145	0.9528298	0.8750863	NaN	0.010839505	NaN
1.1181368	1.0368601	0.5897117	NaN	0.175077969	NaN
1.2859385	1.6638836	2.1061510	NaN	0.212098550	NaN
1.2570460	1.4842794	1.2777416	NaN	0.067186527	NaN
1.0581061	0.8940622	0.5658339	NaN	0.313443050	NaN
0.6946125	0.7462681	0.5042843	NaN	-0.145659620	NaN
0.7349748	0.9445335	0.9219202	NaN	-0.099531037	NaN
0.8421628	1.1564793	1.1060746	NaN	-0.146477104	NaN
0.8200020	0.7052043	1.1989139	NaN	0.143948127	NaN
0.9881076	0.7745277	1.1080853	NaN	0.200658047	NaN
0.6268654	0.8599589	0.5286230	NaN	0.271557624	NaN
0.5234598	0.4971253	0.6616545	NaN	-0.183215436	NaN
0.6087858	1.0096879	0.5671925	NaN	0.297476889	NaN
0.5736472	0.7515106	0.6595242	NaN	-0.188504339	NaN

There are some NAN though, I think the reason is data not that good, because when we calculate option price, they produce zero. Hence, according to b star formula, we have 0/0, that produce NANs.

Problem B. A portfolio construction problem:

(1) Comment: I choose the tickers as follow: 'ACC','JPM','WFC','BAC','C','GS','USB','CB','MS','AXP','AADR','ACFC','AAME','AAT','AAXJ','ACNB','ABCB','ABE','ABR','BRK-B'.

Then I perform a principal component analysis for these 20 equities, the results shows as follow:

```
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4     Comp.5     Comp.6     Comp.7     Comp.8     Comp.9
Standard deviation     2.7473002 1.26097223 1.06329633 1.01312238 1.00659205 0.98004400 0.95014806 0.91814997 0.85472554
Proportion of Variance 0.3773829 0.07950255 0.05652995 0.05132085 0.05066138 0.04802431 0.04513907 0.04214997 0.03652779
Cumulative Proportion  0.3773829 0.45688548 0.51341543 0.56473628 0.61539766 0.66342197 0.70856103 0.75071100 0.78723879
                          Comp.10    Comp.11    Comp.12    Comp.13    Comp.14    Comp.15    Comp.16    Comp.17    Comp.18
Standard deviation     0.84696561 0.76376451 0.72954379 0.67676703 0.64153459 0.59936428 0.57939432 0.51257926 0.48874012
Proportion of Variance 0.03586754 0.02916681 0.02661171 0.02290068 0.02057833 0.01796188 0.01678489 0.01313687 0.01194335
Cumulative Proportion  0.82310633 0.85227314 0.87888485 0.90178553 0.92236386 0.94032574 0.95711062 0.97024750 0.98219084
                          Comp.19     Comp.20
Standard deviation     0.43732664 0.406113919
Proportion of Variance 0.00956273 0.008246426
Cumulative Proportion  0.99175357 1.000000000
```

From this result, we can see clearly that until component 10, we get the variance cumulative proportion over 80%, then I use principal function to do a principal components analysis (PCA) for n principal components with covariance matrix, here is the result:

Loadings:

	RC1	RC10	RC2	RC3	RC8	RC5	RC6	RC4	RC7	RC9
ACC		0.128	0.871							
JPM	0.853	0.209								
WFC	0.716	0.396								0.138
BAC	0.871									0.116
C	0.886	0.153		0.107						
GS	0.843	0.156								
USB	0.684	0.431								0.137
CB	0.310	0.752	0.191							
MS	0.856	0.129								
AXP	0.462	0.502		0.111						0.250
AADR					0.984					
ACFC								0.997		
AAME						0.997				
AAT	0.162	0.143	0.826							0.136
AAXJ	0.464	0.460	0.271	0.386	0.195					
ACNB							0.994			
ABCB	0.338	0.154	0.104							0.895
ABE	0.163			0.957						
ABR	0.119		0.115						0.980	
BRK-B	0.508	0.648	0.157							

	RC1	RC10	RC2	RC3	RC8	RC5	RC6	RC4	RC7	RC9
SS loadings	5.675	1.986	1.652	1.127	1.031	1.008	1.008	1.006	0.999	0.970
Proportion Var	0.284	0.099	0.083	0.056	0.052	0.050	0.050	0.050	0.050	0.049
Cumulative Var	0.284	0.383	0.466	0.522	0.574	0.624	0.674	0.725	0.775	0.823

From the loadings, we can find the equity WFC,C,AAT,ABCB,AAXJ appear frequently, So I think these equities influence the most these PCA's.

(2) We can see that equity AXP,AAXJ,ABCB,AAT appear more times than other equities, so I choose these four equities.

Best:	model 2	model 2	model 1	model 2
	AXP AIC	AAXJ AIC	ABCB AIC	AAT AIC
mod1	3581.156	2560.986	948.9715	1492.980
mod2	3564.292	2442.088	951.0931	1481.295
mod3	3591.116	2490.142	1405.8583	1490.609
mod4	3566.447	2532.256	989.4801	1482.735
mod 5	7045.305	5074.003	8.000	6781.681

(3) Correlation matrix for the 4 stocks based on historical data:

	AXP.Close	AAXJ.Close	ABCB.Close	AAT.Close
AXP.Close	1.0000000	0.6128595	0.2896446	0.4300944
AAXJ.Close	0.6128595	1.0000000	0.1482991	0.3169818
ABCB.Close	0.2896446	0.1482991	1.0000000	0.8412138
AAT.Close	0.4300944	0.3169818	0.8412138	1.0000000

(4) the result table as follow:

	ST.mean	ST.sd	ST.skewness	ST.kurtosis
AXP	48.93597	0.7808147	-0.11862724	2.887843
AAXJ	52.12484	0.6661133	-0.17973874	3.155740
ABCB	10.58095	0.1709692	-0.02665338	2.966082
AAT	21.45780	0.7597919	0.05594974	3.105620

(5) Use ETF data to estimate the coefficient of geometric Brownian motion, the result that I get as follow: μ=0.0004877079 σ=-0.01145874.

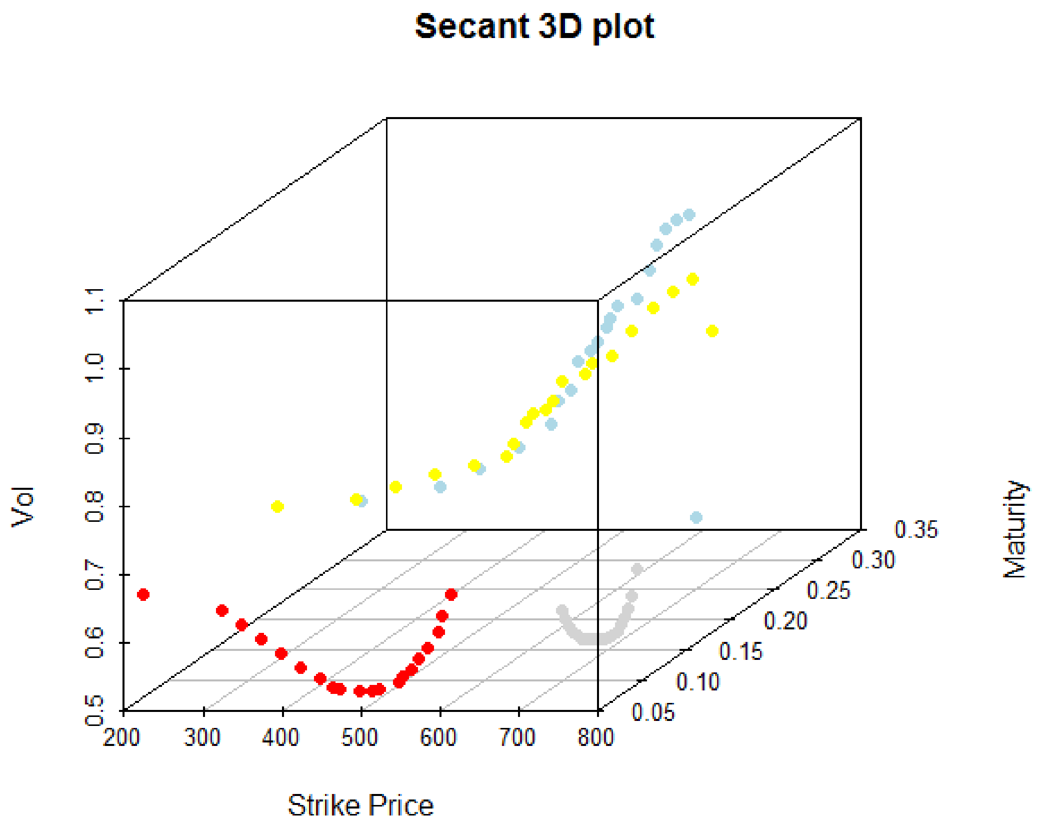
(6) The multivariate regression coefficients as follow:

(Intercept)	AXP.Close	AAXJ.Close	ABCB.Close	AAT.Close
0.354383942	0.001210863	-0.005288692	-0.011007444	0.009486602

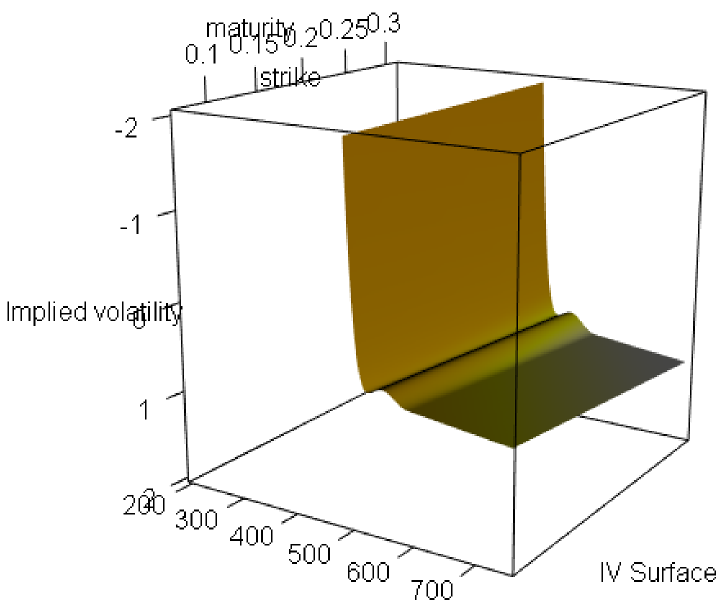
(7) The option price of exchange 1 ETF share with a weighted average of the 4 stocks is 0
The option price of exchange the weighted average for the ETF is 23.69033

Problem C. Local volatility:

(a) Comment: In this part, I use secant method to compute the implied volatility from the Black-Scholes model, then I choose 4 different time to maturity with 20 strike price to each time to maturity. I use these data to get a plot of the points in 3-dimensions in the space (K,T,σ), the 3D plot as follow:



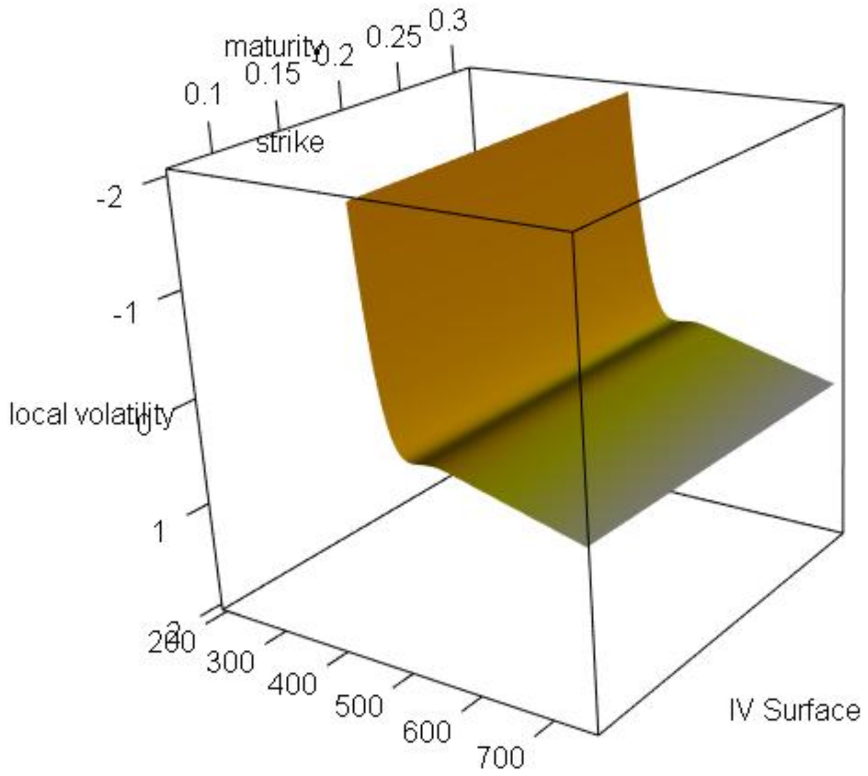
(b) Comment: apply a cubic spline interpolation in 2 dimensions with the points based on the 80 data that I choose in last part, then use these interpolated data to plot the implied volatility surface. The plot as follow:



(c) Comment: For the points on the surface that I just calculated hold the non-arbitrage condition, because implied volatility is calculated by BS model, it should be constrained by the absence of arbitrage and from the plot we can see our implied volatility is vibrate in fixed interval.

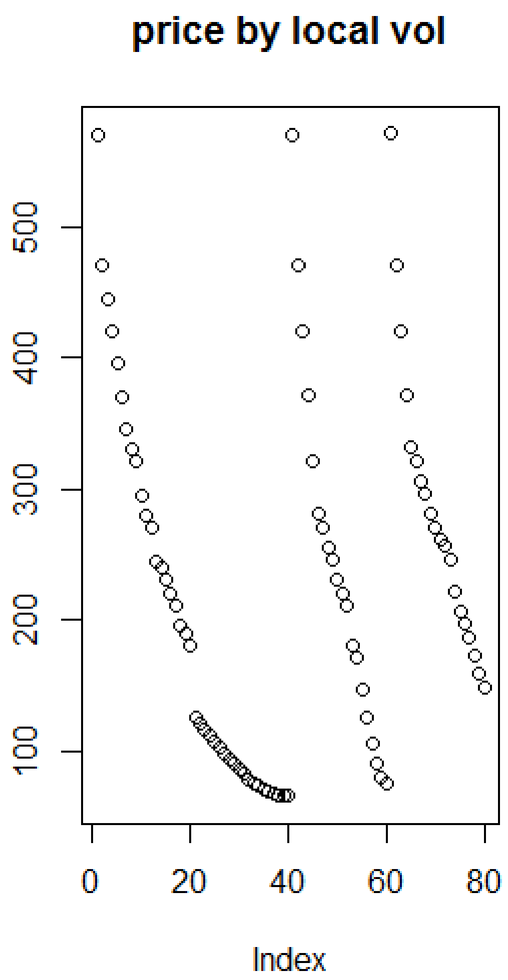
(d) Comment: I write a function to calculate Dupire's local volatility, then use the same method to interpolate , then I got plot as follow:

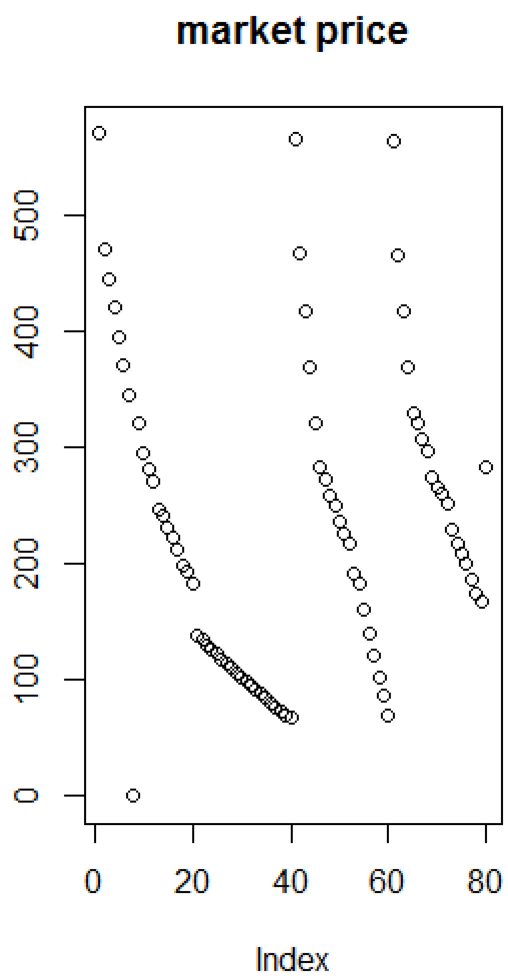
Plot for local volatility is similar to implied volatility, but it is more gentle and stable than implied volatility plot.



(e) Comment: I take local volatility that I calculated in last part into the B-S model function, then I got European option price. Then I plot price of local volatility and market price as follow:

From the plot, we can see that the price of local volatility and market price is very similar to each other. The price by local volatility see(g)





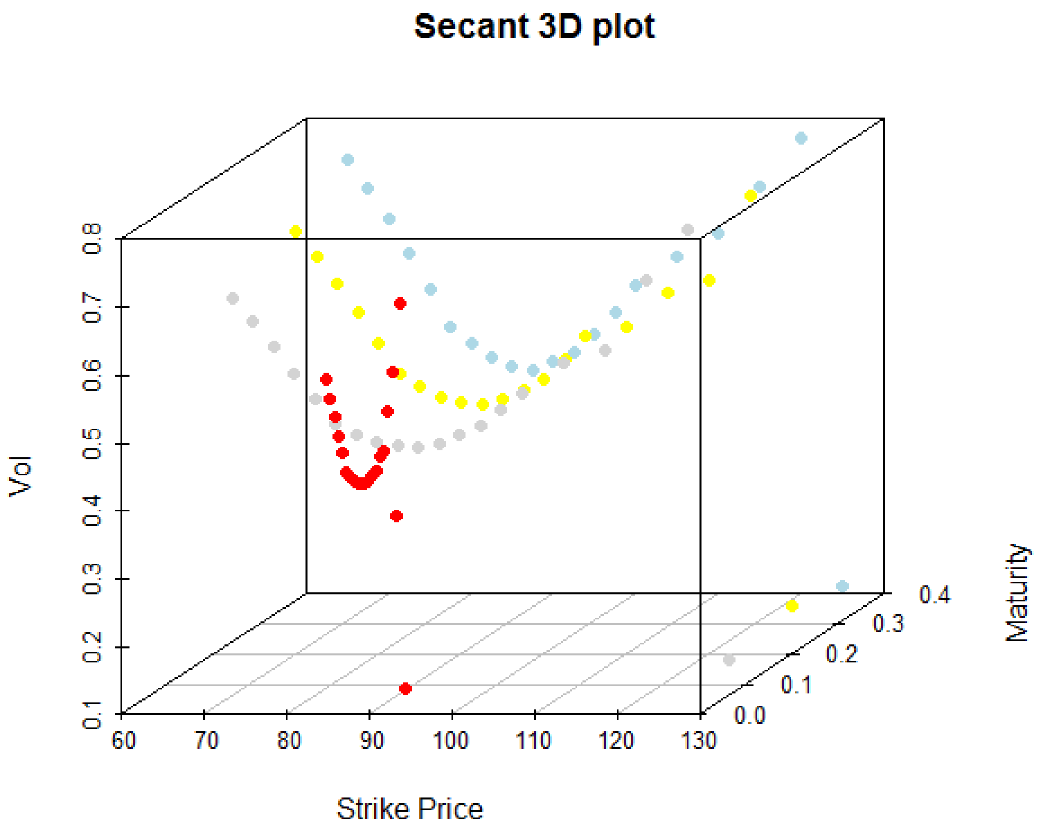
(f) Comment: the results in a table with the following columns: time to maturity, strike price, option market price, implied volatility, local volatility, price obtained using Dupire's local volatility as follow: From the table, we can see that local volatility is smaller than implied volatility, and the price calculated by local volatility is very close to market price.

	Time	Strike	Market price	Implied vol	Local vol	Price by local
1	0.0712329	200	569.80	0.6513775	0.1133606	570.14401
2	0.0712329	300	469.95	0.6299351	0.1055683	470.19101
3	0.0712329	325	444.95	0.6081591	0.1033042	445.20276
4	0.0712329	350	420.05	0.5865037	0.1016273	420.21451
5	0.0712329	375	395.15	0.5653869	0.1005452	395.22626
6	0.0712329	400	370.25	0.5459891	0.1002053	370.23801
7	0.0712329	425	345.35	0.5294503	0.1007943	345.24976
8	0.0712329	440	0.00	0.5171704	0.1010518	330.25681
9	0.0712329	450	320.55	0.5131452	0.1019728	320.26151
10	0.0712329	475	295.75	0.5107900	0.1060067	295.27326
11	0.0712329	490	280.95	0.5106017	0.1090554	280.28031
12	0.0712329	500	271.10	0.5124106	0.1116197	270.28501
13	0.0712329	525	246.55	0.5230706	0.1200016	245.29676
14	0.0712329	530	241.65	0.5318480	0.1230447	240.29911
15	0.0712329	540	231.85	0.5433682	0.1283182	230.30381
16	0.0712329	550	222.15	0.5580706	0.1345411	220.30851
17	0.0712329	560	212.45	0.5749126	0.1415914	210.31321

18	0.0712329	575	197.95	0.5978896	0.1526603	195.32027
19	0.0712329	580	193.15	0.6217566	0.1597774	190.32262
20	0.0712329	590	183.65	0.6527819	0.1713244	180.32732
21	0.1479450	645	138.95	0.5595244	0.2156395	126.00972
22	0.1479450	650	134.75	0.5478004	0.2181472	121.14685
23	0.1479450	655	130.55	0.5378906	0.2212456	116.33431
24	0.1479450	660	126.45	0.5293722	0.2248893	111.58651
25	0.1479450	665	122.35	0.5232531	0.2293687	106.92866
26	0.1479450	670	118.35	0.5195186	0.2347505	102.38750
27	0.1479450	675	114.35	0.5184605	0.2411794	97.99722
28	0.1479450	680	110.35	0.5183332	0.2482901	93.77379
29	0.1479450	685	106.45	0.5180702	0.2558667	89.73052
30	0.1479450	690	102.65	0.5180378	0.2640566	85.89458
31	0.1479450	695	98.85	0.5179923	0.2728593	82.28413
32	0.1479450	700	95.05	0.5194237	0.2827481	78.94946
33	0.1479450	705	91.35	0.5220731	0.2937499	75.91236
34	0.1479450	710	87.75	0.5231910	0.3052124	73.12916
35	0.1479450	715	84.15	0.5299865	0.3192358	70.79506
36	0.1479450	720	80.65	0.5389017	0.3349643	68.85014
37	0.1479450	725	77.15	0.5507422	0.3527298	67.33186
38	0.1479450	730	73.65	0.5644410	0.3723456	66.22055
39	0.1479450	735	70.25	0.5809432	0.3941415	65.54488
40	0.1479450	740	67.05	0.6204872	0.4224807	65.78349
41	0.2246580	200	566.25	0.6442456	0.1440182	570.34633
42	0.2246580	300	466.95	0.6556306	0.1382043	470.49449
43	0.2246580	350	417.65	0.6731916	0.1423319	420.56858
44	0.2246580	400	368.65	0.6919622	0.1497457	370.64266
45	0.2246580	450	320.35	0.7045776	0.1598589	320.71674
46	0.2246580	490	282.25	0.7169045	0.1712442	280.77601
47	0.2246580	500	272.75	0.7358678	0.1765231	270.79082
48	0.2246580	515	258.75	0.7674731	0.1851409	255.81307
49	0.2246580	525	249.45	0.7800769	0.1901543	245.82798
50	0.2246580	540	235.65	0.7862001	0.1967194	230.85102
51	0.2246580	550	226.55	0.7985199	0.2023649	220.86821
52	0.2246580	560	217.55	0.8276403	0.2100355	210.89118
53	0.2246580	590	190.95	0.8389145	0.2271829	181.07524
54	0.2246580	600	182.35	0.8549142	0.2348576	171.25429
55	0.2246580	625	161.25	0.8650907	0.2534838	147.32323
56	0.2246580	650	140.85	0.9015624	0.2776237	125.24887
57	0.2246580	675	121.35	0.9352367	0.3054650	106.12791
58	0.2246580	700	103.05	0.9575251	0.3384014	90.77679
59	0.2246580	725	85.95	0.9760327	0.3792638	79.67824
60	0.2246580	750	70.15	0.8996850	0.4457146	75.02320
61	0.3205480	200	564.15	0.5659972	0.1481471	570.47268
62	0.3205480	300	465.45	0.5878096	0.1434566	470.68401
63	0.3205480	350	416.55	0.6142074	0.1486926	420.78968
64	0.3205480	400	368.35	0.6451555	0.1575407	370.89535

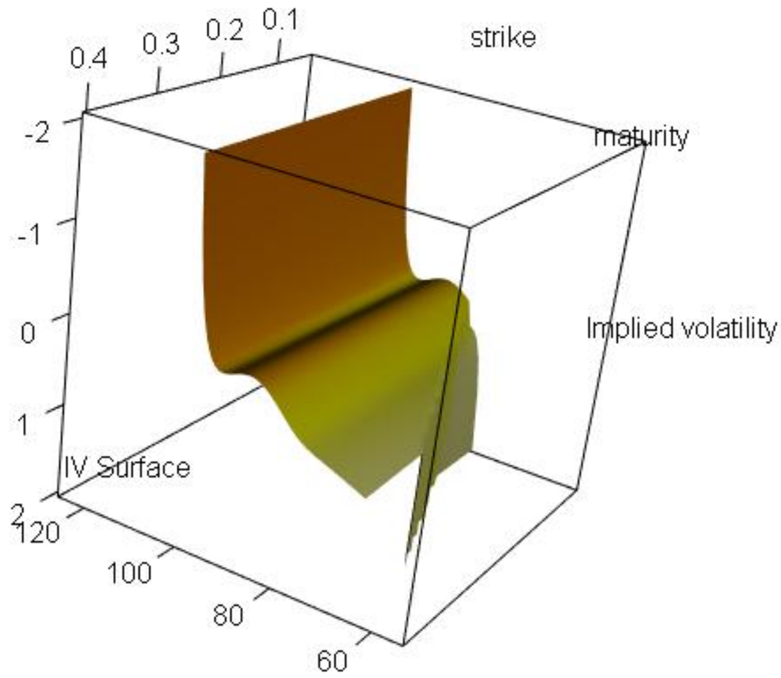
65	0.3205480	440	330.25	0.6789485	0.1679845	330.97989
66	0.3205480	450	320.85	0.7140846	0.1738479	321.00102
67	0.3205480	465	306.85	0.7282114	0.1786128	306.03272
68	0.3205480	475	297.55	0.7715945	0.1852714	296.05388
69	0.3205480	490	274.65	0.7855788	0.1904411	281.08574
70	0.3205480	500	265.55	0.7994274	0.1944722	271.10732
71	0.3205480	510	261.05	0.8196741	0.1991462	261.12987
72	0.3205480	515	252.05	0.8327753	0.2017346	256.14196
73	0.3205480	525	229.95	0.8512477	0.2063630	246.16907
74	0.3205480	550	216.95	0.8634326	0.2163465	221.28043
75	0.3205480	565	208.45	0.9045207	0.2247711	206.43991
76	0.3205480	575	199.95	0.9414885	0.2306532	196.62910
77	0.3205480	585	187.35	0.9648625	0.2359056	186.90804
78	0.3205480	600	175.15	0.9771742	0.2434428	172.56763
79	0.3205480	615	167.15	0.9853374	0.2515124	158.65431
80	0.3205480	625	283.75	0.5422825	0.2295134	148.35273

(g) Comment: In this part, I use secant method to compute the implied volatility from the Black-Scholes model, then I choose 4 different time to marturity with 20 strike price to each time to marturity. I use these data to get a plot of the points in 3-dimensions in the space (K,T,σ), the 3D plot as follow:

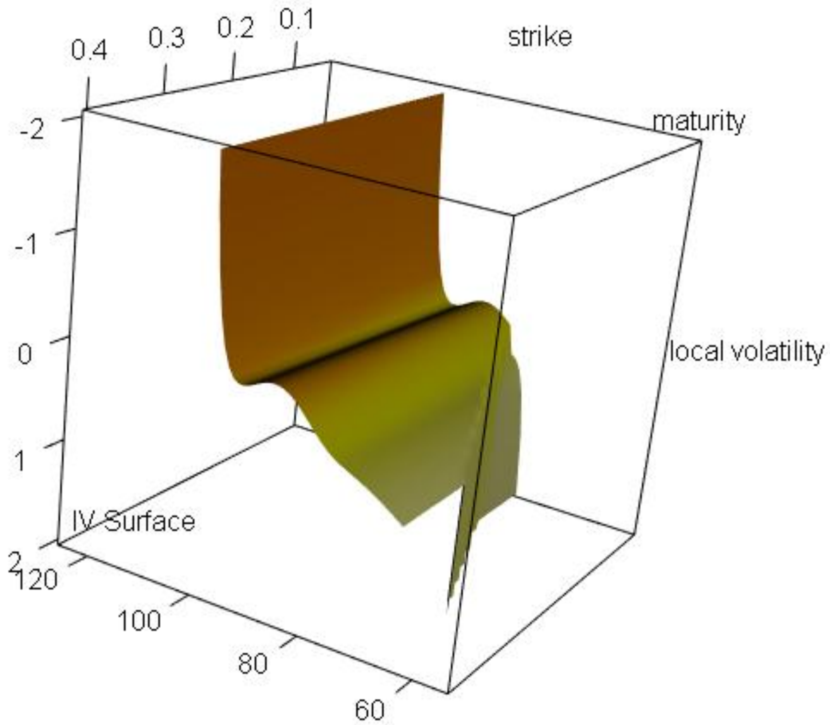


apply a cubic spline interpolation in 2 dimensions with the points based on the 80 data that I choosed in last part, then use these interpolated data to plot the implied volatility surface. The plot as follow:

For the points on the surface that I just calculated hold the non-arbitrage condition, because implied volatility is calculated by BS model, it shoud be constrained by the absence of arbitrage and from the plot we can see our implied volatility is vibrate in fixed interval.



The plot of Dupire's local volatility surface as follow:



Comment: the results in a table with the following columns: time to maturity, strike price, option market price, implied volatility, local volatility, price obtained using Dupire's local volatility as follow:

	time	Strike	price	Implied vol	Local vol	Market price
1	0.0396	82.5	4.715	0.5759753	0.57242003	6.411508e+00
2	0.0396	83.0	4.200	0.5474236	0.54268037	5.905746e+00
3	0.0396	83.5	3.525	0.5197315	0.51389685	5.408424e+00
4	0.0396	84.0	3.100	0.4919040	0.48505845	4.914051e+00
5	0.0396	84.5	2.670	0.4671214	0.45937721	4.444265e+00
6	0.0396	85.0	2.165	0.4400597	0.43147038	3.965635e+00
7	0.0396	85.5	1.795	0.4316572	0.42242523	3.620208e+00
8	0.0396	86.0	1.510	0.4258770	0.41603798	3.304689e+00
9	0.0396	86.5	1.110	0.4223200	0.41189673	3.017564e+00
10	0.0396	87.0	0.785	0.4225807	0.41159079	2.770559e+00
11	0.0396	87.5	0.585	0.4265114	0.41495660	2.562790e+00
12	0.0396	88.0	0.380	0.4331552	0.42102269	2.387042e+00
13	0.0396	88.5	0.245	0.4419957	0.42926271	2.238686e+00
14	0.0396	89.0	0.165	0.4616109	0.44822321	2.173767e+00
15	0.0396	89.5	0.105	0.4698530	0.45580254	2.042100e+00
16	0.0396	90.0	0.070	0.5281015	0.51312106	2.246266e+00
17	0.0396	90.5	0.060	0.5873993	0.57136813	2.459370e+00
18	0.0396	91.0	0.030	0.3750396	0.35987286	9.889409e-01
19	0.0396	91.5	0.025	0.6874056	0.66905812	2.769725e+00
20	0.0396	92.0	0.025	0.1198772	0.10672342	1.975410e-03
21	0.1500	65.0	0.000	0.6476439	0.64166267	2.302752e+01
22	0.1500	67.5	23.175	0.6118603	0.60010382	2.066598e+01
23	0.1500	70.0	20.075	0.5755053	0.55824436	1.831955e+01
24	0.1500	72.5	17.175	0.5364526	0.51388649	1.597480e+01
25	0.1500	75.0	14.950	0.4991969	0.47182586	1.367504e+01
26	0.1500	77.5	12.250	0.4600754	0.42823582	1.139902e+01
27	0.1500	80.0	9.930	0.4455590	0.41053650	9.461527e+00
28	0.1500	82.5	7.325	0.4345000	0.39652174	7.702073e+00
29	0.1500	85.0	4.965	0.4289586	0.38823457	6.179281e+00
30	0.1500	87.5	3.045	0.4276642	0.38428292	4.898854e+00
31	0.1500	90.0	1.455	0.4319648	0.38594276	3.881476e+00
32	0.1500	92.5	0.560	0.4465511	0.39779436	3.168634e+00
33	0.1500	95.0	0.175	0.4582503	0.40654537	2.558321e+00
34	0.1500	97.5	0.055	0.4824127	0.42740425	2.201632e+00
35	0.1500	100.0	0.045	0.5056907	0.44704359	1.908599e+00
36	0.1500	105.0	0.020	0.5519619	0.48503937	1.478232e+00
37	0.1500	110.0	0.040	0.5691486	0.49393307	9.842791e-01
38	0.1500	115.0	0.015	0.6741703	0.58520954	1.179966e+00
39	0.1500	120.0	0.050	0.7478243	0.64473554	1.178211e+00
40	0.1500	125.0	0.035	0.1148216	0.05296161	2.801058e-71
41	0.2890	65.0	0.000	0.6844950	0.64015931	2.482545e+01
42	0.2890	67.5	23.175	0.6472156	0.59614211	2.246739e+01
43	0.2890	70.0	20.075	0.6071680	0.54944537	2.007826e+01
44	0.2890	72.5	17.050	0.5631593	0.49887147	1.763871e+01
45	0.2890	75.0	14.845	0.5187381	0.44832688	1.519872e+01
46	0.2890	77.5	12.400	0.4730171	0.39686237	1.274995e+01

47	0.2890	80.0	9.950	0.4549239	0.37519352	1.079533e+01
48	0.2890	82.5	7.725	0.4396665	0.35668129	8.988713e+00
49	0.2890	85.0	5.675	0.4308147	0.34503604	7.424445e+00
50	0.2890	87.5	4.000	0.4285232	0.34031107	6.131414e+00
51	0.2890	90.0	2.345	0.4356971	0.34539179	5.172449e+00
52	0.2890	92.5	1.360	0.4508015	0.35849325	4.500171e+00
53	0.2890	95.0	0.680	0.4651045	0.37052466	3.923777e+00
54	0.2890	97.5	0.330	0.4962255	0.39936698	3.728694e+00
55	0.2890	100.0	0.160	0.5282520	0.42866507	3.598885e+00
56	0.2890	105.0	0.080	0.5437103	0.43759852	2.704333e+00
57	0.2890	110.0	0.030	0.5939213	0.48013765	2.490765e+00
58	0.2890	115.0	0.045	0.6127792	0.49091801	1.965542e+00
59	0.2890	120.0	0.065	0.7368691	0.60265140	2.780216e+00
60	0.2890	125.0	0.060	0.1322597	0.05546610	9.531270e-35
61	0.4000	65.0	0.000	0.7389485	0.66593807	2.655900e+01
62	0.4000	67.5	24.150	0.6989656	0.61779703	2.414412e+01
63	0.4000	70.0	19.225	0.6533735	0.56376284	2.162356e+01
64	0.4000	72.5	17.375	0.6031615	0.50508521	1.900722e+01
65	0.4000	75.0	14.475	0.5503591	0.44406999	1.632767e+01
66	0.4000	77.5	12.730	0.4947144	0.38036994	1.357685e+01
67	0.4000	80.0	10.375	0.4693379	0.35021653	1.144973e+01
68	0.4000	82.5	8.000	0.4481437	0.32479678	9.479795e+00
69	0.4000	85.0	6.250	0.4352622	0.30859731	7.797554e+00
70	0.4000	87.5	4.450	0.4301206	0.30092042	6.427859e+00
71	0.4000	90.0	2.970	0.4432461	0.31300908	5.634249e+00
72	0.4000	92.5	1.900	0.4572610	0.32578221	4.985828e+00
73	0.4000	95.0	1.130	0.4828772	0.35053618	4.697712e+00
74	0.4000	97.5	0.660	0.5149221	0.38152822	4.613549e+00
75	0.4000	100.0	0.335	0.5557985	0.42113912	4.757487e+00
76	0.4000	105.0	0.175	0.5980458	0.45814990	4.318309e+00
77	0.4000	110.0	0.065	0.6324118	0.48600751	3.858048e+00
78	0.4000	115.0	0.065	0.6990953	0.54548933	4.075373e+00
79	0.4000	120.0	0.035	0.7717020	0.60976702	4.436240e+00
80	0.4000	125.0	0.030	0.1107835	0.09954726	5.531638e-09

APPENDIX

Problem A: Asian Option Pricing using Monte Carlo Control Variate

(a)

```
# Function to calculate opntion price by BS formular
BS.Asian.geo <- function(S0, K, T, r, sigma,N){

  dt = T/N
  adjust.sigma <- sigma * sqrt((2*N+1)/(6*(N+1)))
  rho = 0.5 * (r-1/2*sigma^2+adjust.sigma^2)
  d1 = (log(S0/K)+(rho+adjust.sigma^2/2)*T) / (adjust.sigma*sqrt(T))
  d2 = (log(S0/K)+(rho-adjust.sigma^2/2)*T) / (adjust.sigma*sqrt(T))

  Price = exp(-r*T)*(S0*exp(rho*T)*pnorm(d1) - K*pnorm(d2))
  return(Price)
}
BS.Asian.geo(S0=100, K=100, T=5, r=0.03, sigma=0.3,N=5*252)
```

(b)

```
MC.Asian.arith <- function(S0, K, T, r, sigma,N,M){

  timestart<-Sys.time()
  #precompute constants
  dt = T/N

  sum_CT = 0
  sum_CT2 = 0

  St <- matrix(0,nrow = N+1,ncol = 1)
  for(j in 1:M){ #for each simulation
    St[1] = S0
    for(i in 1:N){ #for each time step
      ybi = rnorm(1)
      St[i+1] = St[i]*exp((r-0.5*sigma^2)*dt+sigma*sqrt(dt)*ybi)
    }
    ST = sum(St)/(N+1)
    CT = max(0, ST - K)

    sum_CT = sum_CT + CT
    sum_CT2 = sum_CT2 + CT^2
  }
  value = sum_CT/M*exp(-r*T) #discount option value to time 0
  SD = sqrt((sum_CT2 - sum_CT^2/M)*exp(-2*r*T)/(M-1))
  SE = SD/sqrt(M) #standard error
  CI = c(value - 1.96*SE,value + 1.96*SE)
  return(list(price = value, CI  = CI))
```

```
  timeend<-Sys.time()
  runningtime<-timeend-timestart
  print(runningtime)
}
MC.Asian.arith(S0=100, K=100, T=5, r=0.03, sig=0.3,N=5*252,M=1e+06)
```

(c)

```
MC.Asian.geo <- function(S0, K, T, r, sigma,N,M){

  #precompute constants
  dt = T/N

  sum_CT = 0

  St <- matrix(0,nrow = N+1,ncol = 1)
  for(j in 1:M){ #for each simulation
    St[1] = S0
    for(i in 1:N){ #for each time step
      ybi = rnorm(1)
      St[i+1] = St[i]*exp((r-0.5*sigma^2)*dt+sigma*sqrt(dt)*ybi)
    }
    ST = prod(St)^(1/(N+1))
    CT = max(0, ST - K)

    sum_CT = sum_CT + CT
  }
  value = sum_CT/M*exp(-r*T) #discount option value to time 0
  return(value)
}
MC.Asian.geo(S0=100, K=100, T=5, r=0.03, sig=0.3,N=5*252,M=1e+06)
```

(d)

```
b.star <- function(S0, K, T, r, sigma,M){

  #precompute constants
  N = 100
  dt = T/N

  X=Y=c();geo.ST=arith.ST=c()
  St <- matrix(0,nrow = N+1,ncol = 1)
  for(j in 1:M){ #for each simulation
    St[1] = S0
    for(i in 1:N){ #for each time step
      ybi = rnorm(1)
      St[i+1] = St[i]*exp((r-0.5*sigma^2)*dt+sigma*sqrt(dt)*ybi)
    }
    ST.arith = sum(St)/(N+1)
    ST.geo = prod(St)^(1/(N+1))
```

```
    X = c(X,max(0,ST.geo - K)) #geometric option price
    Y = c(Y,max(0, ST.arith - K)) #arithmetic option price
    geo.ST = c(geo.ST,ST.geo)
    arith.ST = c(arith.ST,ST.arith)
  }
  b = sum((X-mean(X))*(Y-mean(Y)))/sum((X-mean(X))^2)
  return(list(b=b,geo.price=geo.ST,arith.price=arith.ST))
}
b <- b.star(S0=100, K=100, T=5, r=0.03, sigma=0.3,M=1e+06)
b$geo.price;b$arith.price;b$b
```

(e)

```
error <- function(S0,K,T,r,sig,N,M){
  simulate.price <- MC.Asian.geo(S0, K, T, r, sig,N,M)
  theo.price <- BS.Asian.geo(S0, K, T, r, sig,N)
  error.geo <- theo.price - simulate.price
  return(error.geo)
}
error(S0=100,K=100,T=5,r=0.03,sig=0.3,N=100,M=1e+03)
```

(f)

```
find.M <- function(S0,K,T,r,sig,N,M){
  price.arith.star = c()
  for(i in 1:length(M)){
    price.arith.simu <- MC.Asian.arith(S0, K, T, r, sig,N,M[i])$price
    star.b <- b.star(S0, K, T, r, sig, M[i])$b
    err <- price.arith.simu - star.b * error(S0,K,T,r,sig,N,M[i])
    price.arith.star <- c(price.arith.star, err)
  }
  return(price.arith.star)
}
price.arith.star <- find.M(S0=100,K=100,T=5,r=0.03,sig=0.3,N=100,M=c(1e+03,1e+04,1e+0
5))
price.arith.star
```

bonus

```
AMZN <- read.csv("amzn.csv")
S0 <- 952.82
K <- AMZN$Strike
T <- AMZN$T
sigma <- AMZN$Volatility

amzn.bs.geo=amzn.arith=amzn.geo=amzn.err=amzn.b=amzn.m=c()
for(i in 1:25){
    amzn.bs.geo <- c(amzn.bs.geo,BS.Asian.geo(S0=S0,K=K[i],T=T[i],r=0.03,sigma=sigma[i],
N=10))
    amzn.arith <- c(amzn.arith,MC.Asian.arith(S0=S0,K=K[i],T=T[i],r=0.03,sig=sigma[i],N
```

```
=10,M=1e+03)$price)
    amzn.geo <- c(amzn.geo,MC.Asian.geo(S0=S0,K=K[i],T=T[i],r=0.03,sig=sigma[i],N=10,M=
1e+03))
    amzn.b <- c(amzn.b,b.star(S0=S0,K=K[i],T=T[i],r=0.03,sigma=sigma[i],M=1e+03)$b)
    amzn.err <- c(amzn.err,error(S0=S0,K=K[i],T=T[i],r=0.03,sig=sigma[i],N=10,M=1e+03))
    amzn.m <- c(amzn.m,find.M(S0=S0,K=K[i],T=T[i],r=0.03,sig=sigma[i],N=100,M=1e+03))
}
data.frame(amzn.bs.geo,amzn.arith,amzn.geo,amzn.b,amzn.err,amzn.m)
```

Problem B: A portfolio construction problem

1

```
library(quantmod)
stockData <- new.env()
XLF.symbols <- c('ACC','JPM','WFC','BAC','C','GS','USB','CB','MS','AXP',
                 'AADR','ACFC','AAME','AAT','AAXJ','ACNB','ABCB','ABE','ABR','BRK-B')
getSymbols(XLF.symbols, from="2012-01-01", env=stockData, src="yahoo")

ReturnMatrix=XLF.data=NULL
for(i in 1:length(XLF.symbols)){
  temp = get(XLF.symbols[i], pos=stockData)
  XLF.data = cbind(XLF.data,Cl(temp))
  ReturnMatrix = cbind(ReturnMatrix, (Cl(temp)-Op(temp)) / Op(temp)   )
  colnames(ReturnMatrix)[i] = XLF.symbols[i]
}
PCA <- princomp(ReturnMatrix,cor = TRUE) #by default R centers the variables. Scale als
o makes then sd=1
summary(PCA)
PCA$loadings
```

2

```
library(psych)
fit <- principal(cor(ReturnMatrix), nfactors=10, rotate="varimax", n.obs=dim(ReturnMat
rix)[1])
fit$loadings

library(Sim.DiffProc)
pick <- c("AXP.Close","AAXJ.Close","ABCB.Close","AAT.Close")
equity.pick <- XLF.data[,pick]

model.select <- function(data){
  #model1 Black-Scholes
  fx1 <- expression(theta[1]*x)
  gx1 <- expression(theta[2]*x)
  mod1 <- fitsde(data=as.ts(data),drift=fx1,diffusion=gx1,
                 start = list(theta1=1,theta2=1),pmle="shoji")

  #model2 mean reverting CEV
```

```r
  fx2 <- expression(theta[1]+theta[2]*x)
  gx2 <- expression(theta[3]*x^theta[4])
  mod2 <- fitsde(data=as.ts(data),drift=fx2,diffusion=gx2,
                 start = list(theta1=1,theta2=1,theta3=1,theta4=1),pmle="shoji")

  #model3 Strange 1
  fx3 <- expression(theta[1]*x)
  gx3 <- expression(theta[2]+theta[3]*x^theta[4])
  mod3 <- fitsde(data=as.ts(data),drift=fx3,diffusion=gx3,
                 start = list(theta1=1,theta2=1,theta3=1,theta4=1),pmle="shoji")

  #model4 particular CEV
  fx4 <- expression(theta[1]*x)
  gx4 <- expression(theta[2]*x^(2/3))
  mod4 <- fitsde(data=as.ts(data),drift=fx4,diffusion=gx4,
                 start = list(theta1=1, theta2=1),pmle="shoji")

  #model5 Strange 2
  fx5 <- expression(theta[1]+theta[2]*x)
  gx5 <- expression((theta[3]+theta[4]*log(x))*x)
  mod5 <- fitsde(data=as.ts(data),drift=fx5,diffusion=gx5,
                 start = list(theta1=1, theta2=1,theta3=1,theta4=1),pmle="shoji")

  ## Computes AIC
  AIC <- c(AIC(mod1),AIC(mod2),AIC(mod3),AIC(mod4),AIC(mod5))
  Test <- data.frame(AIC,row.names = c("mod1","mod2","mod3","mod4","mod5"))
  Bestmod <- rownames(Test)[which.min(Test[,1])]
  cat("best model: ",Bestmod,"\n")
  print(Test)
}
col1.AIC <- model.select(equity.pick[,1])  #mod2 AXP
col2.AIC <- model.select(equity.pick[,2])  #mod2 AAXJ
col3.AIC <- model.select(equity.pick[,3])  #mod1 ABCB
col4.AIC <- model.select(equity.pick[,4])  #mod2 AAT
```

3

```r
mat.cor <- cor(equity.pick)
mat.cor
```

4

```r
#model1 Black-Scholes
fx1 <- expression(theta[1]*x)
gx1 <- expression(theta[2]*x)
mod1 <- fitsde(data=as.ts(equity.pick[,3]),drift=fx1,diffusion=gx1,
               start = list(theta1=1,theta2=1),pmle="shoji")
ABCB.coef <- coef(mod1)

#model2 mean reverting CEV
```

```
fx2 <- expression(theta[1]+theta[2]*x)
gx2 <- expression(theta[3]*x^theta[4])
mod21 <- fitsde(data=as.ts(equity.pick[,1]),drift=fx2,diffusion=gx2,
              start = list(theta1=1,theta2=1,theta3=1,theta4=1),pmle="shoji") #AXP
mod22 <- fitsde(data=as.ts(equity.pick[,2]),drift=fx2,diffusion=gx2,
              start = list(theta1=1,theta2=1,theta3=1,theta4=1),pmle="shoji") #AAXJ
mod23 <- fitsde(data=as.ts(equity.pick[,1]),drift=fx2,diffusion=gx2,
              start = list(theta1=1,theta2=1,theta3=1,theta4=1),pmle="shoji") #AAT
AXP.coef <- coef(mod21)
AAXJ.coef <- coef(mod22)
AAT.coef <- coef(mod23)

MC.mod <- function(S0, T){
  #precompute constants
  N = 255
  M = 1e+03 #1000 paths
  dt = T/N

  set.seed(1)
  ST1=ST2=ST3=ST4=c()
  for(j in 1:M){ #for each simulation
    x <- matrix(rnorm(4*N),nrow=N,ncol=4)
    ep <- x%*%chol(mat.cor)
    e1 <- append(0,ep[,1])
    e2 <- append(0,ep[,2])
    e3 <- append(0,ep[,3])
    e4 <- append(0,ep[,4])
    S1=S0[,1] #initial value for asset 1
    S2=S0[,2] #initial value for asset 2
    S3=S0[,3] #initial value for asset 3
    S4=S0[,4] #initial value for asset 3
    for(i in 1:(N+1)){ #for each time step
      S1 <- S1 + (AXP.coef[1]-AXP.coef[2]*S2)*dt + e1[i]*(AXP.coef[3]*S2^AXP.coef[4])*s
qrt(dt) #AXP
      S2 <- S2 + (AAXJ.coef[1]-AAXJ.coef[2]*S2)*dt + e2[i]*(AAXJ.coef[3]*S2^AAXJ.coef
[4])*sqrt(dt) #AAXJ
      S3 <- S3*exp((ABCB.coef[1]-0.5*ABCB.coef[2]^2)*dt + e3[i]*ABCB.coef[2]*sqrt(dt))
#AAT
      S4 <- S4 + (AAT.coef[1]-AAT.coef[2]*S2)*dt + e4[i]*(AAT.coef[3]*S2^AAT.coef[4])*s
qrt(dt) #AXP #ABCB
    }
    ST1 <- c(ST1,S1)
    ST2 <- c(ST2,S2)
    ST3 <- c(ST3,S3)
    ST4 <- c(ST4,S4)
  }
  ST.mean <- c(mean(ST1),mean(ST2),mean(ST3),mean(ST4))
  ST.sd <- c(sd(ST1),sd(ST2),sd(ST3),sd(ST4))
  ST.skewness <- c(skewness(ST1),skewness(ST2),skewness(ST3),skewness(ST4))
```

```
  ST.kurtosis <- c(kurtosis(ST1),kurtosis(ST2),kurtosis(ST3),kurtosis(ST4))
  show <- data.frame(ST.mean,ST.sd,ST.skewness,ST.kurtosis)
  print(show)
}
MC.mod(S0=equity.pick[1,],T=1)
```

5

```
XLF <- read.csv("XLF.csv")
fx <- expression(theta[1]*x)
gx <- expression(theta[2]*x)
mod <- fitsde(data=as.ts(XLF$Close),drift=fx,diffusion=gx,
              start = list(theta1=1, theta2=1),pmle="shoji")
mu <- coef(mod)[1]
sigma <- coef(mod)[2]
mu;sigma;
```

6

```
XLF.return <- (XLF$Close-XLF$Open) / XLF$Open
newdata <- data.frame(XLF.return,equity.pick[,1],equity.pick[,2],equity.pick[,3],equit
y.pick[,4])
fit <- lm(XLF.return~., data = newdata)
weight <- coef(fit)
weight
```

7

```
basket <- function(type,s0,ST,T,m,n){

  nu = mu-sigma^2/2
  nu = as.numeric(nu)
  sum_CT = 0
  dt = T/n
  for(i in 1:m){
    x <- matrix(rnorm(400),nrow=100,ncol=4)
    ep <- x%*%chol(mat.cor)
    e1 <- append(0,ep[,1])
    e2 <- append(0,ep[,2])
    e3 <- append(0,ep[,3])
    e4 <- append(0,ep[,4])
    s1=s2=s3=s4=c()
    S1=s0[1] #initial value for asset 1
    S2=s0[2] #initial value for asset 2
    S3=s0[3] #initial value for asset 3
    S4=s0[4] #initial value for asset 3
    for(j in 1:(n+1)){
      S1 <- S1*exp(nu*dt+e1[j]*as.numeric(sigma)*sqrt(dt))
      S2 <- S2*exp(nu*dt+e2[j]*as.numeric(sigma)*sqrt(dt))
      S3 <- S3*exp(nu*dt+e3[j]*as.numeric(sigma)*sqrt(dt))
```

```
      S4 <- S4*exp(nu*dt+e4[j]*as.numeric(sigma)*sqrt(dt))
      s1 <- c(s1,S1)
      s2 <- c(s2,S2)
      s3 <- c(s3,S3)
      s4 <- c(s4,S4)
    }

    U = weight[2]*s1+weight[3]*s2+weight[4]*s3+weight[5]*s4 #price of basket stocks
    if(type=="ETF"){
      CT = max(0,U[101]-ST)
    }
    else if(type=="equity"){
      CT = max(0,ST - U[101])
    }
    sum_CT = sum_CT + CT
  }
  value = sum_CT/m
  return(value)
}
n=length(XLF$Close)
basket(type = "ETF",s0 = as.numeric(equity.pick[1,]),ST = XLF$Close[n],T=1,m=1000,n=100)
basket(type = "equity",s0 = as.numeric(equity.pick[1,]),ST = XLF$Close[n],T=1,m=1000,n=
100)
```

Problem C. Local volatility

(a)

```
library(readxl)
setwd("E:/621 computational method/Final")
data <- read_excel("SPX.xls",na = "")
row1 <- names(data)
S0 <- as.numeric(row1[2])
r <- as.numeric(row1[3])/100
data <- data.frame(data)
data <- data[!is.na(data[,1]),]
SPX.data <- data[-1,2:4]
SPX.data<- SPX.data[!duplicated(SPX.data[,1:2]),]
colnames(SPX.data) <- c('muturity','strike','price')

T <- as.numeric(SPX.data[,1])
K <- as.numeric(SPX.data[,2])
MPrice <- as.numeric(SPX.data[,3])

# Function to calculate opntion price by BS formular
BS <- function(S0, K, tau, r, sigma,div){

  d1 <- (log(S0/K)+(r-div+sigma^2/2)*tau) / (sigma*sqrt(tau))
```

```
  d2 <- d1 - sigma*sqrt(tau)
  Price <- S0*exp(-div*tau)*pnorm(d1) - K*exp(-r*tau)*pnorm(d2)
  return(Price)
}

#Function to calculate error between BS and Market price
err <- function(S0,K,r,tau,sig,div,MPrice){
  BS(S0,K,tau,r,sig,div) - MPrice
}

# Function to find BS Implied Vol using Secant Method
secant <- function(S0,K,r,tau,div,MPrice){
  sig <- c()
  #loop for every strike price and market price
  for(i in 1:length(K)){
    x0 <- -1
    x1 <- 1
    err0 <- err(S0,K[i],r,tau[i],x0,div,MPrice[i])
    err1 <- err(S0,K[i],r,tau[i],x1,div,MPrice[i])
    #Loop until that the value of function to sigma is less than tolerance level 1e-4
    while(abs(x1-x0) > 1e-4){
      x <- x1 - err1*(x1 - x0) / (err1 - err0) # Calculate the new x value
      x0 <- x1
      x1 <- x
    }
    sig <- c(x,sig)
  }
  return(sig)
}
Ivol <- secant(S0=S0,K=K,r=r,tau=T,div=0,MPrice=MPrice) #one month

t <- unique(T)
t1 <- rep(t[1],20)
t2 <- rep(t[2],20)
t3 <- rep(t[3],20)
t4 <- rep(t[4],20)
k1=k2=k3=k4=c()
for(i in 1:nrow(SPX.data)){
  if(T[i] == t[1]){
    k1 <- c(k1,SPX.data[i,2])
  }
  if(T[i] == t[2]){
    k2 <- c(k2,SPX.data[i,2])
  }
  if(T[i] == t[3]){
    k3 <- c(k3,SPX.data[i,2])
  }
  if(T[i] == t[4]){
    k4 <- c(k4,SPX.data[i,2])
```

```
  }
}
k1 <- sort(as.numeric(unique(k1)))[1:20]
k2 <- sort(as.numeric(unique(k2)))[33:52]
k3 <- sort(as.numeric(unique(k3)))[1:20]
k4 <- sort(as.numeric(unique(k4)))[1:20]
newdata <- (data.frame(K,T,Ivol))
Inv1=Inv2=Inv3=Inv4=c()
mprice1=mprice2=mprice3=mprice4=c()
for(i in 1:length(Ivol)){
  if(T[i] == t[1]){
    for(j in 1:20){
      if(K[i] == k1[j]){
        Inv1 = c(Inv1,Ivol[i])
        mprice1 = c(mprice1,SPX.data[i,3])
      }
    }
  }
  if(T[i] == t[2]){
    for(j in 1:20){
      if(K[i] == k2[j]){
        Inv2 = c(Inv2,Ivol[i])
        mprice2 = c(mprice2,SPX.data[i,3])
      }
    }
  }
  if(T[i] == t[3]){
    for(j in 1:20){
      if(K[i] == k3[j]){
        Inv3 = c(Inv3,Ivol[i])
        mprice3 = c(mprice3,SPX.data[i,3])
      }
    }
  }
  if(T[i] == t[4]){
    for(j in 1:20){
      if(K[i] == k4[j]){
        Inv4 = c(Inv4,Ivol[i])
        mprice4 = c(mprice4,SPX.data[i,3])
      }
    }
  }
}

# Creat 3D plot of volatilities versus K & T
library(scatterplot3d)
COLOR=c(rep("red",20),rep("lightgrey",20),rep("yellow",20),rep("lightblue",20))
scatterplot3d(c(k1,k2,k3,k4),c(t1,t2,t3,t4),c(Inv1,Inv2,Inv3,Inv4),main="Secant 3D plo
t",
```

```
                xlab="Strike Price",ylab = "Maturity",zlab = "Vol",color = COLOR,pch=16,ty
pe = "p")
```

(b)

```
library(rgl)
library(akima)

k.pick <- c(k1,k2,k3,k4)
t.pick <- c(t1,t2,t3,t4)
Inv <- c(Inv1,Inv2,Inv3,Inv4)
mprice <- as.numeric(c(mprice1,mprice2,mprice3,mprice4))
call <- data.frame(K,T,Ivol)

t.spline <- seq(min(t.pick),max(t.pick),0.01)
k.spline <- seq(min(k.pick),max(k.pick),1)

# By setting to use less points, it will "stretch" the surface over those points.
xyz <- with(call, interp(x=t.pick, y=k.pick, z=Inv,
                          xo=t.spline, yo=k.spline,
                          extrap=TRUE,linear = FALSE,duplicate = "mean" ))

with(xyz, persp3d(x,y,z, col=heat.colors(length(z))[rank(z)],ylim = c(min(k.pick),max
(k.pick)),zlim = c(-2,2),xlab='maturity',
                  ylab='strike', zlab='Implied volatility', main='IV Surface'))
```

(d)

```
#using implied volatility to calculate local volatility
local.vol <- function(S0,K,tau,r,q,sig){
  dt=0.01
  dk=1
  dsig.dt = (xyz$z[length(xyz$z)/2+1]-xyz$z[length(xyz$z)/2])/(xyz$x[length(xyz$x)/2+1]
-xyz$x[length(xyz$x)/2])
  dsig.dk = (xyz$z[length(xyz$z)/2+1]-xyz$z[length(xyz$z)/2])/(xyz$y[length(xyz$y)/2+1]
-xyz$y[length(xyz$y)/2])
  dsig.dk2 = (xyz$z[length(xyz$z)/2]-xyz$z[length(xyz$z)/2])/(xyz$y[length(xyz$y)/2+1]
-xyz$y[length(xyz$y)/2])^2
  d1 = (log(S0/K)+(r-q+1/2*sig^2)*tau)/sqrt(tau)
  term1 = 2*sig*dsig.dt*tau
  term2 = 2*sig*(r-q)*tau*K*dsig.dk
  dividend = term1+sig^2+term2
  term3 = (1 + K*d1*dsig.dk*sqrt(tau))^2
  term4 = K^2*tau*sig*(dsig.dk2-d1*dsig.dk2^2*sqrt(tau))
  divider = term3+term4
  x = sqrt(abs(dividend/divider))
  return(x)
}
lv <- local.vol(S0=S0,K=k.pick,tau = t.pick,r=r,q=0,sig = Inv)
```

```r
xyz1 <- with(call, interp(x=t.pick, y=k.pick, z=lv,
                         xo=t.spline, yo=k.spline,
                         extrap=TRUE,linear = FALSE,duplicate = "mean" ))

with(xyz1, persp3d(x,y,z, col=heat.colors(length(z))[rank(z)],ylim = c(min(k.pick),max
(k.pick)),zlim = c(-2,2),xlab='maturity',
                  ylab='strike', zlab='Iocal volatility', main='IV Surface'))
```

(e)

```r
price.local <- BS(S0=S0, K=k.pick, tau=t.pick, r=r, sigma=lv,div=0)

par(mfrow=c(1,2))
plot(price.local,ylab = "option price",main = "price by local vol")
plot(mprice,ylab = "option price",main = "market price")
```

(f)

```r
table1 <- data.frame(t.pick,k.pick,mprice,Inv,lv,price.local)
colnames(table1) <- c('time to maturity','Strike price','market price','implied volatil
ity','local volatility','price by localVol')
table1

price.bs <- BS(S0=S0, K=k.pick, tau=t.pick, r=r, sigma=Inv,div=0)
table2 <- data.frame(Inv,lv,price.bs,price.local)
colnames(table1) <- c('implied volatility','local volatility','Black Scholes price','pr
ice by localVol')
library(xlsx)
write.csv(table2, file="SPXvolitality.csv")
```

(g)

```r
data <- read_excel("JPM.xls",na = "")
row1 <- names(data)
S0 <- as.numeric(row1[2])
r <- as.numeric(row1[3])/100
data <- data.frame(data)
data <- data[!is.na(data[,1]),]
SPX.data <- data[-1,2:4]
SPX.data<- SPX.data[!duplicated(SPX.data[,1:2]),]
colnames(SPX.data) <- c('muturity','strike','price')
T <- as.numeric(SPX.data[,1])
K <- as.numeric(SPX.data[,2])
MPrice <- as.numeric(SPX.data[,3])
Ivol <- secant(S0=S0,K=K,r=r,tau=T,div=0,MPrice=MPrice) #one month
t <- unique(T)
t1 <- rep(t[1],20)
t2 <- rep(t[2],20)
t3 <- rep(t[3],20)
t4 <- rep(t[4],20)
```

```
k1=k2=k3=k4=c()
for(i in 1:nrow(SPX.data)){
  if(T[i] == t[1]){
    k1 <- c(k1,SPX.data[i,2])
  }
  if(T[i] == t[2]){
    k2 <- c(k2,SPX.data[i,2])
  }
  if(T[i] == t[3]){
    k3 <- c(k3,SPX.data[i,2])
  }
  if(T[i] == t[4]){
    k4 <- c(k4,SPX.data[i,2])
  }
}
k1 <- sort(as.numeric(unique(k1)))[1:20]
k2 <- sort(as.numeric(unique(k2)))[1:20]
k3 <- sort(as.numeric(unique(k3)))[1:20]
k4 <- sort(as.numeric(unique(k4)))[1:20]
newdata <- (data.frame(K,T,Ivol))
Inv1=Inv2=Inv3=Inv4=c()
mprice1=mprice2=mprice3=mprice4=c()
for(i in 1:length(Ivol)){
  if(T[i] == t[1]){
    for(j in 1:20){
      if(K[i] == k1[j]){
        Inv1 = c(Inv1,Ivol[i])
        mprice1 = c(mprice1,SPX.data[i,3])
      }
    }
  }
  if(T[i] == t[2]){
    for(j in 1:20){
      if(K[i] == k2[j]){
        Inv2 = c(Inv2,Ivol[i])
        mprice2 = c(mprice2,SPX.data[i,3])
      }
    }
  }
  if(T[i] == t[3]){
    for(j in 1:20){
      if(K[i] == k3[j]){
        Inv3 = c(Inv3,Ivol[i])
        mprice3 = c(mprice3,SPX.data[i,3])
      }
    }
  }
  if(T[i] == t[4]){
    for(j in 1:20){
```

```
      if(K[i] == k4[j]){
        Inv4 = c(Inv4,Ivol[i])
        mprice4 = c(mprice4,SPX.data[i,3])
      }
    }
  }
}

# Creat 3D plot of volatilities versus K & T
library(scatterplot3d)
COLOR=c(rep("red",20),rep("lightgrey",20),rep("yellow",20),rep("lightblue",20))
scatterplot3d(c(k1,k2,k3,k4),c(t1,t2,t3,t4),c(Inv1,Inv2,Inv3,Inv4),main="Secant 3D plo
t",
              xlab="Strike Price",ylab = "Maturity",zlab = "Vol",color = COLOR,pch=16,ty
pe = "p")

##(b)
k.pick <- c(k1,k2,k3,k4)
t.pick <- c(t1,t2,t3,t4)
Inv <- c(Inv1,Inv2,Inv3,Inv4)
mprice <- as.numeric(c(mprice1,mprice2,mprice3,mprice4))
call <- data.frame(K,T,Ivol)
t.spline <- seq(min(t.pick),max(t.pick),0.01)
k.spline <- seq(min(k.pick),max(k.pick),1)
xyz <- with(call, interp(x=t.pick, y=k.pick, z=Inv,
                         xo=t.spline, yo=k.spline,
                         extrap=TRUE,linear = FALSE,duplicate = "mean" ))
with(xyz, persp3d(x,y,z, col=heat.colors(length(z))[rank(z)],ylim = c(min(k.pick),max
(k.pick)),zlim = c(-2,2),xlab='maturity',
                  ylab='strike', zlab='Implied volatility', main='IV Surface'))

##(d)
lv <- local.vol(S0=S0,K=k.pick,tau = t.pick,r=r,q=0,sig = Inv)
xyz1 <- with(call, interp(x=t.pick, y=k.pick, z=lv,
                          xo=t.spline, yo=k.spline,
                          extrap=TRUE,linear = FALSE,duplicate = "mean" ))

with(xyz1, persp3d(x,y,z, col=heat.colors(length(z))[rank(z)],ylim = c(min(k.pick),max
(k.pick)),zlim = c(-2,2),xlab='maturity',
                   ylab='strike', zlab='Iocal volatility', main='IV Surface'))
##(e)
price.local <- BS(S0=S0, K=k.pick, tau=t.pick, r=r, sigma=lv,div=0)
##(f)
table1 <- data.frame(t.pick,k.pick,mprice,Inv,lv,price.local)
colnames(table1) <- c('time to maturity','Strike price','market price','implied volatil
ity','local volatility','price by localVol')
table1

price.bs <- BS(S0=S0, K=k.pick, tau=t.pick, r=r, sigma=Inv,div=0)
```

```r
table2 <- data.frame(Inv,lv,price.bs,price.local)
colnames(table1) <- c('implied volatility','local volatility','Black Scholes price','pr
ice by localVol')
library(xlsx)
write.xlsx(table2, file="JPMvolitality.xls")
```