Project1

Gargiulo,Santonicola,Ambrosio

28/9/2020

Contents

Applicazione Shiny interattiva: clicca qui

Serie storiche: API calling

La serie storiche giornaliere delle azioni US scelte sono state reperite attraverso il servizio API offerto da alphavantage e il pacchetto alphavantager, che permette con funzioni semplici di interrogare il database di alphavantage e scaricare le serie storiche desiderate.

Dopo un'elaborazione preliminare dei dati, i prezzi di chiusura si presentano come segue:

```
knitr::kable(data[1:20,],caption = "Prezzi di chiusura", digits = 4, "pipe")
```

Table 1: Prezzi di chiusura

timestamp	AMD	LMT	MSFT	NEM	PFE	UTX
2019-04-17	27.49	310.50	121.77	34.18	39.88	135.79
2019-04-18	27.68	314.26	123.37	33.04	39.38	137.00
2019-04-22	28.18	315.26	123.76	32.55	38.98	136.91
2019-04-23	27.97	333.10	125.44	32.37	39.42	140.02
2019-04-24	28.46	330.12	125.01	32.20	39.37	139.59
2019-04-25	27.66	328.87	129.15	31.63	39.61	139.73
2019-04-26	27.88	328.33	129.89	31.46	39.97	140.78
2019-04-29	27.69	328.59	129.77	30.76	39.59	142.12
2019-04-30	27.63	333.33	130.60	31.06	40.61	142.61
2019-05-01	26.81	331.85	127.88	30.32	40.77	141.58
2019-05-02	28.29	332.55	126.21	30.10	41.01	139.80
2019-05-03	28.22	334.07	128.90	30.22	41.39	141.63
2019-05-06	27.42	335.72	128.15	29.94	41.65	140.20
2019-05-07	26.66	330.90	125.52	30.72	40.83	135.44
2019-05-08	27.09	334.18	125.51	30.20	40.95	137.07
2019-05-09	27.21	339.36	125.50	29.93	40.64	136.86
2019-05-10	27.96	341.35	127.13	30.15	40.72	137.17
2019-05-13	26.24	335.37	123.35	30.91	40.57	131.96
2019-05-14	27.32	336.17	124.73	30.83	40.66	133.98
2019-05-15	27.58	334.15	126.02	30.69	41.15	133.98

Log-rendimenti e statistiche

La sequenza di rendimenti target è la seguente (l'output di cui sotto si limita solo ai primi 10 elementi):

```
lo.tar=0.005
up.tar=0.5
target <- seq(lo.tar,up.tar,by=0.005)
target[1:10]
```

```
##  [1] 0.005 0.010 0.015 0.020 0.025 0.030 0.035 0.040 0.045 0.050
```

Essa va da 0.005 a 0.5, in step di 0.005.

A questo punto possiamo calcolare i log-rendimenti giornalieri.

```
R <- diff(as.matrix(log(data[,2:ncol(data)])))
time <- data[,1]
R <- data.frame(time=time[-1],R)
knitr::kable(R[1:10,],caption = "Log-rendimenti", digits = 4, "pipe")
```

Table 2: Log-rendimenti

time	AMD	LMT	MSFT	NEM	PFE	UTX
2019-04-18	0.0069	0.0120	0.0131	-0.0339	-0.0126	0.0089
2019-04-22	0.0179	0.0032	0.0032	-0.0149	-0.0102	-0.0007
2019-04-23	-0.0075	0.0550	0.0135	-0.0055	0.0112	0.0225
2019-04-24	0.0174	-0.0090	-0.0034	-0.0053	-0.0013	-0.0031
2019-04-25	-0.0285	-0.0038	0.0326	-0.0179	0.0061	0.0010
2019-04-26	0.0079	-0.0016	0.0057	-0.0054	0.0090	0.0075
2019-04-29	-0.0068	0.0008	-0.0009	-0.0225	-0.0096	0.0095
2019-04-30	-0.0022	0.0143	0.0064	0.0097	0.0254	0.0034
2019-05-01	-0.0301	-0.0044	-0.0210	-0.0241	0.0039	-0.0072
2019-05-02	0.0537	0.0021	-0.0131	-0.0073	0.0059	-0.0127

Calcoliamo rendimenti attesi μ e matrice Σ

```
R <- diff(as.matrix(log(data[,2:ncol(data)])))
e.r <- matrix(apply(R,2,mean)) #Vettore dei rendimenti attesi
rownames(e.r) <- names(data[2:length(data)])
colnames(e.r) <- "mu"
s.d <-matrix(apply(R,2,sd)) # Deviazione standard rendimenti
Sig <- cov(R) #Matrice varianze covarianze (S nella parte teorica allegata)

knitr::kable(e.r,caption = "mu", digits = 4, "pipe")
```

Table 3: mu

	mu
AMD	0.0029
LMT	0.0006
MSFT	0.0015
NEM	0.0016
PFE	-0.0003
UTX	-0.0023

```
knitr::kable(Sig,caption = "var-cov", digits = 8, "pipe")
```

Table 4: var-cov

	AMD	LMT	MSFT	NEM	PFE	UTX
AMD	0.00130114	0.00029699	0.00056485	0.00021049	0.00026718	0.00043842
LMT	0.00029699	0.00049298	0.00030227	0.00009003	0.00025357	0.00041327
MSFT	0.00056485	0.00030227	0.00055575	0.00011539	0.00025263	0.00039575
NEM	0.00021049	0.00009003	0.00011539	0.00059609	0.00005288	0.00016283
PFE	0.00026718	0.00025357	0.00025263	0.00005288	0.00034774	0.00021739
UTX	0.00043842	0.00041327	0.00039575	0.00016283	0.00021739	0.00165818

Utilizziamo poi la funzione solve() per calcolare l'inversa della matrice Σ e creiamo altresì un vettore unitario con lunghezza pari al numero di società presenti nel portafoglio.

```
S.inv <- solve(Sig)   #Inversa della Matrice delle varianze e covarianze
unit <- matrix(rep(1,length(e.r))) #Vettore unitario di lunghezza pari al
                                   #numero di STOCK
```

Calcolo vettore ottimale di pesi

Adesso procediamo a calcolare i coefficienti a, b, c necessari per ottenere il vettore di pesi ottimale w^*:

$$a = \mu' \mathbf{S}^{-1} \mu$$
$$b = \mu' \mathbf{S}^{-1} \mathbf{1}$$
$$c = \mathbf{1}' \mathbf{S}^{-1} \mathbf{1}$$

```
e.r <- matrix(apply(R,2,mean))
A<-t(e.r)%*%S.inv%*%e.r
B<-t(e.r)%*%S.inv%*%unit
C<-t(unit)%*%S.inv%*%unit
data.frame(A=A,B=B,C=C)
```

```
##           A        B        C
## 1 0.0234492 2.184202 4286.679
```

Fissato allora un valore del rendimento target possiamo procedere al calcolo del vettore w^*, che minimizza la varianza del portafoglio dato un livello del rendimento target.

$$w^* = \mathbf{S}^{-1} \frac{(mc - b)\mu \ + \ (a - \mu b)\mathbf{1}}{ac - b^2}$$

Possiamo adesso calcolare la varianza corrispondente ad un portafoglio formato dai titoli con pesi pari a w^*

$$\sigma_R^2 = w^{*'} \mathbf{S} w^*$$

Scegliamo ad esempio $m = 0.05$ e calcoliamo il vettore dei pesi con la corrispondente varianza:

```
m = 0.05 #Fisso rendimento target

num <- e.r%*%(m*C-B)+unit%*%(A-m*B)   #Numeratore della formula per il
                                      #calcolo del vettore dei pesi

den <- A*C - B^2                      #Denominatore

w.star <- S.inv%*%num%*%(1/den)    #Vettore dei pesi

var.star <- t(w.star)%*%Sig%*%w.star #Calcoliamo la varianza del portafoglio
                                     #dati i pesi ottimali w*

colnames(w.star)="weights"

res <- list("weights" =w.star,
            "variance"=var.star)
res
```

```
## $weights
##         weights
## AMD    4.377913
## LMT    5.805366
## MSFT   6.365009
## NEM    3.826951
## PFE  -12.978931
## UTX   -6.396308
##
## $variance
##           [,1]
## [1,] 0.1098893
```

La procedura di cui sopra è stata applicata per ogni livello della sequenza di rendimenti target attraverso una funzione personalizzata in R mediante l'utilizzo di un for loop. La funzione è la seguente:

```
calc <- function(data,up.tar=0.5,lo.tar=0.005){

  target <- seq(lo.tar,up.tar,by=0.005)  # Crea la sequenza di rendimenti target
R <- diff(as.matrix(log(data[,2:ncol(data)]))) #Rendimenti daily logaritmici
                                                #(r nel file di teoria)

e.r <- matrix(apply(R,2,mean)) #Vettore dei rendimenti attesi

s.d <-matrix(apply(R,2,sd)) # Deviazione standard rendimenti

Sig <- cov(R) #Matrice varianze covarianze
              #(S nella parte teorica allegata)

S.inv <- solve(Sig)   #Inversa della Matrice delle varianze e covarianze

unit <- matrix(rep(1,length(e.r))) #Vettore unitario di lunghezza
                                   # pari al numero di azioni presenti




#Coefficienti a,b,c
A<-t(e.r)%*%S.inv%*%e.r
B<-t(e.r)%*%S.inv%*%unit
C<-t(unit)%*%S.inv%*%unit


#res <- matrix(0,ncol=ncol(data)-1,nrow=length(target))

all.w <- list() #Lista vuota ove confluiranno i risultati del for loop
```

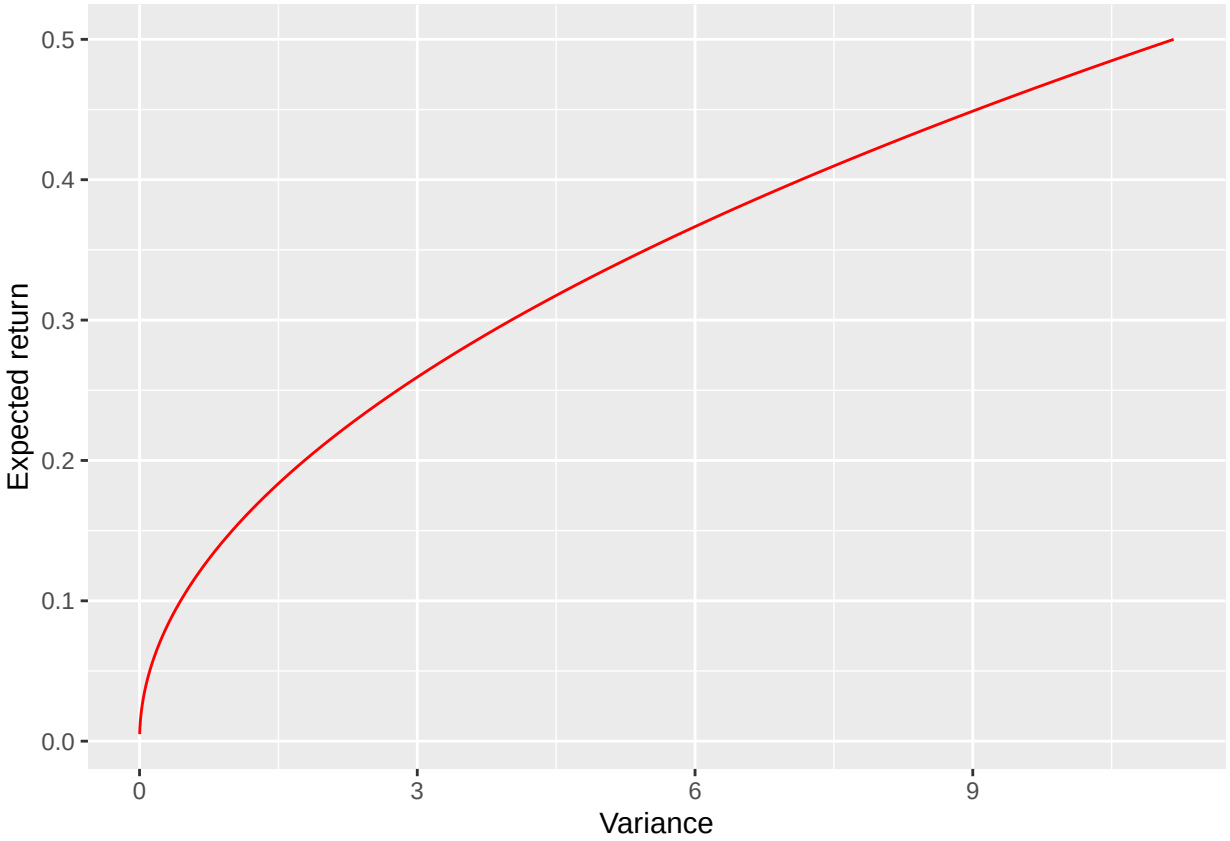
```
var.vec <- vector("numeric",length=length(target))
#vettore vuoto dove confluiranno i risultati del for loop


# FOR LOOP

for (i in 1:length(target)){

m <- target[i] #Rendimento target


num <- e.r%*%(m*C-B)+unit%*%(A-m*B) #Numeratore della formula per il calcolo
                                    #del vettore dei pesi

den <- A*C - B^2                        #Denominatore

w.star <- S.inv%*%num%*%(1/den)    #Vettore dei pesi


var.star <- t(w.star)%*%Sig%*%w.star
#Calcoliamo la varianza del portafoglio dati i pesi ottimali w*


res <- list("weights" =w.star,
            "variance"=var.star)

all.w[[i]] <- res
# Conterrà i pesi ottimali e la varianza per ogni possibile rendimento target


var.vec[i] <- var.star
#Conterrà solo la varianza per ogni possibile rendimento target

#t(w.star)%*%e.r    #prova che il vettore ottimale dei pesi trasposto * per i
#sum(w.star)      #prova che la somma dei pesi è 1
#t(w.star)%*%unit   #prova che la somma dei pesi è 1
}

plot.data <- cbind(target,var.vec) #Creare dataframe da cui plottare grafico

names(all.w) <- as.character(target) #Associare nomi alla lista

invisible(list("all.w"=all.w,"plot"=plot.data))}
```

La funzione permette anche di caratterizzare l'intervallo entro cui calcolare i rendimenti target. L'output della funzione è una lista che comprende a sua volta una lista e un data.frame. La prima

contiene il vettore w^* e la rispettiva varianza per ogni livello del rendimento target. Il data.frame, invece, serve per il grafico. Infatti contiene sia i livelli target sia la varianza minima corrispondente associata a tale livello.

Plot

```
output <- calc(data)
xx <- as.data.frame(output[["plot"]])

xxx <- matrix(0,nrow=6,ncol=2)

for(i in 1:6){

  xxx[i,1] = mean(R[,i])
  xxx[i,2] = sd(R[,i])^2

}
xxx <- as.data.frame(xxx)
pick <- c("AMD","LMT","MSFT","NEM","PFE","UTX")
colnames(xxx) <- c("mu","sig")
rownames(xxx) <- pick


ggplot(data=xx, aes(x=var.vec, y=target))+geom_line(color="red")+
  xlab("Variance")+ylab("Expected return")
```


0.5
0.4
0.3
0.2
0.1
0.0
Expected return
0
3
6
9
Variance

Output

Infine abbiamo costruito una tabella che sintetizza al meglio i risultati ottenuti.

```
df.pick <- as.matrix(data[,pick])


dat <- as.data.frame(output[["plot"]])

weig <- array(NA, dim= c(length(pick),1,length(output$all.w))) #creiamo array
#vuoto in cui estrarre il vettore dei pesi per ogni livello del rend. target

for(i in 1:length(output$all.w)){

  weig[,1,i] <- output %>% extract2(c(1,i,1)) #con extract2 estraggo gli elementi
  #dalla lista e li salvo nella posizione i-esima dell'array

}

weig.f <- adply(weig, c(2,3))[,c(-1,-2)] #trasforma l'array in un data.frame

names(weig.f) <- pick

dat.f <- cbind(dat,weig.f) #Uniamo i due data.frame

dat.f[,c(-1)] <- round(dat.f[,c(-1)], digits = 4) #Arrotondiamo i valori dei pesi

knitr::kable(dat.f,caption = "Risultati","pipe") #Creo tabella
```

Table 5: Risultati

target	var.vec	AMD	LMT	MSFT	NEM	PFE	UTX
0.005	0.0011	0.3497	0.6386	0.6839	0.6431	-0.7351	-0.5802
0.010	0.0043	0.7973	1.2127	1.3152	0.9968	-2.0956	-1.2265
0.015	0.0096	1.2449	1.7868	1.9464	1.3506	-3.4560	-1.8727
0.020	0.0172	1.6925	2.3609	2.5776	1.7044	-4.8164	-2.5189
0.025	0.0271	2.1400	2.9350	3.2089	2.0581	-6.1768	-3.1652
0.030	0.0392	2.5876	3.5090	3.8401	2.4119	-7.5372	-3.8114
0.035	0.0535	3.0352	4.0831	4.4713	2.7657	-8.8977	-4.4576
0.040	0.0701	3.4828	4.6572	5.1025	3.1194	-10.2581	-5.1038
0.045	0.0889	3.9303	5.2313	5.7338	3.4732	-11.6185	-5.7501
0.050	0.1099	4.3779	5.8054	6.3650	3.8270	-12.9789	-6.3963
0.055	0.1332	4.8255	6.3794	6.9962	4.1807	-14.3394	-7.0425
0.060	0.1587	5.2731	6.9535	7.6275	4.5345	-15.6998	-7.6888
0.065	0.1864	5.7206	7.5276	8.2587	4.8883	-17.0602	-8.3350
0.070	0.2164	6.1682	8.1017	8.8899	5.2420	-18.4206	-8.9812
0.075	0.2487	6.6158	8.6758	9.5212	5.5958	-19.7810	-9.6275

target	var.vec	AMD	LMT	MSFT	NEM	PFE	UTX
0.080	0.2831	7.0634	9.2499	10.1524	5.9496	-21.1415	-10.2737
0.085	0.3198	7.5109	9.8239	10.7836	6.3033	-22.5019	-10.9199
0.090	0.3588	7.9585	10.3980	11.4149	6.6571	-23.8623	-11.5661
0.095	0.4000	8.4061	10.9721	12.0461	7.0108	-25.2227	-12.2124
0.100	0.4434	8.8537	11.5462	12.6773	7.3646	-26.5831	-12.8586
0.105	0.4890	9.3012	12.1203	13.3085	7.7184	-27.9436	-13.5048
0.110	0.5369	9.7488	12.6943	13.9398	8.0721	-29.3040	-14.1511
0.115	0.5871	10.1964	13.2684	14.5710	8.4259	-30.6644	-14.7973
0.120	0.6395	10.6439	13.8425	15.2022	8.7797	-32.0248	-15.4435
0.125	0.6941	11.0915	14.4166	15.8335	9.1334	-33.3853	-16.0898
0.130	0.7509	11.5391	14.9907	16.4647	9.4872	-34.7457	-16.7360
0.135	0.8100	11.9867	15.5647	17.0959	9.8410	-36.1061	-17.3822
0.140	0.8714	12.4342	16.1388	17.7272	10.1947	-37.4665	-18.0285
0.145	0.9349	12.8818	16.7129	18.3584	10.5485	-38.8269	-18.6747
0.150	1.0007	13.3294	17.2870	18.9896	10.9023	-40.1874	-19.3209
0.155	1.0688	13.7770	17.8611	19.6208	11.2560	-41.5478	-19.9671
0.160	1.1391	14.2245	18.4352	20.2521	11.6098	-42.9082	-20.6134
0.165	1.2116	14.6721	19.0092	20.8833	11.9636	-44.2686	-21.2596
0.170	1.2863	15.1197	19.5833	21.5145	12.3173	-45.6290	-21.9058
0.175	1.3633	15.5673	20.1574	22.1458	12.6711	-46.9895	-22.5521
0.180	1.4426	16.0148	20.7315	22.7770	13.0249	-48.3499	-23.1983
0.185	1.5241	16.4624	21.3056	23.4082	13.3786	-49.7103	-23.8445
0.190	1.6078	16.9100	21.8796	24.0395	13.7324	-51.0707	-24.4908
0.195	1.6937	17.3576	22.4537	24.6707	14.0862	-52.4312	-25.1370
0.200	1.7819	17.8051	23.0278	25.3019	14.4399	-53.7916	-25.7832
0.205	1.8724	18.2527	23.6019	25.9331	14.7937	-55.1520	-26.4294
0.210	1.9650	18.7003	24.1760	26.5644	15.1475	-56.5124	-27.0757
0.215	2.0599	19.1478	24.7500	27.1956	15.5012	-57.8728	-27.7219
0.220	2.1571	19.5954	25.3241	27.8268	15.8550	-59.2333	-28.3681
0.225	2.2565	20.0430	25.8982	28.4581	16.2088	-60.5937	-29.0144
0.230	2.3581	20.4906	26.4723	29.0893	16.5625	-61.9541	-29.6606
0.235	2.4620	20.9381	27.0464	29.7205	16.9163	-63.3145	-30.3068
0.240	2.5681	21.3857	27.6205	30.3518	17.2701	-64.6750	-30.9531
0.245	2.6764	21.8333	28.1945	30.9830	17.6238	-66.0354	-31.5993
0.250	2.7870	22.2809	28.7686	31.6142	17.9776	-67.3958	-32.2455
0.255	2.8998	22.7284	29.3427	32.2454	18.3314	-68.7562	-32.8917
0.260	3.0148	23.1760	29.9168	32.8767	18.6851	-70.1166	-33.5380
0.265	3.1321	23.6236	30.4909	33.5079	19.0389	-71.4771	-34.1842
0.270	3.2517	24.0712	31.0649	34.1391	19.3927	-72.8375	-34.8304
0.275	3.3734	24.5187	31.6390	34.7704	19.7464	-74.1979	-35.4767
0.280	3.4975	24.9663	32.2131	35.4016	20.1002	-75.5583	-36.1229
0.285	3.6237	25.4139	32.7872	36.0328	20.4540	-76.9187	-36.7691
0.290	3.7522	25.8615	33.3613	36.6641	20.8077	-78.2792	-37.4154
0.295	3.8829	26.3090	33.9353	37.2953	21.1615	-79.6396	-38.0616
0.300	4.0159	26.7566	34.5094	37.9265	21.5153	-81.0000	-38.7078
0.305	4.1511	27.2042	35.0835	38.5577	21.8690	-82.3604	-39.3540

target	var.vec	AMD	LMT	MSFT	NEM	PFE	UTX
0.310	4.2885	27.6517	35.6576	39.1890	22.2228	-83.7209	-40.0003
0.315	4.4282	28.0993	36.2317	39.8202	22.5766	-85.0813	-40.6465
0.320	4.5701	28.5469	36.8058	40.4514	22.9303	-86.4417	-41.2927
0.325	4.7143	28.9945	37.3798	41.0827	23.2841	-87.8021	-41.9390
0.330	4.8607	29.4420	37.9539	41.7139	23.6379	-89.1625	-42.5852
0.335	5.0093	29.8896	38.5280	42.3451	23.9916	-90.5230	-43.2314
0.340	5.1602	30.3372	39.1021	42.9764	24.3454	-91.8834	-43.8777
0.345	5.3133	30.7848	39.6762	43.6076	24.6992	-93.2438	-44.5239
0.350	5.4686	31.2323	40.2502	44.2388	25.0529	-94.6042	-45.1701
0.355	5.6262	31.6799	40.8243	44.8701	25.4067	-95.9646	-45.8163
0.360	5.7860	32.1275	41.3984	45.5013	25.7605	-97.3251	-46.4626
0.365	5.9481	32.5751	41.9725	46.1325	26.1142	-98.6855	-47.1088
0.370	6.1124	33.0226	42.5466	46.7637	26.4680	-100.0459	-47.7550
0.375	6.2789	33.4702	43.1206	47.3950	26.8218	-101.4063	-48.4013
0.380	6.4477	33.9178	43.6947	48.0262	27.1755	-102.7668	-49.0475
0.385	6.6187	34.3654	44.2688	48.6574	27.5293	-104.1272	-49.6937
0.390	6.7920	34.8129	44.8429	49.2887	27.8831	-105.4876	-50.3400
0.395	6.9675	35.2605	45.4170	49.9199	28.2368	-106.8480	-50.9862
0.400	7.1452	35.7081	45.9911	50.5511	28.5906	-108.2084	-51.6324
0.405	7.3252	36.1556	46.5651	51.1824	28.9444	-109.5689	-52.2787
0.410	7.5074	36.6032	47.1392	51.8136	29.2981	-110.9293	-52.9249
0.415	7.6919	37.0508	47.7133	52.4448	29.6519	-112.2897	-53.5711
0.420	7.8786	37.4984	48.2874	53.0760	30.0057	-113.6501	-54.2173
0.425	8.0675	37.9459	48.8615	53.7073	30.3594	-115.0106	-54.8636
0.430	8.2586	38.3935	49.4355	54.3385	30.7132	-116.3710	-55.5098
0.435	8.4520	38.8411	50.0096	54.9697	31.0670	-117.7314	-56.1560
0.440	8.6477	39.2887	50.5837	55.6010	31.4207	-119.0918	-56.8023
0.445	8.8456	39.7362	51.1578	56.2322	31.7745	-120.4522	-57.4485
0.450	9.0457	40.1838	51.7319	56.8634	32.1283	-121.8127	-58.0947
0.455	9.2480	40.6314	52.3060	57.4947	32.4820	-123.1731	-58.7410
0.460	9.4526	41.0790	52.8800	58.1259	32.8358	-124.5335	-59.3872
0.465	9.6595	41.5265	53.4541	58.7571	33.1896	-125.8939	-60.0334
0.470	9.8685	41.9741	54.0282	59.3883	33.5433	-127.2543	-60.6796
0.475	10.0799	42.4217	54.6023	60.0196	33.8971	-128.6148	-61.3259
0.480	10.2934	42.8693	55.1764	60.6508	34.2509	-129.9752	-61.9721
0.485	10.5092	43.3168	55.7504	61.2820	34.6046	-131.3356	-62.6183
0.490	10.7272	43.7644	56.3245	61.9133	34.9584	-132.6960	-63.2646
0.495	10.9475	44.2120	56.8986	62.5445	35.3122	-134.0565	-63.9108
0.500	11.1700	44.6595	57.4727	63.1757	35.6659	-135.4169	-64.5570