Практическая реализация риск метрик: VaR & ES.

Петраков Сергей э-301.

Я собираюсь начать свой проект оценки различных подходов в построении метрик риска, но сначала я собираюсь определить портфель ценных бумаг в рамках, котрого я буду тестировать VaR и ES.

Всю свою работу я предоставлю с кодом R, в котором я реализовал идеи.

Импортирую библиотеки для дальнейшей работы

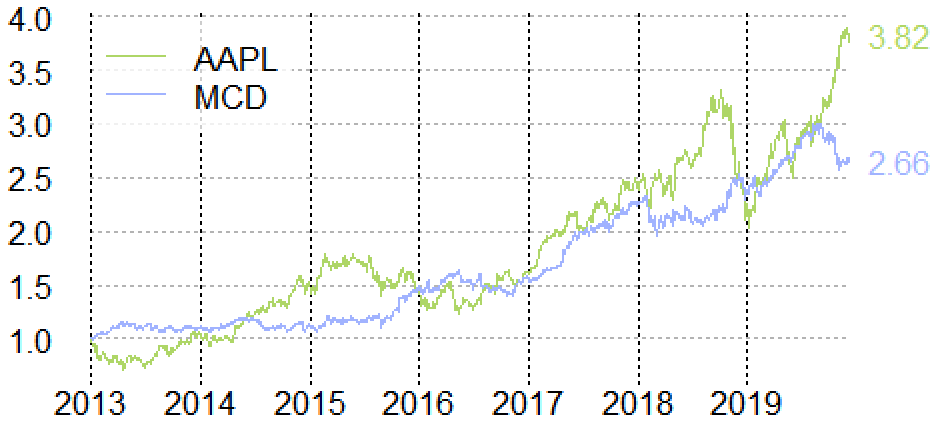
```
library(QuantTools) # Для загрузки рыночных данных
library(data.table) # Для работы с данными
library(ghyp)       # Обобщённое гиперболическое распределение
library(copula)     # Копула
library(fGarch)     # GARCH
library(evd)        # Распределение экстримальных значений
```

Загрузка данных

Создадим портфель, на котором будем тестировать меры риска VaR и ES. Этот подход более сложен, чем просто тестирование VaR и ES на одной конкретной акции. Я собираюсь работать с акциями Apple и акциями McDonalds, это компании из разных секторов экономики, поэтому их движение относительно независимо, что само по себе уже диверсифицирует портфель, по сравнению с ситуацией, когда только одна акция. Возьмём данные за 7 последних лет.

```
# загрузка данных
stock_1 = get_yahoo_data( "AAPL", from = "2013-01-02", to = "2020-01-01" )
stock_2 = get_yahoo_data( "MCD" , from = "2013-01-02", to = "2020-01-01" )
# рассчёт доходностей
stock_1 = stock_1[, .( date, return_1 = close / shift( close, fill = close[1]
) - 1 )]
stock_2 = stock_2[, .( date, return_2 = close / shift( close, fill = close[1]
) - 1 )]
# формируем портфельиз доходностей
portfolio = merge( stock_1, stock_2, by = "date" )
data = as.vector( portfolio[, .( return_1, return_2 ) ] )
# расчёт накопленного pnl (profit and losses)
```

```
plot_dts( portfolio[ , .(date, AAPL = cumprod(1+return_1), MCD = cumprod(1+re
turn_2)) ] )
```



Оптимизация портфеля

В этом разделе будет рассматриваться статическая оптимизация портфеля по рыночным метрикам.

Статическая оптимизация

Напишем функцию для расчета статистики pnl, максимальной просадки за период (max_dd) и коэффициента Шарпа (sharpe)

```
calc_perfomance = function( returns ){

  # Cumulative Products
  pnl = cumprod( returns + 1 )

  summary = data.table(
    pnl    = prod( returns + 1 ) - 1,
    max_dd = min( pnl / cummax( pnl ) - 1 ) * 100,
    sharpe = mean( returns ) / sd( returns ) * sqrt(252)
  )

  round( summary, 2 )

}
```

Посчитаем статистику для наших акций

```
calc_perfomance( portfolio$return_1 )
```

```
##     pnl max_dd sharpe
## 1: 2.82 -38.52    0.9
```

```
calc_perfomance( portfolio$return_2 )
```

```
##     pnl max_dd sharpe
## 1: 1.66 -16.34   0.98
```

Определим сетку весов для формирования весов портфеля.

```
# weights
weights = CJ(
  w1 = seq( 0.0, 1, 0.05),
  w2 = seq( 0.0, 1, 0.05),
)


weights = weights[ w1+w2 == 1 ]
```

Посчитаем статистику для каждого набора весов

1. Поиск таких весов, при которых наибольший pnl.

```
result = weights[, calc_perfomance( portfolio$return_1*w1 + portfolio$return_
2*w2), by = .( w1, w2,) ]

result[order(-pnl)][1:5]
```

```
     w1   w2  pnl max_dd sharpe
1: 1.00 0.00 2.82 -38.52   0.90
2: 0.95 0.05 2.79 -36.71   0.92
3: 0.90 0.10 2.76 -34.87   0.95
4: 0.85 0.15 2.73 -32.98   0.98
5: 0.80 0.20 2.69 -31.04   1.00
```

2. Поиск таких весов, при которых наименьшая максимальная просадка портфеля

```
result[order(-max_dd)][1:5]
```

```
     w1   w2  pnl max_dd sharpe
1: 0.20 0.80 1.97 -12.69   1.15
2: 0.25 0.75 2.04 -13.56   1.17
3: 0.15 0.85 1.89 -13.61   1.12
4: 0.10 0.90 1.82 -14.53   1.07
5: 0.30 0.70 2.12 -14.70   1.19
```

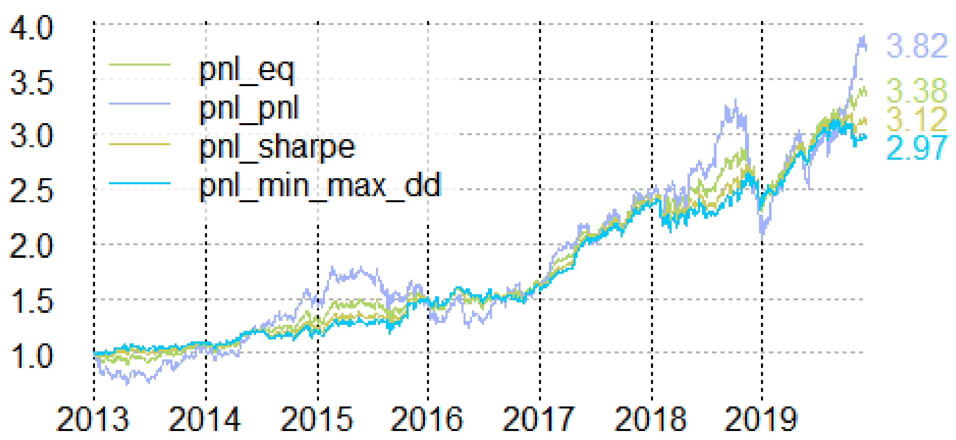
3. Поиск таких весов, при которых максимален коэффициент Шарпа

```
result[order(-sharpe)][1:5]
```

```
     w1   w2  pnl max_dd sharpe
1: 0.30 0.70 2.12 -14.70   1.19
2: 0.35 0.65 2.19 -15.84   1.19
3: 0.40 0.60 2.26 -16.97   1.19
4: 0.45 0.55 2.32 -18.17   1.18
5: 0.25 0.75 2.04 -13.56   1.17
```

Построим портфели с равными весами, с макс доходностью (pnl), максимальным коэффициентом Шарпа и минимальной максимальной просадкой.

```
# sum returns
portf_eq = portfolio[, .(date, pnl_eq = return_1*0.5 + return_2*0.5)]
portf_max_pnl = portfolio[, .(date, pnl_pnl    = return_1*1 + return_2*0)]
portf_max_sharpe = portfolio[, .(date, pnl_sharpe = return_1*0.3 + return_2*0
.7 )]
portf_min_max_dd = portfolio[, .(date, pnl_min_max_dd = return_1*0.2 + return
_2*0.8 ) ]
# calc pnl
portf_eq[, pnl_eq:= cumprod(1 + pnl_eq)]
portf_max_pnl[, pnl_pnl:= cumprod(1 + pnl_pnl)]
portf_max_sharpe[, pnl_sharpe:= cumprod(1 + pnl_sharpe)]
portf_min_max_dd[, pnl_min_max_dd:= cumprod(1 + pnl_min_max_dd)]
plot_dts(
  portf_eq,
  portf_max_pnl,
  portf_max_sharpe,
  portf_min_max_dd
)
```



Оценка риска портфеля.

В этом разделе будут рассмотрены варианты оценки таких метрик риска, как VaR и ES, я рассмотрю 4 подхода, на которых будет определяться модерирование:

1. Обобщённое гиперболическое распределение
2. Копулы
3. GARCH
4. Метод экстримальных значений.

Ghyp (Generalized Hyperbolic distribution)

Выбираем лучшую модель из Обобщённого гиперболического распределения по критерию Акаике.

```
aic.mv = stepAIC.ghyp( data, dist = c( "gauss", "t", "ghyp" ), symmetric=NULL
, silent=T )
## Currently fitting: asymmetric t
## Currently fitting: asymmetric ghyp
## Currently fitting: symmetric t
## Currently fitting: symmetric ghyp
## Currently fitting: gauss
aic.mv$best.model@model[1]
```

[1] "Symmetric Student-t"

Получили, что симметричное распределение Стьюдента лучшего всего описывает доходность акций портфеля.

Подгоняем модель и считаем VaR и ES. Будем использовать портфель с равными весами, поскольку он показывает вторую среди 4 рассмотренных доходность (первый по доходности портфель состоит только из акций Apple, которые сильно выросли в период релиза новой модели Iphone, не обязательно такое повторится), при этом такой портфель не такой волатильный и из содержательных соображений диверсификации распределение средств в равной доли такде имеет смысл.

Расчёт риск метрик сделаем следующим образом: смоделируем 10^6 теоретических значений стоимости портфеля на основании подобранной модели ОГР "Symmetric Student-t". После чего отсортируем их по возрастанию, и соответственно значение с порядковым номером 0.05*10^6 и будет VaR, а среднее среди тех, что левее его будет ES, по определению. Всё это будем совершать при уровне значимости в 5%.

```
prt.fit = fit.tmv( data, symmetric=T, silent=T )
w     = c(0.5,0.5)
N     = 10^6
alpha = 0.05
sim = rghyp( n = N, object = prt.fit )
prt.sim = w[1] * sim[,1] + w[2] * sim[,2]
prt.sim = sort(prt.sim)
VaR = prt.sim[ alpha*N ]
ES  = mean( prt.sim[ 1:( alpha * N-1 ) ] )

data.table( VaR, ES)
```

```
##            VaR          ES
## 1: -0.01475454 -0.02308559
```

Получили, что при 95% вероятности доходность будет выше -1,48% (VaR), а среднее значение доходности в 5% наихудших случаев составляет -2,31% (ES).

Пример оптимизации портфеля (подбор весов)

```
opt = portfolio.optimize(prt.fit, risk.measure= "value.at.risk",
                         type="minimum.risk", silent=T)
opt$opt.weights
w = opt$opt.weights
prt.sim = w[1]*sim[,1]+w[2]*sim[,2]
prt.sim = sort(prt.sim)

VaR = prt.sim[ alpha*N ]
ES  = mean( prt.sim[ 1:( alpha * N-1 ) ] )

data.table( VaR, ES )
```

```
##            VaR          ES
## 1: -0.01313751 -0.02056765
```

Как можно заметить, значения VaR и ES улучшились.

Копулы

Копулы: определение и свойства

Копула $C(\vec{u}),\ \vec{u} = (u_1, \dots, u_d)$ — функция $C: [0;1]^d \to [0;1]$ со следующими свойствами:

- $\exists u_i = 0,\ i \in \{1; \dots; d\} \Rightarrow C(\vec{u}) = 0;$
- $C(1,1, \dots, u_i, \dots, 1,1) = u_i;$
- $\forall u_{i,1} \le u_{i,2}\ \ \forall w_i \in \{u_{i,1}; u_{i,2}\}$
  $\sum_{\forall \vec{w}} \left(C(\vec{w}) \prod_{i=1}^{d} sgn\left(2w_i - u_{i,1} - u_{i,2}\right)\right) \ge 0$

Копула — совместная функция распределения *d* стандартных равномерных случайных величин:

$C(\vec{u}) = P(r_1 < u_1; \dots; r_d < u_d),\ r_i \sim U[0;1]$

Копула и совместная функция распределения

Пусть $\xi \sim F_\xi(u)$, тогда $r_1 = F_\xi(\xi) \sim U[0;1]$ и $F_\xi^{-1}(r_1) = \xi$

$$C\left(F_{\xi_1}(u_1), \dots, F_{\xi_d}(u_d)\right) = P\left(r_1 < F_{\xi_1}(u_1); \dots; r_d < F_{\xi_d}(u_d)\right) =$$
$$= P\left(F_{\xi_1}^{-1}(r_1) < u_1; \dots; F_{\xi_d}^{-1}(r_d) < u_d\right) = P(\xi_1 < u_1; \dots; \xi_d < u_d) =$$
$$= F_{\xi_1, \dots, \xi_d}(u_1, \dots, u_d)$$

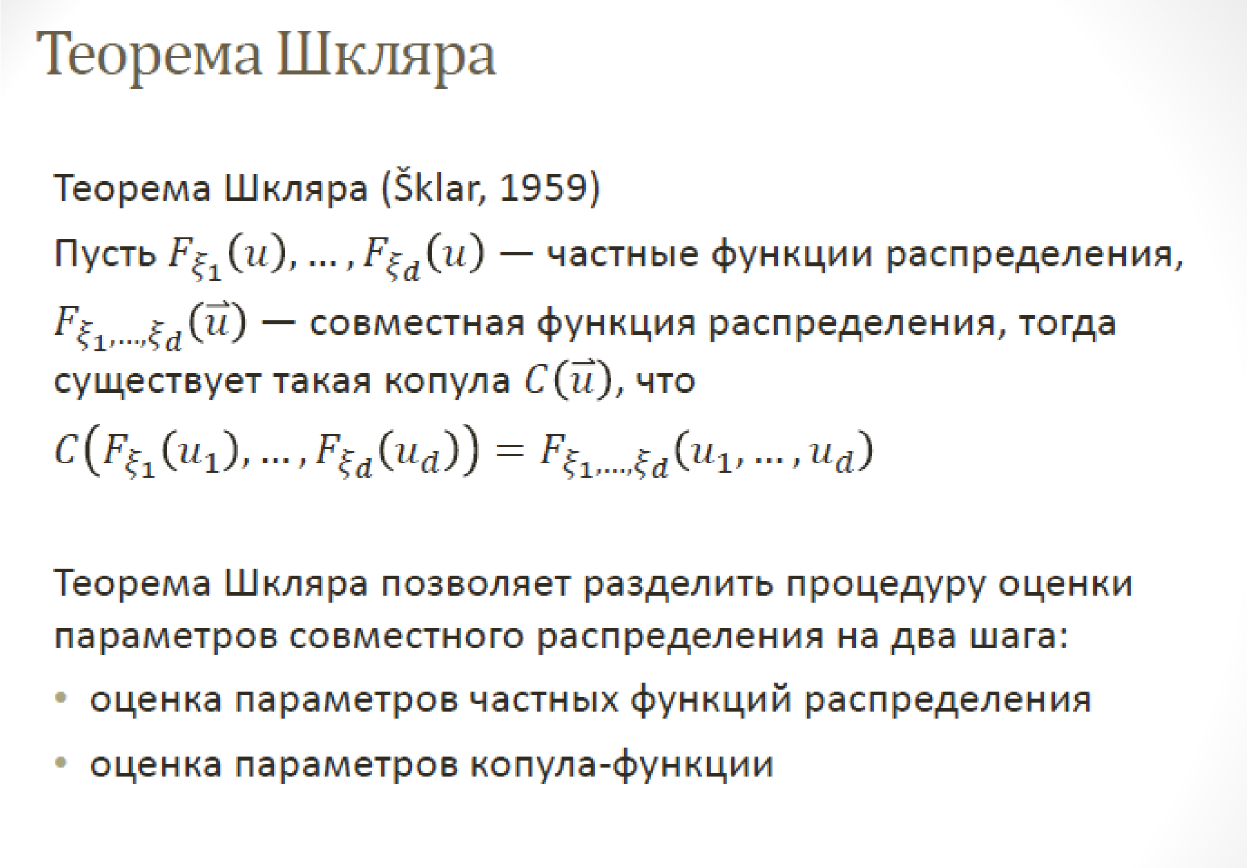
Таким образом, при подстановке в копулу значений частных функций распределения случайных величин мы получим их совместную функцию распределения

Плотностью $c(\vec{u})$ копулы $C(\vec{u})$ называется отношение

$$c(\vec{u}) = \frac{\partial^d C(u_1, \dots, u_d)}{\partial u_1 \dots \partial u_d}$$

Если случайные величины $\xi_1, \dots, \xi_d$ непрерывны, то

$$c\left(F_{\xi_1}(u_1), \dots, F_{\xi_d}(u_d)\right) = \frac{f_{\xi_1, \dots, \xi_d}(u_1, \dots, u_d)}{f_{\xi_1}(u_1) \dots f_{\xi_d}(u_d)}$$

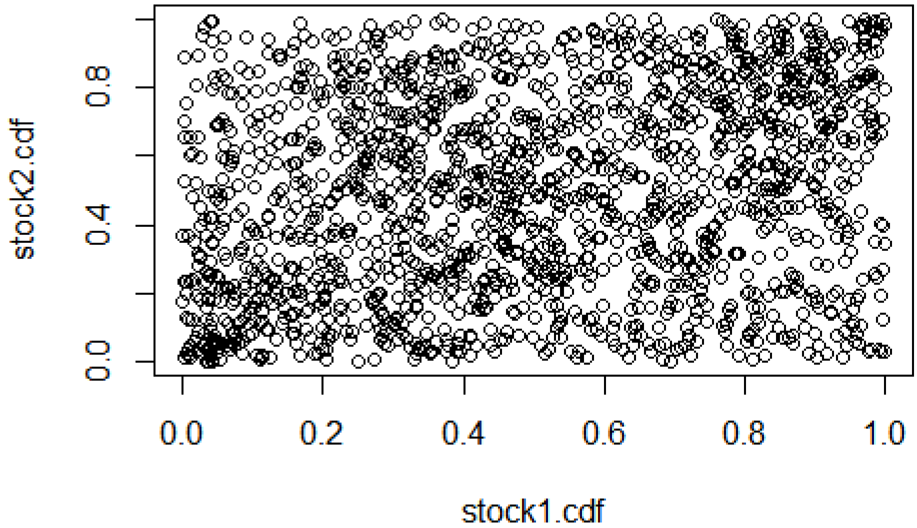


Выбираем лучшую модель для каждой бумаги

```
stock1.fit = stepAIC.ghyp(data$return_1, dist = c("gauss", "t", "ghyp"),
                             symmetric=NULL, silent=T)$best.model
## Currently fitting: asymmetric t
## Currently fitting: asymmetric ghyp
## Currently fitting: symmetric t
## Currently fitting: symmetric ghyp
## Currently fitting: gauss
stock2.fit = stepAIC.ghyp(data$return_2, dist = c("gauss", "t", "ghyp"),
                             symmetric=NULL, silent=T)$best.model
## Currently fitting: asymmetric t
## Currently fitting: asymmetric ghyp
## Currently fitting: symmetric t
## Currently fitting: symmetric ghyp
## Currently fitting: gauss
stock1.fit@model[1]
## [1] "Symmetric Generalized Hyperbolic"
stock2.fit@model[1]
## [1] "Symmetric Student-t"
```

Получаим эмперические чатсные распределения доходностей и построим их на плоскости.

```
stock1.cdf = pghyp( data$return_1, object = stock1.fit )
stock2.cdf = pghyp( data$return_2, object = stock2.fit )
cdf = cbind( stock1.cdf, stock2.cdf, stock3.cdf )
plot( cdf )
```

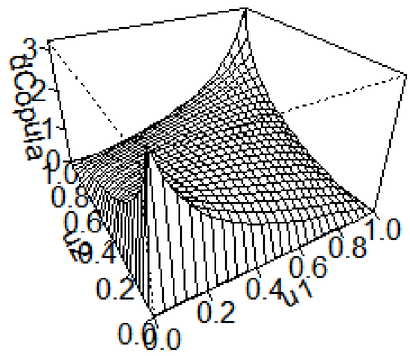


Определяем копулы (Гаусса, Стьюдента, Гумбеля) и находим лучшую модель на данных для каждого из вариантов. Попутно построим копулу Гаусса, копулу Стьюдента и копулу Гумбеля и с помощью критерия log likelihood определим лучшую из них.

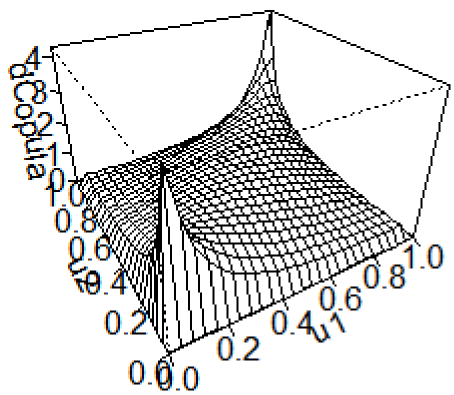
```
# define copulas
norm.cop = normalCopula( dim=2, param=0.5, dispstr="un" )
stud.cop = tCopula( dim=2, param=0.5 )
gumb.cop = gumbelCopula( dim=2, param=2 )
# Copula plot
persp( norm.cop, dCopula )
persp( stud.cop, dCopula )
persp( gumb.cop, dCopula )
# Copula fiting
norm.fit = fitCopula(cdf,copula=norm.cop)
stud.fit = fitCopula(cdf,copula=stud.cop)
gumb.fit = fitCopula(cdf,copula=gumb.cop)
# best model
data.table(
  norm = norm.fit@loglik,
  stud = stud.fit@loglik,
```

```
  gumb = gumb.fit@loglik
)
```

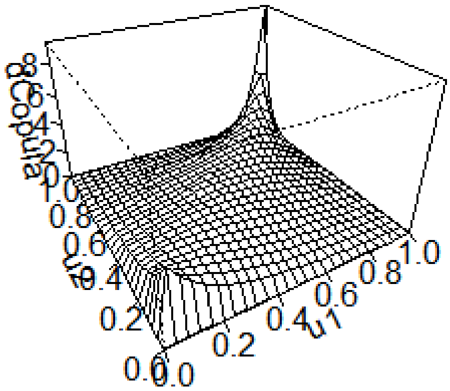
```
##       norm     stud     gumb
## 1: 128.5104 131.805 95.47168
```



Копула Гаусса.



Копула Стьюдента.



Копула Гумбеля.

Получаем, что копула Стьюдента лучшая среди всех по значению loglik.

Считаем Var и ES портфеля на основе копулы и совместных распределений. После того, как совместное распределение по теореме Шкляра будет получено, можно применить подход из предыдущего пункта: на базе эмпирически построенной оценки совместного распределения доходностей, создадим 10^3 значений из этого распределения, отсортируем их по возрастанию. Затем отсечём 5%, доходность, по которой проходит отсечение – это VaR (уровень значимости 5%). Соответственно среднее среди тех значений доходности, которые левее точки отсечения – это ES.

```
N = 10^3
stud.sim = rCopula( n=N, copula = stud.fit@copula )


stock1.sim = qghyp( stud.sim[,1], object=stock1.fit )
stock2.sim = qghyp( stud.sim[,2], object=stock2.fit )


w = c(0.5,0.5)
prt.sim = w[1]*stock1.sim + w[2]*stock2.sim


alpha = 0.05
prt.sim = sort(prt.sim)
VaR = prt.sim[alpha*N]
ES = mean(prt.sim[1:(alpha*N-1)])


data.table( VaR, ES )
```

```
##             VaR          ES
## 1: -0.01811316 -0.02379313
```

Результат: VaR = -1,82% при уровне значимости 5%, ES = -2,38%

GARCH

Общая идея:

1. Подгоняем GARCH модели (частные GARCH модели).
2. Расчёт условных стандартизованных остатков $z_{i,t}$
3. Моделирование многомерной величины z_t

```
#1. Подгонка GARCH модели
stock1.gfit = garchFit( data = data$return_1, formula =~garch(1,1),
                        cond.dist="ged", trace=F )

stock2.gfit = garchFit( data = data$return_2, formula =~garch(1,1),
                        cond.dist="ged", trace=F )

#2. Стандартизация остатков
z = matrix(nrow=nrow(data), ncol=2)
z[,1] = stock1.gfit@residuals / stock1.gfit@sigma.t
z[,2] = stock2.gfit@residuals / stock2.gfit@sigma.t

#3. Функция распределения остатков
mean = c(0,0); sd = c(1,1); nu = c(1.3,1.3)
xi = c(1, 1 )
cdf = matrix( nrow=nrow(data), ncol=2 )

cdf[,1] = psged( z[,1], mean=mean[1], sd=sd[1], nu=nu[1], xi=xi[1] )
cdf[,2] = psged (z[,2], mean=mean[2], sd=sd[2], nu=nu[2], xi=xi[2] )
```

Подгонка копул

```
# Подгонка копул
norm.fit = fitCopula( cdf, copula=norm.cop )
stud.fit = fitCopula( cdf, copula=stud.cop )
gumb.fit = fitCopula( cdf, copula=gumb.cop )
# определение наилучшей модели по критерию log likelihood
data.table(
  norm = norm.fit@loglik,
```

```
  stud = stud.fit@loglik,
  gumb = gumb.fit@loglik)
```

```
##        norm     stud     gumb
## 1: 46.13617 47.88665 32.29711
```

Получаем, что наилучшей моделью тут является копула Стьюдента.

Оцениваем риск

```
#По методу Monte-Carlo оцениваем риск
cdf.sim = rCopula(n=N, copula=norm.fit@copula)
z.sim = matrix(nrow=N, ncol=3)
for (i in 1:3)
  z.sim[,i] = qsged(cdf.sim[,i], mean=mean[i], sd=sd[i], nu=nu[i], xi=xi[i])

frc1 = predict( stock1.gfit, n.ahead=1 )
frc2 = predict( stock2.gfit, n.ahead=1 )
frc3 = predict( stock3.gfit, n.ahead=1 )
mu = c(frc1[,1],frc2[,1],frc3[,1])
sigma = c(frc1[,3],frc2[,3],frc3[,3])



prt.sim = w[1]*(mu[1]+sigma[1]*z.sim[,1]) +
  w[2]*(mu[2]+sigma[2]*z.sim[,2])+w[3]*(mu[3]+sigma[3]*z.sim[,3])

prt.sim = sort(prt.sim)
VaR = prt.sim[alpha*N]
ES = mean(prt.sim[1:(alpha*N-1)])

data.table( VaR, ES )
```

```
##            VaR          ES
## 1: -0.01341181 -0.01724423
```

Метод экстремальных значений

Рассмотрим случай для выборки значений, превышающих многомерный порог

Метод оценивает абсолютные значения, поэтому меняем знак у векторов доходности. Выбираем значения, превышающих многомерный порог

```
ESM = -as.matrix(data)


alpha = 1 - 0.10
```

```
T = nrow( data )
u = c(sort(ESM[,1])[alpha*T],sort(ESM[,2])[alpha*T])
t.ESM = ESM[ ( ESM[,1] > u[1] ) & ( ESM[,2] > u[2] ), ]
```

Подгоняем модели GPD (Generalized pareto distribution) и получаем значения частных функций

```
fit1 = fpot( t.ESM[,1],threshold=u[1], model="gpd" )
fit2 = fpot( t.ESM[,2],threshold=u[2], model="gpd" )
# CDF
cdf1 = pgpd( t.ESM[,1],loc=u[1],scale=fit1$par[1], shape=fit1$par[2] )
cdf2 = pgpd( t.ESM[,2],loc=u[2],scale=fit2$par[1], shape=fit2$par[2] )
cdf = cbind( cdf1, cdf2 )
```

Подгонка копулы

```
gumb.cop = gumbelCopula( dim=3, param = 3 )
gal.cop = galambosCopula( 3 )

gumb.fit = fitCopula( cdf,?copula = gumb.cop )
#gal.fit  = fitCopula( cdf, copula = gal.cop  )

data.table(
  gumb = gumb.fit@loglik#,
  #gal  = gal.fit@loglik
)
```

```
##          gumb
## 1: 95.47168
```

Моделируем значения и расчитываем меры риска

```
cdf.sim = rCopula(n=N,copula=gumb.fit@copula)
sim1 = qgpd(cdf.sim[,1],loc=u[1],scale=fit1$par[1],
            shape=fit1$par[2])
sim2 = qgpd(cdf.sim[,2],loc=u[2],scale=fit2$par[1],
            shape=fit2$par[2])

loss = sort(w[1]*sim1+w[2]*sim2)
```

```
Fu = nrow(t.ESM)/T

alpha = 1-1/(260*Fu)

VaR = loss[alpha*N]

ES = mean(loss[(alpha*N+1):N])


data.table( -VaR, -ES )
```

```
##            V1          V2

## 1: -0.02498317 -0.03350854
```

Таким образом, VaR = -2,50% а ES = -3,35%

Вывод

Как правило, метод экстремальных значений даёт менее точные оценки, по сравнению с GARCH, Копулой и GHYP, поэтому стоит больше ориентироваться а первые три варианта.