Анализ доходности и волатильности финансовых активов. Многомерный случай

ЦМФ. Математические финансы

Многомерное обобщённое гиперболическое распределение

Двумерный случай

```
smi <- EuStockMarkets[,"SMI"]
smi <- smi[2:(T+1)]/smi[1:T] - 1
# доходности портфеля из двух активов
prt <- array(c(dax,smi),dim=c(T,2))

# оценка параметров модели
prt.fit <- fit.[…]mv(prt,symmetric=FALSE,silent=TRUE)
aic.mv <- stepAIC.ghyp(prt, […])

# оценки риска
prt.fit <- fit.ghypmv(prt,symmetric=FALSE,silent=TRUE)
w <- c(0.5,0.5)   # веса активов в портфеле
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)])
```

VaR	-0.009
ES	-0.017

Оптимизация портфеля

```
# выбор оптимальных весов активов в портфеле
opt <- portfolio.optimize(prt.fit,
risk.measure="value.at.risk",type="minimum.risk",
target.return=NULL,risk.free=NULL,level=0.95,silent=TRUE)
```

- ***risk.measure*** определяет целевой измеритель риска
  "sd", "value.at.risk", "expected.shortfall"
- ***type*** — вид оптимизации
  "minimum.risk" — по минимальному риску
  "tangency" — по соотношению "(доходность – безрисковая ставка) / риск"
  "target.return" — минимальный риск при заданной доходности

`opt$opt.weights` # искомые веса

Копула-функции

Копулы: определение и свойства

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

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

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

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

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

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

Теорема Шкляра

Теорема Шкляра (Šklar, 1959)

Пусть $F_{\xi_1}(u), \dots, F_{\xi_d}(u)$ — частные функции распределения,

$F_{\xi_1,\dots,\xi_d}(\vec{u})$ — совместная функция распределения, тогда существует такая копула $C(\vec{u})$, что

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

Теорема Шкляра позволяет разделить процедуру оценки параметров совместного распределения на два шага:

- оценка параметров частных функций распределения
- оценка параметров копула-функции

Виды копул

Виды копула-функций:

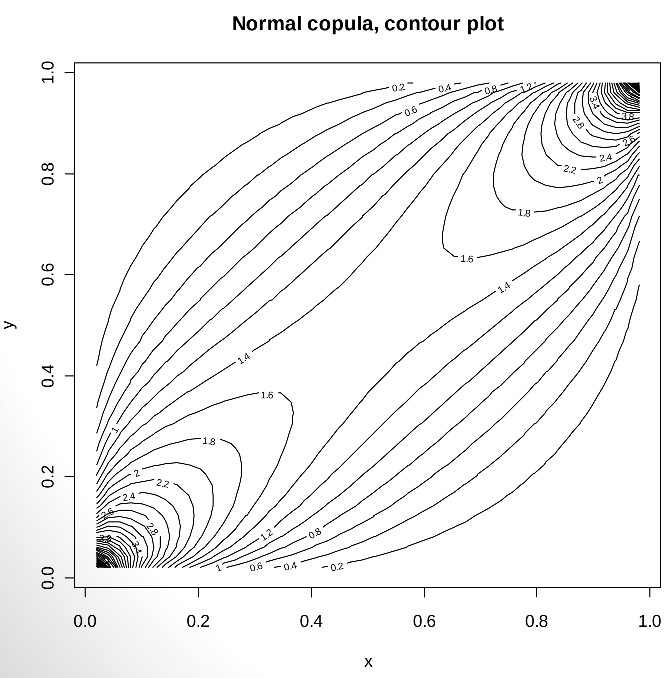
- эллиптические — строятся на основе известных функций распределения (нормальная, Стьюдента, Коши, Лапласа и другие);
- архимедовы — строятся на основе функции-генератора (Гумбеля, Клейтона, Франка и другие);
- экстремальные копулы (Гумбеля, Галамбоса и другие);
- непараметрические копулы

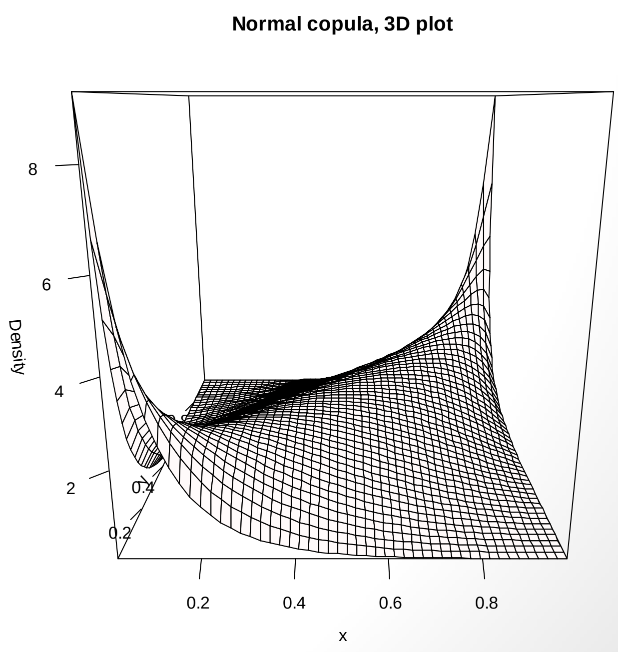
Эллиптические копулы (1:2)

Копула Гаусса (нормальная копула)

$$C_N = \Phi_{\rho_{\xi\eta}}(\Phi^{-1}(x);\Phi^{-1}(y))$$

$$c_N = \frac{1}{\sqrt{1-\rho^2}} \cdot e^{\left(\frac{\Phi^{-2}(u_1)+\Phi^{-2}(u_2)}{2}+\frac{2\rho\Phi^{-1}(u_1)\Phi^{-1}(u_2)-\Phi^{-2}(u_1)-\Phi^{-2}(u_2)}{2(1-\rho^2)}\right)}$$





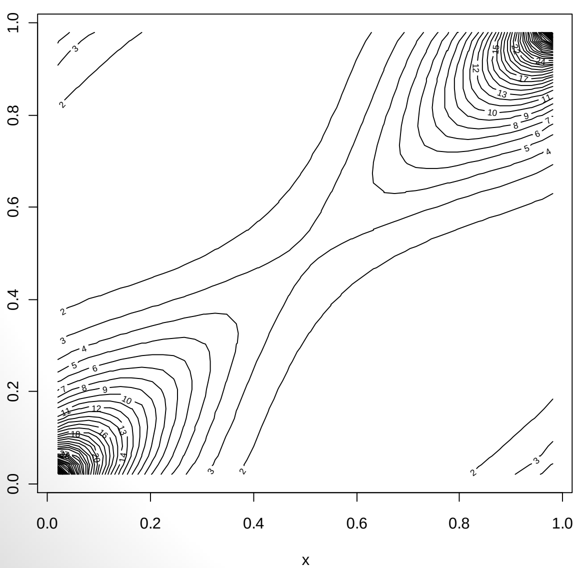
Эллиптические копулы (2:2)

Копула Стьюдента (t-копула)

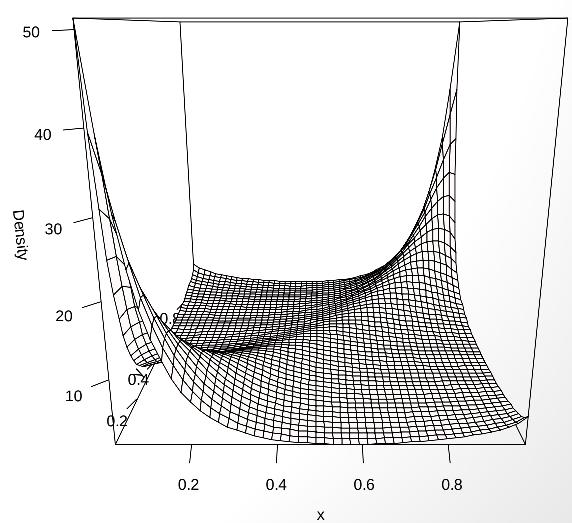
$$C_T = t_{\nu,\rho}\left(t_{1,\nu}^{-1}(u_1), t_{2,\nu}^{-1}(u_2)\right)$$

$$c_T = \frac{\Gamma\left(\frac{\nu+2}{2}\right)\Gamma\left(\frac{\nu}{2}\right)}{\sqrt{\rho}\,\Gamma^2\left(\frac{\nu+1}{2}\right)} \cdot \frac{\left(1+\frac{t_{1,\nu}^{-2}(u_1)+t_{2,\nu}^{-2}(u_2)-2\rho t_{1,\nu}^{-1}(u_1)t_{2,\nu}^{-1}(u_2)}{\nu(1-\rho^2)}\right)^{-\frac{\nu+2}{2}}}{\left(\left(1+\frac{t_{1,\nu}^{-2}(u_1)}{\nu}\right)\left(1+\frac{t_{2,\nu}^{-2}(u_2)}{\nu}\right)\right)^{-\frac{\nu+2}{2}}}$$

Student's copula, contour plot



Student's copula, 3D plot

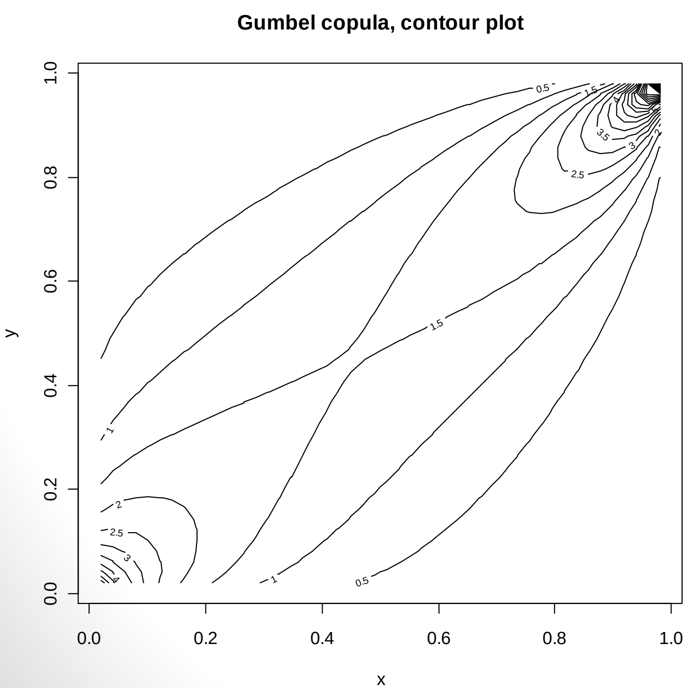


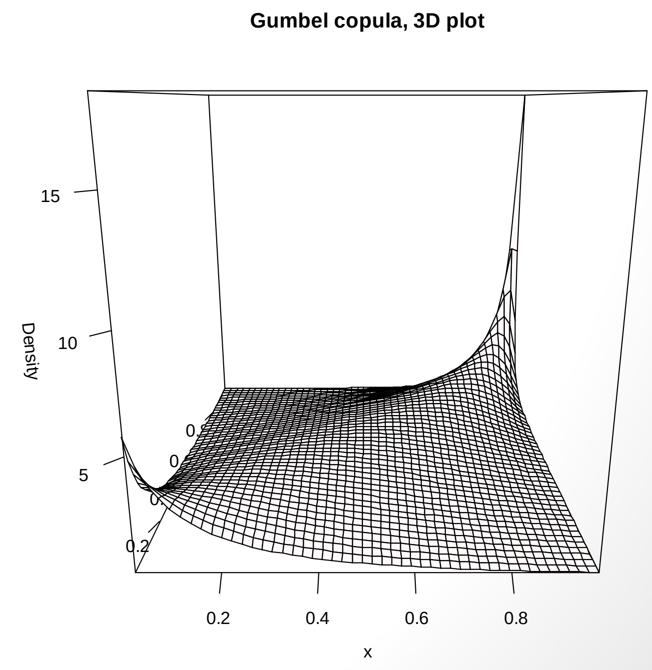
Архимедовы копулы (1:2)

Копула Гумбеля

$$C_G = \exp\left(-\left((-\ln u_1)^{\alpha} + (-\ln u_2)^{\alpha}\right)^{\frac{1}{\alpha}}\right),\ \varphi = (-\ln t)^{\alpha}$$

$$c_G = \frac{-\varphi''(C_G(u_1,u_2))\varphi'(u_1)\varphi'(u_2)}{(\varphi'(C_G(u_1,u_2)))^3},\ \alpha \in [1;+\infty)$$



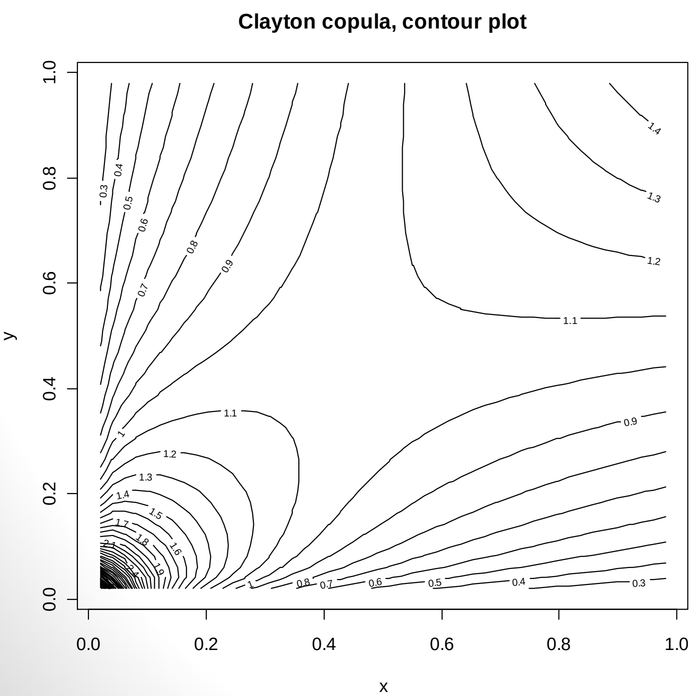


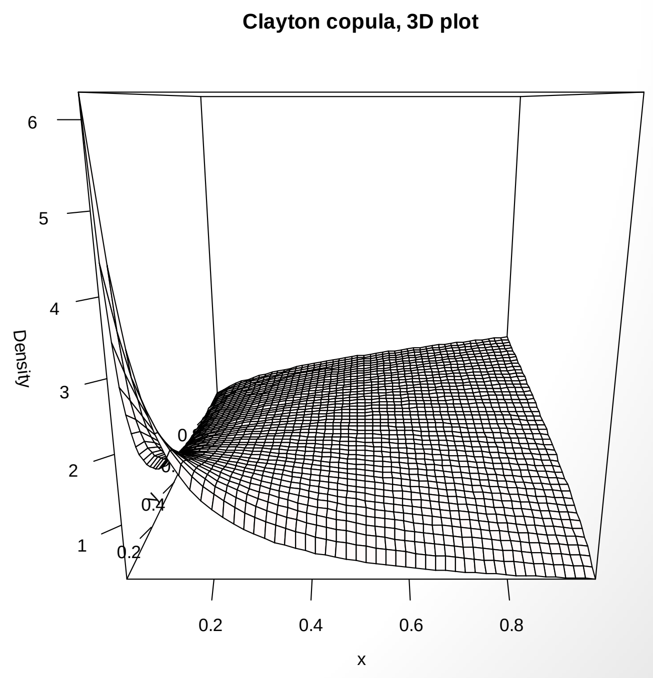
Архимедовы копулы (2:2)

Копула Клейтона

$$C_C = \max\left((u_1^{-\alpha} + u_2^{-\alpha} - 1)^{-\frac{1}{\alpha}}, 0\right),\ \varphi = \frac{1}{\alpha}(t^{-\alpha} - 1)$$

$$c_C = \frac{-\varphi''(C_C(u_1,u_2))\varphi'(u_1)\varphi'(u_2)}{(\varphi'(C_C(u_1,u_2)))^3},\ \alpha \in [-1;0) \cup (0;+\infty)$$





Исходные данные

```
library(datasets)

T <- nrow(EuStockMarkets) - 1

dax <- EuStockMarkets[,"DAX"]
dax <- dax[2:(T+1)]/dax[1:T] - 1

smi <- EuStockMarkets[,"SMI"]
smi <- smi[2:(T+1)]/smi[1:T] - 1
```

Моделирование частных функций распределения

```
library(ghyp)
```

\# моделирование частных функций распределения

```
dax.fit <- stepAIC.ghyp(dax,dist=c("gauss","t","ghyp"),
  symmetric=NULL,silent=TRUE)$best.model
smi.fit <- stepAIC.ghyp(smi,dist=c("gauss","t","ghyp"),
  symmetric=NULL,silent=TRUE)$best.model
```

\# расчёт значений $F_1(u)$ и $F_2(u)$

```
dax.cdf <- pghyp(dax,object=dax.fit)
smi.cdf <- pghyp(smi,object=smi.fit)
cdf <- array(c(dax.cdf,smi.cdf),dim=c(T,2))
```

Моделирование копулы

```
library(copula)
```

объявление копул

```
norm.cop <- normalCopula(dim=2,param=0.5,dispstr="un")
stud.cop <- tCopula(dim=2,param=0.5,df=5,
  df.fixed=TRUE,dispstr="un")
gumb.cop <- gumbelCopula(dim=2,param=2)
clay.cop <- claytonCopula(dim=2,param=2)
```

подгонка копулы

```
norm.fit <- fitCopula(cdf,copula=norm.cop)
stud.fit <- fitCopula(cdf,copula=stud.cop)
gumb.fit <- fitCopula(cdf,copula=gumb.cop)
clay.fit <- fitCopula(cdf,copula=clay.cop)
```

norm.fit@loglik	558.4
stud.fit@loglik	595.0
gumb.fit@loglik	533.3
clay.fit@loglik	486.3

Оценка финансового риска

\# значения частных функций распределения

```
N <- 10^4
stud.sim <- rcopula(n=N,copula=stud.fit@copula)
```

\# доходности активов

```
dax.sim <- qghyp(stud.sim[,1],object=dax.fit)
smi.sim <- qghyp(stud.sim[,2],object=smi.fit)

w <- c(0.5,0.5)
prt.sim <- w[1]*dax.sim + w[2]*smi.sim
```

\# измерители риска

```
alpha <- 0.1
prt.sim <- sort(prt.sim)
VaR <- prt.sim[alpha*N]
ES <- mean(prt.sim[1:(alpha*N-1)])
```

VaR	-0.009
ES	-0.016

Модель Copula-GARCH

Формализация модели

Уравнения для дисперсии по частным GARCH-моделям:

$$\varepsilon_{i,t} = z_{i,t}\sigma_{i,t}, \qquad z_{i,t} \sim idd(0;1)$$

$$\sigma_{i,t}^2 = \omega_i + \sum_{k=1}^{p} \alpha_{i,k}\varepsilon_{i,t-k}^2 + \sum_{k=1}^{q} \beta_{i,k}\sigma_{i,t-k}^2$$

$$i \in \{1; \dots; d\}$$

Этапы моделирования:

1. Оценка частных GARCH-моделей;
2. Расчёт условных стандартизированных остатков $z_{i,t}$
3. Моделирование многомерной величины z_t

Модель «copula–GARCH» в R

одномерные GARCH-модели

```
library(fGarch)
dax.gfit <- garchFit(data=dax,formula=~garch(1,1),
  shape=1.25,include.shape=F,cond.dist="ged",trace=F)
smi.gfit <- garchFit(data=smi,formula=~garch(1,1),
  shape=1.3,include.shape=F,cond.dist="sged",trace=F)
```

стандартизированные остатки

```
z <- matrix(nrow=T,ncol=2)
z[,1] <- dax.gfit@residuals / dax.gfit@sigma.t
z[,2] <- smi.gfit@residuals / smi.gfit@sigma.t
```

частные распределения остатков

```
mean <- c(0,0); sd <- c(1,1); nu <- c(1.25,1.3)
xi <- c(1,smi.gfit@fit$par["skew"])

cdf <- matrix(nrow=T,ncol=2)
for (i in 1:2) cdf[,i] <- psged(z[,i],mean=mean[i],
  sd=sd[i],nu=nu[i],xi=xi[i])
```

Модель «copula–GARCH» в R

подгонка копул

```
norm.fit <- fitCopula(cdf,copula=norm.cop)
stud.fit <- fitCopula(cdf,copula=stud.cop)
gumb.fit <- fitCopula(cdf,copula=gumb.cop)
clay.fit <- fitCopula(cdf,copula=clay.cop)
```

метод Монте-Карло

```
cdf.sim <- rcopula(n=N,copula=stud.fit@copula)

z.sim <- matrix(nrow=N,ncol=2)
for (i in 1:2) z.sim[,i] <- qsged(cdf.sim[,i],
  mean=mean[i],sd=sd[i],nu=nu[i],xi=xi[i])

frc1 <- predict(dax.gfit,n.ahead=1)
frc2 <- predict(smi.gfit,n.ahead=1)

mu <- c(frc1[,1],frc2[,1])
sigma <- c(frc1[,3],frc2[,3])
```

Оценка финансового риска

```
# модельные доходности портфеля
prt.sim <- w[1]*(mu[1]+sigma[1]*z.sim[,1]) +
           w[2]*(mu[2]+sigma[2]*z.sim[,2])

# измерители риска
prt.sim <- sort(prt.sim)
VaR <- prt.sim[alpha*N]
ES <- mean(prt.sim[1:(alpha*N-1)])
```

VaR	-0.017
ES	-0.026

Теория экстремальных значений (многомерный случай)

Многомерная максима

Пусть $\vec{x}_1, \dots, \vec{x}_n \sim F,\ \ \vec{x}_i \in R^d,\ \ \vec{x}_i \sim F_i$

$\vec{x}_i = \left(x_{i,1}, \dots, x_{i,d}\right)^T$ — убытки различных видов

$M_{n,j} = \max\left(x_{1,j}, \dots, x_{n,j}\right)$

$M_n = \left(M_{n,1}, \dots, M_{n,d}\right)^T$ — покомпонентная блочная максима

Нас интересует сходимость нормализованной максимы:

$\frac{M_n - d_n}{c_n} = \left(\frac{M_{n,1} - d_{n,1}}{c_{n,1}}, \dots, \frac{M_{n,d} - d_{n,d}}{c_{n,d}}\right)^T \xrightarrow{n} H,\ \ c_n > 0$

Пусть $\frac{M_n - d_n}{c_n}$ сходится к некоторой векторной случайной величине с совместной функцией распределения H, тогда

$\lim_n P\left(\frac{M_n - d_n}{c_n} \le \vec{x}\right) = \lim_n F^n(c_n \vec{x} + d_n) = H(\vec{x})$, т.е. $F \in MDA(H)$

Экстремальная копула

Если у H есть невырожденные частные функции распределения, то они должны быть Фреше, Гумбеля или Вейбулла. По теореме Шкляра существует копула $C(F_1(x_1), \dots, F_d(x_d)) = H(\vec{x})$

<u>Теорема</u> о копуле эктремальных значений

Пусть $F \in MDA(H)$ и $H_i \sim GEV$, тогда $C(\vec{u}^t) = C^t(\vec{u}),\ \forall t > 0$

<u>Теорема</u> о представлении Пикандса

Копула C — экстремальная тогда и только тогда, когда её можно представить в виде

$C(\vec{u}) = e^{B\left(\frac{\ln u_1}{\sum_{k=1}^{d} \ln u_k}, \dots, \frac{\ln u_d}{\sum_{k=1}^{d} \ln u_k}\right) \sum_{i=1}^{d} \ln u_i}$, где

$B(\vec{w}) = \int_{S_d} \max(x_1 w_1, \dots, x_d w_d)\, dH(\vec{x})$,

$S_d = \left\{\vec{x} : x_i \geq 0,\ i = 1, \dots, d,\ \sum_{i=1}^{d} x_i = 1\right\}$

Примеры

Пример 3. Копула Гумбеля

$$C^{Gu}_{\theta,\alpha,\beta}(u_1,u_2) = u_1^{1-\alpha}u_2^{1-\beta}e^{-\left((-\alpha\ln u_1)^{\theta}+(-\beta\ln u_2)^{\theta}\right)^{\frac{1}{\theta}}}$$

Из (2) имеем: $A(w) = (1-\alpha)w + (1-\beta)(1-w) + \left((\alpha w)^{\theta} + \left(\beta(1-w)\right)^{\theta}\right)^{\frac{1}{\theta}}$

Пример 4. Копула Галамбоса

Пусть $A(w) = 1 - \left((\alpha w)^{-\theta} + \left(\beta(1-w)\right)^{-\theta}\right)^{\frac{1}{\theta}}$, где

$\alpha,\beta \in [0;1],\ \ \theta > 0$, тогда с помощью (1) можно сконструировать копулу:

$$C^{Gal}_{\theta,\alpha,\beta}(u_1,u_2) = u_1u_2e^{\left((-\alpha\ln u_1)^{-\theta}+(-\beta\ln u_2)^{-\theta}\right)^{-\frac{1}{\theta}}}$$

MGEV в R

Практический пример 2. Биржевые индексы DAX и FTSE

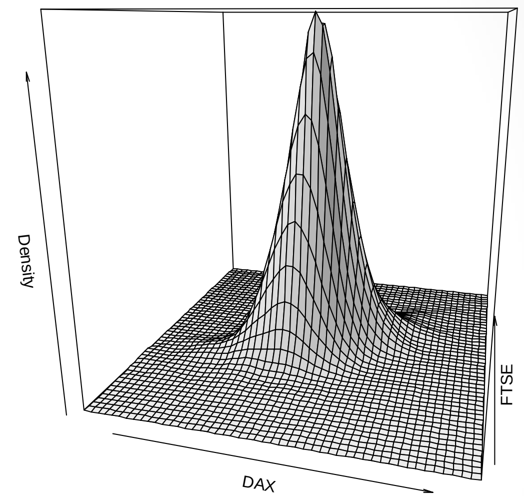
загрузка данных

```
ftse <- EuStockMarkets[,4]
ftse <- ftse[2:(T+1)]/ftse[1:T]-1
ftse <- - ftse
ESM <- cbind(dax,ftse)
```

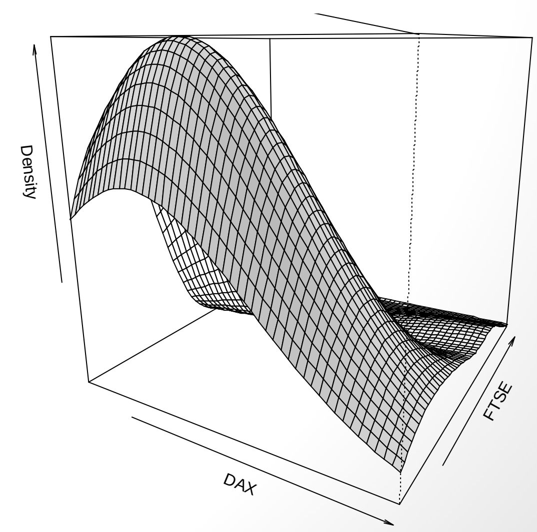
расчёт максим

```
Mn <- rep(0,times=m*2)
dim(Mn) <- c(m,2)
for (i in 1:2) {
  for (j in 1:m)
  Mn[j,i] <- max(ESM[((j-1)*n+1):(j*n),i])
}
```

Эмпирическое распределение доходностей DAX и FTSE



Эмпирическое распределение максим



MGEV в R

\# частные распределения на основе GED

```
fit1 <- fgev(Mn[,1])
fit2 <- fgev(Mn[,2])
```

\# экстремальные копулы

```
library(copula)
gumb.cop <- gumbelCopula(2)
gal.cop <- galambosCopula(2)
```

\# значения частных функций распределения

```
cdf1 <- pgev(Mn[,1],loc=fit1$estimate[1],
    scale=fit1$estimate[2],shape=fit1$estimate[3])
cdf2 <- pgev(Mn[,2],loc=fit2$estimate[1],
    scale=fit2$estimate[2],shape=fit2$estimate[3])
cdf <- cbind(cdf1,cdf2)
```

MGEV в R

подгонка копулы

```
gumb.fit <- fitCopula(cdf,copula=gumb.cop)
gal.fit <- fitCopula(cdf,copula=gal.cop)
```

gumb.fit@loglik	5.798
gal.fit@loglik	5.846

модельные значения максим

```
N <- 10^5
cdf.sim <- rcopula(n=N,copula=gal.fit@copula)
sim1 <- qgev(cdf.sim[,1],loc=fit1$estimate[1],
    scale=fit1$estimate[2],shape=fit1$estimate[3])
sim2 <- qgev(cdf.sim[,2],loc=fit2$estimate[1],
    scale=fit2$estimate[2],shape=fit2$estimate[3])
```

MGEV в R

\# модельные убытки

```
w <- c(0.5,0.5)
loss <- sort(w[1]*sim1+w[2]*sim2)
```

\# расчёт мер риска

```
k <- 4
alpha <- 1-1/k
VaR <- loss[alpha*N]
ES <- mean(loss[(alpha*N+1):N])
```

VaR	**0.029**
ES	**0.043**

Превышение многомерного порога

Пусть $\vec{x}_1, \dots, \vec{x}_n \sim F(\vec{x}) = C(F_1(x_1), \dots, F_d(x_d)) \in MDA(MGEV)$

Согласно теории для одномерного случая частные распределения величин, превышающих многомерный порог $\vec{u} = (u_1, \dots, u_d)$ имеет вид:

$$\tilde{F}_i(x_i) = 1 - \bar{F}_i(u_i)\left(1 + \frac{\xi_i(x_i - \mu_i)}{\beta_i}\right)^{\frac{1}{\xi_i}},\ \vec{x} \ge \vec{u}$$

Для многомерного случая используется приближение

$$F(\vec{x}) \approx C\left(\tilde{F}_1(x_1), \dots, \tilde{F}_d(x_d)\right),\ \vec{x} \ge \vec{u}$$

Поскольку исходное распределение $F(\vec{x})$ неизвестно, копулу *C(.)* также нужно аппроксимировать

Для этого применяется экстремальная копула:

$$F(\vec{x}) \approx C_0\left(\tilde{F}_1(x_1), \dots, \tilde{F}_d(x_d)\right),\ \vec{x} \ge \vec{u}$$

Превышение многомерного порога в R

\# выборка значений, превышающих многомерный порог

```
u <- c(sort(dax)[0.9*T],sort(ftse)[0.9*T])
t.ESM <- ESM[(ESM[,1]>u[1])&(ESM[,2]>u[2]),]
```

\# частные распределения на основе GED

```
fit1 <- fpot(t.ESM[,1],threshold=u[1],
    model="gpd",method="SANN")
fit2 <- fpot(t.ESM[,2],threshold=u[2],
    model="gpd",method="SANN")
```

\# значения частных функций распределения

```
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)
```

Превышение многомерного порога в R

\# подгонка копулы

```
gumb.fit <- fitCopula(cdf,copula=gumb.cop)
gal.fit <- fitCopula(cdf,copula=gal.cop)
```

gumb.fit@loglik	**12.27**
gal.fit@loglik	**12.69**

\# модельные значения убытков

```
cdf.sim <- rcopula(n=N,copula=gal.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])
```

Превышение многомерного порога в R

\# убытки по портфелю

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

\# расчёт мер риска

```
Fu <- nrow(t.ESM)/T
alpha <- 1-1/(260*Fu)
VaR <- loss[alpha*N]
ES <- mean(loss[(alpha*N+1):N])
```

VaR	0.029
ES	0.037

Непараметрическое моделирование (многомерный случай)

Оценки плотности

Простая оценка плотности в двумерной точке (y_1, y_2):

$$\hat{f}(y_1, y_2) = \frac{1}{nh^2}\sum_{i=1}^{n}\left(I\left(y_1 - \frac{h}{2} < y_{i,1} < y_1 + \frac{h}{2}\right) \cdot I\left(y_2 - \frac{h}{2} < y_{i,2} < y_2 + \frac{h}{2}\right)\right) \quad (26)$$

Заменим индикаторы на ядерные функции:

$$\hat{f}(y_1, y_2) = \frac{1}{nh^2}\sum_{i=1}^{n}\left(K\left(\frac{y_{1,i} - y_1}{h}\right)K\left(\frac{y_{2,i} - y_2}{h}\right)\right) \quad (27)$$

Оценка (27) не обязательно даёт одинаковый результат для всех точек, равноудалённых от пары (y_1, y_2)

Проблема решается с помощью многомерного ядра:

$$\hat{f}(y_1, y_2) = \frac{1}{nh^2}\sum_{i=1}^{n} K\left(\frac{y_{1,i} - y_1}{h}, \frac{y_{2,i} - y_2}{h}\right) \quad (28)$$

В качестве $K(x_1, x_2)$ обычно берутся одномодальные симметричные многомерные функции плотности

Двумерные ядерные функции

Ядро Гаусса:

$$K_G(x_1, x_2) = \frac{1}{2\pi} \exp\left(-\frac{x_1^2 + x_2^2}{2}\right) \quad (29)$$

Двумерное ядро Гаусса в точности равно произведению двух одномерных:

$$K_G(x_1, x_2) \equiv K_G(x_1) \cdot K_G(x_2)$$

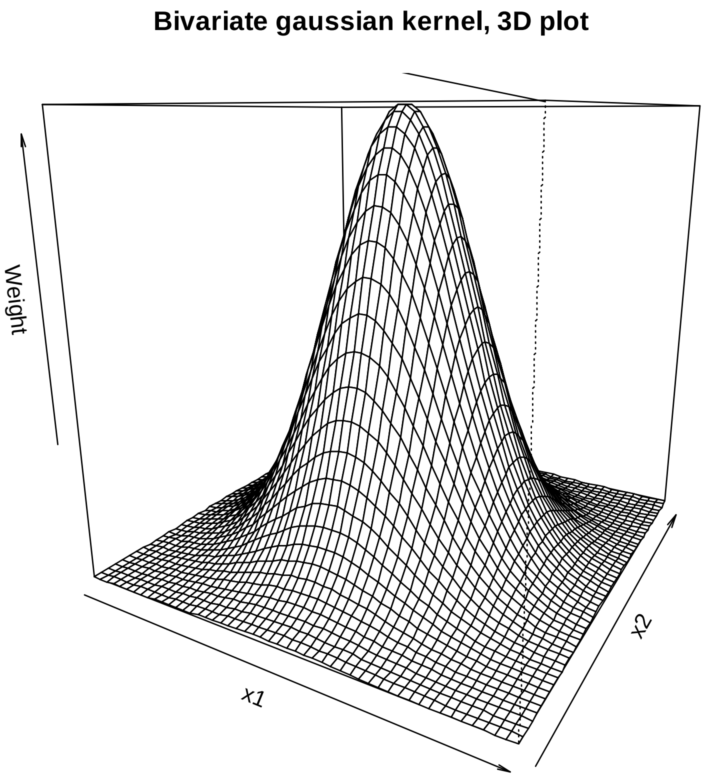
Ядро Епанечникова:

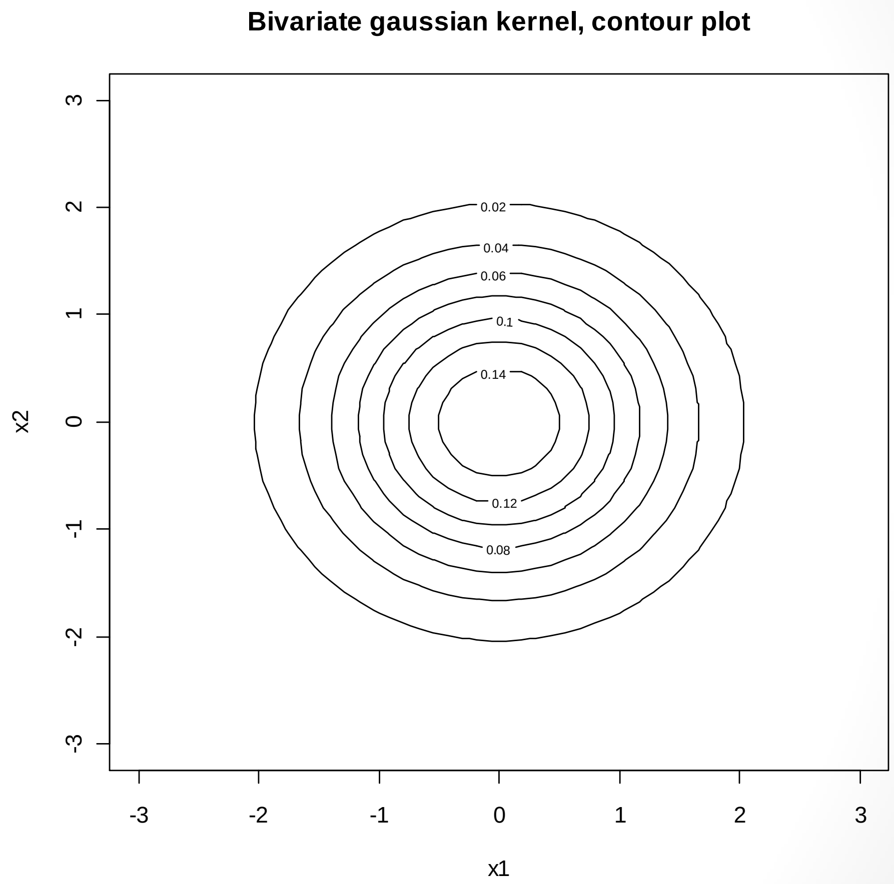
$$K_E(x_1, x_2) = \frac{2}{\pi}(1 - x_1^2 - x_2^2) \cdot I(x_1^2 + x_2^2 < 1) \;\; (30)$$

Двумерное ядро не равно произведению двух одномерных:

$$K_E(x_1, x_2) \neq K_E(x_1) \cdot K_E(x_2)$$

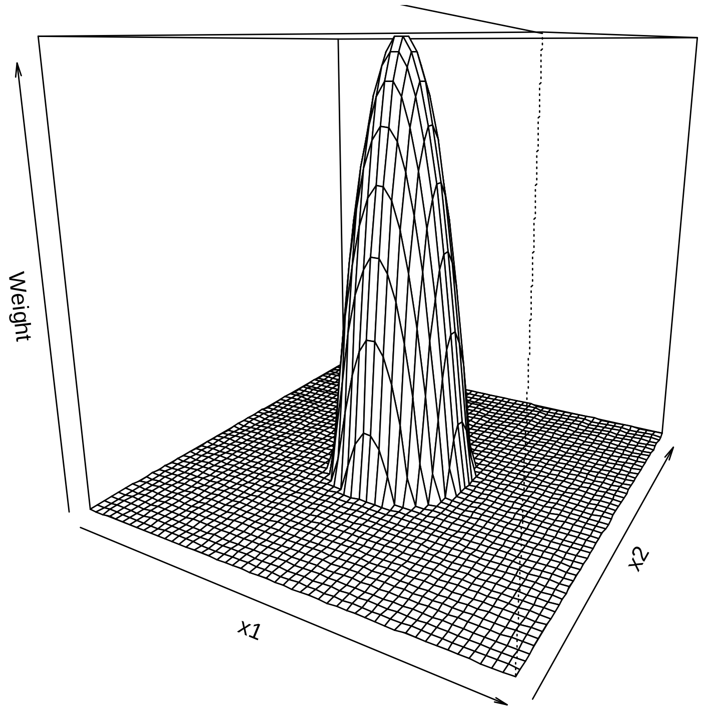
Двумерное гауссовское ядро



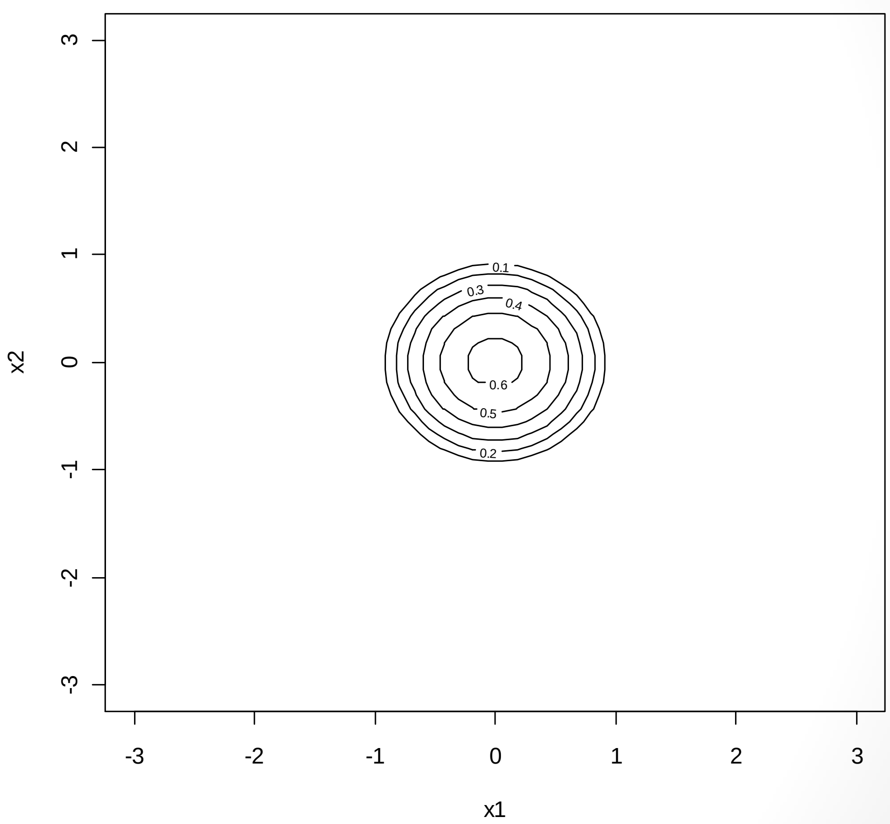


Двумерное ядро Епанечникова

Bivariate Epanechnikov kernel, 3D plot



Bivariate Epanechnikov kernel, contour plot

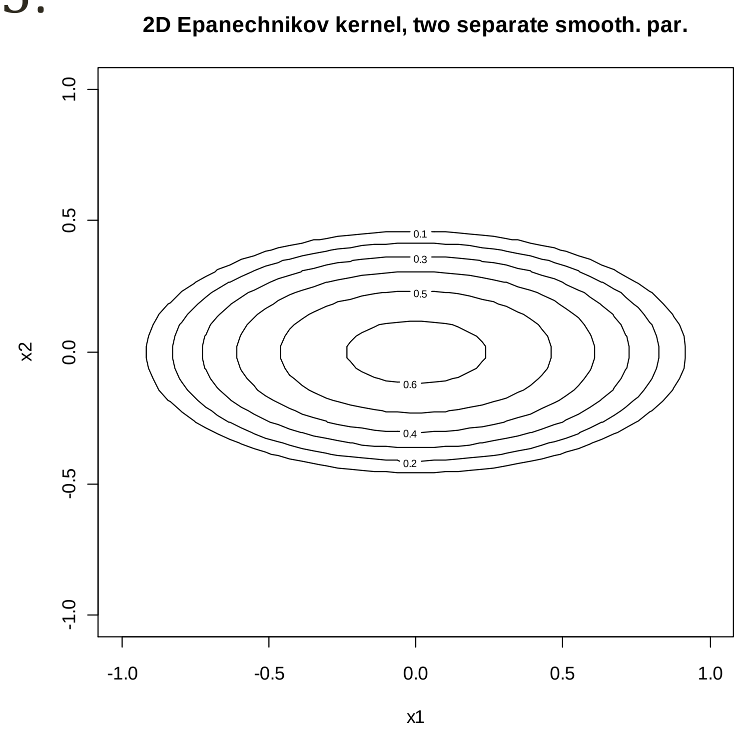


Двумерные ядерные функции

Различные сглаживающие параметры

Если разброс данных в первой и во второй выборке сильно отличаются, то можно использовать для этих выборок разные сглаживающие параметры $h_1 \neq h_2$

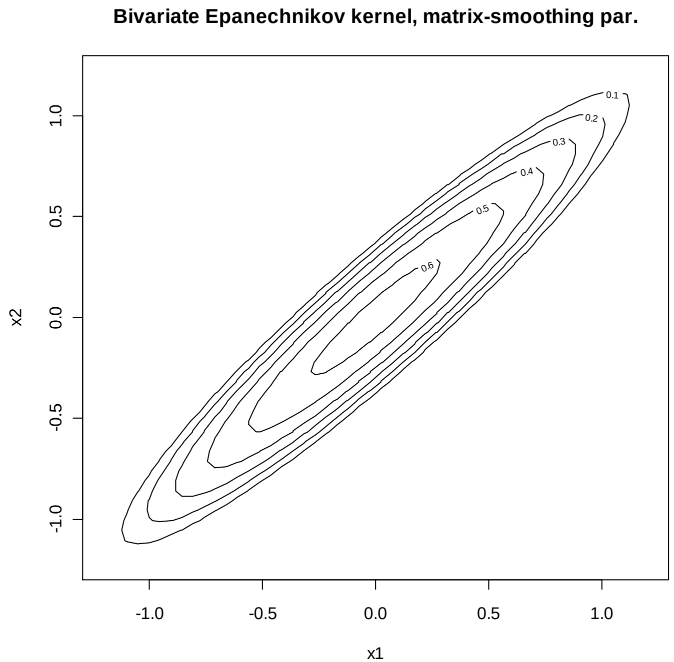
На рисунке ниже представлено двумерное ядро Епанечникова $K\left(\frac{x_1}{h_1}, \frac{x_2}{h_2}\right)$ со сглаживающими параметрами $h_1 = 1,\ h_2 = 0.5$:



Сглаживающая матрица

Если рассматриваемые величины коррелируют, это учитывается с помощью симметричной положительно определённой сглаживающей матрицы (matrix-smoothing parameter) *H*, которая в двумерном случае состоит их 4-х элементов: $H = \begin{pmatrix} h_1 & h_{12} \\ h_{21} & h_2 \end{pmatrix},\ h_{12} = h_{21}$

Для $h_1 = h_2 = 1,\ h_{12} = h_{21} = \sqrt{0.5}$, получим:



Общий случай

Ядерная оценка *d*-мерной плотности:

$$\hat{f}(\vec{y}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|H|}K(H^{-1}(\vec{y}-\vec{y}_i)), \quad (31)$$

$\vec{y} = (y_1, \dots, y_d)$, $|H|$ — определитель матрицы H

Пусть $\vec{x} = (x_1, \dots, x_d)$, тогда *d*-мерные ядра Гаусса и Епанечникова запишутся как

$$K_G(\vec{x}) = (2\pi)^{-\frac{d}{2}}\exp\left(-\frac{\vec{x}\vec{x}'}{2}\right), \quad (32)$$

$$K_E(\vec{x}) = (2c_d)^{-1}(d+2)(1-\vec{x}\vec{x}')\cdot I(\vec{x}\vec{x}' < 1), \quad (33)$$

c_d — объём единичного *d*-мерного шара

Правило подстановки

Если в качестве подставляемого распределения использовать нормальное $N(\mu, \Sigma)$, $\Sigma = \left(\sigma_1^2, \dots, \sigma_d^2\right) \cdot$ ב, ב$_{[d \times d]}$ — единичная матрица, то по критерию *MISE*, оптимальная диагональная матрица *H* состоит из элементов

$$h_j = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} n^{-\frac{1}{d+4}} \sigma_j \quad (34)$$

Так как первый множитель при любых *d* приблизительно равен единице на практике используют правило

$$\hat{h} = n^{-\frac{1}{d+4}} \hat{\sigma}_j \quad (35)$$

Обобщённое правило подстановки:

$$\hat{H} = n^{-\frac{1}{d+4}} \hat{\Sigma}^{\frac{1}{2}} \quad (36)$$

Правило подстановки

На практике обобщённое правило подстановки (36) используют следующим образом:

- данные преобразуются так, чтобы они имели единичную ковариационную матрицу $\Sigma = ב$;
- строится оценка плотности с единственным сглаживающим параметром, $\widehat{H} = \hat{h} \cdot ב$, $\hat{h} = n^{-\frac{1}{d+4}}$;
- выполняется обратное преобразование для полученной оценки

Метод перекрёстной проверки

Метод перекрёстной проверки также может быть обобщён на многомерный случай, однако при этом он становится достаточно сложным, требующим ресурсоёмких вычислений

Алгоритм практического применения метода аналогичен правилу подстановки, но в этом случае, при предположении, что $K(\vec{x})$ — симметричная функция, оценка $\hat{h}$ находится путём минимизации выражения

$$CV(h) = \frac{1}{n^2 h^d} \sum_{i=1}^{n} \sum_{j=1}^{n} K^* \left(\frac{\vec{y}_i - \vec{y}_j}{h} \right) + \frac{2}{n h^d} K\left(0_{[1 \times d]}\right), \quad (37)$$

$$K^*(\vec{x}) = \int_{R^d} K\left(\vec{t}\right) K\left(\vec{x} - \vec{t}\right) d\vec{t} - 2K(\vec{x}),$$

$$\vec{t} = (t_1, \dots, t_d)$$

Обобщённый метод ближайших соседей

Пусть $d_k(\vec{y})$ — евклидово расстояние от точки $\vec{y}$ до *k*-го ближайшего наблюдения в выборке, $V_k(\vec{y})$ — объём *d*-мерного шара радиусом $d_k(\vec{y}), V_k(\vec{y}) = c_d d_k^d(\vec{y})$

В этом случае простая оценка равна

$$\hat{f}(\vec{y}) = \frac{k}{nV_k(\vec{y})} = \frac{k}{nc_d d_k^d(\vec{y})}, \quad (38)$$

что аналогично одномерной оценке (22), так как $c_1 = 2$

Оценка (38) может быть обобщена с помощью ядер:

$$\hat{f}(\vec{y}) = \frac{1}{c_d n d_k^d(\vec{y})} \sum_{i=1}^{n} K\left(\frac{\vec{y}-\vec{y}_i}{c_d d_k(\vec{y})}\right), \quad (39)$$

что аналогично одномерной оценке (23)

Адаптивный метод ближайших соседей

Рассмотрим $d_k(\vec{y}_i)$ — расстояние от элемента $\vec{y}_i$ до k-го ближайшего элемента выборки

Оценка плотности по адаптивному методу равна

$$\hat{f}(\vec{y}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d d_k^d(\vec{y}_i)} K\left(\frac{\vec{y}-\vec{y}_i}{h d_k(\vec{y}_i)}\right), \quad (40)$$

что аналогично одномерной оценке (24)

Как и в одномерном случае показатель локальной концентрации наблюдений $d_k(\vec{y}_i)$ часто заменяется на величину

$$\lambda_i = \left(\frac{g}{\tilde{f}(\vec{y}_i)}\right)^{\alpha}, \ \alpha \in [0; 1], \quad (41)$$

$g = \left(\prod_{i=1}^{n} \tilde{f}(y_i)\right)^{\frac{1}{n}}$ — геометрическое среднее пилотных оценок плотности

Пример 2. Старый служака

```
y <- faithful; N <- nrow(y)
```

\# сетка для расчёта оценок плотности

```
L <- 50; u <- seq(0,7,length=L); v <- seq(30,110,length=L)
uv <- expand.grid(u,v)
```

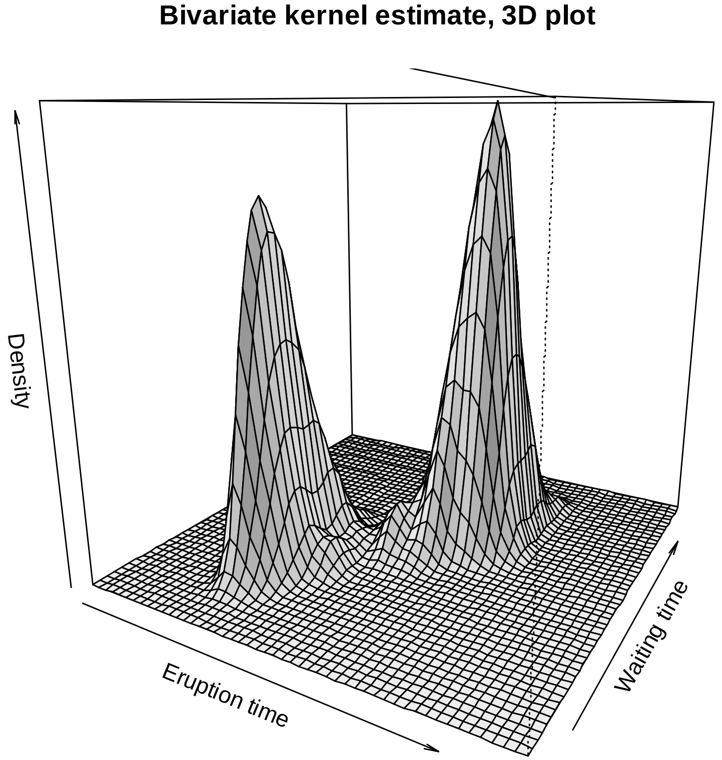
\# оценка плотности

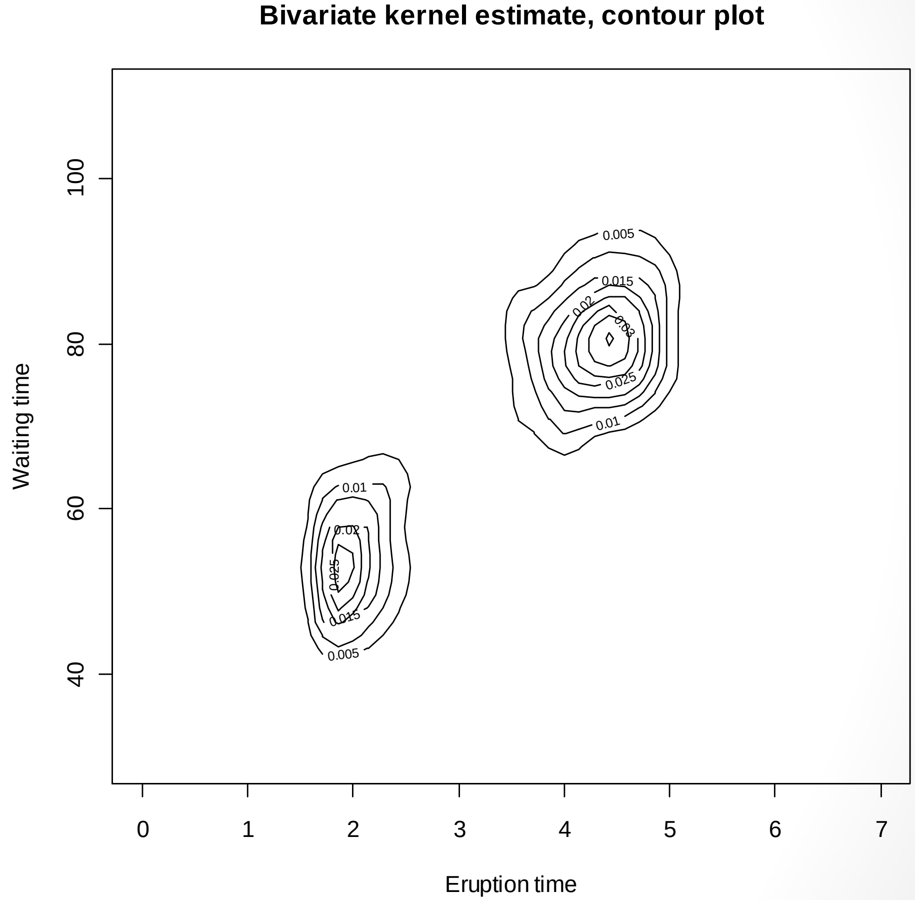
```
f.fix <- npudens(tdat=y,edat=uv,ckertype="gaussian",bwtype="fixed")
```

\# графики оценки

```
w <- f.fix$dens; dim(w) <- c(L,L)

persp(u,v,w,theta=30,main="Bivariate kernel estimate, 3D plot",
xlab="Eruption time",ylab="Waiting time",zlab="Density")

contour(u,v,w,nlevel=7,
main="Bivariate kernel estimate, contour plot",
xlab="Eruption time",ylab="Waiting time")
```

Пример 2. Старый служака



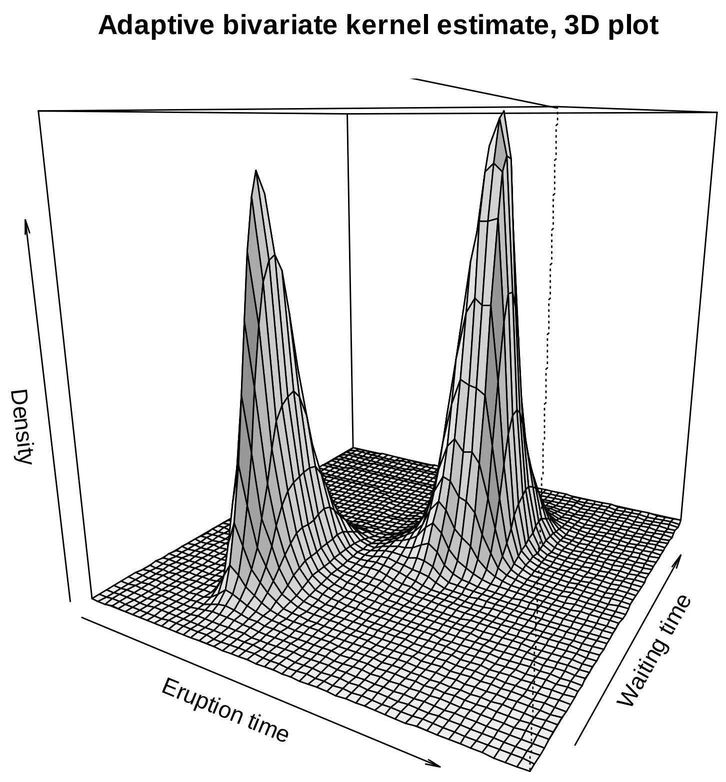


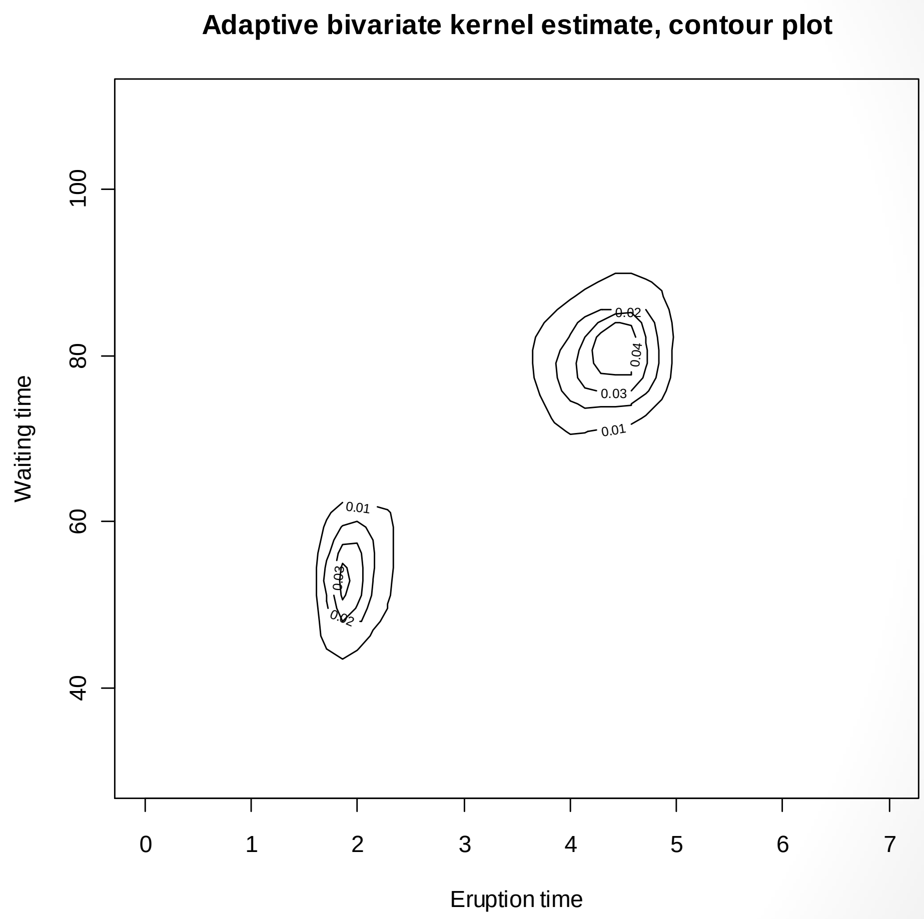
Пример 2. Старый служака

Адаптивный метод с λ_i, аналогично одномерному случаю

```
pilot <- npudens(tdat=y,ckertype="gaussian",bwtype="fixed")
h <- pilot$bws$bw

g <- 1
for (i in 1:N) g <- g*pilot$dens[i]^(1/N)

alpha <- 0.5
lmbd <- (g/pilot$dens)^alpha

kern <- function(x) exp(-(x[1]^2+x[2]^2)/2)/(2*pi)

f <- rep(0,times=L^2)
for (i in 1:(L^2)) {
  f[i] <- sum(kern((uv[i,]-y)/(h*lmbd))/lmbd^2
}
f <- f / (N*h[1]*h[2])
```

Пример 2. Старый служака





Пример 2. Старый служака

Значения логарифмической функции правдоподобия

\# оценки плотности в точках y_i

```
f.fix.llh <- npudens(tdat=y,ckertype="gaussian",bwtype="fixed")
llh.fix <- sum(log(f.fix.llh$dens))
```

\# для адаптивного метода

```
f.llh <- rep(0,times=N)
for (i in 1:N) {
  for (j in 1:N) f.llh[i] <- f.llh[i]+kern((y[i,]-
    y[j,])/(h*lmbd[j]))/lmbd[j]^2
  f.llh[i] <- f.llh[i]/(N*h[1]*h[2])
}
llh.ada <- sum(log(f.llh))
```

llh.fix	**-1106**
llh.ada	**-1114**

Пример 2. Старый служака

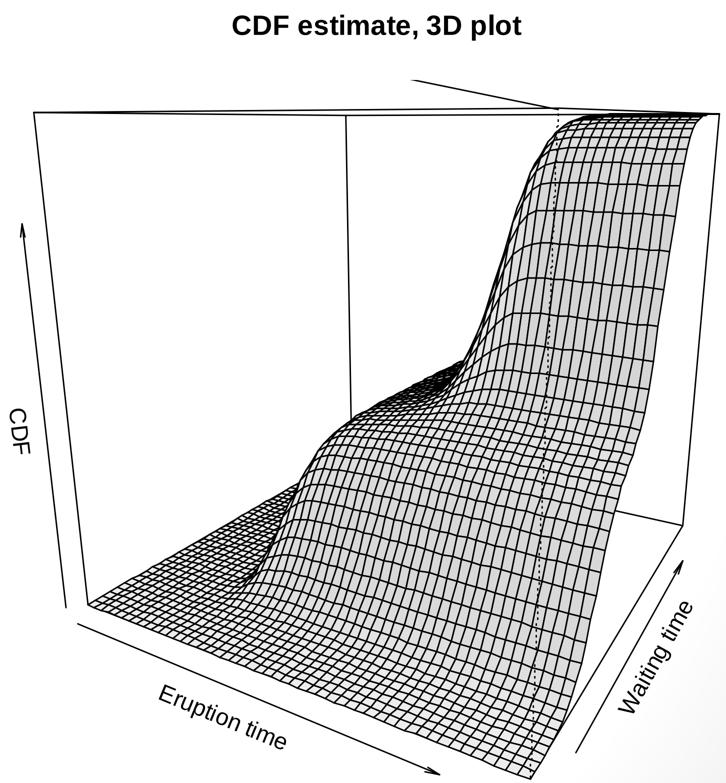
Расчёт функций распределения

фиксированный метод

```
F.fix <- npudist(tdat=y,edat=uv,ckertype="gaussian",bwtype="fixed")
```

адаптивный метод

```
du <- u[2]-u[1]; dv <- v[2]-v[1]
w <- f; dim(w) <- c(L,L)
F <- rep(0,times=L^2)
for (i in 1:L) {
  for (j in 1:L) F[j+(i-1)*L] <-
  sum(w[1:j,1:i])*du*dv
}
```

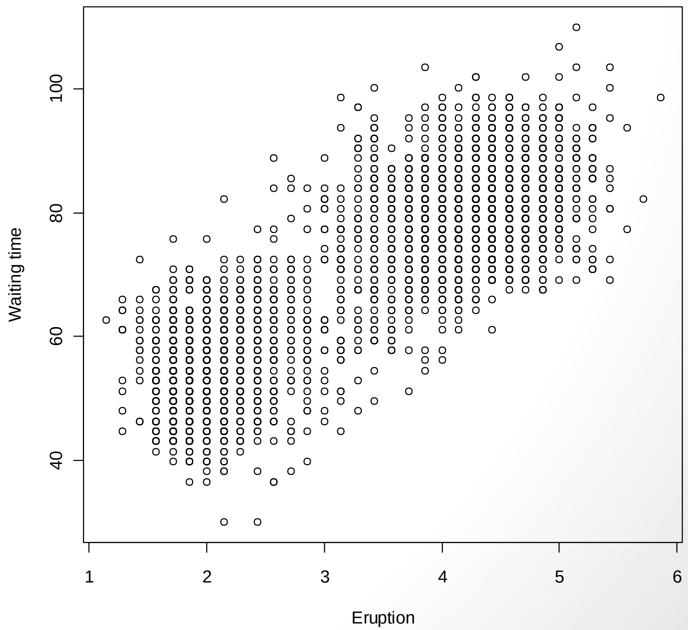


Пример 2. Старый служака

Генератор случайных чисел

для адаптивного метода

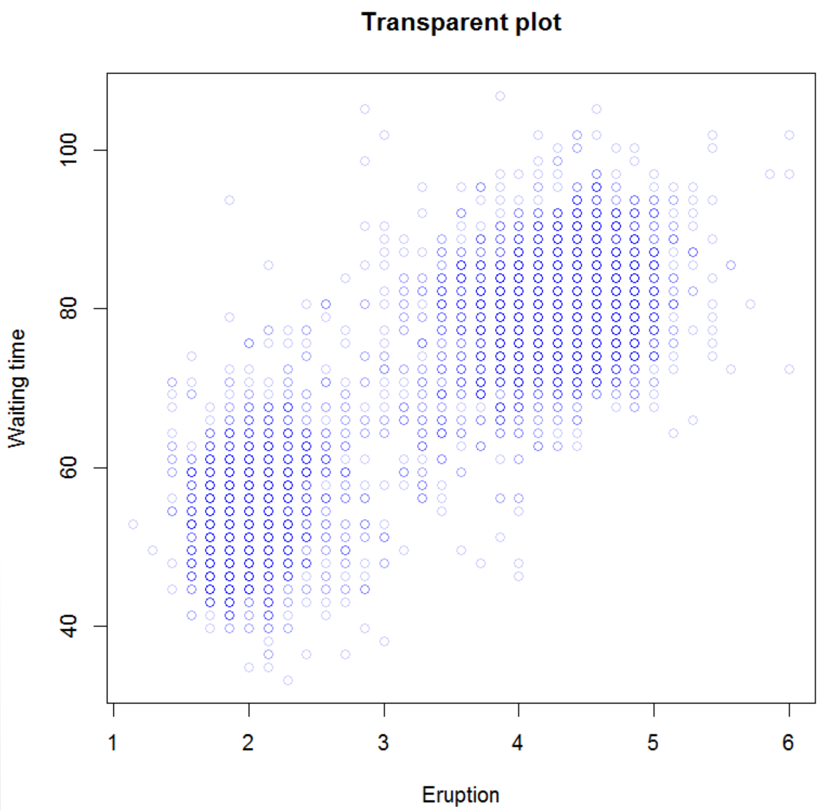
```
alpha <- 0.99
M <- 5000
smpl.ind <- sample(1:(L^2),prob=f,size=M,replace=TRUE)
y.ada.sim <- uv[smpl.ind,]
plot(y.ada.sim,xlab="Eruption",ylab="Waiting time")
```

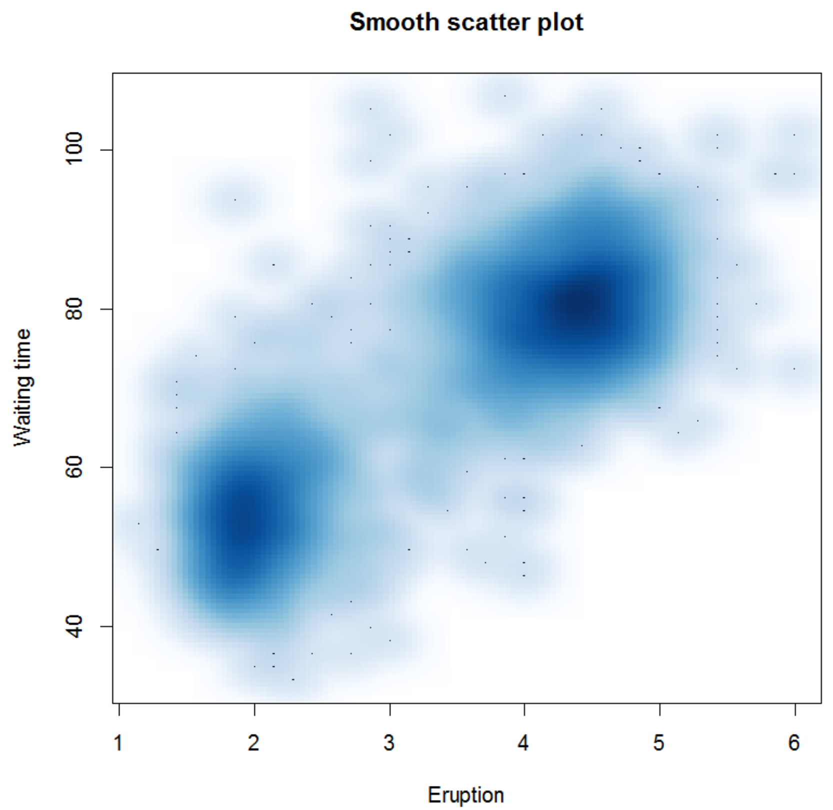


Пример 2. Старый служака

Рисование графиков с перекрывающими друг друга точками

```
plot(y.ada.sim,col=rgb(0,0,1,alpha=0.2))
smoothScatter(y.ada.sim)
```





Домашнее задание

- рассчитать оценки риска для портфеля из акций или биржевых индексов по всей совокупности наблюдений на основе многомерных вариантов моделей семейства GARCH, ОГР, а также с помощью инструментария ТЭЗ и непараметрического моделирования
- построить кривые VaR для указанных моделей и проверить качество оценок риска

Исходные данные — котировки с сайта finam.ru, finance.yahoo.com и др.