Лекция 2 - Кривая VaR, пакет ghyp и Монте-Карло

Илья Езепов

25 сентября 2015

Введение

Сегодняшняя лекция будет состоять из нескольких частей, которые весьма независимы, но на самом деле связаны. Пожалуйста, прерывайте меня, если какая-либо часть кода не будет вам понятна. Или не будет понятно какое-либо из произнесенных мной слов. Договорились? Начнем.

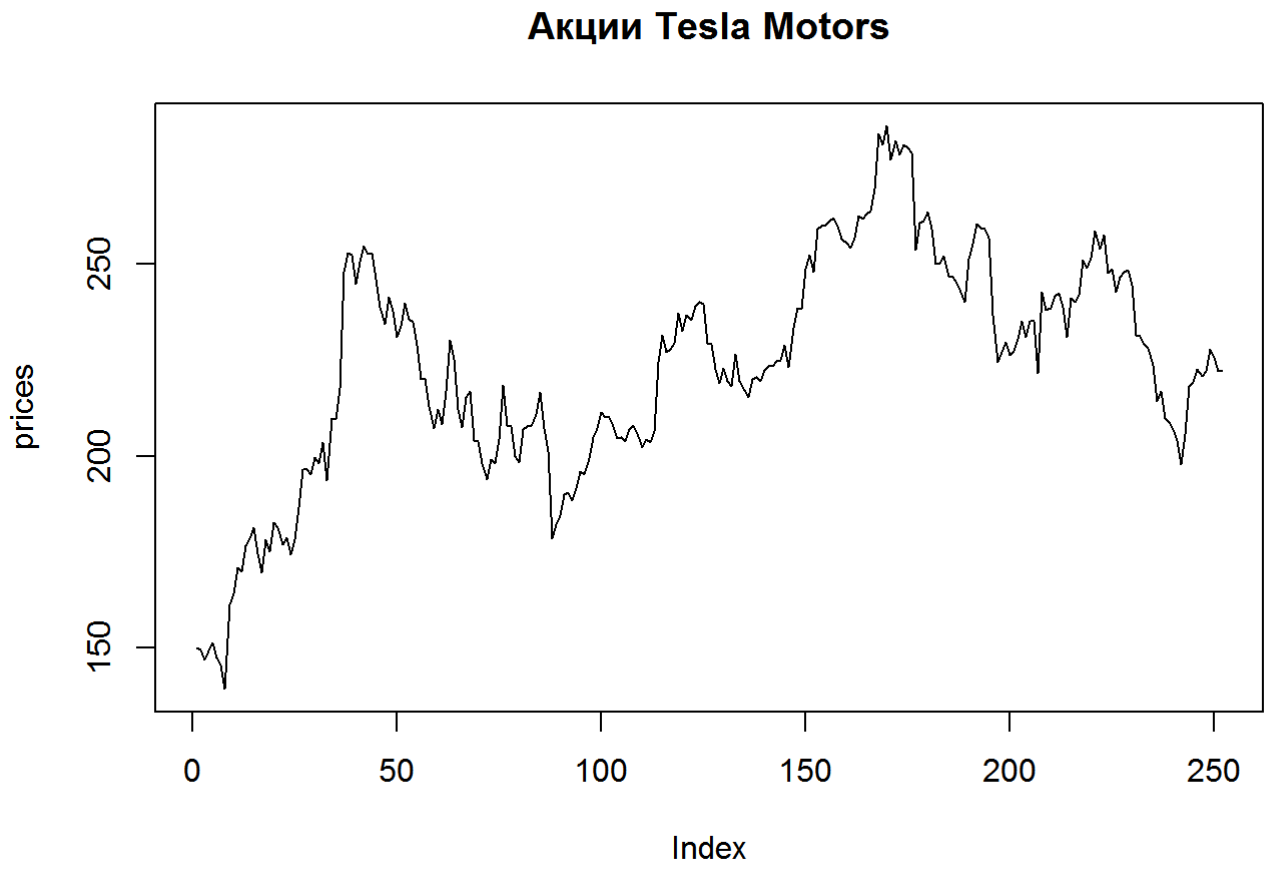
Часть первая, в которой читатель узнает, как измерить риск финансовго актива

Финансовые продукты характеризуются двумя основными характеристикам – доходностью и риском. С доходностью все довольно понятно – это процентное изменение стоимости за некоторый промежуток времени. Чтобы не быть голословным, будем изучать все на каких-нибудь примерах. Возьмем котировки компании Tesla Motors за 2014 год:

```
library(quantmod)
getSymbols("TSLA", from="2014-01-01", to="2015-01-01")
```

```
## [1] "TSLA"
```

```
prices <- as.numeric(Ad(TSLA))
plot(prices, type="l", main="Акции Tesla Motors")
```



Как найти годовую доходность? Это просто процентное изменение последней точки к первой:

```
tail(prices, 1)/head(prices, 1) - 1
```

```
## [1] 0.4817455
```

Это много? Почему?

Но вот что считать мерой риска? Мы рассмотрим 4 популярные характеристики, а дальше сосредоточимся на двух.

Стандартное отклонение доходности

Название говорит само за себя. Возьмем ряд доходностей, найдем его стандартное отклонение.

```
ret <- diff(prices)/head(prices, -1)
mean(ret)
```

```
## [1] 0.002024531
```

```
sd(ret)
```

```
## [1] 0.03049802
```

А это много? :)

NB: Сравнивать акции на основе стандартного отклонения можно при равных доходностях. А чтобы сравнивать разные акции нужно использовать комбинированные метрики. Например, Коэффициент Шарпа – отношение ожидаемой доходности к ее стандартному отклонению:

```
mean(ret)/sd(ret)
```

```
## [1] 0.06638238
```

Такой метод оценки риска хорош, так как прост и понятен, но есть пара минусов. Во-первых стандартное отклонение ничего не говорит об экстремальности распределения. Во-вторых, стандартное отклонение не естественно, то есть на интуитивном уровне не понятно.

Максимальная просадка

Максимальная просадка (maximum drawdown) – это то, насколько сильно просел наш актив после максимального влета. Для графика выше максимальная просадка - это разница между максимальным значением около 170 и минимальным около 250. В R есть готовые способы найти максимальную просадку. Но мы же хотим все прочувствовать сами. Алгоритм:

1. говорим, что максимальная просадка равна 0.
2. Берем первую точку и объявляем ее максимумом цены.
3. Идем к следующей точке.

- Если она больше максимума, то обновляем максимум.
- Если меньше, то считаем в ней просадку. Если просадка больше максимальной, то обновляем максимальную

4. Повторяем шаг 3, пока не дойдем до конца.

```
maxDD <- 0
maxPr <- prices[1]

for (i in 2:length(prices)) { # Проходим по всем точкам, начиная со второй
  if (prices[i] > maxPr) { # Проверяем не больше ли она текущего максимума
    maxPr <- prices[i] # Если больше, то меняем максимум на эту точку
  } else {
    DD <- (maxPr - prices[i])/maxPr # А иначе считаем просадку
    if (DD > maxDD) { # И смотрим не больше ли эта просадка максимальной
      maxDD <- DD # А если больше, то меняем значение максимальной просадки
      }
  }
}
maxPr
```

```
## [1] 286.04
```

```
maxDD
```

```
## [1] 0.3084534
```

Чем хороша эта мера? А чем плоха?

Очень важно понять, как работает цикл `for` и оператор `if` . Дальше их будет все больше, а циклы будут работать часами.

Value-at-Risk

… или статистика 16:15. Value-at-Risk на каком-либо уровне (скажем, 5%) – это просто квантиль на уровне 5%. То есть это такая доходность, что в 95% случаев у нас дела будут лучше. Вот если сейчас не понятно, то точно стоит меня прервать. Считаем ее элементарно:

```
quantile(ret, 0.05)
```

```
##          5%
## -0.04236016
```

И чуть более продвинуто:

```
sort(ret)[0.05*length(ret)]
```

```
## [1] -0.04307281
```

VaR хорош, так как не требует гипотез о хоть какой-либо форме распределения и описывает именно ту часть распределения, которую мы боимся. Но VaR - это лучшее из 5% худших случаев. А что лежит в тех 5%?…

Expected shortfall

Он же ES, conditional value at risk (CVaR), average value at risk (AVaR), и expected tail loss (ETL). Но мы будем говорить ES. ES - Это среднее значение по всем тем точкам, что оказались левее VaR, То есть среднее по худшим 5%. Считаем:

```
mean(ret[ret <= quantile(ret, 0.05)]) # Попросите меня разобрать эту формулу подробне
e.
```

```
## [1] -0.06072323
```

Чем ES хуже VaR? А чем лучше?

Кто побеждает?

Да никто. Все рассуждения о риске – это попытки запихнуть функцию распределения доходностей в одно единственное число. Поэтому информация потеряется и все метрики будут не идеальны. В лучшем случае посредственны. А к чему это приведет, можно прочитать в этой книге (http://www.ozon.ru/context/detail/id/22685935/) (обязательно к прочтению для тех, кто видит себя трейдером, и занятное чтиво для остальных).

А чем будем пользоваться мы? VaR и иногда ES. И еще пару метрик появятся по ходу изложения курса.

Часть вторая, в которой мы будем оценивать качество разных вещей

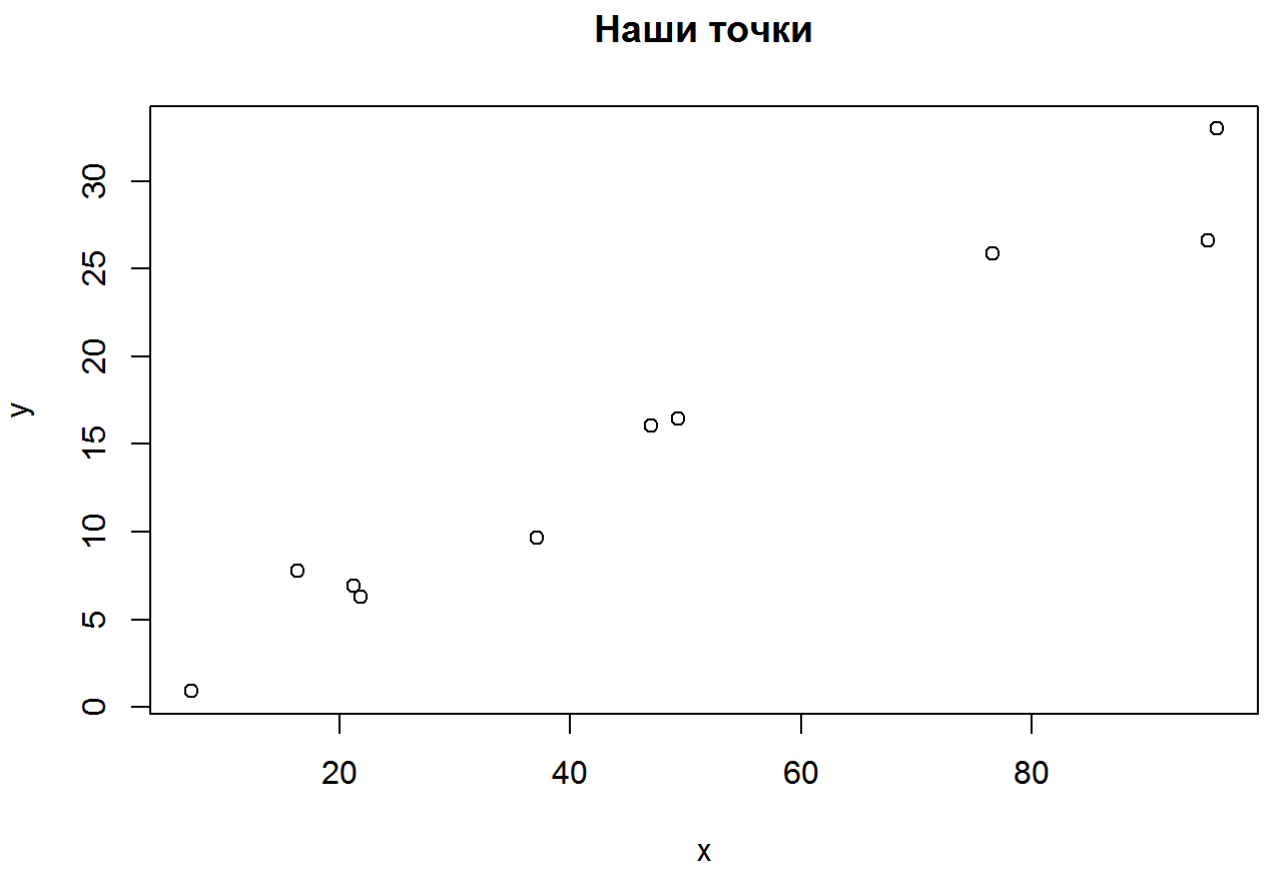
Вот мы уже умеем находим VaR и понимаем, что это такое. Взяли цены акций за год, нашли квантиль 5%. И говорим, что в **будущем** акции будут вести себя так же. То есть занимаемся моделированием, предсказанием, алхимией. А значит нужно научиться проверять наши прогнозы.

Тренировочная и тестовая выборка

Но сначала о методологии анализа данных. Если вы занимаетесь моделированием чего угодно на данных, то вам нужно иметь две выборки: тренировочную и тестовую. На тренировочной вы строите модель, а на тестовой должны проверить ее качество. Давайте на примерах.

Возьмем вектор из 10 случайных чисел от 0 до 100. И поставим другой вектор в линейную зависимость от этих точек. Ну и добавим шум.

```
x <- runif(10, min = 0, max = 100)
y <- 0.3*x + rnorm(10, sd = 2)
plot(x, y, main="Наши точки")
```



Все готово к построению парной регрессии! Кстати, в R это очень просто.

```
model <- lm(y ~ x)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7506 -1.2663  0.4018  1.3685  2.4899
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0837     1.2394   0.068    0.948
## x             0.3178     0.0221  14.381 5.34e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.16 on 8 degrees of freedom
## Multiple R-squared:  0.9628, Adjusted R-squared:  0.9581
## F-statistic: 206.8 on 1 and 8 DF,  p-value: 5.341e-07
```

```
plot(x, y, main="А теперь спрямой, полученной  МНК")
abline(model)
```

Но я ведь могу описать лучше!

```
model2 <- lm(y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8)) # Не хочешь немног
о полиномов?
summary(model2)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) +
##     I(x^7) + I(x^8))
##
## Residuals:
##          1          2          3          4          5          6
##  0.0006373  0.0134281  0.0004969 -0.0258080 -0.0002272  0.2039455
##          7          8          9         10
## -0.0171726 -0.0004412 -0.1932479  0.0183890
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.593e+01  2.508e+01   1.034    0.489
## x           -1.161e+01  8.140e+00  -1.427    0.389
## I(x^2)       1.886e+00  9.894e-01   1.906    0.308
## I(x^3)      -1.374e-01  6.071e-02  -2.263    0.265
## I(x^4)       5.283e-03  2.090e-03   2.528    0.240
## I(x^5)      -1.142e-04  4.184e-05  -2.729    0.224
## I(x^6)       1.392e-06  4.809e-07   2.895    0.212
## I(x^7)      -8.928e-09  2.933e-09  -3.044    0.202
## I(x^8)       2.340e-11  7.337e-12   3.189    0.193
##
## Residual standard error: 0.2836 on 1 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9993
## F-statistic:  1558 on 8 and 1 DF,  p-value: 0.01959
```

```
plot(x,y, main="Еще более классная модель")
lines(1:100, predict(model2, newdata = data.frame(x=1:100)) )
```

Сравним ошибки на тренировочных выборках:

```
sum((predict(model)-y)^2)/10
```

```
## [1] 3.732779
```

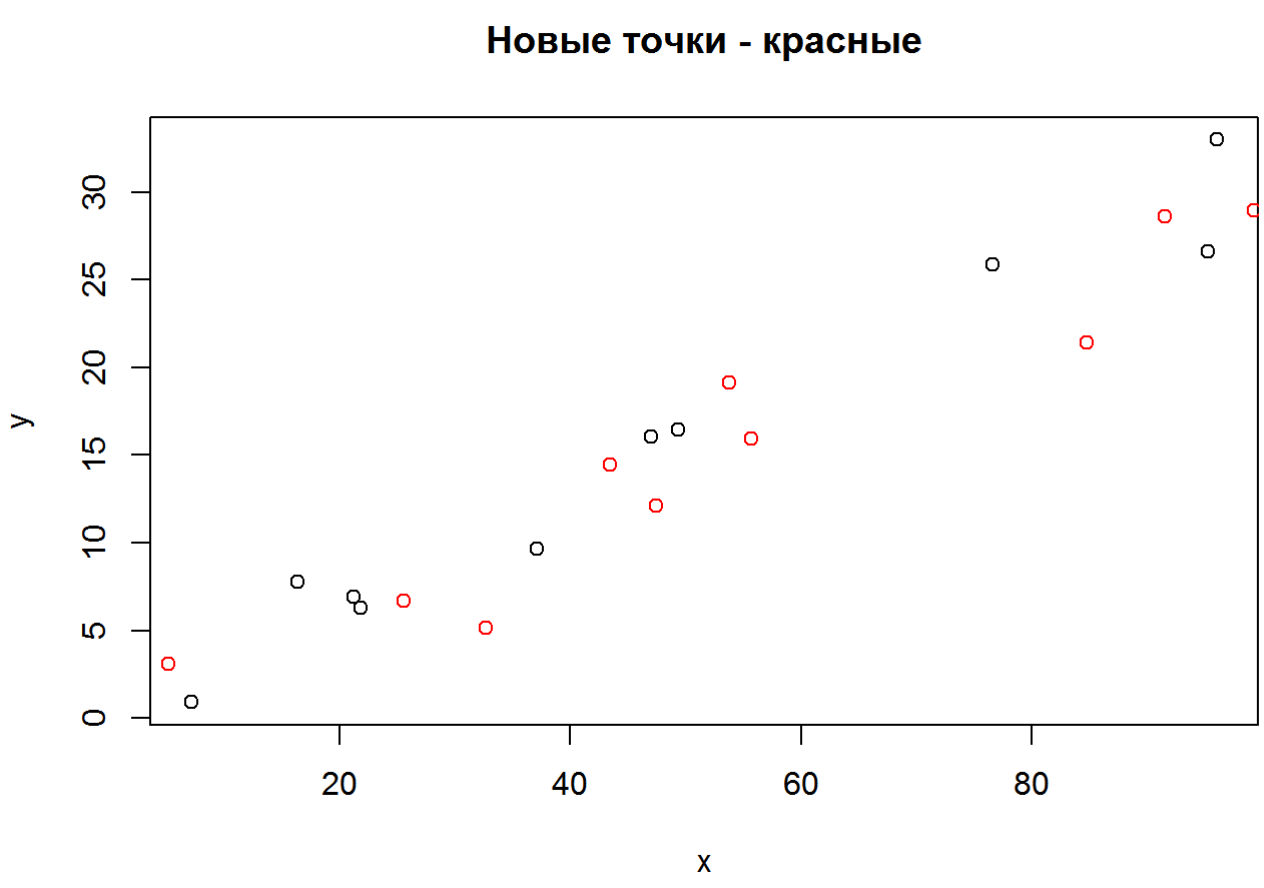
```
sum((predict(model2)-y)^2)/10
```

```
## [1] 0.008041885
```

Вторая модель лучше! Как ни крути.

Вот для этого и нужна тестовая выборка. Пусть у нас появились новые точки, которые мы не использовали в обучении.

```
x_test <- runif(10, min = 0, max = 100)
y_test <- 0.3*x_test + rnorm(10, sd = 2)
plot(x,y,main="Новые точки - красные")
points(x_test, y_test, col=2)
```



И сравним теперь ошибки на тестовой выборке:

```
sum((predict(model, newdata = data.frame(x = x_test)) - y_test)^2)/10
```

```
## [1] 8.796043
```

```
sum((predict(model2, newdata = data.frame(x = x_test)) - y_test)^2)/10
```

```
## [1] 400.4834
```

Кривая VaR

Поэтому когда мы найдем значение VaR, то верифицировать его мы должны будем на другой части наблюдений. У нас есть

```
length(ret)
```

```
## [1] 251
```

наблюдение. Давайте на первых 150-ти наблюдениях найдем VaR, а на второй его оценим.

Разбиваем на две выборки:

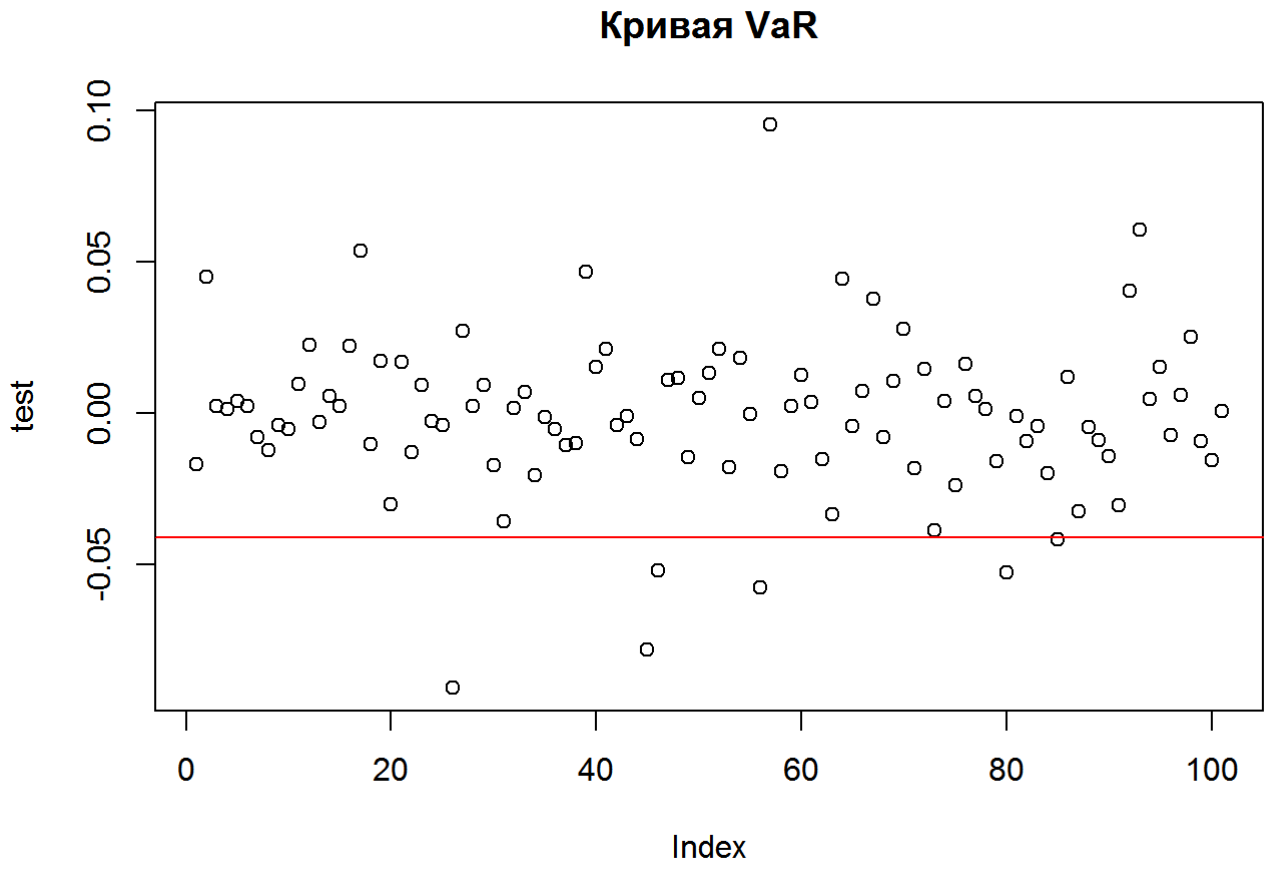
```
train <- ret[1:150]
test  <- ret[151:length(ret)]
```

Теперь найдем VaR как умеем на первой выборке:

```
var <- quantile(train, 0.05)
```

И нарисуем **кривую VaR**:

```
plot(test, type="p", main = "Кривая VaR")
abline(h = var, col="red")
```



Мы видим, что на деле 6 точек попали ниже кривой, а должны были:

```
0.05*length(test)
```

```
## [1] 5.05
```

Вопрос: можно ли считать прогноз точным?

Тест Купика

А вот и наш тест. Он проверяет совпадает ли число пробитий с ожидаемым. Пусть число пробитий $K = \sum(X_t < VaR_t)$, а доля пробитий $\alpha_0 = \frac{K}{L}$, где L - число наблюдений в тестовой выборке. Гипотеза теста: $H_0 : \alpha_0 = \alpha$. Статистика:
$S = 2\ln((1-\alpha_0)^{T-L} \cdot \alpha_0^L) - 2\ln((1-\alpha)^{T-L} \cdot \alpha^L) \sim \chi^2(1)$. Нет, знать статистику наизусть к экзамену не надо. Знать, как она распределена неплохо бы.

Теперь мы вооружены и можем провести тест сами:

```
L <- length(test)
K <- sum(test < var)
a0 <- K/L
a <- 0.05
S <- 2*log( (1-a0)^(L-K) * a0^K ) - 2*log( (1-a)^(L-K) * a^K )
pval <- 1 - pchisq(S, 1)
pval
```

```
## [1] 0.6731912
```

P-значение велико. А значит не отвергаем нулевую гипотезу. А значит VaR найден хорошо.

А почему все это плохо?

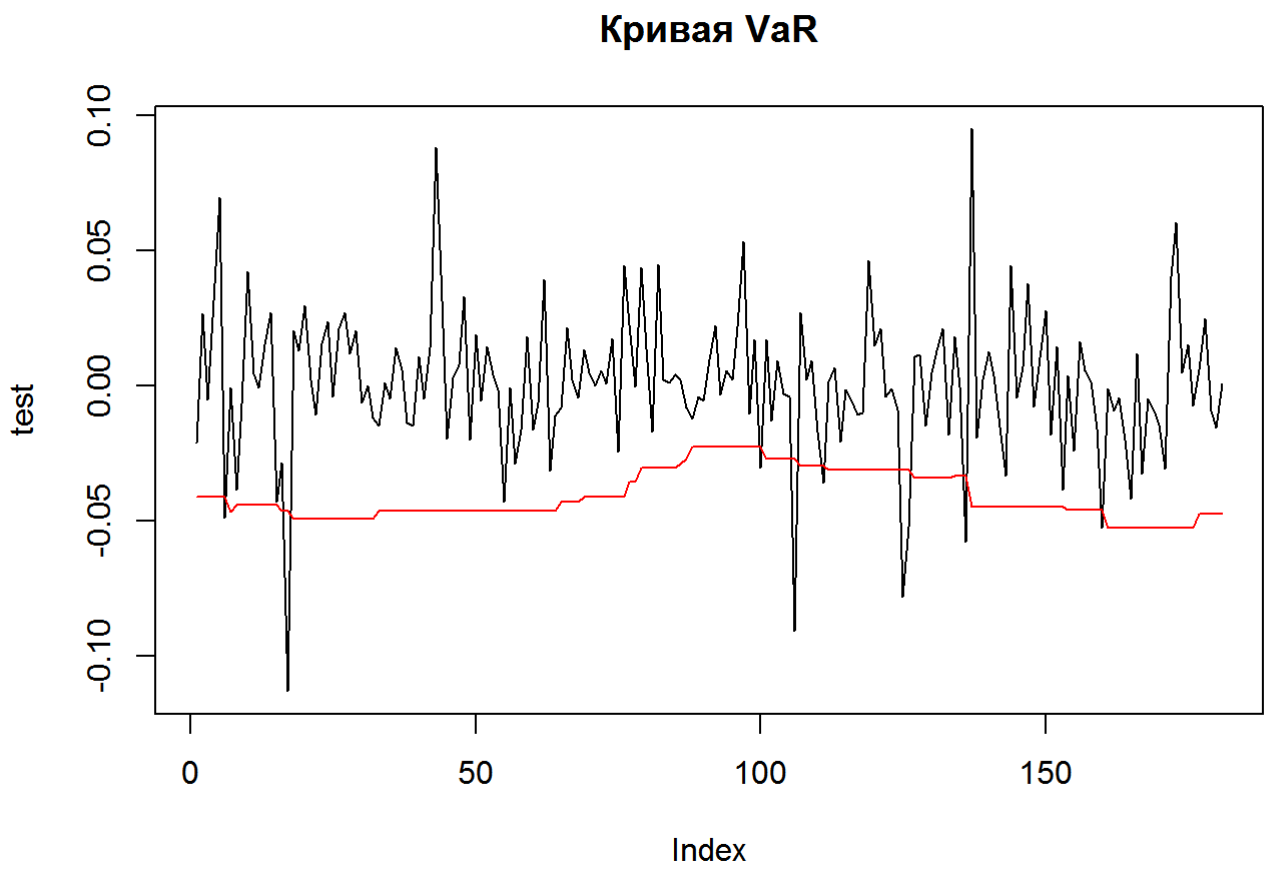
А потому, что мы не расширяем обучающую выборку, то есть не учитываем новую информацию. Мы один раз предсказали VaR и так с ним и сидим. Давайте сделаем хорошо.

Теперь мы выделим N точек, найдем по ним VaR для дня N+1. Потом сдвинем N точек вправо и найдем VaR по ним для точки N+2. И так далее. VaR будет в каждой точке разный. В коде будет понятнее:

```
N <- 70 # Длина тренировочной выборки. возьмем меньше, чтобы отличалось.
test <- ret[(N+1):length(ret)] # Тестовая выборка
VaR <- rep(0, length(test)) # Запишем сюда пока нули.
for (i in (N+1):length(ret)){
  train <- ret[(i-N):(i-1)]
  VaR[i-N] <- quantile(train, 0.05)
}
```

А теперь нарисуем:

```
plot(test, type="l", main = "Кривая VaR")
lines(VaR, col="red")
```



Что нужно сделать теперь? Верно, провести тест Купика. Не хочу загромождать текст, скажу, что получается p = 0.98. Лучше чем было раньше. *Ожидаемо? Поячему?*

Часть третья, в которой читатель узнает про то, зачем была нужна первая лекция

То, что мы делали до этого момента, называется историческим способом оценки VaR. Нужен квантиль 5%? Вот и бери 5% худших значений. Работает? да. Но как-то неинтересно. Давайте теперь по-настоящему моделировать. Например, я решу, что мое распределение доходностей нормальное. С первой лекции Вы должны уметь проверить так ли это на самом деле. Но я проверять не буду, так как я плохой исследователь и не проверяю предпосылки (только сегодня :)).

Тогда мне нужно искать VaR в каждой точке чуть-чуть по другому: взять последние N точек, подогнать под них нормальное распределение и найти его квантиль. Подогнать нормальное распределение просто: нужно взять распределение с таким же средним и дисперсией. Утверждается, что оно максимально хорошо описывает точки. С чего бы это?

Сначала сравним VaR новым и старым методом на всех данных:

```
quantile(ret, 0.05)
```

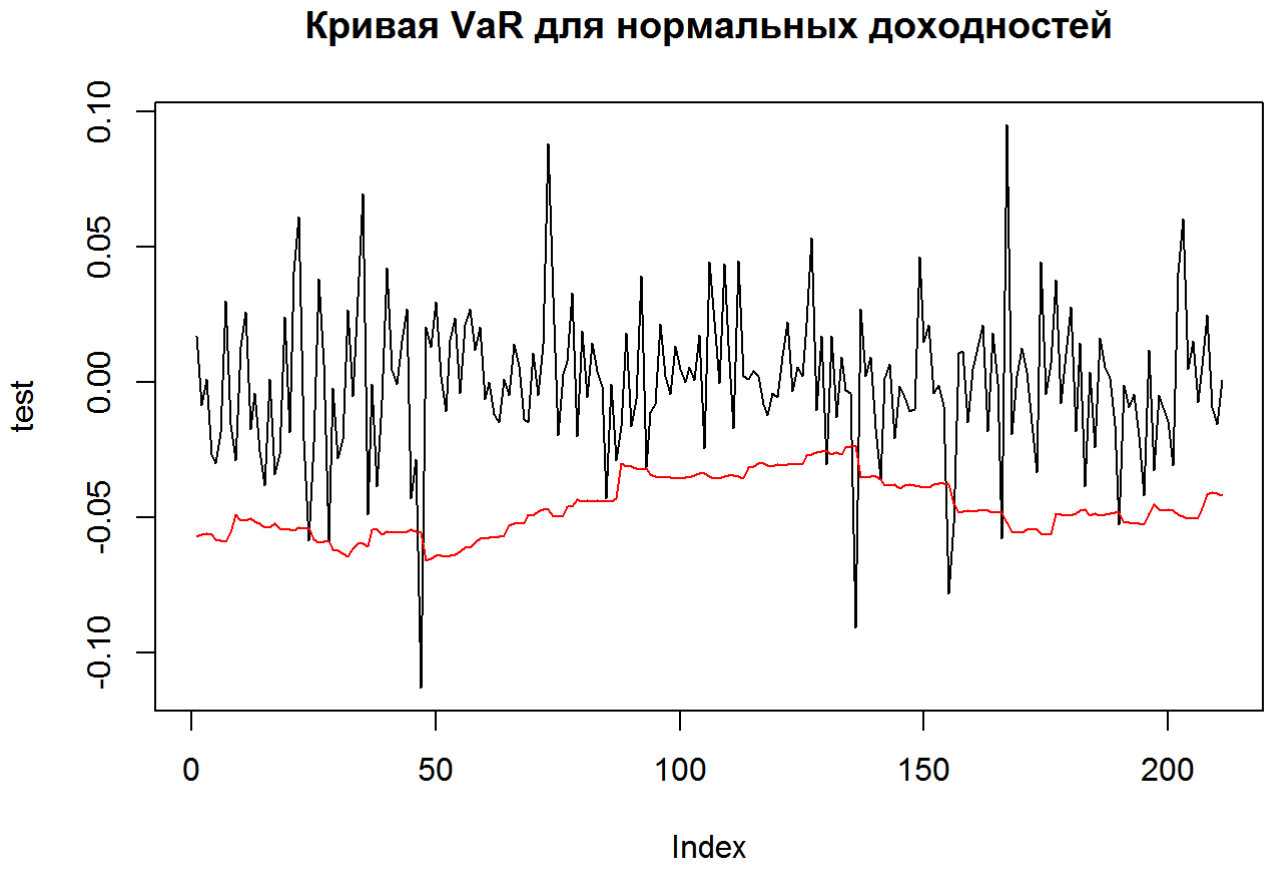
```
##          5%
## -0.04236016
```

```
qnorm(0.05, mean=mean(ret), sd=sd(ret))
```

```
## [1] -0.04814024
```

Похоже? Нет? Теперь строим кривую VaR:

```
N <- 40
test <- ret[(N+1):length(ret)]
VaR <- rep(0, length(test))
for (i in (N+1):length(ret)){
  train <- ret[(i-N):(i-1)]
  VaR[i-N] <- qnorm(0.05, mean=mean(train), sd=sd(train))
}
plot(test, type="l", main = "Кривая VaR для нормальных доходностей")
lines(VaR, col="red")
```



И тест Купика!

```
L <- length(test)
K <- sum(test < VaR)
a0 <- K/L
a <- 0.05
S <- 2*log( (1-a0)^(L-K) * a0^K ) - 2*log( (1-a)^(L-K) * a^K )
pval <- 1 - pchisq(S, 1)
pval
```

```
## [1] 0.8609284
```

Чем метод лучше исторического? А чем хуже? Дополнительный вопрос для тех, что читал “Черного Лебедя”: *Что вообще про все это думает Талеб?*

Часть четвертая, где мы боремся с ограми

Это огр:

А это **ОГР**:

$$f(x) = \frac{(\gamma/\delta)^\lambda}{\sqrt{2\pi}K_\lambda(\delta\gamma)}\ e^{\beta(x-\mu)} \times \frac{K_{\lambda-1/2}\left(\alpha\sqrt{\delta^2+(x-\mu)^2}\right)}{\left(\sqrt{\delta^2+(x-\mu)^2}/\alpha\right)^{1/2-\lambda}}$$

где

$$K_a(z) = \frac{\pi}{2\sin(a\pi)}[I_{-a}(z) - I_a(z)]$$

где

$$I_a(z) = \sum_{k=0}^{\infty} \frac{\left(\frac{z}{2}\right)^{2k+a}}{k!\Gamma(k+a+1)}$$

где

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$$

Это формула плотности вероятности Обобщенного Гиперболического Распределения.

Какая была проблема с гауссианой? Хвосты плохо описывает да симметрию не поддерживает. А вот ОГР умеет почти все. У него 5 параметров: $\lambda, \alpha, \beta, \delta, \mu$ и $\gamma = \sqrt{\alpha^2 - \beta^2}$ для простоты формулы (хаха).

Собственно Гаусс, Стьюдент, обратный Гаусс, variance-gamma, обратное Хи-квадрат и многое другое – это все частные случаи ОГР. И раз мы не смогли хорошо подогнать гауссиану, то ОГР-то точно сможем!

За работу с этим огромным классом распределений в R отвечает пакет `ghyp`. Его нужно установить или подключить.

```
#install.packages("ghyp")
library(ghyp)
```

В нем есть много хороших функций. Функции для подгонки: `fit.*uv` – для подгонки одномерных распределений, `fit.*mv` – для подгонки многомерных распределений. Где * – это `gauss`, `ghyp`, `hyp`, `NIG`, `t` и `VG` для разных распределений.

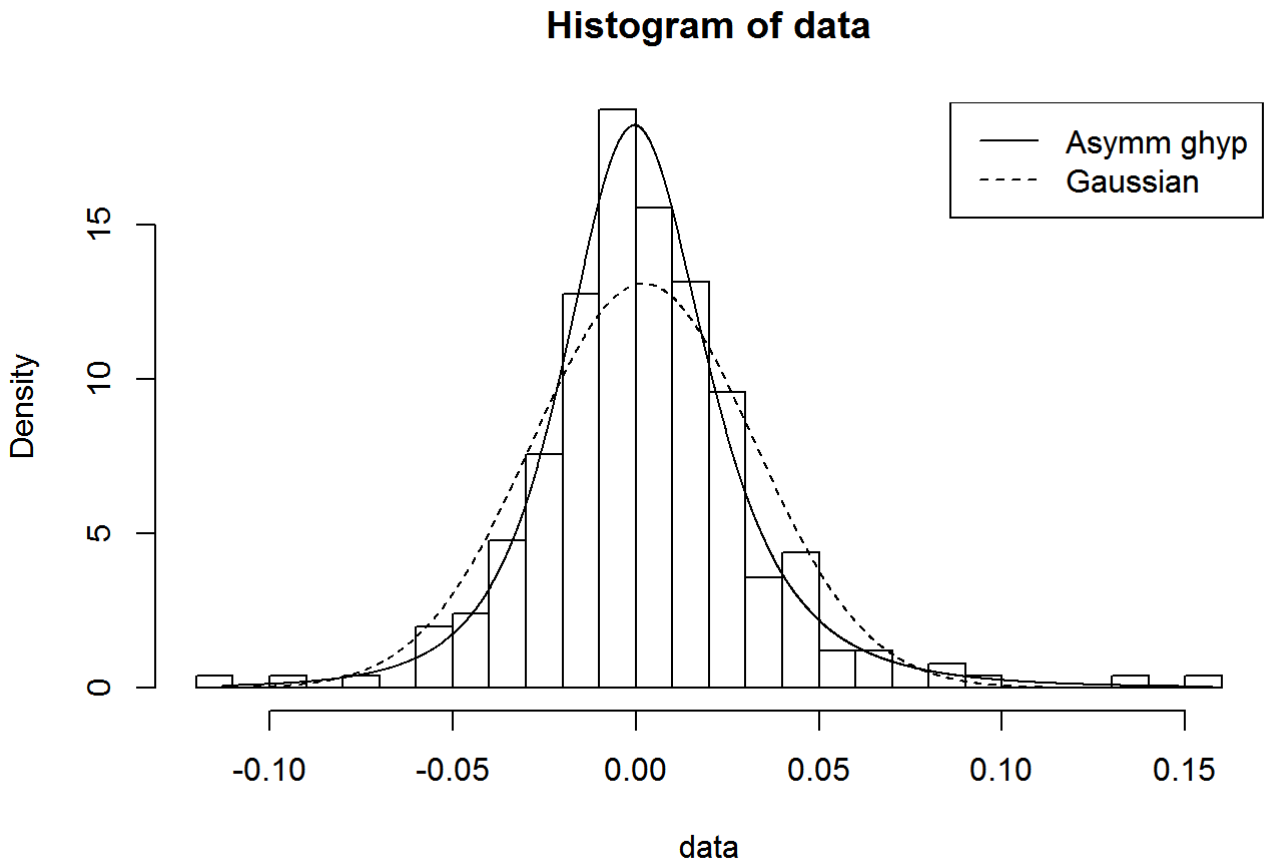
Так как мы очень крутые, то подгоним сразу самое общее распределение под наши данные:

```
ghyp_dist <- fit.ghypuv(ret, silent = TRUE)
summary(ghyp_dist)
```

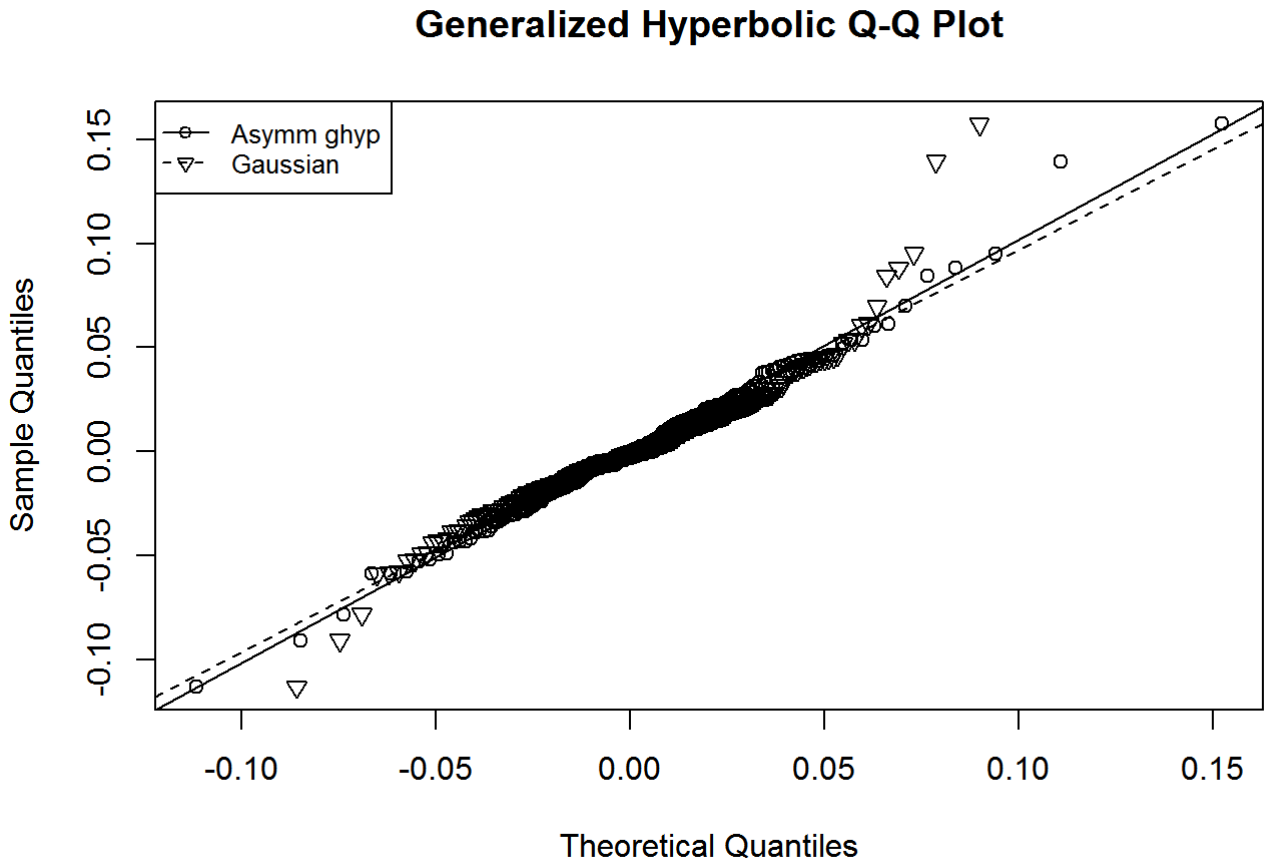
```
## Warning: fitting procedure did not converge!
##
## Asymmetric Generalized Hyperbolic Distribution:
##
## Parameters:
##       lambda    alpha.bar           mu        sigma        gamma
## -1.405501496  0.483864256 -0.001719425  0.030065296  0.003723467
##
## Call:
## fit.ghypuv(data = ret, silent = TRUE)
##
## Optimization information:
## log-Likelihood:                543.1607
## AIC:                           -1076.321
## Fitted parameters:             lambda, alpha.bar, mu, sigma, gamma;  (Number: 5)
## Number of iterations:          502
## Converged:                     FALSE
## Error code:                    1
## Error message:
```

Что-то оно не сошлось. Но это не значит, что параметры не определены. Смотрим на результат:

```
hist(ghyp_dist)
```




```
qqghyp(ghyp_dist)
```



Но может ОГР это слишком? И Стьюдент был бы лучше? Или NIG? Для этого есть информационный критерий Акаике. $AIC = 2k - 2\ln(L)$, где k - количество параметров, а L - правдоподобие.

```
aic <- stepAIC.ghyp(ret, dist=c("ghyp", "hyp", "t", "gauss"), silent=TRUE)
```

```
## Currently fitting: asymmetric ghyp
## Currently fitting: asymmetric hyp
## Currently fitting: asymmetric t
## Currently fitting: symmetric ghyp
## Currently fitting: symmetric hyp
## Currently fitting: symmetric t
## Currently fitting: gauss
```

```
aic$best.model
```

```
## Symmetric Student-t Distribution:
##
## Parameters:
##           nu           mu        sigma        gamma
## 3.5001763833 0.0007733997 0.0314808581 0.0000000000
##
## log-likelihood:
## 542.4663
##
##
## Call:
## stepAIC.ghyp(data = ret, dist = c("ghyp", "hyp", "t", "gauss"),     silent = TRUE)
```

Используем ОГР

Но как тогда найти VaR? В пакете `ghyp` есть готовые функции для поиска квантилей. На вход они принимают вероятность и меодель.

```
qghyp(0.05, object = aic$best.model)
```

```
## [1] -0.04502962
```

Построим кривую? Только теперь в каждой точке нам нужно: а) переопределять лучшее распределение. б) Находить его квантиль:

Внимание! Этот цикл на Intel Core i7 выполнялся около минуты. А еще генерируется куча текста, поэтому в этой презентации этот и следующий блшок просто выключены.

```
N <- 100
test <- ret[(N+1):length(ret)]
VaR <- rep(0, length(test))
for (i in (N+1):length(ret)){
  train <- ret[(i-N):(i-1)]
  model <- stepAIC.ghyp(train, dist=c("ghyp", "hyp", "t", "gauss"), silent=T)$best.mo
del
  VaR[i-N] <- qghyp(0.05, object = model)
}
plot(test, type="l", main = "Кривая VaR для нормальных доходностей")
lines(VaR, col="red")
```

Надеюсь, что у вас все получилось красиво. Сразу к тесту.

```
L <- length(test)
K <- sum(test < VaR)
a0 <- K/L
a <- 0.05
S <- 2*log( (1-a0)^(L-K) * a0^K ) - 2*log( (1-a)^(L-K) * a^K )
pval <- 1 - pchisq(S, 1)
pval
```

Ваши выводы?

Часть пятая, последняя, где мы отправляемся в Монте-Карло

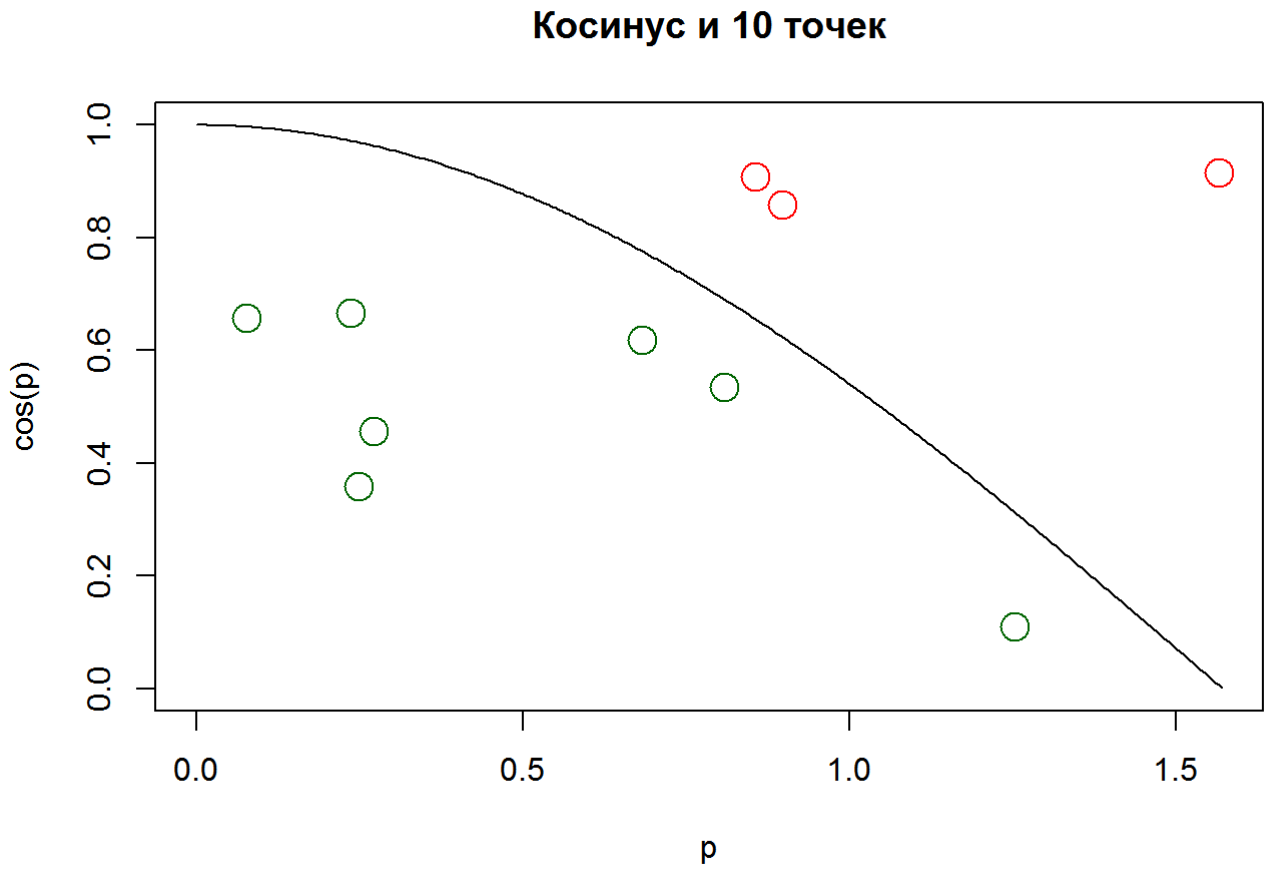
Мóнте-Ка́рло (монегасский Monte-Carlu, фр. Monte-Carlo) — административная территория княжества Монако, крупнейший район страны, расположенный на территории одноименной коммуны в Монако. Город известен своими казино, пляжами и пользуется популярностью у представителей высшего общества. В Монте-Карло начинается и завершается ежегодное ралли «Монте-Карло».

Так говорит Википедия. Но мы будем говорить про метод работы со случайными числами, которые раньше генерировали рулеткой, коих много в Монте-Карло.

Метод Монте-Карло для поиска площади фигуры

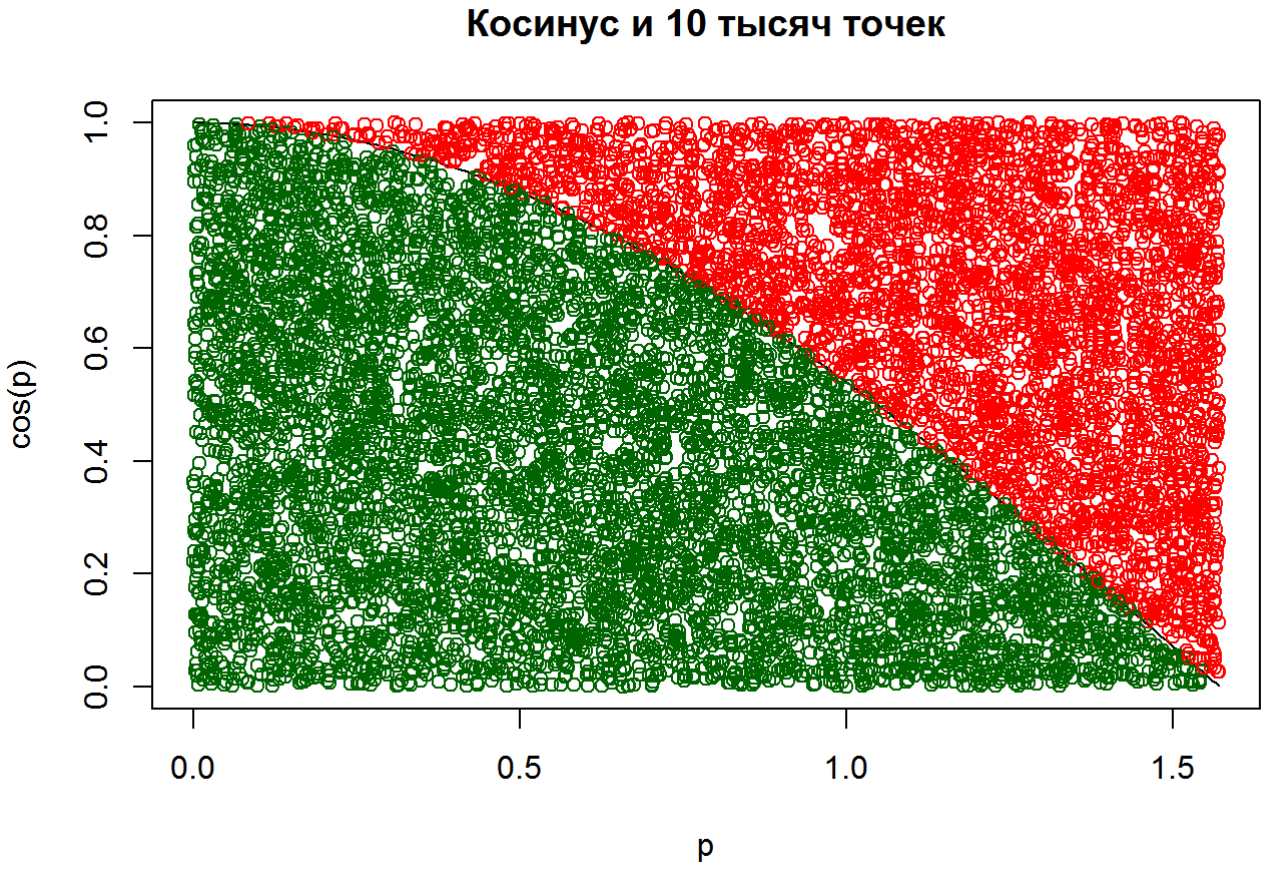
Как найти площадь под графиком $\cos(x)$ от 0 до $\pi/2$. Можно вспомнить, что это $sin(\pi/2) - sin(0) = 1$. Но допустим, все лекции по интегралам вы прогуляли. Тогда можно поступить так: Впишем график в прямоугольник $(0,0) : (\pi/2, 1)$, он точно весь туда помещается. И теперь сгенерируем пару точек в этом прямоугольнике

```
p <- seq(0,pi/2, by=0.005)
plot(p, cos(p), type="l", main = "Косинус и 10 точек") # график
x <- runif(10, min=0, max=pi/2)
y <- runif(10, min=0, max=1)
points(x, y, col=ifelse(y > cos(x),"red","darkgreen"), cex=2)
```



Площадь всего прямоугольника = $1 * \pi/2 = \pi/2$. Заметим, что только часть точек попала под кривую. Утверждается, что доля точек, лежащих под кривой, умноженная на площадь прямоугольника стремится к истинной площади под кривой. Продемонстрирую свою мысль:

```
plot(p, cos(p), type="l", main = "Косинус и 10 тысяч точек")
x <- runif(10^4, min=0, max=pi/2)
y <- runif(10^4, min=0, max=1)
points(x, y, col=ifelse(y > cos(x),"red","darkgreen"))
```




```
sum(y<cos(x))/10^4 * pi/2
```

```
## [1] 0.9952566
```

А зачем это все?

А затем, что мы умеем находить VaR для ОГР, но не ES. Еще раз VaR:

```
qghyp(0.05, object = aic$best.model)
```

```
## [1] -0.04502962
```

А теперь ES:

```
N = 10^6 # Количество точек для Монте-Карло.
mean(sort(rghyp(N, object = aic$best.model))[1:N*0.05])
```

```
## [1] -0.07052255
```

Если понятно, почему ES считается именно так, то круто! Если нет, но вы читаете это на лекции, то сейчас я все объясню. Если не понятно, а лекцию вы пропустили, то очень жаль :(

На этом на сегодня все. Теперь вы все умеете.

Домашнее задание

1. Кто еще не сделал первую работу, то самое время за нее сесть. Дедлайн - 23:59, 3 октября, суббота.
2. ДЗ по этой лекции: взять котировки на свой вкус. Точек не менее 300. Лучше более 500. Периодичность дневная. Построить VaR историческим методом и через подгонку ОГР. Нужны красивые картинки, результаты тестов Купика, ваши выводы и исходные коды. Дедлайн - 23:59, 3 октября, суббота.
3. Если кто-то хочет дополнительных баллов и/или отличных знаний, то курс Try R (http://tryr.codeschool.com) все еще ждет вас.

Те, кто пропустят дедлайн будут оштрафованы. Опоздать можно на неделю и получить штраф в 50% от максимума. Работы, присланные через неделю после дедлайна не засчитываются. В конце курса две самые плохие работы можно будет пересдать.

Отправлять каждое ДЗ можно неограниченное число раз, ибо нет предела совершенству.

Напоследок, книга для тех, кто желает изучить R серьезно: R в действии. Анализ и визуализация данных на языке R (http://dmkpress.com/catalog/computer/programming/978-5-94074-912-7/). Там 600 страниц, куча статистики и советов по визуализации. Очень подойдет будущим статистикам, эконометристам и всякого рода аналитикам. Ссылка на сайт издательства, pdf легко найти в сети.

P.S. Вот вам еще фото Монте-Карло.