Patrick Hénaff

Topics in Empirical Finance

with R and Rmetrics

v 2.0

An Open Access Publication

Contents

Preface to the second edition

Ten years after the first edition, it is time to revisit this text, not only to update the data sets, but also the add material that didn't make it the first time around.

So much has taken place since the 2011. Publishing open source material has become dramatically simpler with the development of the `bookdown` package and the possibility of generating an online version of the text.

Alas, there was also tragedy. In 2016 Diethelm Wurtz and his wife Barbara were involved in a fatal accident. Diethelm was the founder of the Rmetrics association, the author of the Rmetrics suite of packages, and the organizer of the R/Rmetrics Summer Schools and Workshops. We hold fond memories of these wonderful workshops organized in Meielisalp, and this second edition is dedicated to the memory of Barbara and Diethelm.

Preface to the first edition

Une totale soumission aux données de
l'expérience
est la règle d'or qui domine toute discipline,
toute activité valable.
—Maurice Allais

This textbook is about empirical finance, and focusses on the pricing and risk management of financial assets: bonds, futures contracts, and other derivative securities.

The emphasis of this text is empirical. We present models, and verify their relevance by testing them of real data. We emphasize:

- an incremental approach to model building, starting from simple models, and building upon that foundation to construct more complex models, as needed,
- a data-driven approach: we will implement all the models that are presented, using the R statistical package and the Rmetrics libraries,
- the systematic use of simulation as a way of validating modeling decisions.

Last but not least, a particular attention is given to model estimation, in order to measure the tradeoff between model complexity and the challenges of a robust calibration.

This course would not be possible without the R statistical program and without the Rmetrics packages. We extend our deep appreciation to the R community and to Diethelm Wuertz and the Rmetrics team.

This book is open access (free as in free beer). It's also open source: feel free to clone and submit additions. You can download a PDF copy

Computational Framework

1 Introduction

This first part is dedicated to the description of the experimental environment used in this book.

As mentioned earlier, our approach is data-driven and empirical. It is thus appropriate to start this text with a description of the sources of data that we will use, and how to fetch publicly available financial data from the Internet.

In the following chapters, we will make repeated use of the Rmetrics pricing libraries, and often compare different models applied to the same data. To facilitate this comparison, the Rmetrics pricing libraries have been wrapped in a object-oriented framework, which hides most of the implementation details, and allows us to focus on the key features of the financial instruments and models. The framework uses the S4 object model (Genolini 2008) of the R language, and is described in Chapter 3.

Simulation is our tool of choice to explore the features of pricing models and test their robustness. To that end, we have developped a framework that facilitates the definition and testing of simple risk management strategies. The core of this simulation framework is the `DataProvider` class, which is presented in Chapter 4.

In addition to the packages found on CRAN, data sets and code used in the text have been gathered into three packages:

empfin. contains all the data sets and several utility functions for simple bond pricing and the manipulation of dates,

fInstrument. provides the `fInstrument` class, that wraps the Rmetrics pricing library and the `DataProvider` class, that acts as a container for market data,

DynamicSimulation. contains tools needed to perform dynamic simulations, such as scenario generators and delta hedging simulators.

2 Data Sources

```
> library(RCurl)
> library(data.table)
> library(fasttime)
> library(fImport)
> library(timeSeries)
> library(empfin)
> library(stringr)
> library(lubridate)
> library(quantmod)
> data(LiborRates)
```

This first chapter describes the data sources and the processing that has been performed to create the data sets found in the package. In the following chapters, we will use data from three asset classes: commodities, equities and fixed income.

2.1 Commodities Data

The CME group (www.cmegroup.com) provides a large quantity of data on the products listed on its exchanges. Commodity prices are particularly interesting in empirical finance because they exhibit high volatility, seasonality, and many other specific features.

The following function downloads settlement data on commodity options and futures traded on the NYMEX.

```
getNymexData <- function(type) {

    url <- "ftp://ftp.cmegroup.com/pub/settle/"
    filename <- paste("nymex_", type, ".csv",
        sep = "")
    fn <- paste(url, filename, sep = "")

    curl <- getCurlHandle(ftp.use.epsv = FALSE)
    x <- getURL(fn, curl = curl)
```

```
    # csv <- strsplit(x, '\n')[[1]]

    con <- textConnection(csv)
    tmp <- data.frame(read.csv(con, header = TRUE))
    close(con)

    if (type == "future") {
        goodColumns <- c("PRODUCT.SYMBOL", "CONTRACT.MONTH",
            "CONTRACT.YEAR", "CONTRACT", "PRODUCT.DESCRIPTION",
            "SETTLE", "EST..VOL", "TRADEDATE")

        goodNames <- c("Symbol", "Month", "Year",
            "Contract", "Description", "Settle",
            "Volume", "TradeDate")

    }

    if (type == "option") {
        goodColumns <- c("PRODUCT.SYMBOL", "CONTRACT.MONTH",
            "CONTRACT.YEAR", "PUT.CALL", "STRIKE",
            "SETTLE", "EST..VOL", "TRADEDATE")

        goodNames <- c("Symbol", "Month", "Year",
            "CP", "Strike", "Settle", "Volume",
            "TradeDate")
    }
    tmp <- tmp[, goodColumns]
    colnames(tmp) <- goodNames
    tmp
}
```

As an illustration, the following script downloads the NYMEX futures data and extracts the Crude Oil (WTI) futures settlement prices. Running the code on January 9, 2019, yields the following result:

```
nymex.futures <- getNymexData('future')
tmp <- nymex.futures[nymex.futures$Symbol == 'CL',]
head(tmp[,c('Month', 'Year', 'Contract', 'Settle')])
```

```
      Month Year Contract Settle
4645     2 2019    CLG19  49.78
4646     3 2019    CLH19  50.11
4647     4 2019    CLJ19  50.49
4648     5 2019    CLK19  50.93
```

```
4649     6 2019    CLM19  51.36
4650     7 2019    CLN19  51.74
```

For the record, the same function executed on October 11, 2012, yielded the following data set:

```
head(tmp[,c('Month', 'Year', 'Contract', 'Settle')])
```

```
  Month Year Contract Settle
1998    11 2012    CLX12  91.25
1999    12 2012    CLZ12  91.64
2000     1 2013    CLF13  92.11
2001     2 2013    CLG13  92.58
2002     3 2013    CLH13  93.00
2003     4 2013    CLJ13  93.30
```

The U.S. Energy Information Administration (www.eia.gov) publishes historical data on spot and future prices. The data set in package `empfin` was obtained from that source.

2.2 *Libor Rates*

The package `empfin` also contains a data set of USD libor and swap rates obtained from the US Federal Reserve archive (www.federalreserve.gov), which provides a data download program for table H15 at `https://www.federalreserve.gov/datadownload/Choose.aspx?rel=H15`.

The time series can be downloaded from a web browser, but can also be downloaded programmatically. The site provides directions on how to construct the URL corresponding to each particular data set.

In the example below, the URL is specific to the H15 table, with all available deposit and swap rates included.

```
get.frb.url <- function(dtStart, dtEnd) {

    # Federal Reserve Board URL Construct
    # this URL at
    # 'http://www.federalreserve.gov/datadownload

    url.1 <- paste("https://www.federalreserve.gov/datadownload/Output.aspx?",
        "rel=H15&series=0b30e7214e1df1ba738a199d707702f1",
        "&lastobs=", sep = "")
    url.2 <- "&filetype=csv&label=include&layout=seriescolumn"
    url.2 <- "&filetype=csv&label=include&layout=seriescolumn"
    url <- paste(url.1, "&from=", as.character(dtStart,
        format = "%m/%d/%Y"), "&to=", as.character(dtEnd,
```

```
        format = "%m/%d/%Y"), url.2, sep = "")
    url
}
```

The data is downloaded from the FRB web site and converted into a object, with more explicit names. The resulting time series has daily observations and 11 columns, with deposit rates of maturities 1, 3, and 6 months, and swap rates with maturities ranging from 1 to 30 years.

```
getLiborRates <- function(dtStart, dtEnd) {

    columns.dic = hash(keys = c("RIFLDIY01_N.B",
        "RIFLDIY02_N.B", "RIFLDIY03_N.B", "RIFLDIY04_N.B",
        "RIFLDIY05_N.B", "RIFLDIY07_N.B", "RIFLDIY10_N.B",
        "RIFLDIY30_N.B", "RILSPDEPM01_N.B", "RILSPDEPM03_N.B",
        "RILSPDEPM06_N.B"), values = c("Swap1Y",
        "Swap2Y", "Swap3Y", "Swap4Y", "Swap5Y",
        "Swap7Y", "Swap10Y", "Swap30Y", "Libor1M",
        "Libor3M", "Libor6M"))

    # non-available data is marked 'ND' or
    # 'NC' in the data set
    good.row <- function(x) length(grep("NC|ND",
        x)) == 0

    url <- get.frb.url(dtStart, dtEnd)

    curl <- getCurlHandle(ftp.use.epsv = FALSE)
    x <- getURL(url, curl = curl)

    csv <- strsplit(x, "\n")[[1]]

    # skip 5 header rows at start of file
    con <- textConnection(csv[6:length(csv)])
    # colClasses='character' needed to avoid
    # conversion to factors
    csv <- read.csv(con, colClasses = "character",
        header = TRUE)
    close(con)

    ncol <- dim(csv)[2]
    indx <- apply(csv, 1, good.row)

    dt <- as.Date(csv$Time.Period[indx])
    vx <- data.frame(csv[indx, 2:ncol])
```

```
    for (i in seq(ncol - 1)) vx[, i] = as.numeric(vx[,
        i])

    colNames = unlist(sapply(names(vx), function(x) columns.dic[[x]]),
        use.names = FALSE)
    timeSeries(vx, dt, format = "%Y-%m-%d", zone = "GMT",
        FinCenter = "GMT", units = colNames)
}
```

Executing this function on January 9, 2019 yields the following result.

```
> dtStart <- myDate('01Jan2010')
> dtEnd <- myDate('01Jan2011')
> ts = getLiborRates(dtStart, dtEnd)
> head(ts[,c('Swap1Y', 'Swap10Y', 'Swap30Y')])
GMT
           Swap1Y Swap10Y Swap30Y
2010-01-04   0.65    3.93    4.51
2010-01-05   0.61    3.87    4.46
2010-01-06   0.60    3.92    4.52
2010-01-07   0.57    3.92    4.53
2010-01-08   0.55    3.94    4.58
2010-01-11   0.53    3.91    4.56
```

The dataset in package has been constructed with this function, and provides daily rates from July 2000 to October 2016.

As of this writing, current and historical Libor rates are available from the Wall Street Journal Market Data Center (www.wsj.com). Swap rates are available, for example, from Skandinaviska Enskilda Banken (www.sebgroup.com).

2.3 *Sovereign Yield Curves*

The European Central Bank publishes fitted curves of euro area AAA sovereign yield. The data can be downloaded as a .csv file from this institution's statistical data warehouse (sdw.ecb.europa.eu). The following script read, parse and display the par AAA euro government yield curve for a set of dates.

```
# read, keep only par yield data
ecb.file <- fread("./data/ecb-data.csv", select = c("DATA_TYPE_FM",
    "TIME_PERIOD", "OBS_VALUE"))[substr(DATA_TYPE_FM,
    1, 2) == "PY"]
```

```
ecb.file$TIME_PERIOD <- as.Date(ecb.file$TIME_PERIOD)

# convert to rectangular table, one column
# per maturity
tmp <- dcast(ecb.file, TIME_PERIOD ~ DATA_TYPE_FM,
    value.var = "OBS_VALUE")
ecb.series <- tmp[, -1]
# row names are characters
row.names(ecb.series) <- tmp$TIME_PERIOD

# decode maturity from column names
tokens <- str_match(names(ecb.series), "^PY_(([0-9]+)Y)?(([0-9]+)M)?")

# maturity expressed in months
mat <- apply(tokens, 1, function(x) {
    tmp <- as.numeric(x[c(3, 5)])
    tmp[is.na(tmp)] <- 0
    12 * tmp[1] + tmp[2]
})

# rearrange columns by maturity
sort.mat <- sort(mat, index.return = TRUE)
setcolorder(ecb.series, neworder = sort.mat$ix)
```

Plotting a few curves let us measure the sharp increase in interest rate during the year 2022.

```
nbObs <- dim(ecb.series)[1]
indx <- seq(1, nbObs, 100)

YC <- ecb.series[indx, ]
dtObs <- as.Date(row.names(ecb.series)[indx])
nbSample <- length(indx)

dtMat = dtObs[1] + months(sort.mat$x)
plot(dtMat, YC[1, ], type = "l", ylim = c(-1,
    4), xlim = c(dtObs[1], dtObs[nbSample] + months(last(sort.mat$x))),
    bty = "n", col = 1, lwd = 2, ylab = "Euro AAA Govt Yield",
    xlab = "Maturity date")
text.legend = format(dtObs[1], "%D")
abline(h = 0, lty = 2, lwd = 1)
for (i in seq(2, nbSample)) {
    dtMat = dtObs[i] + months(sort.mat$x)
    lines(dtMat, YC[i, ], type = "l", col = i,
        lwd = 2)
```

```
    text.legend <- append(text.legend, list(format(dtObs[i],
        "%D")))
}

legend(x = "topleft", legend = text.legend, lty = 1,
    col = seq(nbSample), cex = 1, box.lty = 0)
```

2.4 *Equities Data*

Time series of stock prices are readily available from free on-line services. The provides a simple function to fetch Open/High/Low/Close/Volume series from Yahoo Finance as an xts time series:

```
getSymbols("F", src = "yahoo")
```

```
## [1] "F"
```

and a good-looking bar chart is obtained by invoking

```
barChart(F)
```

For illustrative purpose, the package provides one time series of daily prices for three stocks: IBM, Coca-Cola (KO) and Trans-Ocean (RIG), and one Exchange Traded Fund: the Nasdaq 100 Trust (QQQQ).

3 *Basic Components*

```
library(fOptions)
library(fExoticOptions)
library(fInstrument)
library(DynamicSimulation)
library(empfin)
library(plotly)
```

This chapter provides a tutorial and explains the design of the object framework that has been build on top of the Rmetrics library. As mentioned earlier, this layer is meant to hide most of the implementation details, and allows us to focus on the key features of the financial instruments and models.

The reader is encouraged to work through the examples, but the sections on design and implementation can be skipped.

The object framework involves two main entities:

- the `fInstrument` class models an abstract financial instrument, and exposes generic methods for computing the net present value (NPV) and the risk indicators (the "greek"). With this class, one can perform calculations on portfolio of instruments, without being concerned about the implementation details specific to each type of asset. This will be illustrated below with the detailed description of a delta-hedging algorithm.
- the `DataProvider` class is a container for market data, derived from the built-in `R environment`. It greatly simplifies the signature of pricing functions. Instead of passing as arguments all the necessary market data, we simply pass one `DataProvider` argument. The pricing methods fetch the necessary data from the `DataProvider`, as needed.

Each entity is now described and illustrated.

3.1 *The `fInstrument` Class*

As mentioned, the purpose of the `fInstrument` class is to create a layer of abstraction over the large variety of pricing models found in Rmetrics. With this class, we can express calculation algorithms in a generic manner. This is best explained by an example.

Consider a portfolio made of two options, a vanilla European call and a binary (cash-or-nothing) option, both written on the same underlying asset. We would like to compute the NPV and delta of this portfolio. Let's contrast the process, first performed with the Rmetrics functions, and then with the `fInstrument`.

Starting with the Rmetrics functions, you first compute the price and delta of the European call:

```
p <- vector(mode = "numeric", length = 2)
d <- vector(mode = "numeric", length = 2)

p <- vector(mode = "numeric", length = 2)
cp <- "c"
Spot <- 100
Strike <- 100
Ttm <- 1
int.rate <- 0.02
div.yield <- 0.02
sigma <- 0.3
p[1] <- GBSOption(TypeFlag = cp, S = Spot, X = Strike,
    Time = Ttm, r = int.rate, b = int.rate - div.yield,
    sigma = sigma)@price
d[1] <- GBSGreeks(Selection = "delta", TypeFlag = cp,
    S = Spot, X = Strike, Time = Ttm, r = int.rate,
    b = int.rate - div.yield, sigma = sigma)
```

Perform the same calculation for the binary option. The delta is computed by finite difference.

```
p <- vector(mode = "numeric", length = 2)
K <- 1
p[2] <- CashOrNothingOption(TypeFlag = cp, S = Spot,
    X = Strike, K = K, Time = Ttm, r = int.rate, b = int.rate -
        div.yield, sigma = sigma)@price
h <- Spot * 0.001
dh <- CashOrNothingOption(TypeFlag = cp, S = c(Spot +
    h, Spot - h), X = Strike, K = K, Time = Ttm, r = int.rate,
    b = int.rate - div.yield, sigma = sigma)@price
d[2] <- diff(dh)/(2 * h)
```

Finally, sum both vectors to get the portfolio NPV and delta.

```
p <- vector(mode = "numeric", length = 2)
print(paste("Price:", round(sum(p), 2), "Delta:", round(sum(d),
    3)))
```

```
## [1] "Price: 0 Delta: 0.536"
```

With the `fInstrument` class, the calculation steps are quite different.

You first create a vanilla instrument with the `fInstrumentFactory` function:

```
dtExpiry <- myDate("01jan2011")
underlying <- "IBM"
Strike <- 100
K <- 1

b <- fInstrumentFactory("vanilla", quantity = 1,
    params = list(cp = "c", strike = Strike, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))
```

Next, use again the `fInstrumentfactory` to create the binary option:

```
v <- fInstrumentFactory("binary", quantity = 1,
    params = list(cp = "c", strike = Strike, dtExpiry = dtExpiry,
        K = K, underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))
```

Insert the relevant market data into a `DataProvider` (this will be explained in the next section):

```
base.env <- DataProvider()
dtCalc <- myDate("01jan2010")
setData(base.env, underlying, "Price", dtCalc,
    100)
setData(base.env, underlying, "DivYield", dtCalc,
    div.yield)
setData(base.env, underlying, "ATMVol", dtCalc,
    sigma)
setData(base.env, "USD.LIBOR", "Yield", dtCalc,
    int.rate)
```

Construct a portfolio, as a list of `fInstrument` objects:

```
portfolio = c(v, b)
```

and finally compute the price and delta of the portfolio:

```
price <- sum(sapply(portfolio, function(a) getValue(a,
    "Price", dtCalc, base.env)))
delta <- sum(sapply(portfolio, function(a) getValue(a,
    "Delta", dtCalc, base.env)))
print(paste("Price:", round(price, 2), "Delta:",
    round(delta, 3)))
```

```
## [1] "Price: 13.31 Delta: 0.599"
```

3.1.1 *Design and Implementation*

`fInstrument` objects are instantiated by the `fInstrumentfactory` class method, which takes as argument:

- the type of instrument being instantiated,
- the quantity or nominal amount of the instrument
- a list of parameters that define the instrument

The `fInstrumentFactory` method is simply a switch that delegates the object instantiation the the concrete subclasses of `fInstrument`. The following code fragment is extracted from the file `fInstrument.r` in package `fInstrument`:

```
fInstrumentFactory <- function(type, quantity,
    params) {
    switch(toupper(type), VANILLA = Vanilla(quantity,
        params), BINARY = Binary(quantity, params),
        ASIAN = Asian(quantity, params), STANDARDBARRIER = StandardBarrier(quantity,
            params))
}
```

There is only one method defined on `fInstrument` objects. This method is `getValue`, and takes as argument:

- the kind of calculation being requested (price, delta)
- the calculation date,
- the data container from which the required market data will be fetched.

Again, this method simply delegates to the concrete classes the requested calculation:

```
setMethod(f = "getValue", signature = signature("fInstrument"),
    definition = function(object, selection, dtCalc,
        env = NULL) {
```

```
        res <- NULL
        res <- switch(toupper(selection), PRICE = object@p(dtCalc,
            env), DELTA = object@d(dtCalc, env),
            GAMMA = object@g(dtCalc, env), VEGA = object@v(dtCalc,
                env))
        return(res * object@quantity)
    })
```

As an illustration, the price calculation for vanilla options is implemented as follows in `Vanilla.r`:

```
getP <- function(dtCalc, env) {
    Spot <- getData(env, Underlying, "Price",
        dtCalc)
    s <- getData(env, Underlying, "ATMVol", dtCalc)
    r <- getData(env, df, "Yield", dtCalc)
    b <- getData(env, Underlying, "DivYield",
        dtCalc)
    t <- tDiff(dtCalc, dtExpiry)
    if (trace) {
        print(paste("Calling GBSOption with Spot=",
            Spot, "Strike=", Strike, "t=", t,
            "r=", r, "b=", b, "sigma=", s))
    }
    GBSOption(TypeFlag = cp, S = Spot, X = Strike,
        Time = t, r = r, b = b, sigma = s)@price
}
```

The actual calculation being performed by the Rmetrics `GBSOption` function. The model can be easily extended to accommodate other instruments.

3.2 *The `DataProvider` Class*

The `DataProvider` class is a container of market data, from which the pricing algorithm will fetch the necessary market information, as illustrated in the code fragment above. We first describe the representation of market data, then the algorithm for searching data in a `DataProvider`.

3.2.1 *The Model for Market Data*

The model for market data is strongly inspired by M. Folwer (1996). To summarize, a piece of market data is modeled as an observed phenomenon on a financial instrument. Therefore, every market data item is identified by three attributes:

- the financial instrument being observed (e.g. a stock)
- item the observed phenomenon (e.g. the implied volatility, or the price)
- the observation date

In order to optimize storage, the data is stored in a hash table. The first two attributes are combined to created the key, and the data for all observation dates is stored as a time series, with one column for actual data, and many additional columns when performing a simulation.

3.2.2 *The Search Algorithm*

The `DataProvider` inherits from the built-in `environement` class. In particular, it inherits the parent/child relationship: if a `DataProvider` has a parent, the data not found in the child environment is fetched from the parent, if possible, or from the grand-parent, and so forth.

This is useful when performing a scenario analysis where only a few variables are modified: The data held constant is stored in the parent scenario, and the modified data is stored in the child scenario which is passed as argument. This scheme is illustrated by the following example.

Let's define a vanilla option:

```
dtExpiry <- myDate("01jan2011")
underlying <- "IBM"
K <- 100
a <- fInstrumentFactory("vanilla", quantity = 1,
    params = list(cp = "c", strike = K, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))
```

and populate a DataProvider with the necessary market data:

```
base.env <- DataProvider()
dtCalc <- myDate("01jan2010")

setData(base.env, underlying, "Price", dtCalc,
    100)
setData(base.env, underlying, "DivYield", dtCalc,
    0.02)
setData(base.env, underlying, "ATMVol", dtCalc,
    0.3)
setData(base.env, "USD.LIBOR", "Yield", dtCalc,
    0.02)
```

The NPV of the derivative is obtained by:

```
getValue(a, "Price", dtCalc, base.env)
[1] 12.82158
```

Next, we investigate the relationship between the underlying price and the value of the derivative, by inserting a set of scenarios for the underlying asset price in a child `DataProvider`:

```
sce <- t(as.matrix(seq(80, 120, length.out = 30)))
sce.env <- DataProvider(parent = base.env)
setData(sce.env, underlying, "Price", dtCalc,
    sce)
```

and compute the NPV of the derivative for each scenario. The relationship between the underlying price and the value of the call is illustrated in figure 3.1.

```
p <- getValue(a, "Price", dtCalc, sce.env)
plot(sce, p, type = "l", lwd = 3, xlab = "Spot",
    ylab = "Price", bty = "n", col = "red")
```

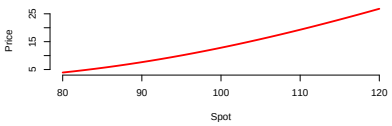


Figure 3.1: Call Price as a function of spot value. Strike: 100, maturity: 1 Year

4 The Simulation Framework

```
library(fInstrument)
library(empfin)
library(DynamicSimulation)
library(kableExtra)
```

{r tufte::newthought("The")} simulation framework has two main components:

1. A simulator, which can generate paths for the variables of interest (price of a asset, implied volatility, term structure of interest rate, etc.), according to a model that relates the variables to stochastic risk factors.
2. A framework for expressing trading strategies, and in particular dynamic hedging policies.

This is again best explained by an example, and we describe next the use of this framework for running a delta-hedging experiment. The main elements of the design are discussed afterwards.

4.1 Scenario Simulator

Assume that you want to simulate 500 price paths over a period of one year, with 100 time steps. The price process will be log-normal with annual volatility of 30%, starting from an initial value of $100.

```
dtStart <- myDate("01jan2010")
dtEnd <- myDate("01jan2011")
nbSteps <- 100
nbPaths <- 500
```

Next, define the sequence of simulation dates and the volatility of the simulated log-normal process:

```
dtSim <- seq(dtStart, dtEnd, length.out = nbSteps +
    1)
sigma <- 0.3
```

Use a sobol sequence as random number generator, with antithetic variates, standardized to unit variance:

```
tSpot <- pathSimulator(dtSim = dtSim, nbPaths = nbPaths,
    innovations.gen = sobolInnovations, path.gen = logNormal,
    path.param = list(mu = 0, sigma = sigma), S0 = 100,
    antithetic = FALSE, standardization = TRUE, trace = FALSE)
print(head(tSpot[, 1:2]))
```

```
## GMT
##                          TS.1     TS.2
## 2010-01-01 00:00:00 100.0000 100.0000
## 2010-01-04 15:36:00 100.1452 103.5384
## 2010-01-08 07:12:00 105.6815 104.4292
## 2010-01-11 22:48:00 108.0174 104.1740
## 2010-01-15 14:24:00 109.7310 106.3975
## 2010-01-19 06:00:00 110.4729 108.0305
```

The output of the path simulator is a `timeSeries`. A plot of the first few paths is displayed in figure 4.1. The function can generate simulated values acording to various statistical processes; this is documented in the vignette of the package.

```
plot(tSpot[, 1:50], plot.type = "single", ylab = "Price",
    format = "%b %d")
```

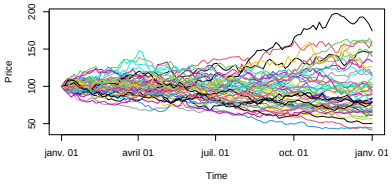


Figure 4.1: Simulated price paths under a log-normal diffusion process

4.2 *A Delta Hedging Experiment*

Having generated some scenarios for the stock price, let's now simulate the dynamic hedging of a European call option written on this stock, using the Black-Scholes pricing model, with the implied volatility and interest rate held constant.

Fist, we define the instrument to be hedged:

```
dtExpiry <- dtEnd

underlying <- "IBM"
K <- 100

a <- fInstrumentFactory("vanilla", quantity = 1, params = list(cp = "c",
    strike = K, dtExpiry = dtExpiry, underlying = underlying,
    discountRef = "USD.LIBOR", trace = FALSE))
```

Next, we define the market data that will be held constant during the simulation, and insert it in a `DataProvider`:

```
base.env <- DataProvider()
setData(base.env, underlying, "Price", dtStart, 100)
setData(base.env, underlying, "DivYield", dtStart,
    0.02)
setData(base.env, underlying, "ATMVol", dtStart, sigma)
setData(base.env, underlying, "discountRef", dtStart,
    "USD.LIBOR")
setData(base.env, "USD.LIBOR", "Yield", dtStart, 0.02)
```

At this stage, we can price the asset as of the start date of the simulation:

```
p <- getValue(a, "Price", dtStart, base.env)
```

which gives a value of $p = 12.82$.

Next, define the simulation parameters: we want to simulate a dynamic hedging policy over 500 paths, and 100 time steps per path:

We use a child `DataProvider` to store the simulated paths. Data not found in the child `DataProvider` will be searched for (and found) in the parent `base.env`.

```
sce.env <- DataProvider(parent = base.env)
setData(sce.env, underlying, "Price", time(tSpot),
    as.matrix(tSpot))
```

We can now run the delta-hedge strategy along each path:

```
assets = list(a)
res <- deltaHedge(assets, sce.env, params = list(dtSim = time(tSpot),
    transaction.cost = 0), trace = FALSE)
```

The result is a data structure that contains the residual wealth (hedging error) per scenario and time step. The distribution of hedging error at expiry is shown in Figure 4.2.

```
hist(tail(res$wealth, 1), 50, xlab = "Residual wealth at expiry",
    main = "")
```

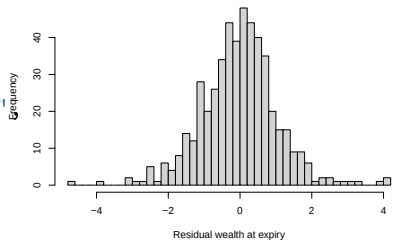


Figure 4.2: Distribution of residual wealth at expiry: delta hedge of a 1 year call option

To better illustrate the hedging policy, let's run a toy example with few time steps. The function `deltaHedge` produces a detailed log of the hedging policy, which is presented in Table 4.1. For each time step, the table show:

- The stock price,
- the option delta,
- the option value,
- the value of the replicating portfolio and the short bond position in that portfolio.

```
dtSim <- time(tSpot)[seq(1, dim(tSpot)[1], 10)]
res <- deltaHedge(assets, sce.env, params = list(dtSim = dtSim,
    transaction.cost = 0), trace = FALSE)
sim.table <- makeTable(1, res)
```

Table 4.1: Simulated value of a call option and its hedge portfolio over time

time	stock price	delta	option	bond pos	hedge port.
1	100.00	0.586	12.82	-46.68	12.82
2	109.18	0.696	18.01	-58.98	18.11
3	112.07	0.732	19.34	-63.02	20.00
4	95.63	0.501	8.28	-40.65	7.85
5	100.39	0.573	10.02	-47.96	10.15
6	95.88	0.482	6.76	-39.13	7.47
7	95.12	0.450	5.48	-36.02	7.03
8	75.11	0.052	0.26	-6.00	-2.04
9	74.66	0.019	0.07	-3.48	-2.07
10	73.62	0.001	0.00	-2.16	-2.10
11	61.39	0.000	0.00	-2.11	-2.11

4.2.1 Design Considerations

Two design features are worth mentioning.

The generation of the scenarios is independent from the expression of the dynamic trading strategies. Remember that every data element stored in a `DataProvider` is a `timeSeries`. Since all the calculations on `fInstrument` are vectorized, there is no difference between performing a calculation on a scalar, or performing a simulation on multiple scenarios.

The second aspect is the use of parent/child relationships among `DataProvider` objects. All the market data that is held constant in the simulation is stored in the parent `DataProvider`. The data that changes from simulation to simulation is stored in the child `DataProvider`, and this is the object that is passed to the simulator. When a piece of data is requested from the child `DataProvider`, the following logic is applied:

1. First look for the data in the current `DataProvider` (the object passed as argument to the simulator)
2. if not found, look for the data in the parent `DataProvider`.
3. and so forth: the logic is recursive.

This behavior is inherited from the built-in `environment`.

Discreet Models

5 *Arbitrage-Free Pricing and Risk Neutral Valuation*

```
library(fBasics)
library(xtable)
library(empfin)
library(magick)
library(tufte)
library(dplyr)
library(ggplot2)
library(ggforce)
library(png)
library(grid)
library(ggraph)
library(igraph)
```

We are exclusively concerned about the pricing of redundant securities, i.e. relative value pricing. Financial economics tries to explain the pricing of underlying securities, mathematical finance is about the relative value pricing of derivatives securities.

5.1 Arbitrage-free Pricing

Consider three produce baskets of apples and oranges.

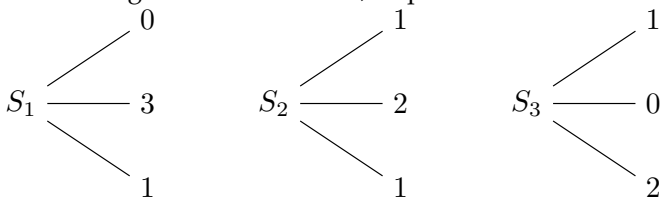
Basket	Apples	Oranges	Price (€)
B1	2	3	4
B2	3	2	5
B3	2	2	?

Is € 3.5 a fair price for basket B3? The answer is no: I can buy 5 baskets B3 and sell 2 baskets B1 and 2 baskets B1 for a riskless profit of €0.5. The fair price for B3 is 3.6 €. We have determined the arbitrage-free price for B3 by constructing a replication out of baskets B1 and B2. This is the essence of derivatives pricing.

5.1.1 *Arrow-Debreu Securities*

Imagine an economy that can evolve between time $t = 0$ and $t = 1$ to take one out of 3 possible states. There is a consensus to attribute probability p_i to the occurence of state i. The interest rate is null.

We next define 3 securities; each one pays a certain pattern of cash-flow according to the future state, as pictured below:



Asset 1 Asset 2 Asset 3

Assume that prices at time $t = 0$ are: $S_1 = 1.3$, $S_2 = 1.25$, $S_3 = 1.3$. The price of these securities is determined by several factors:

1. the preference of market participants for earning a payoff in one state versus another: in a risk-adverse economy, earning 1 euro when the aggregate wealth is low is more valuable than earning the same amount when the aggregate wealth is high
2. the preference for holding money today rather than at time T
3. the likelihood of each state

We are now interested in the price of securities that pays 1 euro if state i is realized, and nothing otherwise. Such securities are called Arrow-Debreu securities, and can be represented by a vector giving the value of such security in each future state of the world.

The value of the first Arrow-Debreu security is determined by solving the following system:

$$\begin{pmatrix} 0 & 1 & 1 \\ 3 & 2 & 0 \\ 1 & 1 & 2 \end{pmatrix} W_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

which yields

```
P <- matrix(c(0, 3, 1, 1, 2, 1, 1, 0, 2), 3, 3)
A1 <- matrix(c(1, 0, 0), 3, 1)
W1 <- inv(P) %*% A1
W1
```

```
##      [,1]
## [1,] -0.8
## [2,]  1.2
## [3,] -0.2
```

The value of the first Arrow-Debreu security is thus:

```
S <- matrix(c(1.3, 1.25, 1.3), 3, 1)
a1 <- t(W1) %*% S
print(a1[1, 1])
```

```
## [1] 0.2
```

Similarly, the prices a_2 and a_3 of the other two Arrow-Debreu securities are computed by:

```
W2 <- inv(P) %*% matrix(c(0, 1, 0), 3, 1)
a2 <- t(W2) %*% S
W3 <- inv(P) %*% matrix(c(0, 0, 1), 3, 1)
a3 <- t(W3) %*% S
```

The replicating portfolios and prices of the 3 A-D assets are summarized below.

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sat Jan 14 22:40:00 2023

	W1	W2	W3	Price
1	-0.80	0.20	0.40	0.20
2	1.20	0.20	-0.60	0.25
3	-0.20	-0.20	0.60	0.55

A portfolio made of the three Arrow-Debreu securities guarantees a payoff of 1 euro. Therefore, this collection can be priced at time $t = 0$ by discounting the payoff at the risk-free rate (0 for now). Thus, we must have:

$$\sum a_i = 1$$

Which is indeed the case:

```
print(a1 + a2 + a3)
```

```
##      [,1]
## [1,]    1
```

5.1.2 *The Price of Traded Securities*

The price of Arrow-Debreu securities is determined by the price of traded securities. We now consider how these price are determined.

The current price of an asset depends on the future payoffs, and on the states in which these payoffs occur: if the payoffs of an asset are positively correlated with the aggregate market value, it will be worth less, everything else being equal, than an asset whose payoffs are negatively correlated with the aggregate market value. The capital asset pricing model formalizes this observation. Let

S_t. Asset price at time t

M_t. Aggregate market wealth at time t

R_s. Return on asset: S_T/S_0

R_m. Market return: M_T/M_0

R_f. Risk-free return: $1+r$

The model relates the expected return of a security to the beta value:

$$E(R_s) = R_f + \beta[E(R_m) - R_f]$$

where

$$\beta = \frac{\text{Cov}(R_s, R_m)}{\text{Var}(R_m)}$$

In that framework, it can be shown that S_0, the current asset price, is given by:

$$S_0 = \frac{E(S_T) - \lambda \text{Cov}(S_T, M_T)}{R_f}$$

where λ is the market price of risk times the current market value M_0:

$$\lambda = \frac{M_0[E(R_m) - R_f]}{\text{Var}(M_T)}$$

In a complete market, the asset and the market portfolio can be expressed in terms of Arrow-Debreu securities:

$$S_0 = \sum_i V_i a_i$$

$$M_0 = \sum_i U_i a_i$$

Assume that the states are ordered in order of increasing aggregate wealth. We have:

$$\begin{aligned} E(S_T) &= \sum_i V_i p_i \\ E(M_T) &= \sum_i U_i p_i \\ E(S_T M_T) &= \sum_i U_i V_i p_i \end{aligned}$$

Substitute in (5.1.2) to get:

$$S_0 = \sum_i V_i d_i p_i$$

where the discount factor d_i is given by:

$$d_i = \frac{1 - \lambda(U_i - E(M_T))}{R_f}$$

Equation (5.1.2) shows that as aggregate wealth U_i increases, the discount factor decreases. An asset is more valuable, everything else being equal, if its payoffs occur in the states where U_i is low, and therefore where the discount factor is high.

To generalize: any factor that affects supply and demand for traded securities, and the market price of risk, will have a direct influence on the Arrow-Debreu prices and therefore on the risk neutral probabilities.

As noted by Derman and Taleb((**Derman2005?**)),

> The Nobel committee upon granting the Bank of Sweden Prize in honour of Alfred Nobel, provided the following citation: "Black, Merton and Scholes made a vital contribution by showing that it is in fact not necessary to use any risk premium when valuing an option. This does not mean that the risk premium disappears; instead it is already included in the stock price." It is for having removed the effect of μ
>
> $$thestockdrift$$
>
> on the value of the option, and not for rendering the option a deterministic and riskless security, that their work is cited.

5.1.3 *Complete Market*

A complete market is a market where all Arrow-Debreu securities can be traded, and therefore any payoff profile can be replicated as a portfolio of Arrow-Debreu securities. The existence of this replicating portfolio imposes a unique arbitrage-free price for any payoff. Going back to the elementary example above, consider now a new security that has the payoff profile illustrated in figure **??**.

$$grow' = right, siblingdistance = 1cm$$

child node 1 child node -0.5 child node 1;

$$grow' = right, siblingdistance = 1cm$$

child node 1 child node -0.5 child node 1;

This security is equivalent to a portfolio of three Arrow-Debreu securities, and is worth

```
V <- a1 - 0.5 * a2 + a3
```

If its market price of S_4 is less than $V = 0.62$, you can earn a riskless profit by buying a unit of S_4 and selling the portfolio P.

In general, a security with payoff X_i in state i is worth:

$$\sum_i a_i X_i$$

with

$$\sum_i a_i = 1$$

and

$$a_i > 0, \ \ \forall i$$

and we can interpret the Arrow-Debreu prices as probabilities. Since preferences no longer play a role, we call them "risk-neutral" probabilities.

5.1.4 *Equivalent Probability Measures*

Two probability measures p and q are equivalent if they are consistent with respect to possible and impossible outcome:

$$p_i > 0 \Leftrightarrow q_i > 0$$

Let p be the real probability measure and q be the risk-neutral measure. It is easy to show that the two measures must be equivalent.

Consider a state i such that $p_i = 0$. then the corresponding Arrow-Debreu security cannot cost anything, or a riskless profit could be gained by selling this security. Thus, $q_i = 0$. A similar argument applies for the case $p_i > 0$.

5.1.5 *The Impact of Interest Rate*

What happens to Arrow-Debreu prices and risk-neutral probabilities when interest rate is not null? In the presence of interest rate, the value of a complete set of Arrow-Debreu securities must be

$$\sum_i a_i = e^{-rT}$$

The risk-neutral probabilities are now defined as:

$$q_i = a_i e^{rT}$$

so that we still have $\sum_i q_i = 1$. As before, an arbitrary security with payoff X_i in state i is worth,

$$P(X) = e^{-rT} \sum_i X_i q_i$$

or,

$$P(X) = e^{-rT} E^Q[X]$$

Risk-neutral probabilities are compounded Arrow-Debreu prices. Here, for expository purpose, we have derived risk-neutral probabilities from state prices, but in practice, we will do the opposite: to obtain state prices from risk-neutral probabilities.

5.1.6 Trading Strategy and Dynamic Completeness

Let's now consider an economy where decisions can be made at various stages. This economy is illustrated in Figure **??**. At the second time step, we have 3 distinct states, i.e. 3 Arrow-Debreu securities. If we only had one time step, we would need 3 linearly independent assets to construct these securities. Now, because of the intermediate trading opportunity, we may be able to construct the Arrow-Debreu securities with fewer (i.e. two) underlying assets. A market in which every Arrow-Debreu security can be constructed with a self-financing trading strategy is called dynamically complete.

5.1.7 Discounted Asset Prices as Martingales

Let's consider again a two-stage economy. What can we say at time o about the expected value of a derivative at a future time t?

$$E_0^Q[P_t(X)], \forall t \geq 0$$

$$E_0^Q[P_t(X)] = e^{-2r} E_0^Q[X]$$

Now let's consider the expected price at $t = 1$:

$$\begin{aligned} E_0^Q[P_1(X)] = &\quad E_0^Q[e^{-r} E_1^Q[X]] \\ E_0^Q[P_1(X)] = &\quad E_0^Q[e^{-r} E_0^Q[X]] \\ E_0^Q[P_1(X)] = &\qquad e^{-r} E_0^Q[X] \end{aligned}$$

Similarly,

$$\begin{aligned} E_0^Q[P_2(X)] = &\quad E_0^Q[E_2^Q[X]] \\ E_0^Q[P_2(X)] = &\qquad E_0^Q[X] \end{aligned}$$

The price of each state-dependent payoff grows at the risk-free rate, simply because this growth rate is incorporated in the definition of each risk-neutral probability with which we weight the state-dependent payoffs.

The expected price of any asset (as seen from time $t = 0$) grows at the risk-free rate. Not the price, but the expectation of the price. Generalizing, we have:

$$P_0(X) = e^{-rt} E_0[P_t(X)], \ \forall t > 0$$

6 Risk-Neutral Pricing in a Binomial Framework

In this chapter we use the binomial tree framework to introduce the key concepts of option valuation.

6.1 Introduction

Consider first a one-period model: An asset S_t is worth \$40 at $t = 0$, and suppose that a month later, at $t = T$, it will be either \$45 or \$35. We are interested in buying a call option struck at $K = 40$, expiring at T. Interest rate is 1% per month. What is the fair price of this option?

The option payoff is

$$c_T = (S_T - K)^+ = \begin{cases} 5 & \text{if } S_T = 45 \\ 0 & \text{if } S_T = 35 \end{cases}$$

Now consider a portfolio made of one unit of stock and 2 call options:

$$\Phi = S - 2c$$

$$S_T - 2c_T = \begin{cases} 35 & \text{if } S_T = 45 \\ 35 & \text{if } S_T = 35 \end{cases}$$

This portfolio is worth today the same as a bank deposit that would provide \$35 in a month, or

$$\frac{35}{1 + 1\%} = 34.65$$

In this simple one-period economy, the option is thus worth:

$$c_o = \frac{40 - 34.65}{2} = 2.695$$

6.1.1 The Binomial Model for Stocks

Consider again the one-period binomial model of the previous section, and introduce notation to characterize the value of the three assets of

interest, today and in the two future states, labeled "Up" and "Down". The notation is summarized in Table

$$tab : binomial$$

.

State	Stock	Bond	Call
Today	S_0	B_0	c_0
Up	$S_T^u = S_0 u$	$(1+rT)B_0$	$c_T^u = (S_T^u - K)^+$
Down	$S_T^d = S_0 d$	$(1+rT)B_0$	$c_T^d = (S_T^d - K)^+$

Construct a risk-free portfolio made of the option and the stock:

$$\Pi_0 = c_0 - \Delta S_0$$

To be riskless, one must have:

$$\Pi_T = (1+rT)\Pi_0$$

In particular, this is true in the two scenarios for S_T:

$$\begin{aligned} c_T^u - \Delta S_0 u = &\quad (1+rT)(c_0 - \Delta S_0) \\ c_T^d - \Delta S_0 d = &\quad (1+rT)(c_0 - \Delta S_0) \end{aligned}$$

Solve for Δ:

$$\Delta = \frac{c_T^u - c_T^d}{S_0(u-d)}$$

Set $(1+rT) = \rho$. The option value at time $t = 0$ is:

$$\begin{aligned} c_0 = &\quad \frac{1}{\rho}\Pi_T + \Delta S_0 \\ = &\quad \frac{1}{\rho}\left(\frac{\rho - d}{u-d}c_T^u + \frac{u-\rho}{u-d}c_T^d\right) \end{aligned}$$

Assume $d < \rho < u$. and define:

$$\begin{aligned} q_u = &\quad \frac{\rho - d}{u-d} \\ q_d = &\quad \frac{u-\rho}{u-d} \end{aligned}$$

Rewrite:

$$c_0 = \frac{1}{\rho}\left(q_u c_T^u + q_d c_T^d\right)$$

One can observe that $0 < q_u, q_d < 1$ and that $q_u + q_d = 1$, and therefore interpret q_u and q_d as probabilities associated with the events

$S_T = S_T^u$ and $S_T = S_T^d$. Let Q be this probability measure. This leads us to write:

$$c_0 = \frac{1}{\rho} E^Q(c_T)$$

The option price is the discounted expected future payoff, under the probability Q.

6.2 *Pricing With Arrow-Debreu Securities*

An alternative derivation of the same result can be obtained with Arrow-Debreu securities. Let's first compute the price a_1 and a_2 of the two Arrow-Debreu securities in this economy. The price of the option will then be, by definition:

$$c_0 = a_1 c_T^u + a_2 c_T^d$$

where a_1 and a_2 are the prices of the Arrow-Debreu securities for the up and down scenarios.

Let's now determine the prices of these Arrrow-Debreu securities. To do so, we construct a portfolio made of x units of stock and y units of bond, that has the same payoff as an Arrrow-Debreu security. Setting $B_0 = 1$, the replicating portfolio for the first Arrow-Debreu security is obtained by solving the system:

$$\begin{pmatrix} S_0 u & \rho \\ S_0 d & \rho \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Which yields:

$$x = \frac{1}{S_0(u-d)}, \quad y = -\frac{d}{\rho(u-d)}$$

The price of the first Arrow-Debreu security is thus:

$$\begin{aligned} a_1 = & \quad xS_0 + yB_0 \\ = & \quad \frac{1}{\rho} \frac{\rho - d}{u - d} \end{aligned}$$

Similarly, the second Arrow-Debreu security is found to be worth:

$$a_2 = \frac{1}{\rho} \frac{\rho - u}{u - d}$$

and we obtain therefore the same option price as in (

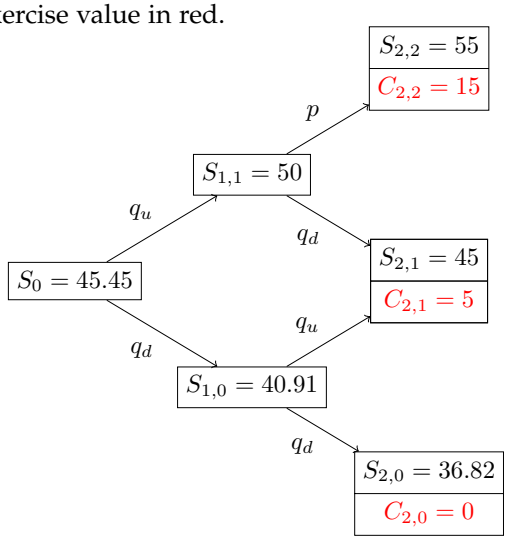
$$eq : cox - ross - 1$$

)

6.3 *Multi-Period Binomial Model*

The logic of the previous sections can be extented to multiple periods. When dealing with multiple periods, it is important in practice to construct recombining trees, i.e. trees where an up move followed by a down move results in the same state than a down move followed by a up move. N steps in a recombining tree generate $N+1$ states, but 2^N states in a non-recombining binomial tree.

The calculation process in a multi-period tree is best explained through an example. Consider a stock priced at 45.45€. At each period, the stock may appreciate or depreciate by 10%. The riskless rate is 5%, and we want to price a call option struck at 40€. The two-period tree is represented in figure **??** with stock price in black and the option exercise value in red.



The risk-neutral probabilities are identical at all nodes:

$$\begin{aligned} q_u &= \frac{\rho - d}{u - d} = \frac{1.05 - .9}{1.1 - .9} = .75 \\ q_d &= .25 \end{aligned}$$

Using these values, the option value one step prior to expiry can be computed using ((6.1.1)):

$$\begin{aligned} C_{1,1} &= \frac{1}{\rho}(q_u C_{2,2} + q_d C_{2,1}) = 11.90 \\ C_{1,0} &= \frac{1}{\rho}(q_u C_{2,1} + q_d C_{2,0}) = 3.57 \end{aligned}$$

The reasoning that led to (

$$eq : cox - ross - 1$$

) applies however to any node, and in particular to node (0,0). The option price at that node is thus:

$$C_{0,0} = \frac{1}{\rho}(q_u C_{1,1} + q_d C_{1,0}) = 9.35$$

The process is pictured in Figure **??**.

6.4 *American Exercise*

American exercise refer to the right to exercise the option at any time before expiry. Clearly, an option with American exercise is worth more than the comparable option with European exercise. To price an American option, we introduce the notion of continuation value. The continuation value V_t^i at node i and time t is the option value, assuming that it will be exercised after time t. At each step, one determines the optimal decision by comparing the value of an immediate exercise to the continuation value. The option value is therefore, for a call:

$$\begin{aligned} C_t^i = & \max(S_t^i - K, V_t^i) \\ = & \max(S_t^i - K, \frac{1}{\rho}(q_u C_{t+1}^{i+1} + q_d C_{t+1}^i)) \end{aligned}$$

Starting from the data of the previous section, we now assume that the stock pays a dividend of 3 € in period 2. The modified tree is represented in figure **??**.

We price an American call struck at 40€ in this tree. Exercise values in period 2 are computed as before, giving the following values:

$$\begin{aligned} C_{2,2} = & \quad 12 \\ C_{2,1} = & \quad 0 \\ C_{2,0} = & \quad 0 \end{aligned}$$

At period 1, the option holder determines the optimal policy: exercise immediately or hold the option until period 2. The resulting values are:

$$\begin{aligned} C_{1,1} = & \quad \max((50-40), \frac{1}{\rho}(q_u 10 + q_d 2)) = 10 \\ C_{1,0} = & \quad \max((40.91-40), \frac{1}{\rho}(q_u 2 + q_d 0)) = 1.42 \end{aligned}$$

The option value today is finally determined to be:

$$\begin{aligned} C_{0,0} = & \quad \max(5.45, \frac{1}{\rho}(q_u 10 + q_d 1.42)) \\ = & \quad 7.48 \end{aligned}$$

Under the same conditions, a European option is worth $C_{0,0} = 6.79$. The difference comes from the early exercise of the American option in node (1,1).

6.5 Calibration of the Binomial Model

With interest rate assumed to be known, option prices are determined by the terms u and d that describe the binomial branching process. How should u and d be chosen?

The time interval $[0, T]$ is divided into N equal steps $[t_j, t_{j+1}]$ of length Δt. Assume that the process for the underlying asset S_t is such that $V_t = \ln(S_t/S_{t-\Delta t})$ are iid random variables with mean $\mu\Delta t$ and variance $\sigma^2\Delta t$.

$$S_{t_j} = S_{t_{j-1}} e^{V_j}$$

Let's determine u and d by matching the mean and variance of V_t and of the binomial process:

$$\begin{aligned} E(V_j) &= p\ln u + (1-p)\ln d \\ &= \mu\Delta t \\ V(V_j) &= E(V_j^2) - E(V_j)^2 \\ &= \sigma^2\Delta t \end{aligned}$$

Which forms a system of 2 equations and three unknown. Without loss of generality, set $p = 1/2$, to obtain:

$$\begin{aligned} \frac{1}{2}(\ln u + \ln d) &= \mu\Delta t \\ \frac{1}{4}(\ln u + \ln d)^2 &= \sigma^2\Delta t \end{aligned}$$

The solution is:

$$\begin{aligned} u &= e^{\mu\Delta t + \sigma\sqrt{\Delta t}} \\ d &= e^{\mu\Delta t - \sigma\sqrt{\Delta t}} \end{aligned}$$

The corresponding value of the risk-neutral up probability, q_u is

$$q_u = \frac{e^{r\Delta t} - e^{\mu\Delta t - \sigma\sqrt{\Delta t}}}{e^{\mu\Delta t + \sigma\sqrt{\Delta t}} - e^{\mu\Delta t - \sigma\sqrt{\Delta t}}}$$

This is the single-period risk-neutral probability, not the objective probability p. We next compute the limit of q_u as $\Delta t \to 0$.

We use the following first order approximation:

$$\begin{aligned} e^{\mu\Delta t + \sigma\sqrt{\Delta t}} &= (1 + \mu\Delta t + \sigma\sqrt{\Delta t} + \frac{1}{2}(\mu\Delta t + \sigma\sqrt{\Delta t})^2 + \dots \\ &= 1 + \mu\Delta t + \sigma\sqrt{\Delta t} + \frac{1}{2}\sigma^2\Delta t + O(\Delta t^{3/2}) \end{aligned}$$

and similarly,

$$e^{\mu\Delta t-\sigma\sqrt{\Delta t}} = 1+\mu\Delta t-\sigma\sqrt{\Delta t}+\frac{1}{2}\sigma^2\Delta t+O(\Delta t^{3/2})$$

Combining these approximations with (

$$eq : qu$$

) yields,

$$\begin{aligned} q_u = & \frac{\sigma\sqrt{\Delta t}+(r-\frac{1}{2}\sigma^2-\mu)\Delta t+O(\Delta t^{3/2})}{2\sigma\sqrt{\Delta t}+O(\Delta t^{3/2})} \\ = & \frac{1}{2}+\frac{r-\frac{1}{2}\sigma^2-\mu}{2\sigma}\sqrt{\Delta t}+O(\Delta t) \end{aligned}$$

Let's now use these expressions for q_u to compute the mean and variance of V_j.

$$E(V_j) = q_u \ln u+(1-q_u)\ln d$$

Use (

$$eq : ud$$

) to get:

$$E(V_j) = q_u(2\sigma\sqrt{\Delta t})+\mu\Delta t-\sigma\sqrt{\Delta t}$$

Substitute q_u by its value (

$$eq : qu$$

) to get:

$$E(V_j) = (r-\frac{1}{2}\sigma^2)\Delta t+O(\Delta t^{3/2})$$

Similarly,

$$Var(V_j) = \sigma^2\Delta t+O(\Delta t^{3/2})$$

The remarkable point of this result is that μ no longer appears.
Extending the reasoning of Section

$$subsec : binomial$$

to multiple periods, we write that under the risk neutral probability q_u, q_d, the option value is the discounted expected value of the payoff:

$$\begin{aligned} P(S_0) = & e^{-rT}E^Q(f(S_T)) \\ = & e^{-rT}E^Q(f(S_0e^{\sum_{i=1}^N V_i})) \end{aligned}$$

where $f(S_T)$ is the payoff at expiry. We next need to compute the limit:

$$\lim_{N\to\infty} \sum_{i=1}^{N} V_i$$

The variables V_i are a function of N, thus the Central Limit Theorem cannot be used as such. However, we can invoke Lindeberg's condition to obtain the same result:

(Lindeberg's Condition) Let $X_k, k \in N$ be independent variables with $E(X_k) = \mu_k$ and $V(X_k) = \sigma_k^2$. Let $s_n^2 = \sum_i^n \sigma_i^2$. If the variables satisfy Lindeberg's condition, then

$$Z_n = \frac{\sum_{k=1}^{n}(X_k - \mu_k)}{s_n} \to N(0,1)$$

To simplify notation, let $E^Q(V_i) = a$, $Var^Q(V_i) = b$, Lindeberg's condition yields:

$$\frac{\sum_i^N V_i - Na}{b\sqrt{N}} \to N(0,1)$$

$$\frac{\sum_i^N V_i - Na}{b\sqrt{N}} = \frac{\sum_i^N V_i - (r - \frac{1}{2}\sigma^2)T + O(N^{-1/2})}{\sigma\sqrt{T} + O(N^{-1/2})}$$

$$\frac{\sum_i^N V_i - (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T})} \to N(0,1)$$

Thus,

$$\sum_i^N V_i \to N\left((r - \frac{1}{2}\sigma^2)T, \sigma^2 T\right)$$

Finally, as $N \to \infty$, (

$$eq : pso$$

) becomes:

$$P(S_0) = \frac{e^{-rT}}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(S_0 e^{(r-\frac{1}{2}\sigma^2)T + \sigma\sqrt{T}} e^{-\frac{1}{2}u^2} du$$

which is the Black-Scholes valuation formula, as derived by Cox, Ross, and Rubinstein (1979). Again, the significance of this result is that μ does not appear in the formula.

6.5.1 Tree Geometry

We now use the result from the previous section to determine the geometry of the tree and the risk-neutral transition probabilities, consistent with the parameters of the diffusion process.

Recall from the previous section that u and d are defined by:

$$\begin{aligned} u &= e^{\mu\Delta t+\sigma\sqrt{\Delta t}} \\ d &= e^{\mu\Delta t-\sigma\sqrt{\Delta t}} \end{aligned}$$

Ultimately, μ does not appear in the valuation formula, it can thus be set to an arbitrary value without loss of generality.

In the original CRR model, $\mu = 0$, which leads to:

$$\begin{aligned} u &= e^{\sigma\sqrt{t}} \\ d &= e^{-\sigma\sqrt{t}} \\ q_u &= \frac{e^{rt} - e^{-\sigma\sqrt{t}}}{e^{\sigma\sqrt{t}} - e^{-\sigma\sqrt{t}}} \end{aligned}$$

There are many other possibilities: a popular choice introduced by Jarrow and Rudd (1993) is to set μ so that transition probabilities are equal to $\frac{1}{2}$. Using (

$$eq : qu - 2$$

), this involves setting $\mu = r - \frac{1}{2}\sigma^2$, and the branching process is then:

$$\begin{aligned} u &= e^{(r-\frac{1}{2}\sigma^2)\Delta t+\sigma\sqrt{\Delta t}} \\ d &= e^{(r-\frac{1}{2}\sigma^2)\Delta t-\sigma\sqrt{\Delta t}} \\ q_u &= \frac{1}{2} \end{aligned}$$

There are many other possible choices, but no significant differences in the convergence rate to the Black-Scholes option value. Most of the models, however, suffer from a form of instability which is now described.

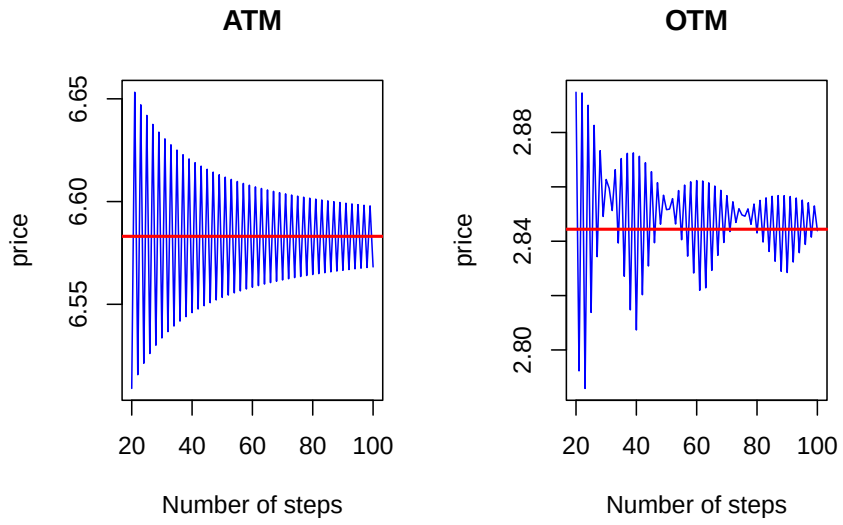
7 *Stability of the Binomial Model*

We would like to verify the convergence of the Cox-Ross model to the Black-Scholes price as the number of steps N increases. This can be investigated with the following script:

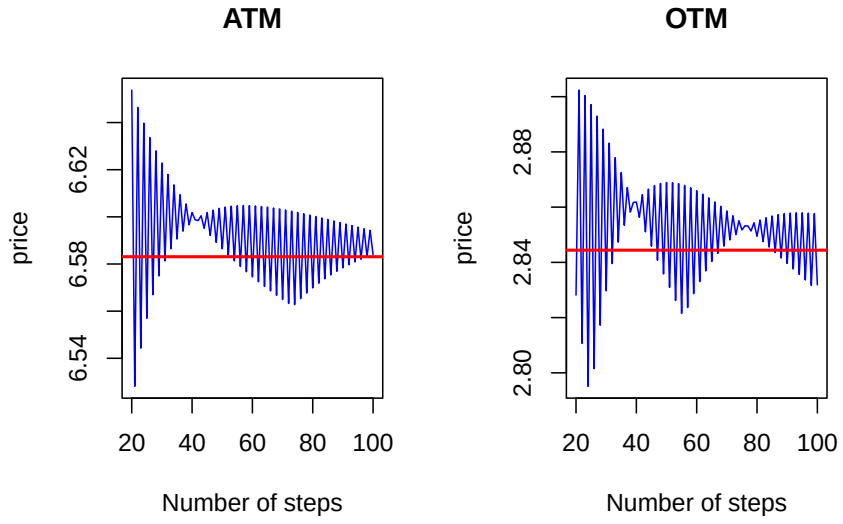
```
Strike <- 100
Spot <- 100
T1 <- 1/4
r <- 0.05
b <- 0.05
sigma <- 0.3

NN <- seq(20, 100, 1)
nb <- length(NN)
res <- matrix(nrow = nb, ncol = 2)
bs <- rep(0, 2)

# The Black-Scholes price
bs[1] <- GBSOption(TypeFlag = "c", S = Spot, X = Strike,
    Time = T1, r = r, b = b, sigma = sigma)@price
bs[2] <- GBSOption(TypeFlag = "c", S = Spot, X = Strike +
    10, Time = T1, r = r, b = b, sigma = sigma)@price

# Binomial price, function of number of
# steps
res[, 1] <- sapply(NN, function(n) CRRBinomialTreeOption(TypeFlag = "ce",
    S = Spot, X = Strike, Time = T1, r = r, b = b,
    sigma = sigma, n)@price)

res[, 2] <- sapply(NN, function(n) CRRBinomialTreeOption(TypeFlag = "ce",
    S = Spot, X = Strike + 10, Time = T1, r = r,
    b = b, sigma = sigma, n)@price)
```

A plot of the prices as a function of the number of steps (Figure **??** shows an oscillating pattern:

```
par(mfrow = c(1, 2))
plot(NN, res[, 1], type = "l", main = "ATM", xlab = "Number of steps",
    ylab = "price", col = "blue", ylim = c(min(res[,
        1]), max(res[, 1])))
abline(h = bs[1], lwd = 2, col = "red")

plot(NN, res[, 2], type = "l", main = "OTM", xlab = "Number of steps",
    ylab = "price", col = "blue", ylim = c(min(res[,
        2]), max(res[, 2])))
abline(h = bs[2], lwd = 2, col = "red")
par(mfrow = c(1, 1))
```



Other binomial algorithms, such as Tian's, exhibit a similar pattern, as evidenced in Figure **??**. The horizontal line marks the Black-Scholes price.



This sawtooth pattern is due to the position of the strike relative to

the sequence of nodes at expiry; we describe below some computational strategies for smoothing these oscillations and speeding up the convergence of binomial trees. See Joshi (2007) for an extensive survey of binomial models with improved convergence properties.

Since the oscillations are caused by the variations in the relative position of the strike with respect to nodes at expiry, a natural strategy, introduced by Leisen and Reimer (1996), is to construct the tree such that the strike coincides with a node. This is achieved by setting

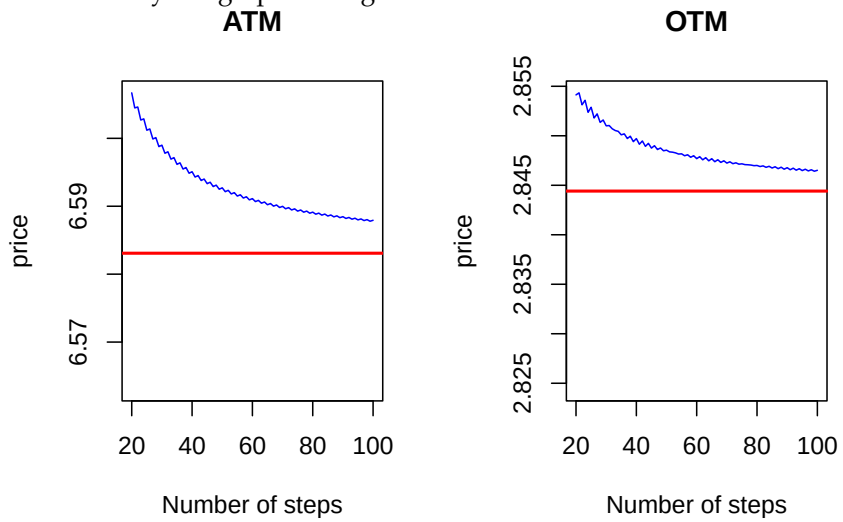
$$\mu = \frac{1}{T} \log\left(\frac{K}{S_0}\right)$$

The resulting tree is centered on K in log space. The pricing method is implemented as follows:

```
CRRWithDrift <- function(TypeFlag = c("ce", "pe",
    "ca", "pa"), S, X, Time, r, mu, sigma, n) {
    TypeFlag = TypeFlag[1]
    z = NA
    if (TypeFlag == "ce" || TypeFlag == "ca")
        z = +1
    if (TypeFlag == "pe" || TypeFlag == "pa")
        z = -1
    if (is.na(z))
        stop("TypeFlag misspecified: ce|ca|pe|pa")
    dt = Time/n
    u = exp(mu * dt + sigma * sqrt(dt))
    d = exp(mu * dt - sigma * sqrt(dt))

    p = (exp(r * dt) - d)/(u - d)
    Df = exp(-r * dt)

    # underlying asset at step N-1
    ST <- S * (d^(n - 1)) * cumprod(c(1, rep((u/d),
        n - 1)))
    # at step (n-1), value an European
    # option of maturity dt
    BSTypeFlag <- substr(TypeFlag, 1, 1)
    OptionValue <- GBSOption(BSTypeFlag, ST, X,
        dt, r, b, sigma)@price

    if (TypeFlag == "ce" || TypeFlag == "pe") {
        for (j in seq(from = n - 2, to = 0, by = -1)) OptionValue <- (p *
            OptionValue[2:(j + 2)] + (1 - p) *
            OptionValue[1:(j + 1)]) * Df
    }
```

```
    if (TypeFlag == "ca" || TypeFlag == "pa") {
        for (j in seq(from = n - 2, to = 0, by = -1)) {
            ContValue <- (p * OptionValue[2:(j +
                2)] + (1 - p) * OptionValue[1:(j +
                1)]) * Df
            ST <- S * (d^j) * cumprod(c(1, rep((u/d),
                j)))
            OptionValue <- sapply(1:(j + 1), function(i) max(ST[i] -
                X, ContValue[i]))
        }
    }

    OptionValue[1]
}
```

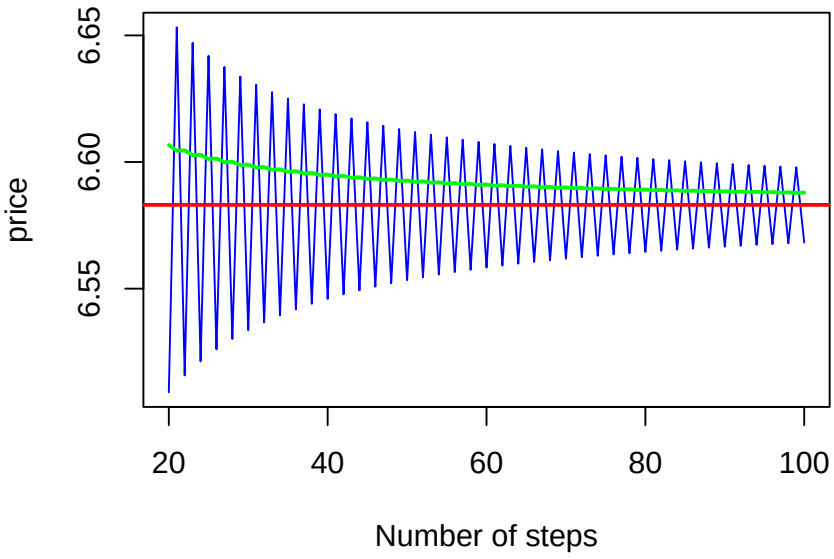
Convergence of the model as N increases is significantly improved, as evidenced by the graphs in Figure **??**.



In the context of an American option, note that if the option has not been exercised at step $N-1$, the option is now European, and can be priced at these nodes with the Black-Scholes model, rather than with the backward recursion from step N (the expiry date). This simple modification smooths the option value at step $N-1$ and cancels the oscillations, as illustrated in figure **??** but at the price of a substantial increase in computation time.

```
CRRWithBS <- function(TypeFlag = c("ce", "pe",
    "ca", "pa"), S, X, Time, r, b, sigma, n) {
    TypeFlag = TypeFlag[1]
    z = NA
    if (TypeFlag == "ce" || TypeFlag == "ca")
```

```
        z = +1
    if (TypeFlag == "pe" || TypeFlag == "pa")
        z = -1
    if (is.na(z))
        stop("TypeFlag misspecified: ce|ca|pe|pa")
    dt = Time/n
    u = exp(sigma * sqrt(dt))
    d = 1/u
    p = (exp(b * dt) - d)/(u - d)
    Df = exp(-r * dt)

    # underlying asset at step N-1
    ST <- S * (d^(n - 1)) * cumprod(c(1, rep((u/d),
        n - 1)))
    # at step (n-1), value an European
    # option of maturity dt
    BSTypeFlag <- substr(TypeFlag, 1, 1)
    OptionValue <- GBSOption(BSTypeFlag, ST, X,
        dt, r, b, sigma)@price

    if (TypeFlag == "ce" || TypeFlag == "pe") {
        for (j in seq(from = n - 2, to = 0, by = -1)) OptionValue <- (p *
            OptionValue[2:(j + 2)] + (1 - p) *
            OptionValue[1:(j + 1)]) * Df
    }

    if (TypeFlag == "ca" || TypeFlag == "pa") {
        for (j in seq(from = n - 2, to = 0, by = -1)) {
            ContValue <- (p * OptionValue[2:(j +
                2)] + (1 - p) * OptionValue[1:(j +
                1)]) * Df
            ST <- S * (d^j) * cumprod(c(1, rep((u/d),
                j)))
            OptionValue <- sapply(1:(j + 1), function(i) max(ST[i] -
                X, ContValue[i]))
        }
    }

    OptionValue[1]
}
```


price
6.55
6.60
6.65
20
40
60
80
100
Number of steps

8 Trinomial Models

A NATURAL EXTENSION of the binomial model is a trinomial model, that is, a model with three possible future states at each time step and current state.

8.1 The Trinomial Tree

The stock price at time t is S_t. From t to $t + \Delta t$, the stock may move up to S_u with probability p_u, down to S_d with probability p_d, or move to a middle state S_m with probability $1 - p_u - p_d$. The probabilities and future states must satisfy the following constraints:

1. The expected value of the stock at $t + \Delta t$ must be the forward price:

$$p_u S_u + p_d S_d + (1 - p_u - p_d) S_m = F = S e^{(r-\delta)\Delta t}$$

2. Variance:

$$p_u (S_u - F)^2 + p_d (S_d - F)^2 + (1 - p_u - p_d)(S_m - F)^2 = S^2 \sigma^2 \Delta t$$

The first method for constructing trinomial trees is simply to combine two steps of any binomial tree.

Recall that a CRR binomial tree is defined by:

$$\begin{aligned} u &= e^{\sigma\sqrt{\Delta t}} \\ d &= e^{-\sigma\sqrt{\Delta t}} \\ p &= \frac{e^{rt} - e^{-\sigma\sqrt{\Delta t}}}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}} \end{aligned}$$

Combining two steps at a time, we obtain a trinomial tree with

$$
\begin{aligned}
S_u &= Se^{\sigma\sqrt{2\Delta t}} \\
S_m &= S \\
S_d &= Se^{-\sigma\sqrt{2\Delta t}} \\
p_u &= \left(\frac{e^{r\Delta t} - e^{-\sigma\sqrt{\Delta t}}}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}}\right)^2 \\
p_d &= (1 - \sqrt{p_u})^2
\end{aligned}
$$

To every binomial tree corresponds a trinomial tree, obtained by aggregating two steps.

Another geometry can be defined by setting the middle node to S (~[1] p. 360):

[1]

$$
\begin{aligned}
S_m &= S \\
S_u &= S_m e^{\sigma\sqrt{3\Delta t}} \\
S_d &= S_m e^{-\sigma\sqrt{3\Delta t}} \\
p_u &= -\sqrt{\frac{\Delta t}{12\sigma^2}}\left(r - \frac{\sigma^2}{2}\right) + \frac{1}{6} \\
p_d &= \sqrt{\frac{\Delta t}{12\sigma^2}}\left(r - \frac{\sigma^2}{2}\right) + \frac{1}{6}
\end{aligned}
$$

Derivatives in the Black-Scholes Framework

9 *Pricing with Black-Scholes*

Recall from the previous section that under the risk-neutral probability, the discounted value is a martingale:

$$S_0 = e^{-rT} E^Q(S_T)$$

where S_T is a log-normal variable that follows the process:

$$S_T = S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}Y}$$

with Y $N(0,1)$. In a seminal paper, F. Black and M. Scholes provided an arbitrage-free analytical expression for the value of a European option.

The value of any derivative is the discounted expected payoff. For a call option, we have:

$$\begin{aligned} c(S,0) = & \quad e^{-rT} E^Q[\left(S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}Y} - K\right)^+] \\ = & \quad \frac{e^{-rT}}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}Y} - K\right)^+ e^{-x^2/2} dx \end{aligned}$$

Now compute the bounds of integration:

$$\begin{aligned} & S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}x} \geq K \\ <=> \quad & e^{\sigma\sqrt{T}x} \geq \frac{K}{S_0} e^{-(r-\frac{\sigma^2}{2})T} \\ <=> \quad & x \geq \frac{1}{\sigma\sqrt{T}} \left(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T\right) \end{aligned}$$

Let

$$T_1 = \frac{1}{\sigma\sqrt{T}} \left(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T\right)$$

$$\begin{aligned} c(S,0) = & \frac{e^{-rT}}{\sqrt{2\pi}} \int_{T_1}^{\infty} \left(S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}x} - K \right) e^{-x^2/2} dx \\ = & \frac{e^{-rT}}{\sqrt{2\pi}} \int_{T_1}^{\infty} S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}x} e^{-x^2/2} dx - K \frac{e^{-rT}}{\sqrt{2\pi}} \int_{T_1}^{\infty} e^{-x^2/2} dx \\ = & A - B \end{aligned}$$

$$B = Ke^{-rT}(1 - N(T_1))$$

Apply the change of variable $y = x - \sigma\sqrt{T}$ to get

$$\begin{aligned} A = & \frac{e^{-rT}}{\sqrt{2\pi}} \int_{T_1-\sigma\sqrt{T}}^{\infty} S_0 e^{-y^2/2} dy \\ = & S_0(1 - N(T_1 - \sigma\sqrt{T})) \end{aligned}$$

where $N(x)$ is the cumulative normal distribution. Using the identity $N(x) + N(-x) = 1$, one gets the Black-Scholes formula in its standard form.

$$c(S,0) = S_0 N(d_1) - Ke^{-rT} N(d_2)$$

where

$$\begin{aligned} d_1 = & \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \\ d_2 = & d_1 - \sigma\sqrt{T}. \end{aligned}$$

The value of the put can be obtained from the call-put parity relationships.

9.1 Relationships Among Greeks

See standard text for derivation of greeks. Recall the Black-Scholes pricing equation:

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf$$

This is true of any derivative, and in particular of a delta hedged portfolio. Set:

$$\begin{aligned} \Theta = & \frac{\partial f}{\partial t} \\ \Gamma = & \frac{\partial^2 f}{\partial S^2} \end{aligned}$$

then,

$$\Theta + \frac{1}{2}\sigma^2 S^2 \Gamma = rf$$

10 Applicability of the Black-Scholes Model

In this section, we investigate the effectiveness of various hedging strategies in the Black-Scholes framework. Indeed, the ultimate measure of risk is the quality of the dynamic replicating strategy, and the distribution of the residual wealth after expiry of the option. Throughout this section, we place ourselves in the situation of a financial institution that sell a derivative and dynamically hedges the risk.

10.1 Self-Financing Replicating Portfolio

Consider the self-financing replicating portfolio defined by:

$$V_t = \delta_t S_t + B_t$$

The auto-financing condition defines the dynamic of the bank account

$$e^{r(t_i - t_{i-1})} B_{t_{i-1}} + \delta_{t_{i-1}} S_{t_i} = B_{t_i} + \delta_{t_i} S_{t_i}$$

$$V_{t_i} - V_{t_{i-1}} = (e^{r(t_i - t_{i-1})} - 1)(V_{t_{i-1}} - \delta_{t_{i-1}} S_{t_{i-1}}) + \delta_{t_{i-1}} (S_{t_i} - S_{t_{i-1}})$$

As $t_i - t_{i-1} = \Delta t \to 0$,

$$dV_t = (V_t - \delta_t S_t) r dt + \delta_t dS_t$$

10.2 Hedging Error due to Discrete Hedging

We first consider the risk related to discreet hedging.

The dynamic of the replicating portfolio, V_t rebalanced at times $t_i, i = 1, \ldots, N$. For $t \in [t_i, t_{i+1})$ is:

$$dV_t = \frac{\partial C(t_i, S_{t_i})}{\partial S} dS_t$$

The dynamic of the option price is:

$$dC_t = \frac{\partial C(t, S_t)}{\partial S} dS_t$$

The hedging error process $\epsilon_t = C_t - V_t$ is therefore:

$$d\epsilon_t = \left(\frac{\partial C(t, S_t)}{\partial S} - \frac{\partial C(t_i, S_{t_i})}{\partial S} \right) dS_t$$

It can be shown that:

$$\lim_{h \to 0} E^Q \left[\frac{\epsilon_T^2}{\Delta t} \right] = E^Q \left[\frac{1}{2} \int_0^T \left(\frac{\partial^2 C(S_t, t)}{\partial S_t^2} \right)^2 S_t^4 \sigma^4 dt \right]$$

It can also be shown that the process ϵ_t converges in distribution to a process that has the following expression:

$$\frac{\epsilon_t}{\sqrt{\Delta t}} \to \frac{1}{\sqrt{2}} \int_0^t \frac{\partial^2 C(S_u, u)}{\partial S^2} \sigma^2 S_u^2 dZ_u$$

This expression highlights the following points:

1. The variance is inversely related to the hedging frequency
2. The variance is directly related to the magnitude of Γ

10.3 Error due to Unknown Volatility

To simplify the notation, assume now that hedging is done continuously, but that the volatility of the underlying asset is unknown. The dynamic of the underlying asset is:

$$dS_t = \mu dt + \sigma_t dW_t$$

The derivative is priced with an estimated volatility Σ.
The dynamic of the hedge portfolio is:

$$dV_t = \frac{\partial C_t^\Sigma}{\partial S_t} dS_t$$

By Itô's lemma:

$$dC(S_t, t) = \left(\frac{\partial C}{\partial t} + \frac{1}{2} \frac{\partial^2 C}{\partial S^2} \sigma_t^2 S_t^2 \right) dt + \frac{\partial C}{\partial S} dS_t$$

The price of the derivative verifies the Black-Scholes EDP:

$$\frac{\partial C}{\partial t} + \frac{1}{2} \Sigma^2 S_t^2 \frac{\partial^2 C}{\partial S^2} = 0$$

this yields,

$$d\epsilon_t = dV_t - dC_t = \frac{1}{2} \left[\Sigma^2 - \sigma_t^2 \right] \frac{\partial^2 C}{\partial S^2} S^2 dt$$

and the hedging error at expiry is thus,

$$\epsilon_T = \frac{1}{2}\int_0^T \left[\Sigma^2 - \sigma_t^2\right] \frac{\partial^2 C}{\partial S^2} S^2 dt$$

We identify three components in the above formulae:

1. the nature of the option, characterized by its gamma.
2. the behavior of the market, characterized by σ_t,
3. the model calibration, summarized here by the Black-Scholes volatility Σ

If the gamma keeps a constant sign, we can compute the fair value of the Black-Scholes volatility that sets the expected hedging error to 0 is

$$\sigma_{BS}^2 = \frac{E\int_0^T \Gamma S^2 \frac{\Delta S}{S}^2}{E\int_0^T \Gamma_{BS} S^2 dt}$$

Going back to our study of various exotic options, we can now determine in which case the basic Black-Scholes model is adapted or not:

Table 10.1: Fitness of the Black-Scholes Model

Payoff	Is BS Adapted
Vanilla	Yes
European Binary	No
Up and Out Barrier	No
Asian	Yes

10.4 Second-Order Hedging

In the previous section, we have determined that hedging error is determined by three factors:

1. the hedging frequency
2. the magnitude of the option gamma
3. the difference between Σ (the BS volatility used for pricing and hedging) and the volatility that is actually experienced, σ_t.

In this section, we investigate strategies for controlling the risk associated with the magnitude of the gamma.

10.4.1 Gamma Smoothing

The idea is to use a static hedge in order to reduce the gamma of the position that is hedged dynamically.

At $t = 0$, sell the option at price $C(S_0, T, \Sigma)$, buy a quantity γ_0 of another option G to smooth the gamma of the position:

The dynamic delta hedge is then applied to the portfolio:

$$-C_t + \gamma_0 G_t$$

Example: partial static hedge of digital option.

10.4.2 Dynamic Delta-Gamma hedge

Most of the time, a static gamma smoothing is not available, so the gamma smoothing strategy must be executed dynamically: At each time t, construct a replicating portfolio:

$$V_t = \gamma_t G_t + \delta_t S_t + \alpha_t B_t$$

with:

$$\begin{aligned} \gamma_t &= \frac{\partial^2 C_t}{\partial S^2} \Big/ \frac{\partial^2 G_t}{\partial S^2} \\ \delta_t &= \frac{\partial C_t}{\partial S} - \gamma_t \frac{\partial G_t}{\partial S} \end{aligned}$$

Illustration: Residual risk of delta hedge vs. delta-gamma hedge.

We can summarize this study of the Black-Scholes model as follows:

1. assuming a delta hedged position, the residual risk is a function of the gamma of the derivative instrument
2. when the gamma does not change sign, the Black-Scholes variance is the average actual variance, weighted by actual gamma.
3. when the gamma changes sign, the Black-Scholes model is no longer an appropriate risk-management tool. The gamma of the position should be smoothed with a partial static hedge before using the Black-Scholes model for risk management.

11 *Price and Greeks in the Black-Scholes Model*

```
## Loading required package: timeDate

## Loading required package: timeSeries

## Loading required package: fBasics

## Loading required package: lubridate

## Loading required package: timechange

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

## Loading required package: xtable

##
## Attaching package: 'xtable'
```

```
## The following objects are masked from 'package:Hmisc':
##
##     label, label<-

## The following object is masked from 'package:timeSeries':
##
##     align

## The following object is masked from 'package:timeDate':
##
##     align

## Loading required package: empfin

## Loading required package: fImport

## Loading required package: RCurl
```

In this chapter, we consider the price and risk indicators of several types of options. We are particularly interested in the higher order risk indicators, such as the Gamma. We will see later that this indicator has a major impact on our ability to dynamically hedge such instruments.

11.1 Solving the Pricing Equation

A fairly rigorous derivation of the Black-Scholes equation can be obtained without using the tools of stochastic calculus. Recall from the previous section that under the risk-neutral probability, the discounted value is a martingale:

$$S_0 = e^{-rT} E^Q(S_T)$$

where S_T is a log-normal variable that follows the process:

$$S_T = S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}Y}$$

with Y $N(0,1)$. The value of any derivative is the discounted expected payoff. For a call option, we have:

$$\begin{aligned} c(S,0) = & \quad e^{-rT} E^Q[\left(S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}Y} - K\right)^+] \\ = & \quad \frac{e^{-rT}}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}Y} - K\right)^+ e^{-x^2/2} dx \end{aligned}$$

Now compute the bounds of integration:

$$
\begin{aligned}
& S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}x} \geq K \\
<=> \quad & e^{\sigma\sqrt{T}x} \geq \frac{K}{S_0} e^{-(r-\frac{\sigma^2}{2})T} \\
<=> \quad & x \geq \frac{1}{\sigma\sqrt{T}} \left(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T \right)
\end{aligned}
$$

Let

$$
T_1 = \frac{1}{\sigma\sqrt{T}} \left(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T \right)
$$

$$
\begin{aligned}
c(S,0) = & \frac{e^{-rT}}{\sqrt{2\pi}} \int_{T_1}^{\infty} \left(S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}x} - K \right) e^{-x^2/2} dx \\
= & \frac{e^{-rT}}{\sqrt{2\pi}} \int_{T_1}^{\infty} S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}x} e^{-x^2/2} dx - K \frac{e^{-rT}}{\sqrt{2\pi}} \int_{T_1}^{\infty} e^{-x^2/2} dx \\
= & A - B
\end{aligned}
$$

$$
B = K e^{-rT} (1 - N(T_1))
$$

Apply the change of variable $y = x - \sigma\sqrt{T}$ to get

$$
\begin{aligned}
A = & \frac{e^{-rT}}{\sqrt{2\pi}} \int_{T_1-\sigma\sqrt{T}}^{\infty} S_0 e^{-y^2/2} dy \\
= & S_0 (1 - N(T_1 - \sigma\sqrt{T}))
\end{aligned}
$$

where $N(x)$ is the cumulative normal distribution. Using the identity $N(x) + N(-x) = 1$, one gets the Black-Scholes formula in its standard form.

$$
C_{\mathrm{BS}}(S,0) = S_0 N(d_1) - K e^{-rT} N(d_2)
$$

with

$$
\begin{aligned}
d_1 = & \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \\
d_2 = & d_1 - \sigma\sqrt{T}.
\end{aligned}
$$

The value of the put can be obtained from the call-put parity relationships.

In the following discussion about the management of market risk of option portfolios, we use three risk indicators, also known as greeks:

- Delta, which is the derivative of price with respect to the price of the underlying asset,
- Gamma, which is the second derivative of price with respect to spot, and

- Vega, which is the derivative of price with respect to the volatility.

See, for example Hull

[@*Hull*1997]

for a detailed discussion of these risk indicators.

11.2 *An Option Dashboard*

We start the chapter by constructing a "dashboard" which illustrates the properties of various types of European options. These displays are inspired by similar figures developped by D. Eddelbuettel

[@*Eddelbuettela*]

for the RQuantLib project. The class provides a generic way of writing such function.

The dashboard has four panels, which display the price, delta, gamma and vega of the option under consideration. Each panel has a family of curves varying in color from yellow to green to blue. This family of curves pictures the evolution of the price and of the greeks for varying levels of volatility. The yellow curves corresponding to a high volatility conditions, and at the other end, the blue line corresponding to low volatility conditions.

```
OptionDashBoard <- function(inst, base.env, dtCalc,
    und.seq, vol.seq, trace = FALSE) {
    sce <- t(as.matrix(und.seq))
    underlying <- inst@params$underlying
    setData(base.env, underlying, "Price", dtCalc,
        sce)

    p <- matrix(nrow = length(und.seq), ncol = length(vol.seq))
    d <- matrix(nrow = length(und.seq), ncol = length(vol.seq))
    g <- matrix(nrow = length(und.seq), ncol = length(vol.seq))
    v <- matrix(nrow = length(und.seq), ncol = length(vol.seq))

    for (i in seq_along(vol.seq)) {
        setData(base.env, underlying, "ATMVol",
            dtCalc, vol.seq[i])
        p[, i] <- getValue(inst, "Price", dtCalc,
            base.env)
        d[, i] <- getValue(inst, "Delta", dtCalc,
            base.env)
        g[, i] <- getValue(inst, "Gamma", dtCalc,
```

```
        base.env)
    v[, i] <- getValue(inst, "Vega", dtCalc,
        base.env)
}

if (trace) {
    print("price")
    print(p)
}

old.par <- par(no.readonly = TRUE)
par(mfrow = c(2, 2), oma = c(5, 0, 0, 0),
    mar = c(2, 2, 2, 1))

# Price
plot(und.seq, p[, 1], type = "n", main = "Price",
    xlab = "", ylab = "")
topocol <- topo.colors(length(vol.seq))
for (i in 2:length(vol.seq)) lines(und.seq,
    p[, i], col = topocol[i])

# Delta
plot(und.seq, d[, 1], type = "n", main = "Delta",
    xlab = "", ylab = "")

for (i in 2:length(vol.seq)) lines(und.seq,
    d[, i], col = topocol[i])

# Gamma
plot(und.seq, g[, 1], type = "n", main = "Gamma",
    xlab = "", ylab = "")
for (i in 2:length(vol.seq)) lines(und.seq,
    g[, i], col = topocol[i])

# Vega
plot(und.seq, v[, 1], type = "n", main = "Vega",
    xlab = "", ylab = "")
for (i in 2:length(vol.seq)) lines(und.seq,
    v[, i], col = topocol[i])

mtext(text = paste(inst@desc, "\nrate: 0.03",
    "Underlying from", und.seq[1], "to", und.seq[length(und.seq)],
    "Volatility  from", vol.seq[1], "to",
    vol.seq[length(vol.seq)]), side = 1, font = 1,
```

```
        outer = TRUE, line = 3)
    par(old.par)
}
```

11.2.1 *Vanilla Option*

European calls and puts are called "vanilla" in the banking industry, presumably due to their simple structures. The payoff of a vanilla option (call) is:

$$(S_T - K)^+$$

and its value (call) is given by:

$$e^{-rT}\left(F_T N(d_1) - K N(d_2)\right)$$

where F_T is the forward value of the underlying asset at T.

For at-the-money options, a simple approximation to the Black-Scholes price is:

$$V = e^{-rT} F_T \frac{\sigma\sqrt{T}}{\sqrt{2\pi}}$$

The formula above makes use of the approximation:

$$N(x) \approx \frac{1}{2} + \frac{x}{\sqrt{2\pi}} + O(x^2)$$

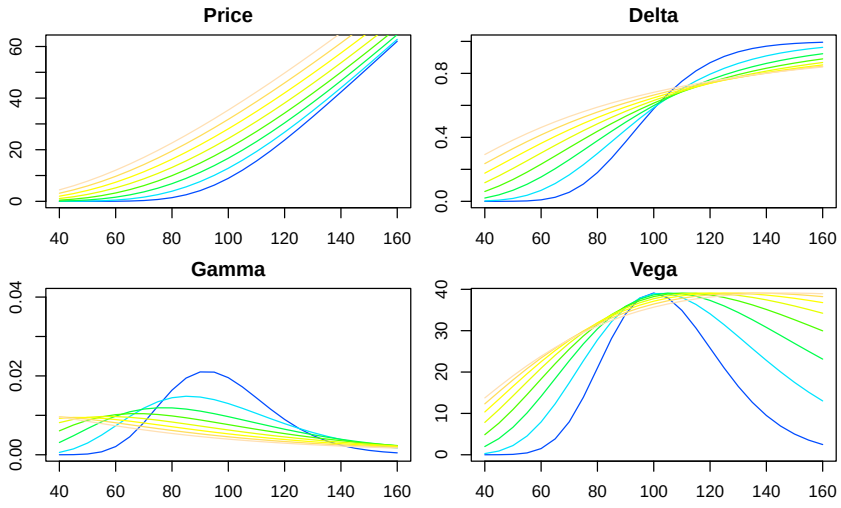
The evolution of price and greeks as a function of the underlying asset and volatility is shown in figure

$$fig : eu - call$$

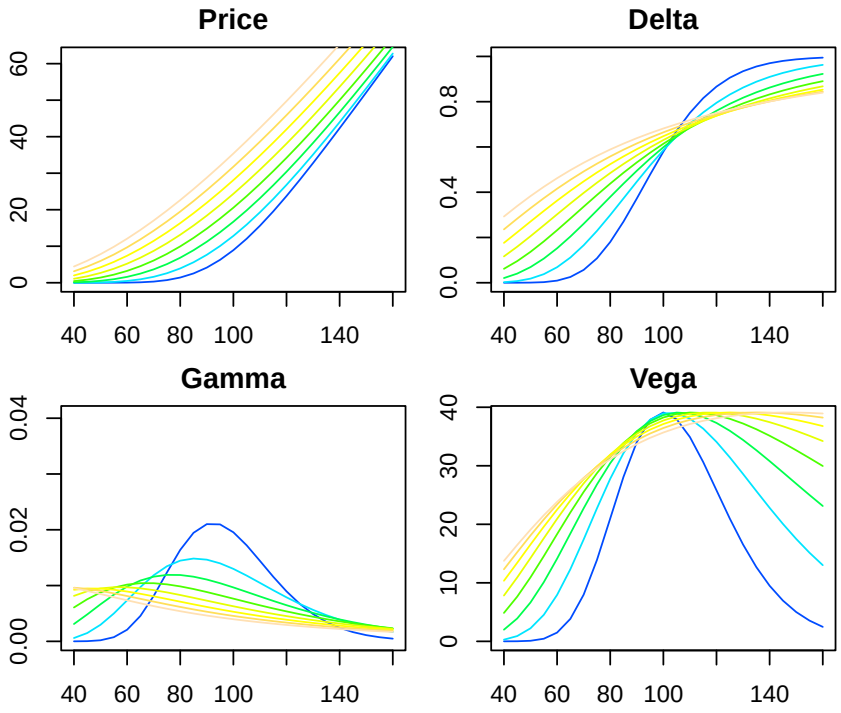
. The graph is obtained with the following script:

```
dtExpiry <- myDate("01jan2011")
underlying <- "IBM"
Strike <- 100
K <- 1

b <- fInstrumentFactory("vanilla", quantity = 1,
    params = list(cp = "c", strike = Strike, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))

# define two vectors for the underlying and
# the volatility all other market data is
# fixed
und.seq <- seq(40, 160, by = 5)
vol.seq <- seq(0.1, 0.9, by = 0.1)
```

```
base.env <- DataProvider()
dtCalc <- myDate("01jan2010")
setData(base.env, underlying, "DivYield", dtCalc,
    0.02)
setData(base.env, "USD.LIBOR", "Yield", dtCalc,
    0.02)

OptionDashBoard(b, base.env, dtCalc, und.seq,
    vol.seq)
```



Vanilla IBM c @ 100 expiry: 2011−01−01
rate: 0.03 Underlying from 40 to 160 Volatility from 0.1 to 0.9



Vanilla IBM c @ 100 expiry: 2011−01−01
rate: 0.03 Underlying from 40 to 160 Volatility from 0.1 to 0.9

Several observations are worth mentioning:

- In high volatility conditions, the delta is almost linear, but takes on more pronounced a "S" shape as volatility decrease.
- The gamma is always positive, reflecting the uniformly convex shape of the premium function.
- The vega is also always positive: an increase in volatility always results in an increase in premium.

We will see shortly that these last two features have important implications for risk management.

11.2.2 Binary Option

A binary option pays a fixed amount if the underlying asset is in the money at expiry.

The payoff is:

$$1_{S_T > K}$$

Recall that $N(d_2)$ is the risk-neutral probability that $S_T > K$ at expiry. The value of a binary call is therefore:

$$e^{-rT} N(d_2)$$

Alternatively, observe that a binary option struck at K can be replicated by a very tight call spread:

$$C_B(K) = \lim_{h \to 0} \frac{C_V(K-h) - C_V(K-h)}{h}$$

where $C_B(K)$ is the price of a binary call struk at $K$and $C_V(K)$ the price of a vanilla call. The value of a binary call is the negative of the derivative of a vanilla call price with respect to strike.

$$\begin{aligned} C_B(K) = & -\frac{\partial C_V(K)}{\partial K} \\ = & e^{-rT} N(d_2) \end{aligned}$$

Recall that the derivative of a vanilla call price with respect to spot is $N(d_1)$, therefore, the shape of the binary price function is similar to the shape of the delta of a European option, and the shape of the delta of a binary call is similar to the shape of the gamma of a vanilla option.

As a consequence, the gamma of a binary option changes sign, and so does the vega. Options with gamma that changes sign are costly to hedge, as simulations will demonstrate in section.

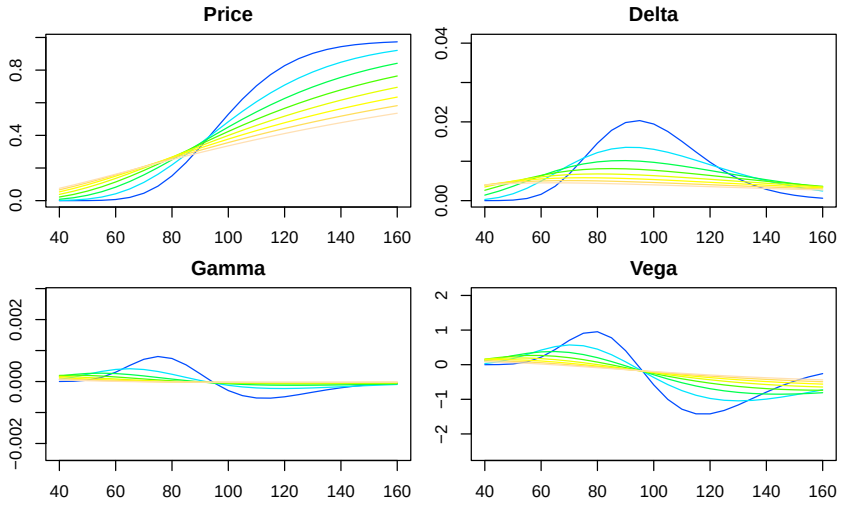
The dashboard in Figure

$$fig : eu - bin - call$$

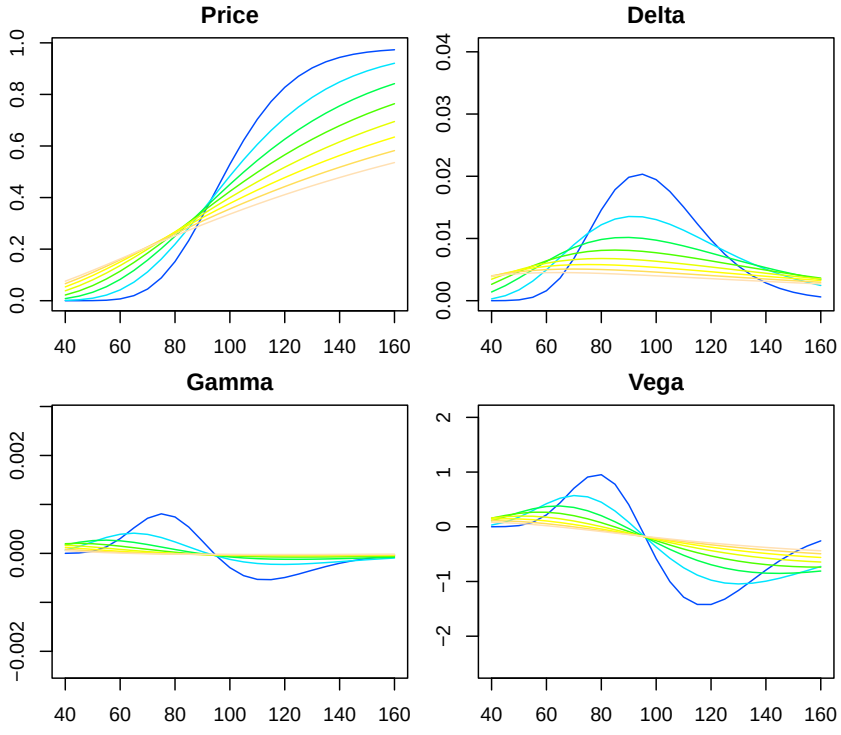
is generated with the following script:

```
dtExpiry <- myDate("01jan2011")
underlying <- "IBM"
Strike <- 100
K <- 1

bi <- fInstrumentFactory("binary", quantity = 1,
    params = list(cp = "c", strike = Strike, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))

# define two vectors for the underlying and
# the volatility all other market data is
# fixed
und.seq <- seq(40, 160, by = 5)
vol.seq <- seq(0.1, 0.9, by = 0.1)

base.env <- DataProvider()
dtCalc <- myDate("01jan2010")
setData(base.env, underlying, "DivYield", dtCalc,
```

```
    0.02)
setData(base.env, "USD.LIBOR", "Yield", dtCalc,
    0.02)

OptionDashBoard(bi, base.env, dtCalc, und.seq,
    vol.seq)
```



Binary c @ 100 expiry: 2011−01−01
rate: 0.03 Underlying from 40 to 160 Volatility from 0.1 to 0.9



Binary c @ 100 expiry: 2011−01−01
rate: 0.03 Underlying from 40 to 160 Volatility from 0.1 to 0.9

11.2.3 *Barrier Option*

There is a large variety of barrier options. Haug

$$[@Haug2006]$$

provides a long list of pricing formulae for standard and complex barrier payoffs. As an illustration we consider here the "up and out" call, which is a regular European call that is canceled if the underlying asset breaches an upper barrier before expiry.

Payoff:

$$(S_T - K)^+ 1_{\max_t S_t < B}$$

This instruments introduces a new twist in the option landscape, since this call option can have a negative delta. As for binary options, the gamma can switch sign, and finally, observe the magnitude of the vega: the maximum value, in absolute term, is twice as large as for european options of comparable strike and maturity.

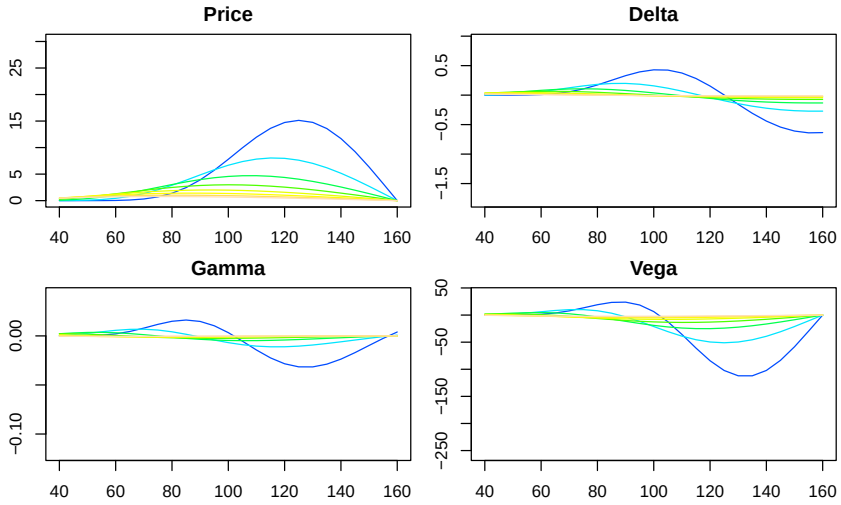
The dashboard in Figure

$$fig : eu - ba - call$$

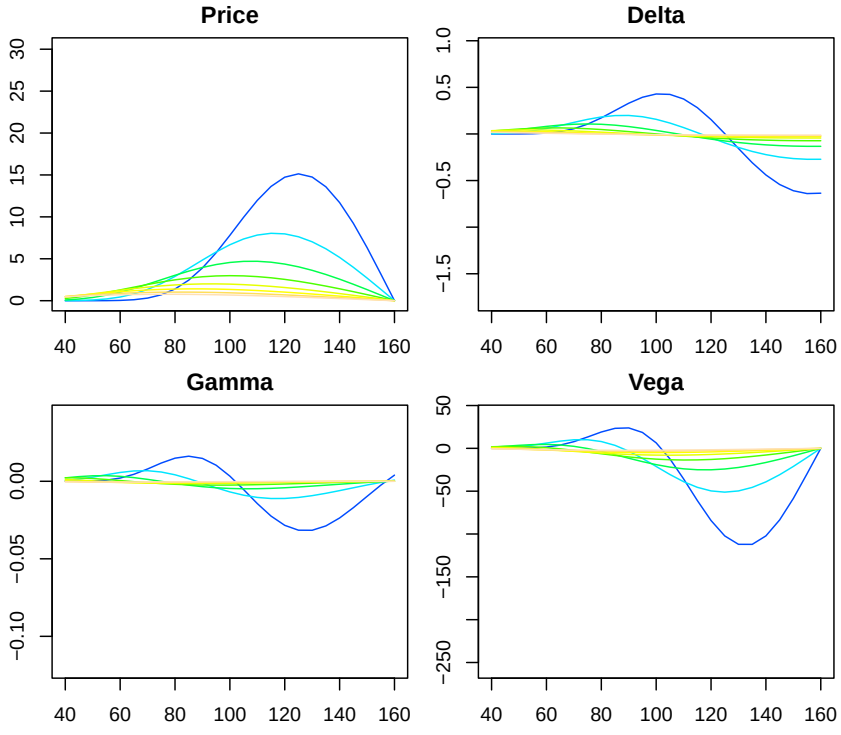
is generated with the following script:

```
dtExpiry <- myDate("01jan2011")
underlying <- "IBM"
Strike <- 100
barrier <- 160

ba <- fInstrumentFactory("standardbarrier", quantity = 1,
    params = list(cp = "cuo", strike = Strike,
        barrier = barrier, rebate = 0, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))

# define two vectors for the underlying and
# the volatility all other market data is
# fixed
und.seq <- seq(40, 160, by = 5)
vol.seq <- seq(0.1, 0.9, by = 0.1)

base.env <- DataProvider()
dtCalc <- myDate("01jan2010")
setData(base.env, underlying, "DivYield", dtCalc,
    0.02)
setData(base.env, "USD.LIBOR", "Yield", dtCalc,
```

```
    0.02)

OptionDashBoard(ba, base.env, dtCalc, und.seq,
    vol.seq)
```



Standard Barrier cuo @ 100 Barrier: 160 Rebate: 0 expiry: 2011−01−01
rate: 0.03 Underlying from 40 to 160 Volatility from 0.1 to 0.9



Standard Barrier cuo @ 100 Barrier: 160 Rebate: 0 expiry: 2011−01−01
rate: 0.03 Underlying from 40 to 160 Volatility from 0.1 to 0.9

11.2.4 *Asian Option*

An asian option is an option on the average value of the underlying asset, computed over a period of time prior to expiry. This type is very popular in commodities markets, because it captures the price risk on a market participant that produces or consumes the asset at a steady rate over time.

In spite of its apparent complexity, asian options are in fact fairly easy to manage. One can think of an Asian option as a regular European option, with an underlying asset that is the average value. The value of an asian call is lower than the value of a comparable european vanilla call, since the volatility of an average is lower than the volatility of the price at expiry. For the same reason, the vega of an Asian option is about half of the vega of its European conterpart.

The dashboard in Figure

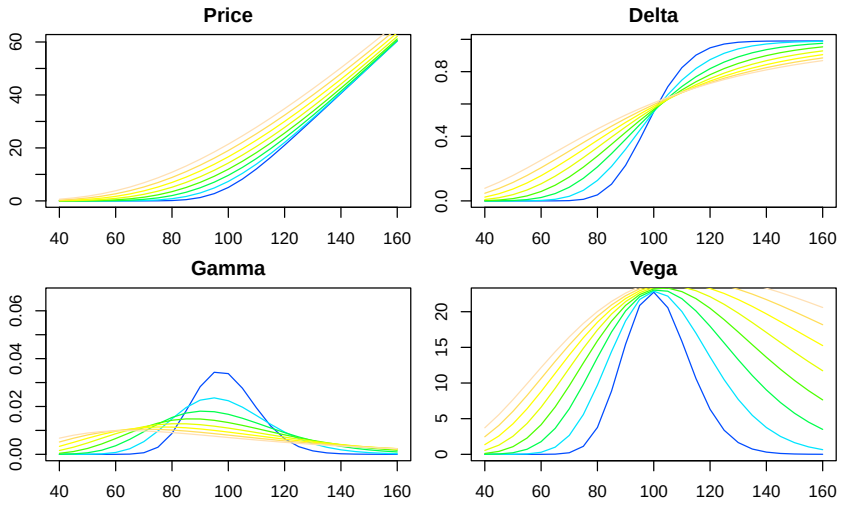
$$fig : eu - as - call$$

is generated with the following script:

```
dtExpiry <- myDate("01jan2011")
underlying <- "IBM"
Strike <- 100

ba <- fInstrumentFactory("asian", quantity = 1,
    params = list(cp = "c", strike = Strike, dtExpiry = dtExpiry,
        dtStart = dtCalc, dtEnd = dtExpiry, underlying = underlying,
        discountRef = "USD.LIBOR", trace = FALSE))

# define two vectors for the underlying and
# the volatility all other market data is
# fixed
und.seq <- seq(40, 160, by = 5)
vol.seq <- seq(0.1, 0.9, by = 0.1)

base.env <- DataProvider()
dtCalc <- myDate("01jan2010")
setData(base.env, underlying, "DivYield", dtCalc,
    0.02)
setData(base.env, "USD.LIBOR", "Yield", dtCalc,
    0.02)

OptionDashBoard(ba, base.env, dtCalc, und.seq,
    vol.seq, TRUE)

## [1] "price"
```

```
##               [,1]         [,2]         [,3]         [,4]         [,5]
##  [1,] 0.000000e+00 2.797267e-15 1.929820e-07 1.676725e-04  0.004835215
##  [2,] 0.000000e+00 5.554081e-12 7.049604e-06 1.414773e-03  0.020537644
##  [3,] 0.000000e+00 2.457959e-09 1.228438e-04 7.848277e-03  0.066480524
##  [4,] 0.000000e+00 3.085715e-07 1.221308e-03 3.167283e-02  0.175041555
##  [5,] 0.000000e+00 1.458436e-05 7.863270e-03 9.988736e-02  0.392541985
##  [6,] 1.093153e-13 3.192058e-04 3.596464e-02 2.593626e-01  0.775341243
##  [7,] 8.008437e-10 3.780066e-03 1.252489e-01 5.765250e-01  1.383044255
##  [8,] 7.712348e-07 2.730738e-02 3.502590e-01 1.130251e+00  2.270945048
##  [9,] 1.429466e-04 1.322786e-01 8.198778e-01 1.999979e+00  3.483681688
## [10,] 6.907294e-03 4.636100e-01 1.660142e+00 3.253068e+00  5.051276086
## [11,] 1.125566e-01 1.251008e+00 2.985769e+00 4.935257e+00  6.987832736
## [12,] 7.770427e-01 2.736296e+00 4.873107e+00 7.066310e+00  9.292515837
## [13,] 2.802681e+00 5.064821e+00 7.345783e+00 9.640811e+00 11.951865448
## [14,] 6.409656e+00 8.224511e+00 1.037587e+01 1.263239e+01 14.942892256
## [15,] 1.096881e+01 1.207104e+01 1.389739e+01 1.600031e+01 18.236455170
## [16,] 1.584710e+01 1.640467e+01 1.782418e+01 1.969573e+01 21.799974968
## [17,] 2.078882e+01 2.104062e+01 2.206648e+01 2.366749e+01 25.599967634
## [18,] 2.573846e+01 2.584361e+01 2.654238e+01 2.786625e+01 29.603825049
## [19,] 3.068876e+01 3.072996e+01 3.118381e+01 3.224716e+01 33.781014330
## [20,] 3.563909e+01 3.565435e+01 3.593816e+01 3.677130e+01 38.103811303
## [21,] 4.058942e+01 4.059479e+01 4.076701e+01 4.140610e+01 42.547654136
## [22,] 4.553975e+01 4.554156e+01 4.564358e+01 4.612506e+01 47.091214686
## [23,] 5.049008e+01 5.049067e+01 5.054994e+01 5.090718e+01 51.716277174
## [24,] 5.544041e+01 5.544060e+01 5.547450e+01 5.573615e+01 56.407497967
## [25,] 6.039075e+01 6.039080e+01 6.040994e+01 6.059951e+01 61.152102719
##              [,6]       [,7]       [,8]       [,9]
##  [1,]  0.03480616  0.1267117  0.3156147  0.6239206
##  [2,]  0.10121228  0.2918643  0.6227507  1.1041871
##  [3,]  0.24293612  0.5821899  1.0977431  1.7834416
##  [4,]  0.50376139  1.0402770  1.7745638  2.6846818
##  [5,]  0.93201347  1.7053818  2.6800762  3.8230890
##  [6,]  1.57498535  2.6101067  3.8331392  5.2066461
##  [7,]  2.47400687  3.7785399  5.2447380  6.8371745
##  [8,]  3.66102069  5.2257150  6.9187684  8.7115201
##  [9,]  5.15690968  6.9580627  8.8531635 10.8227185
## [10,]  6.97138399  8.9744954 11.0411348 13.1610503
## [11,]  9.10402282 11.2678131 13.4723744 15.7148985
## [12,] 11.54600354 13.8261298 16.1340354 18.4714647
## [13,] 14.28193766 16.6341983 19.0117344 21.4174042
## [14,] 17.29194560 19.6746992 22.0902349 24.5392082
## [15,] 20.55337745 22.9291544 25.3540045 27.8235366
## [16,] 24.04229320 26.3788177 28.7876870 31.2574540
## [17,] 27.73457010 30.0051604 32.3764300 34.8286327
```

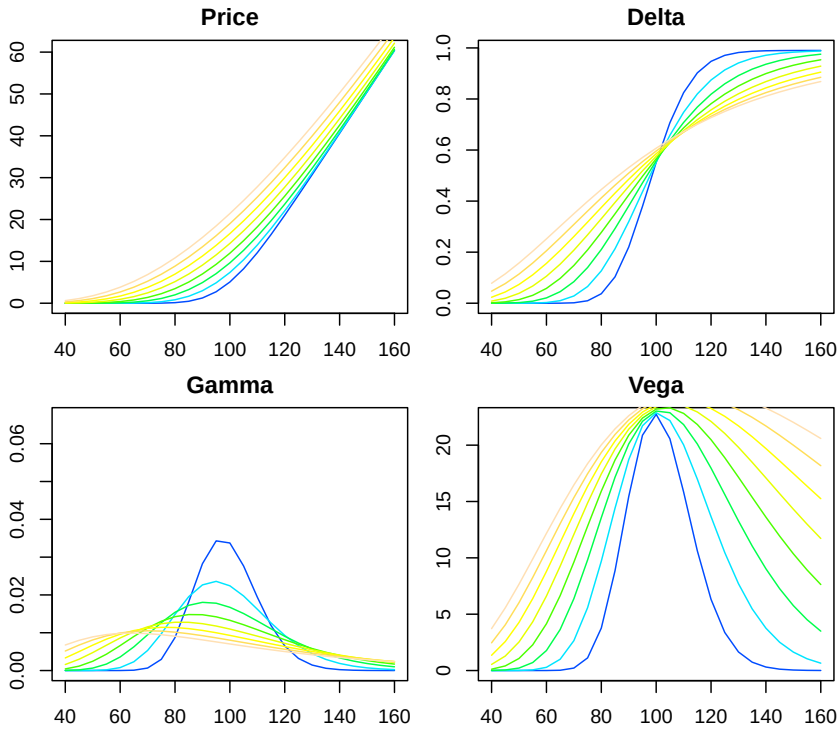
```
## [18,] 31.60683215 33.7903670 36.1060349 38.5254089
## [19,] 35.63701167 37.7176264 39.9631518 42.3368376
## [20,] 39.80470827 41.7713001 43.9353497 46.2527415
## [21,] 44.09138129 45.9370103 48.0111454 50.2637062
## [22,] 48.48041400 50.2016664 52.1800019 54.3610623
## [23,] 52.95708545 54.5534465 56.4323043 58.5368549
## [24,] 57.50847966 58.9817460 60.7593200 62.7838078
## [25,] 62.12335609 63.4771065 65.1531485 67.0952819
```



Asian c @ 100 expiry: 2011−01−01
rate: 0.03 Underlying from 40 to 160 Volatility from 0.1 to 0.9

```
## [1] "price"
##               [,1]         [,2]         [,3]         [,4]         [,5]
##  [1,] 0.000000e+00 2.797267e-15 1.929820e-07 1.676725e-04  0.004835215
##  [2,] 0.000000e+00 5.554081e-12 7.049604e-06 1.414773e-03  0.020537644
##  [3,] 0.000000e+00 2.457959e-09 1.228438e-04 7.848277e-03  0.066480524
##  [4,] 0.000000e+00 3.085715e-07 1.221308e-03 3.167283e-02  0.175041555
##  [5,] 0.000000e+00 1.458436e-05 7.863270e-03 9.988736e-02  0.392541985
##  [6,] 1.093153e-13 3.192058e-04 3.596464e-02 2.593626e-01  0.775341243
##  [7,] 8.008437e-10 3.780066e-03 1.252489e-01 5.765250e-01  1.383044255
##  [8,] 7.712348e-07 2.730738e-02 3.502590e-01 1.130251e+00  2.270945048
##  [9,] 1.429466e-04 1.322786e-01 8.198778e-01 1.999979e+00  3.483681688
## [10,] 6.907294e-03 4.636100e-01 1.660142e+00 3.253068e+00  5.051276086
## [11,] 1.125566e-01 1.251008e+00 2.985769e+00 4.935257e+00  6.987832736
## [12,] 7.770427e-01 2.736296e+00 4.873107e+00 7.066310e+00  9.292515837
## [13,] 2.802681e+00 5.064821e+00 7.345783e+00 9.640811e+00 11.951865448
## [14,] 6.409656e+00 8.224511e+00 1.037587e+01 1.263239e+01 14.942892256
## [15,] 1.096881e+01 1.207104e+01 1.389739e+01 1.600031e+01 18.236455170
## [16,] 1.584710e+01 1.640467e+01 1.782418e+01 1.969573e+01 21.799974968
## [17,] 2.078882e+01 2.104062e+01 2.206648e+01 2.366749e+01 25.599967634
```

```
## [18,] 2.573846e+01 2.584361e+01 2.654238e+01 2.786625e+01 29.603825049
## [19,] 3.068876e+01 3.072996e+01 3.118381e+01 3.224716e+01 33.781014330
## [20,] 3.563909e+01 3.565435e+01 3.593816e+01 3.677130e+01 38.103811303
## [21,] 4.058942e+01 4.059479e+01 4.076701e+01 4.140610e+01 42.547654136
## [22,] 4.553975e+01 4.554156e+01 4.564358e+01 4.612506e+01 47.091214686
## [23,] 5.049008e+01 5.049067e+01 5.054994e+01 5.090718e+01 51.716277174
## [24,] 5.544041e+01 5.544060e+01 5.547450e+01 5.573615e+01 56.407497967
## [25,] 6.039075e+01 6.039080e+01 6.040994e+01 6.059951e+01 61.152102719
##              [,6]       [,7]       [,8]       [,9]
##  [1,]  0.03480616  0.1267117  0.3156147  0.6239206
##  [2,]  0.10121228  0.2918643  0.6227507  1.1041871
##  [3,]  0.24293612  0.5821899  1.0977431  1.7834416
##  [4,]  0.50376139  1.0402770  1.7745638  2.6846818
##  [5,]  0.93201347  1.7053818  2.6800762  3.8230890
##  [6,]  1.57498535  2.6101067  3.8331392  5.2066461
##  [7,]  2.47400687  3.7785399  5.2447380  6.8371745
##  [8,]  3.66102069  5.2257150  6.9187684  8.7115201
##  [9,]  5.15690968  6.9580627  8.8531635 10.8227185
## [10,]  6.97138399  8.9744954 11.0411348 13.1610503
## [11,]  9.10402282 11.2678131 13.4723744 15.7148985
## [12,] 11.54600354 13.8261298 16.1340354 18.4714647
## [13,] 14.28193766 16.6341983 19.0117344 21.4174042
## [14,] 17.29194560 19.6746992 22.0902349 24.5392082
## [15,] 20.55337745 22.9291544 25.3540045 27.8235366
## [16,] 24.04229320 26.3788177 28.7876870 31.2574540
## [17,] 27.73457010 30.0051604 32.3764300 34.8286327
## [18,] 31.60683215 33.7903670 36.1060349 38.5254089
## [19,] 35.63701167 37.7176264 39.9631518 42.3368376
## [20,] 39.80470827 41.7713001 43.9353497 46.2527415
## [21,] 44.09138129 45.9370103 48.0111454 50.2637062
## [22,] 48.48041400 50.2016664 52.1800019 54.3610623
## [23,] 52.95708545 54.5534465 56.4323043 58.5368549
## [24,] 57.50847966 58.9817460 60.7593200 62.7838078
## [25,] 62.12335609 63.4771065 65.1531485 67.0952819
```



Asian c @ 100 expiry: 2011−01−01
rate: 0.03 Underlying from 40 to 160 Volatility from 0.1 to 0.9

12 Dynamic Hedging with the Black-Scholes Model

In this section, we investigate the effectiveness of various hedging strategies in the Black-Scholes framework. Indeed, the ultimate measure of risk is the quality of the dynamic replicating strategy, and the distribution of the residual wealth after expiry of the option. Throughout this section, we place ourselves in the situation of a financial institution that sell a derivative and dynamically hedges the risk.

12.1 Self-Financing Replicating Portfolio

Consider the self-financing replicating portfolio defined by:

$$V_t = \delta_t S_t + B_t$$

The auto-financing condition defines the dynamic of the bank account

$$e^{r(t_i - t_{i-1})} B_{t_{i-1}} + \delta_{t_{i-1}} S_{t_i} = B_{t_i} + \delta_{t_i} S_{t_i}$$

$$V_{t_i} - V_{t_{i-1}} = (e^{r(t_i - t_{i-1})} - 1)(V_{t_{i-1}} - \delta_{t_{i-1}} S_{t_{i-1}}) + \delta_{t_{i-1}} (S_{t_i} - S_{t_{i-1}})$$

As $t_i - t_{i-1} = \Delta t \to 0$,

$$dV_t = (V_t - \delta_t S_t) r dt + \delta_t dS_t$$

12.2 Hedging Error due to Discrete Hedging

We first consider the risk related to discreet hedging.

The dynamic of the replicating portfolio, V_t rebalanced at times $t_i, i = 1, \ldots, N$. For $t \in [t_i, t_{i+1})$ is:

$$dV_t = \frac{\partial C(t_i, S_{t_i})}{\partial S} dS_t$$

The dynamic of the option price is:

$$dC_t = \frac{\partial C(t, S_t)}{\partial S} dS_t$$

The hedging error process $\epsilon_t = C_t - V_t$ is therefore:

$$d\epsilon_t = \left(\frac{\partial C(t, S_t)}{\partial S} - \frac{\partial C(t_i, S_{t_i})}{\partial S} \right) dS_t$$

It can be shown that:

$$\lim_{h \to 0} E^Q \left[\frac{\epsilon_T^2}{\Delta t} \right] = E^Q \left[\frac{1}{2} \int_0^T \left(\frac{\partial^2 C(S_t, t)}{\partial S_t^2} \right)^2 S_t^4 \sigma^4 dt \right]$$

It can also be shown that the process ϵ_t converges in distribution to a process that has the following expression:

$$\frac{\epsilon_t}{\sqrt{\Delta t}} \to \frac{1}{\sqrt{2}} \int_0^t \frac{\partial^2 C(S_u, u)}{\partial S^2} \sigma^2 S_u^2 dZ_u$$

This expression highlights the following points:

1. The variance is inversely related to the hedging frequency
2. The variance is directly related to the magnitude of Γ

12.3 Error due to Unknown Volatility

To simplify the notation, assume now that hedging is done continuously, but that the volatility of the underlying asset is unknown. The dynamic of the underlying asset is:

$$dS_t = \mu dt + \sigma_t dW_t$$

The derivative is priced with an estimated volatility Σ.
The dynamic of the hedge portfolio is:

$$dV_t = \frac{\partial C_t^\Sigma}{\partial S_t} dS_t$$

By Itô's lemma:

$$dC(S_t, t) = \left(\frac{\partial C}{\partial t} + \frac{1}{2} \frac{\partial^2 C}{\partial S^2} \sigma_t^2 S_t^2 \right) dt + \frac{\partial C}{\partial S} dS_t$$

The price of the derivative verifies the Black-Scholes EDP:

$$\frac{\partial C}{\partial t} + \frac{1}{2} \Sigma^2 S_t^2 \frac{\partial^2 C}{\partial S^2} = 0$$

this yields,

$$d\epsilon_t = dV_t - dC_t = \frac{1}{2} \left[\Sigma^2 - \sigma_t^2 \right] \frac{\partial^2 C}{\partial S^2} S^2 dt$$

and the hedging error at expiry is thus,

$$\epsilon_T = \frac{1}{2}\int_0^T \left[\Sigma^2 - \sigma_t^2\right] \frac{\partial^2 C}{\partial S^2} S^2 dt$$

We identify three components in the above formulae:

1. the nature of the option, characterized by its gamma.
2. the behavior of the market, characterized by σ_t,
3. the model calibration, summarized here by the Black-Scholes volatility Σ

If the gamma keeps a constant sign, we can compute the fair value of the Black-Scholes volatility that sets the expected hedging error to o is

$$\sigma_{BS}^2 = \frac{E\int_0^T \Gamma S^2 \frac{\Delta S}{S}^2}{E\int_0^T \Gamma_{BS} S^2 dt}$$

Going back to our study of various exotic options, we can now determine in which case the basic Black-Scholes model is adapted or not:

Table 12.1: Fitness of the Black-Scholes Model

Payoff	Is BS Adapted
Vanilla	Yes
European Binary	No
Up and Out Barrier	No
Asian	Yes

12.4 Second-Order Hedging

In the previous section, we have determined that hedging error is determined by three factors:

1. the hedging frequency
2. the magnitude of the option gamma
3. the difference between Σ (the BS volatility used for pricing and hedging) and the volatility that is actually experienced, σ_t.

In this section, we investigate strategies for controlling the risk associated with the magnitude of the gamma.

12.4.1 Gamma Smoothing

The idea is to use a static hedge in order to reduce the gamma of the position that is hedged dynamically.

At $t = 0$, sell the option at price $C(S_0, T, \Sigma)$, buy a quantity γ_0 of another option G to smooth the gamma of the position:

The dynamic delta hedge is then applied to the portfolio:

$$-C_t + \gamma_0 G_t$$

Example: partial static hedge of digital option.

12.4.2 Dynamic Delta-Gamma hedge

Most of the time, a static gamma smoothing is not available, so the gamma smoothing strategy must be executed dynamically: At each time t, construct a replicating portfolio:

$$V_t = \gamma_t G_t + \delta_t S_t + \alpha_t B_t$$

with:

$$\begin{aligned} \gamma_t &= \frac{\partial^2 C_t}{\partial S^2} \Big/ \frac{\partial^2 G_t}{\partial S^2} \\ \delta_t &= \frac{\partial C_t}{\partial S} - \gamma_t \frac{\partial G_t}{\partial S} \end{aligned}$$

Illustration: Residual risk of delta hedge vs. delta-gamma hedge.

13 *Dynamic Hedging in the Black-Scholes Framework*

```
library(fInstrument)
```

```
## Loading required package: fOptions
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```
## Loading required package: fBasics
```

```
## Loading required package: fExoticOptions
```

```
## Loading required package: fAsianOptions
```

```
## Loading required package: lubridate
```

```
## Loading required package: timechange
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(DynamicSimulation)
```

```
## Loading required package: Hmisc
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units

## Loading required package: xtable

##
## Attaching package: 'xtable'

## The following objects are masked from 'package:Hmisc':
##
##     label, label<-

## The following object is masked from 'package:timeSeries':
##
##     align

## The following object is masked from 'package:timeDate':
##
##     align

## Loading required package: empfin

## Loading required package: fImport

## Loading required package: RCurl

library(empfin)
library(gplots)

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

library(BB)
library(splines)
library(SuppDists)
library(moments)

##
## Attaching package: 'moments'

## The following objects are masked from 'package:fBasics':
##
##     kurtosis, skewness
```

```
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
```

In this chapter, we investigate the effectiveness of hedging strategies in the Black-Scholes framework, and assess the impact of departures from the assumptions of the Black-Scholes model. Indeed, the ultimate measure of risk is the quality of the dynamic replicating strategy, and the distribution of the residual wealth at expiry of the option. Throughout this chapter, we place ourselves in the situation of a financial institution that sells a derivative and dynamically hedges the risk.

13.1 Self-Financing Replicating Portfolio

13.1.1 Continuous rebalancing

Let $C(S_t, t)$ be the price of a call option at time $t, 0 \leq t \leq T$, with the spot price being S_t. We first show that we can construct a trading strategy involving the underlying asset and a bank account, which will replicate exactly the option payoff $\max(S_T - K, 0)$ at time T . This strategy is self-financing, meaning that it does not require any cash inflow, nor does it generate any cash outflow after $t = 0$. Because of this self-financing property, the value of the option must be equal to the initial value of this replicating portfolio. Otherwise, a sure profit could be gained by trading the option against the replicating portfolio. The replicating portfolio P is made of α_t shares and a notional amount β_t of risk-free bond. The portfolio value is thus:

$$P_t = \alpha_t S_t + \beta_t B_t$$

where B_t is the price of the risk-free bond at time t:

$$B_t = e^{-r(T-t)}$$

and $\alpha_t = \frac{\partial C}{\partial S}$. At $t = 0$ the portfolio has by definition the same value as the option:

$$P_0 = C(S_0, 0)$$

The residual wealth, V_t is the difference between the replicating portfolio and the option:

$$V_t = P_t - C(S_t, t)$$

We show that in fact $V_t = 0,\ 0 \leq t \leq T$. Using the definition of the hedge portfolio, we have:

$$\begin{aligned} V_t = & \frac{\partial C}{\partial S} S_t + \beta_t B_t - C(S_t, t) \\ = & \Pi_t + \beta_t B_t \beta_t B_t \end{aligned}$$

Differentiate:

$$dV_t = d\Pi_t + \beta_t dB_t$$

From the derivation of the Black-Scholes equation (see Appendix or Hull (1997)), we know that the portfolio Π_t is riskless:

$$d\Pi_t = r\Pi dt$$

therefore,

$$\begin{aligned} dV_t = & r\Pi_t dt + \beta_t r B dt \\ = & r V_t dt \end{aligned}$$

Since $V_0 = 0$, equation (13.1.1) has a unique solution which is:

$$V_t = 0,\ 0 \leq t \leq T$$

and we conclude, that portfolio P replicates the option at all time, and in particular at expiry. The dynamic of the replication is governed by the assumptions of the Black-Scholes framework,which are recalled here:

- The underlying asset follows a geometric Brownian motion process, with constant and known volatility. The actual drift does not matter.
- The interest rate is constant.
- Trading is frictionless: there are no trading costs, shares can be purchase in arbitrary fractions, trading can take place continuously.

Clearly, none of these assumptions are verified in real life, and we expect that departure from these assumptions will affect the quality of the dynamic replication described by the Black-Scholes model. The purpose of this chapter is to investigate the effect of departures from these assumptions, and address the question: how useful is the Black-Scholes model in a environment where Black-Scholes assumptions are not verified?

We will adress this question by means of simulations that are now described.

13.1.2 *Discreet rebalancing*

Since trading does not occur continuously, a replicating strategy must consist in a decision rule that is implemented at regular intervals. In the case of the delta-hedging strategy the rule will be:

1. Define a rebalancing interval Δt. Trading dates with therefore be $\Delta t, 2\Delta t, \ldots, T - \Delta t$.
2. At each trading date t, buy or sell the underlying stock so that the hedge portfolio holds a quantity of risky asset equal to $\frac{\partial C_t}{\partial S_t}$.

To describe the process step by step, we use the following notation:

C_t. Value of derivative at time t
V_t. Value of hedge portfolio
B_t. Value of a pure discount bond maturing at T
α_t. Delta of derivative, also the quantity of risky asset in the replicating portfolio
β_t. Quantity of riskless asset in hedge.

At $t = 0$, the derivative is sold at price C_0 and the proceeds are used to purchase a hedge portfolio. The initial hedge is $\alpha_0 S_0 + \beta_0 B_0$, where β_0 is computed from the acounting identity:

$$C_0 = \alpha_0 S_0 + \beta_0 B_0$$

At trading time t, the decision rule is as follows:

1. Compute the value of the hedge portfolio formed at the previous time step:

$$V_t = \alpha_{t-\Delta t} S_t + \beta_{t-\Delta t} B_t$$

2. Compute the amount of risky security to hold:

$$\alpha_t = \frac{\partial C_t}{\partial S_t}$$

3. Since the strategy must be self-financing, determine the quantity of bond to be lent or borrowed:

$$\beta_t = \frac{V_t - \alpha_t S_t}{B_t}$$

At expiry of the derivative, the bond is worth par, and the residual wealth is:

$$-C_T + \alpha_{T-\Delta t} S_T + \beta_{T-\Delta t}$$

The quality of a model is ultimately measured by the residual error: the liquidation value of the replicating portfolio, less the exercise value of the option: $V_T - C_T$.

13.1.3 Illustration

Let's illustrate this rebalancing process through a simple example that will also demonstrate the self-financing nature of the portfolio. A financial institution writes (sells) an at-the-money option on a stock that does not pay any dividend, and worth €100. The option expires in two months, and for illustrative purpose, the hedge will be rebalanced every week (in practice, of course, dynamic rebalancing is done much more often). Interest rate is 2% and volatility 30%.

```
r <- 0.02
T <- 2/12
S0 <- 100
K <- 100
sigma <- 0.3

p <- GBSOption(TypeFlag = "c", S = S0, X = K,
    Time = T, r = r, b = r, sigma)@price
delta <- GBSGreeks("delta", TypeFlag = "c", S = S0,
    X = K, Time = T, r = r, b = r, sigma)
```

The initial hedge portfolio is constructed as follows:

- The value of the hedge portfolio is the value of the derivative: $P_0 = 5.04$.
- The delta of the option is:

$$\frac{\partial C_0}{\partial S_0} = 0.54$$

  Thus the hedge portfolio must also hold $\alpha_0 = 0.54$ units of stocks, for a cost of $\alpha_0 \times 100 = 53.52$.
- this position is funded by a riskless borrowing of $\beta_0 B = P_0 - 100\alpha_0 = -48.48$.
- Since the zero-coupon bond is worth $B_0 = e^{-rT} = 0.997$, we get $\beta_0 = -48.64$.

At every time step, the portfolio is adjusted acording to the algorithm of the previous section.

This calculation is easily performed with the supplied packages:

```
dtStart <- myDate("01jun2010")
dtObs <- dtStart
dtExpiry <- myDate("01aug2010")
underlying <- "IBM"
```

```
K <- 100
sigma <- 0.3

a <- fInstrumentFactory("vanilla", quantity = 1,
    params = list(cp = "c", strike = K, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))

base.env <- DataProvider()

setData(base.env, underlying, "Price", dtObs,
    100)
setData(base.env, underlying, "DivYield", dtObs,
    0)
setData(base.env, underlying, "ATMVol", dtObs,
    sigma)
setData(base.env, underlying, "discountRef", dtObs,
    "USD.LIBOR")
setData(base.env, "USD.LIBOR", "Yield", dtObs,
    0.02)
```

At this stage, we can price the asset with

```
p <- getValue(a, "Price", dtStart, base.env)
```

which gives a value of $p = 4.87$. Next, define the parameters to simulate a dynamic hedging policy with weekly rebalancing and two price paths:

```
nbSteps <- 8
nbPaths <- 2
dtSim <- seq(dtObs, dtExpiry, length.out = nbSteps +
    1)
```

The simulated price path is generated by

```
tSpot <- pathSimulator(dtSim = dtSim, nbPaths = nbPaths,
    innovations.gen = sobolInnovations, path.gen = logNormal,
    path.param = list(mu = 0, sigma = sigma),
    S0 = 100, antithetic = F, standardization = TRUE,
    trace = FALSE)
```

and the simulated paths are placed in a new data provider:

```
sce.env <- DataProvider(parent = base.env)
setData(sce.env, underlying, "Price", time(tSpot),
    as.matrix(tSpot))
```

We can now run the delta-hedge strategy along each path:

```
assets = list(a)
res <- deltaHedge(assets, sce.env, params = list(dtSim = time(tSpot),
    transaction.cost = 0), trace = FALSE)
```

The result is a data structure that containts the residual wealth (hedging error) per scenario.

Tables **??** and **??** illustrate how the hedge portfolios evolve over time, based on the path of the underlying asset. In Table **??**, the option expires out of the money, and the hedge portfolio ends up being exclusively cash. In Table **??**, the option expires in the money, and the hedge portfolio holds at expiry one unit of stock. Note the amount of leverage required by delta hedging when the option expires in the money: the option payoff is 45 cents, and this is replicated by a portfolio holding €100.45 worth of stock, funded by borrowing €99.00.

The data shown in both tables are extracted from the simulation results and formated with the following function.

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sun Jan 15 20:51:28 2023

	time	stock price	delta	option	bond pos	hedge port.
1	1.00	100.00	0.52	4.87	-47.56	4.87
2	2.00	93.48	0.30	1.90	-26.40	1.45
3	3.00	90.07	0.18	0.86	-15.40	0.42
4	4.00	84.08	0.04	0.13	-4.08	-0.64
5	5.00	82.91	0.02	0.04	-2.10	-0.68
6	6.00	82.71	0.01	0.01	-1.22	-0.69
7	7.00	89.44	0.04	0.08	-3.93	-0.65
8	8.00	93.07	0.05	0.08	-5.26	-0.52
9	9.00	93.08	0.00	0.00	-0.52	-0.52

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sun Jan 15 20:51:28 2023

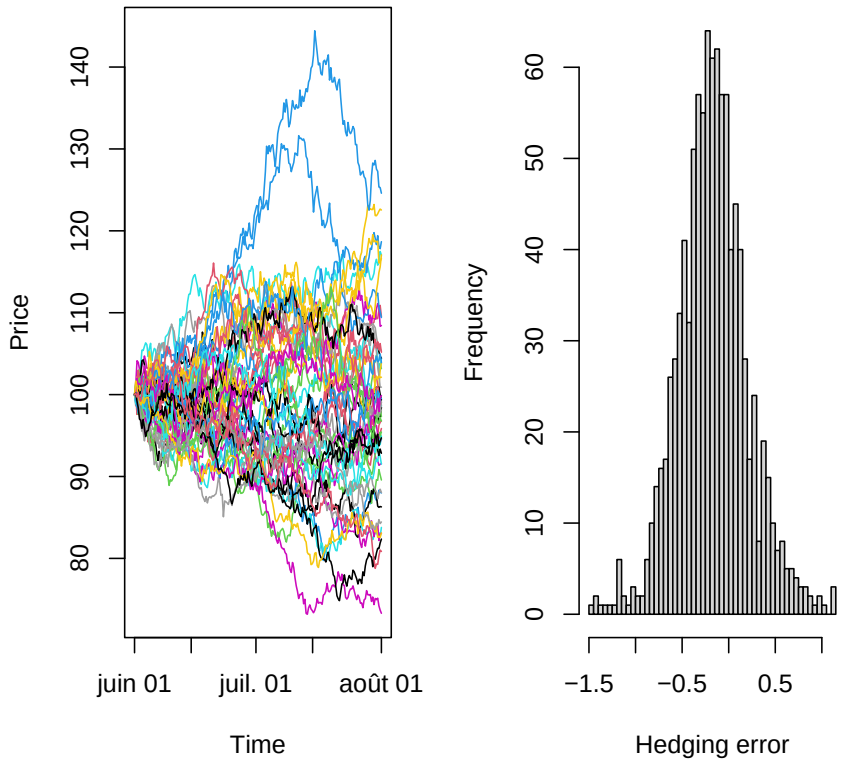
	time	stock price	delta	option	bond pos	hedge port.
1	1.00	100.00	0.52	4.87	-47.56	4.87
2	2.00	101.78	0.58	5.55	-53.61	5.79
3	3.00	98.59	0.47	3.53	-42.21	3.90
4	4.00	102.55	0.62	5.31	-57.95	5.74
5	5.00	98.34	0.44	2.66	-40.22	3.10
6	6.00	101.71	0.60	3.95	-56.87	4.57
7	7.00	107.75	0.89	8.08	-88.12	8.18
8	8.00	108.28	0.97	8.33	-96.21	8.62
9	9.00	105.83	1.00	5.83	-99.62	6.21

13.1.4 *A More Realistic Example*

More time steps and scenarios are needed in order to accurately assess the distribution of residual wealth. The next experiment uses the parameters of the previous section, with 200 time steps and 1000 sceanrios.

Figure **??** displays a few simulated paths for the underlying asset, and an histogram of the residual wealth.

```
nbSteps <- 200
nbPaths <- 1000
dtSim <- seq(dtObs, dtExpiry, length.out = nbSteps +
    1)
tSpot <- pathSimulator(dtSim = dtSim, nbPaths = nbPaths,
    innovations.gen = sobolInnovations, path.gen = logNormal,
    path.param = list(mu = 0, sigma = sigma),
    S0 = 100, antithetic = F, standardization = TRUE,
    trace = FALSE)
sim.env <- DataProvider(parent = base.env)
setData(sim.env, underlying, "Price", time(tSpot),
    as.matrix(tSpot))
sim.deltaHedge <- deltaHedge(assets, sim.env,
    params = list(dtSim = time(tSpot), transaction.cost = 0),
    trace = FALSE)
```



As expected, the distribution of residual wealth is centered, and has a standard deviation of

```
sd.wealth <- sd(t(tail(sim.deltaHedge$wealth,
    1)))
```

A confidence interval (say, 80%) for the P&L of the delta-hedging strategy is readily computed:

```
pl.confidence <- qnorm(0.2, mean = 0, sd = sd.wealth)
```

which gives a value of -0.32. In other words, there is a 80% probability that this delta-hedging strategy will yield a P&L greater than -0.32 per € of nominal. In the next couple of sections, we will provide a more rigorous analysis of these observations.

With this simulation tool, we will now consider the effect on hedging effectiveness of:

- discrete rather than continuous hedging,
- transaction costs, and
- unknown and stochastic volatility.

13.2 *Hedging Error due to Discrete Hedging*

We first consider the risk related to discreet hedging. The dynamic of the replicating portfolio, V_t rebalanced at times $t_i, i = 1, \ldots, N$. For $t \in [t_i, t_{i+1})$ is:

$$dV_t = \frac{\partial C(t_i, S_{t_i})}{\partial S} dS_t$$

The dynamic of the option price is:

$$dC_t = \frac{\partial C(t, S_t)}{\partial S} dS_t$$

The difference between the two processes being that in the hedge portfolio, the quantity of stock $\partial C(t_i, S_{t_i})/\partial S_{t_i}$ (Eq.

$$eq : dV$$

), is held constant between two rebalancing dates.

The hedging error process $\epsilon_t = C_t - V_t$ is therefore:

$$d\epsilon_t = \left(\frac{\partial C(t, S_t)}{\partial S} - \frac{\partial C(t_i, S_{t_i})}{\partial S} \right) dS_t$$

It can be shown that:

$$\lim_{h \to 0} E^Q \left[\frac{\epsilon_T^2}{\Delta t} \right] = E^Q \left[\frac{1}{2} \int_0^T \left(\frac{\partial^2 C(S_t, t)}{\partial S_t^2} \right)^2 S_t^4 \sigma^4 dt \right]$$

It can also be shown that the process ϵ_t converges in distribution to a process that has the following expression:

$$\frac{\epsilon_t}{\sqrt{\Delta t}} \rightarrow \frac{1}{\sqrt{2}} \int_0^t \frac{\partial^2 C(S_u, u)}{\partial S^2} \sigma^2 S_u^2 dZ_u$$

This expression highlights the following points:

1. The variance of the hedging is inversely related to the hedging frequency, and
2. is directly related to the magnitude of $\frac{\partial^2 C(S_u,u)}{\partial S^2}$, which is the Gamma of the option.

13.2.1 Empirical Test

To test the impact of hedging frequency on the distribution of delta-hedging error, we simulate the dynamic delta hedging of a ATM option. The price paths are generated with the same volatility as the one used for pricing, and there are no transaction costs. Therefore, the only source of hedging error is the discrete hedging policy. The simulation involves several steps that are outlined next.

First, define the derivative to be dynamically hedged:

```
dtExpiry <- mytDate("01jan2011")
underlying <- "IBM"
K <- 100
a <- fInstrumentFactory("vanilla", quantity = 1,
    params = list(cp = "c", strike = K, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))
```

Next, we define the market data that we be used in the simulation. The base environment holds data that will not change from one simulation to the next.

```
dtStart <- mytDate("01jan2010")
nbPaths <- 1000
sigma <- 0.3

base.env <- DataProvider()
setData(base.env, underlying, "Price", dtStart,
    100)
setData(base.env, underlying, "DivYield", dtStart,
    0.02)
setData(base.env, underlying, "ATMVol", dtStart,
    sigma)
setData(base.env, underlying, "discountRef", dtStart,
```

```
    "USD.LIBOR")
setData(base.env, "USD.LIBOR", "Yield", dtStart,
    0.02)
```

We next perform a series of dynamic hedging simulations, while varying the number of time steps.

```
nb.range <- seq(10, 500, by = 50)
mean.error <- 0 * nb.range
sd.error <- 0 * nb.range

for (i in seq_along(nb.range)) {

    nbSteps <- nb.range[i]

    dtSim <- seq(as.timeDate(dtStart), as.timeDate(dtExpiry),
        length.out = nbSteps + 1)

    # price paths
    tSpot <- pathSimulator(dtSim = dtSim, nbPaths = nbPaths,
        innovations.gen = sobolInnovations, path.gen = logNormal,
        path.param = list(mu = 0, sigma = sigma),
        S0 = 100, antithetic = F, standardization = TRUE,
        trace = FALSE)

    # derived environment for scenario
    # analysis

    sce.env <- DataProvider(parent = base.env)
    setData(sce.env, underlying, "Price", time(tSpot),
        as.matrix(tSpot))

    # simulate a delta-hedge strategy along
    # each path

    assets = list(a)
    res <- deltaHedge(assets, sce.env, params = list(dtSim = time(tSpot),
        transaction.cost = 0), trace = FALSE)

    expiry.error <- tail(res$wealth, 1)

    mean.error[i] <- mean(expiry.error)
    sd.error[i] <- sd(as.vector(expiry.error))
}
```

Figure **??** illustrates the mean and standard deviation of the hedging

error as a function of the number of rehedging dates. The figure also shows the curve

$$y = a + b\frac{1}{\sqrt{N}}$$

fitted to the standard deviation of the hedging error. The theory predicts that the standard deviation of the hedging error is proportional to $\frac{1}{\sqrt{N}}$, where N is the number of re-hedging dates.

```
plotCI(x = nb.range, y = mean.error, uiw = sd.error/2,
    liw = sd.error/2, ylim = c(min(mean.error -
        sd.error/2), max(mean.error + sd.error/2)),
    xlab = "Nb of rebalancings", ylab = "Hedging error")

q <- lm((mean.error + sd.error/2) ~ sqrt(1/nb.range))
yy <- predict(interpSpline(nb.range, q$fitted.values))
# lines(nb.range, q$fitted.values, lwd=2,
# col='red')
lines(yy, lwd = 2, col = "red")
```

To investigate the effect of gamma on hedging error, we will compare the effectiveness of delta hedging, applied to two options that initially have the same price and delta. The initial Gammas are within 10% of each other. However, the options differ substantially in one aspect: the first option is a European call, and its Gamma varishes when the option gets deep in the money, while the second option is a binary call, whose Gamma remains large and actually changes sign under the same conditions.

To perform the experiment, we first define the data environment: the spot price is 100, option maturity will be 1 year.

```
dtExpiry <- myDate("01jan2011")
dtStart <- myDate("01jan2010")
underlying <- "IBM"
sigma <- 0.3

base.env <- DataProvider()
setData(base.env, underlying, "Price", dtStart,
    100)
setData(base.env, underlying, "DivYield", dtStart,
    0.02)
setData(base.env, underlying, "ATMVol", dtStart,
    sigma)
setData(base.env, underlying, "discountRef", dtStart,
    "USD.LIBOR")
setData(base.env, "USD.LIBOR", "Yield", dtStart,
    0.02)
```

The strike of the European option is set to $K = 130$.

```
K <- 130
op.eu <- fInstrumentFactory("vanilla", quantity = 1,
    params = list(cp = "c", strike = K, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))

P <- getValue(op.eu, "Price", dtStart, base.env)
D <- getValue(op.eu, "Delta", dtStart, base.env)
```

The price is $P = 3.99$, delta is $D = 0.26$.

The strike and payoff of the binary option are set in order to match the price and delta of the European call, so as to make a meaningful comparison of hedging error. This involves solving a set of two non-linear equations (price and delta) in two unknown (payoff and strike). These equations are solved by means of the following function:

```
price.delta.error <- function(x, base.env) {
    Q <- x[1]
    K.b <- x[2]

    op.bi <- fInstrumentFactory("binary", quantity = Q,
        params = list(cp = "c", strike = K.b,
            dtExpiry = dtExpiry, underlying = underlying,
            discountRef = "USD.LIBOR", trace = FALSE))

    Pa <- getValue(op.bi, "Price", dtStart, base.env)
    Da <- getValue(op.bi, "Delta", dtStart, base.env)

    f <- rep(NA, 2)
    f[1] <- (P - Pa)
    f[2] <- (D - Da)
    f
}

ans <- dfsane(par = c(100, 100), fn = price.delta.error,
    base.env = base.env)
```

```
## Iteration:  0  ||F(x0)||:  31.38328
## iteration:  10  ||F(xn)|| =   0.01664841
## iteration:  20  ||F(xn)|| =   0.1147432
## iteration:  30  ||F(xn)|| =   0.04243755
## iteration:  40  ||F(xn)|| =   0.009005428
## iteration:  50  ||F(xn)|| =   0.001379282
## iteration:  60  ||F(xn)|| =   6.250002e-06
```

```
Q <- ans$par[1]
K.b <- ans$par[2]
```

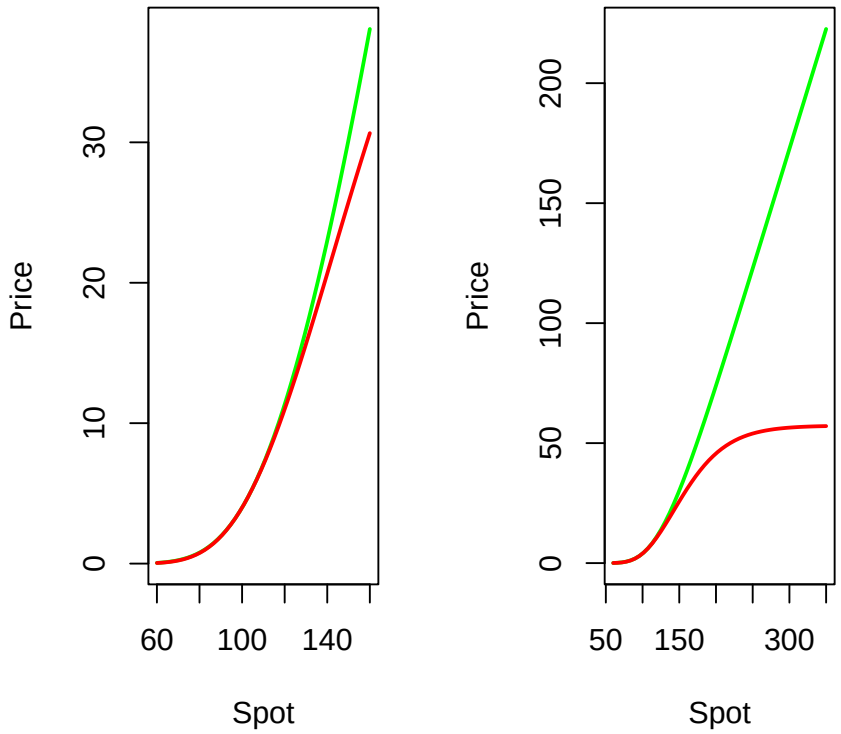
The solution is a binary call with payoff $Q = 58.45$ and strike $K = 155.06$.

```
op.bi <- fInstrumentFactory("binary", quantity = Q,
    params = list(cp = "c", strike = K.b, dtExpiry = dtExpiry,
        underlying = underlying, discountRef = "USD.LIBOR",
        trace = FALSE))

G.bi <- getValue(op.bi, "Gamma", dtStart, base.env)
G.eu <- getValue(op.eu, "Gamma", dtStart, base.env)
```

The Gammas of the two options are not identical, but within 10% of each others: 0.0107 for European option, 0.01 for the binary option.

To illustrate the similarity between the two derivatives, we plot in Figure**??** the prices of both options for a range of underlying values, with a zoom around the current spot price.

```
sce.1 <- t(as.matrix(seq(60, 160, length.out = 50)))
sce.2 <- t(as.matrix(seq(60, 350, length.out = 50)))

# derived data provider with price scenarii
sim.env.1 <- DataProvider(parent = base.env)
setData(sim.env.1, underlying, "Price", dtStart,
    sce.1)

sim.env.2 <- DataProvider(parent = base.env)
setData(sim.env.2, underlying, "Price", dtStart,
    sce.2)

# Compute and plot NPV per scenario for
# underlying spot price
p.eu.1 <- getValue(op.eu, "Price", dtStart, sim.env.1)
p.bi.1 <- getValue(op.bi, "Price", dtStart, sim.env.1)

p.eu.2 <- getValue(op.eu, "Price", dtStart, sim.env.2)
p.bi.2 <- getValue(op.bi, "Price", dtStart, sim.env.2)
```

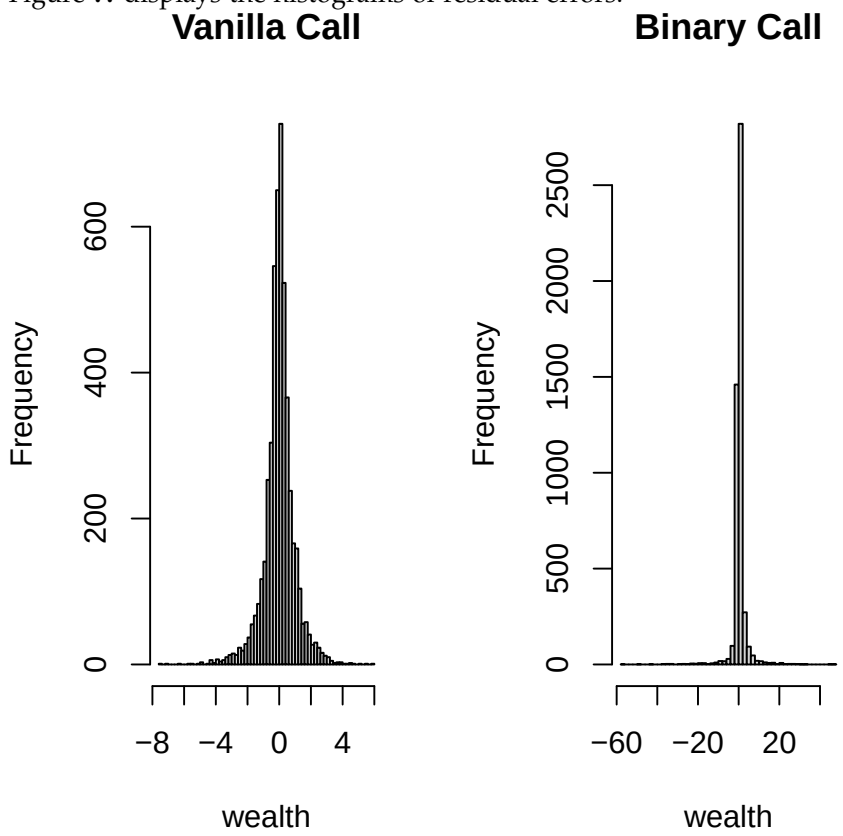


A delta hedging strategy is next simulated for both derivatives, using 100 steps, 5000 paths.

```
nbSteps <- 100
nbPaths <- 5000

dtSim <- seq(as.timeDate(dtStart), as.timeDate(dtExpiry),
    length.out = nbSteps + 1)

# price paths
tSpot <- pathSimulator(dtSim = dtSim, nbPaths = nbPaths,
    innovations.gen = sobolInnovations, path.gen = logNormal,
    path.param = list(mu = 0, sigma = sigma),
    S0 = 100, antithetic = F, standardization = TRUE,
    trace = FALSE)

# derived environment for scenario analysis
sce.env <- DataProvider(parent = base.env)
setData(sce.env, underlying, "Price", time(tSpot),
    as.matrix(tSpot))

assets = list(op.eu)
res1 <- deltaHedge(assets, sce.env, params = list(dtSim = time(tSpot),
    transaction.cost = 0), trace = FALSE)
```

```
assets = list(op.bi)
res2 <- deltaHedge(assets, sce.env, params = list(dtSim = time(tSpot),
    transaction.cost = 0), trace = FALSE)
```

Figure **??** displays the histograms of residual errors.



Statistics on the hedging error at expiry are summarized in Table

tab : *euro* − *bin*

. The standard deviation of the binary hedging error is about three times larger than for the vanilla option. Most importantly, the excess kurtosis is almost 10 times larger, indicating the possibilily that applying a standard delta-hedging strategy to a binary option can result in extreme losses (or gains!). All of this in a context where the only departure from the Black-Scholes assumptions is a discrete rather than continuous hedging. We next investigate the impact of transaction costs.

```
w1 <- as.vector(tail(res1$wealth, 1))
w2 <- as.vector(tail(res2$wealth, 2))
df.tmp <- data.frame(Vanilla = w1, Binary = w2)
funs <- list(mean, stdev, kurtosis, skewness)
res <- sapply(funs, mapply, df.tmp)
```

```
colnames(res) <- c("mean", "$\\sigma$", "skewness",
    "kurtosis")
xm <- xtable(res)
```

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sun Jan 15 20:52:45 2023

	mean	σ	skewness	kurtosis
Vanilla	0.00	0.99	8.32	-0.46
Binary	0.43	3.09	68.23	-1.57

13.3 *Hedging Error due to Transaction Cost*

Let S_t be the mid price of the underlying asset and k the proportional transaction cost: the hedger buys at the ask price:

$$S_t^a = S_t(1 + \frac{k}{2})$$

and sells at the bid:

$$S_t^a = S_t(1 - \frac{k}{2})$$

Clearly, transaction costs increase the cost of replication. To compensate for that, one needs to start with a higher budget, that is, the option seller should change a higher option premium than the one given by the Black-Scholes model. This higher option premium can be obtained in the Black-Scholes framework by increasing volatility. Leland (1985) shows that one can adjust the Black-Scholes volatility as follows:

$$\sigma^{+2} = \sigma^2 \left(1 + \sqrt{\frac{2}{\pi}} \frac{k}{\sigma\sqrt{\Delta t}}\right)$$

where σ is the Black-Scholes volatility, and $\sigma+$ the volatility adjusted for transaction costs.

13.3.1 *Empirical Test*

To test Leland's formula, we simulate the delta-hedging of a at-the-money European option, struck at 100, for transaction costs k ranging from 0 to 1%. The simulation is performed as follows:

```
tr.range <- seq(0, 1, 0.1)/100
nbSteps <- 250

dtSim <- seq(as.timeDate(dtStart), as.timeDate(dtExpiry),
    length.out = nbSteps + 1)
```

```
dt <- as.double(dtExpiry - dtStart)/(365 * nbSteps)

# price paths
tSpot <- pathSimulator(dtSim = dtSim, nbPaths = nbPaths,
    innovations.gen = sobolInnovations, path.gen = logNormal,
    path.param = list(mu = 0, sigma = sigma),
    S0 = 100, antithetic = FALSE, standardization = TRUE,
    trace = FALSE)

# derived environment for scenario analysis
sce.env <- DataProvider(parent = base.env)
setData(sce.env, underlying, "Price", time(tSpot),
    as.matrix(tSpot))

mean.error <- 0 * tr.range
sd.error <- 0 * tr.range

for (i in seq_along(tr.range)) {

    # simulate a delta-hedge strategy along
    # each path

    assets = list(a)
    res <- deltaHedge(assets, sce.env, params = list(dtSim = time(tSpot),
        transaction.cost = tr.range[i]/2), trace = FALSE)

    expiry.error <- tail(res$wealth, 1)

    mean.error[i] <- mean(expiry.error)
    sd.error[i] <- sd(as.vector(expiry.error))
}
```

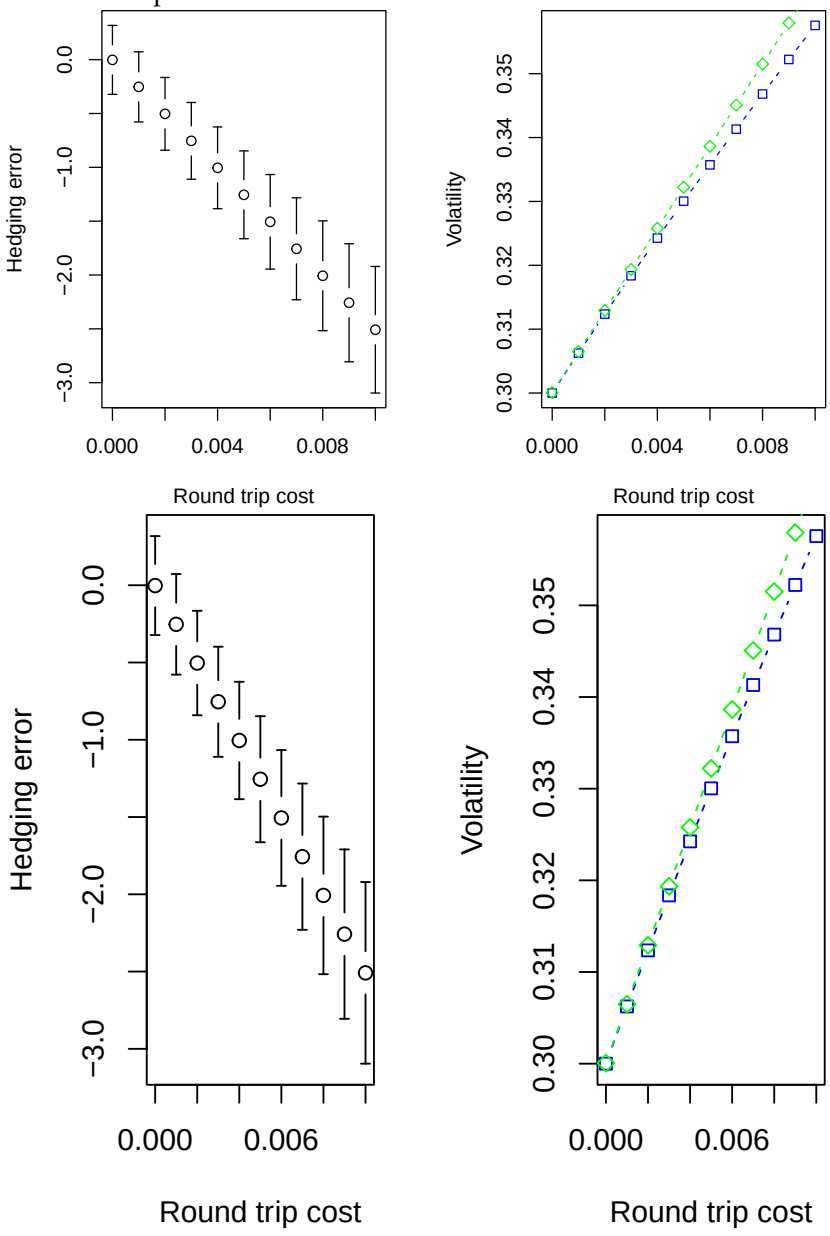
We next compare the expected hedging error to the value predicted by Leland's formula. To do so, we convert the expected hedging error into a volatility adjustment, using the first order approximation:

$$P(\sigma) = P(\sigma^*) + (\sigma - \sigma^*)\frac{\partial P}{\partial \sigma}$$

where $P(\sigma)$ is the price of the option with volatility σ.

```
sigma.leland <- sigma * sqrt(1 + sqrt(2/pi) *
    (tr.range/(sigma * sqrt(dt))))

vega <- getValue(a, "Vega", dtStart, base.env)
sigma.sim <- sigma - mean.error/vega
```

The results are presented in Figure **??**. We find very good agreement between the empirical result and Leland's formula.



13.4 *Hedging Error due to Unknown Volatility*

The strongest assumption of the Black-Scholes model is arguably the assumption that volatility is known and constant. we next investigate the impact of unknown volatility of hedging error, following a landmark paper by (**Karoui1998?**).

To simplify the notation, assume now that hedging is done continuously, but that the volatility of the underlying asset is unknown. The

dynamic of the underlying asset is:

$$dS_t = \mu dt + \sigma_t dW_t$$

The derivative is priced with an estimated volatility Σ.
The dynamic of the hedge portfolio is:

$$dV_t = \frac{\partial C_t^{\Sigma}}{\partial S_t} dS_t$$

By Itô's lemma:

$$dC(S_t, t) = \left(\frac{\partial C}{\partial t} + \frac{1}{2} \frac{\partial^2 C}{\partial S^2} \sigma_t^2 S_t^2 \right) dt + \frac{\partial C}{\partial S} dS_t$$

The price of the derivative verifies the Black-Scholes EDP:

$$\frac{\partial C}{\partial t} + \frac{1}{2} \Sigma^2 S_t^2 \frac{\partial^2 C}{\partial S^2} = 0$$

this yields,

$$d\epsilon_t = dV_t - dC_t = \frac{1}{2} \left[\Sigma^2 - \sigma_t^2 \right] \frac{\partial^2 C}{\partial S^2} S^2 dt$$

and the hedging error at expiry is thus,

$$\epsilon_T = \frac{1}{2} \int_0^T \left[\Sigma^2 - \sigma_t^2 \right] \frac{\partial^2 C}{\partial S^2} S^2 dt$$

We identify three components in the above equation:

1. the nature of the option, characterized by its Gamma.
2. the behavior of the market, characterized by σ_t,
3. the model calibration, summarized here by the Black-Scholes volatility Σ

If the Gamma keeps a constant sign, we can compute the fair value of the Black-Scholes volatility that sets the expected hedging error to o is

$$\sigma_{BS}^2 = \frac{E \int_0^T \Gamma S^2 \frac{\Delta S}{S}^2}{E \int_0^T \Gamma_{BS} S^2 dt}$$

Equation (

$$eq : ElKaroui - JB - S$$

) also shows that, when the gamma keeps a constant sign, we can choose a pricing volatility Σ such that $E(\epsilon_T)$ is positive.

13.5 Conclusion

The experiments of this chapter highlight the crucial role of Gamma in dynamic hedging. In short, the original Black-Scholes framework remains adapted for options for which the Gamma does not change sign. This is summarized in Table

$$tab : bs - adapted$$

. In such case, a positive expected hedging error can be ensured by setting the pricing volatility sufficiently high or low, as the case may be. Moreover, the Gamma of these options often vanishes in deep in the money or out of the money scenarios, and this renders of course the delta hedging strategy very effective. This may explain why the Black-Scholes model remains so widely used, despite its well documented shortcomings. Next chapter describes a framework for dealing with Gamma risk, while remainning in the Black-Scholes framework.

Payoff	Is BS Adapted?
Vanilla	Yes
European Binary	No
Up and Out Barrier	No
Asian	Yes

13.6 Second-Order Hedging

In the previous section, we have determined that hedging error is determined by three factors: the hedging frequency, the magnitude of the option gamma, and finally the difference between Σ (the BS volatility used for pricing and hedging) and the volatility that is actually experienced, σ_t.

In this section, we investigate strategies for controlling the risk associated with the magnitude of the gamma.

14 *The implied volatility*

```
## Loading required package: timeDate

## Loading required package: timeSeries

## Loading required package: fBasics

## Loading required package: gsw

## Loading required package: fExoticOptions

## Loading required package: fAsianOptions

## Loading required package: fImport

## Loading required package: RCurl

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

Listed options are quoted in price. Implied volatility is the volatility that recovers that quoted price, using the Black-Scholes model. Computing implied volatility amounts to inverting the BS formula, given an option price. This chapter discusses the issues involved in such calculation, and how to represent the result.

14.1 Market Data

Option settlement prices are published every day by exchanges. Figure **??** shows a sample of the data file published every day by the NYMEX.

NEW YORK MERCANTILE EXCHANGE
NYMEX OPTIONS CONTRACT LISTING FOR 12/29/2008

--------CONTRACT--------				TODAY'S SETTLE	PREVIOUS SETTLE	ESTIMATED VOLUME
LC	02 09	P	30.00	.53	.85	0
LC	02 09	P	35.00	1.58	2.28	0
LC	02 09	P	37.50	2.44	3.45	0
LC	02 09	C	40.00	3.65	2.61	10
LC	02 09	P	40.00	3.63	4.90	0
LC	02 09	P	42.00	4.78	6.23	0
LC	02 09	C	42.50	2.61	1.80	0
LC	02 09	C	43.00	2.43	1.66	0
LC	02 09	P	43.00	5.41	6.95	100

NEW YORK MERCANTILE EXCHANGE NYMEX OPTIONS CONTRACT LISTING FOR 12/29/2008

TODAY'S PREVIOUS ESTIMATED ——CONTRACT—— SETTLE SETTLE VOLUME LC 02 09 P 30.00 .53 .85 0 LC 02 09 P 35.00 1.58 2.28 0 LC 02 09 P 37.50 2.44 3.45 0 LC 02 09 C 40.00 3.65 2.61 10 LC 02 09 P 40.00 3.63 4.90 0 LC 02 09 P 42.00 4.78 6.23 0 LC 02 09 C 42.50 2.61 1.80 0 LC 02 09 C 43.00 2.43 1.66 0 LC 02 09 P 43.00 5.41 6.95 100

Notice that settlement prices are provided even for options that have not been traded today. Providing a price for all strikes and maturities is necessary in order to compute margin calls on open positions. The exchange uses an interpolation method for estimating these settlement prices in the absence of actual transactions.

14.2 Calculation of implied volatility

Given an observed price C^*, compute the volatility σ such that:

$$\begin{aligned} C^* &= C(S, K, T, r, \sigma) \\ &= C(\sigma) \end{aligned}$$

Observe that there is a change in convexity in $f(\sigma)$:

$$\frac{\partial C}{\partial \sigma} = Sn(d_1)\sqrt{T}$$

$$\frac{\partial^2 C}{\partial \sigma^2} = S\sqrt{T}n(d_1)\frac{1}{\sigma}\left[\frac{1}{\sigma^2 T}\ln(\frac{F}{K})^2 - \frac{1}{4}\sigma^2 T\right]$$

with $F = Se^{rT}$.

Thus, $C(\sigma)$ is convex on the interval $(0, \sqrt{\frac{2|\ln(F/K)|}{T}}]$, and concave otherwise.

14.2.1 Newton's method

Let C^* be the price of a call option. We look for the volatility σ^* such that

$$C^* = C(S, K, T, r, \sigma^*)$$

To ensure convergence of Newton's method, one must carefully choose the initial point.

Let f be defined on the interval $[a, b]$ and assume that:

1. $f(x^*) = 0$ for some $x^* \in [a, b]$
2. $f'(x) > 0$
3. $f''(x) \geq 0$

Then Newton's method converges monotonically from $x_0 = b$. If

1. $f(x^*) = 0$ for some $x^* \in [a, b]$
2. $f'(x) > 0$
3. $f''(x) \leq 0$

Then Newton's method converges monotonically from $x_0 = a$.

We prove the first assertion. For all $x_n \in [a, b]$,

$$f(x^*) = f(x_n) + (x^* - x_n)\frac{\partial f}{\partial x} + K$$

with $K > 0$. Therefore,

$$x^* < x_n - \frac{\partial f}{\partial x}$$

or,

$$x^* <= x_{n+1} < x_n$$

Newton's method generates a monotonic sequence that must converge to x^*.

Consider now Newton's method started at

$$\sigma_0 = \sqrt{\frac{2|\ln(F/K)|}{T}}$$

If $f(\sigma_0) > 0$, we are in case I of the above theorem, and Newton's method generates a decreasing sequence that converges to σ^*. If $f(\sigma_0) < 0$ we are in case II, and Newton's method also converges. The algorithm is as follows:

14.2.2 *Implied Volatility by Newton's Method*

The following algorithm generates a monotonic series (σ_n):

1. Set $\sigma_0 = \sqrt{\frac{2|\ln(F/K)|}{T}}$
2. While $|C(\sigma_n) - C^*| > \epsilon$:
   1. Let
   $$\sigma_{n+1} = \sigma_n + \frac{C^* - C(\sigma_n)}{\frac{\partial C}{\partial \sigma}}$$
   2. $n \leftarrow n + 1$

The algorithm is implemented as follows:

```
ImpliedVolNewton <- function(p, TypeFlag, S, X, Time, r, b, tol, maxiter=50) {
  F <- S * exp((r-b)*T)
  s <- sqrt(2*log(F/X)/T)
  not_converged <- T
  vega <- GBSGreeks(Selection="vega", TypeFlag, S, X, Time, r, b, s)
  i <- 1
  while(not_converged & (i<maxiter)) {
    err <- (p-GBSOption(TypeFlag, S, X, Time, r, b, s)@price)
    s <- s + err/vega
    not_converged <- (abs(err/vega) > tol)
    i <- i+1
  }
s
}
```

A timing test demonstrate the advantage of choosing the initial point described above:

```
TypeFlag <- 'c'
S <- 100
X <- 100
Time <- 1
r <- .03
b <- .01
sigma <- .314
tol <- 1e-6

p <- GBSOption(TypeFlag, S, X, Time, r, b, sigma)@price
```

We perform 100 replications of the same calculation.

```
t1 <- function(n=100) {
for (i in 1:n) {
si <- GBSVolatility(p, TypeFlag, S, X, Time, r, b, tol=tol, maxiter=50)
}}

t2 <- function(n=100) {
for (i in 1:n) {
si <- ImpliedVolNewton(p, TypeFlag, S, X, Time, r, b, tol) }}
```

Statistics for the default implied volatility calculation are:

```
system.time(t1(100))
```

```
##    user  system elapsed
##   0.184   0.000   0.185
```

and with Newton's method and the initial point computed as above:

```
system.time(t2(100))
```

```
##    user  system elapsed
##   0.096   0.000   0.095
```

14.2.3 Secant Method

For deep in the money or out of the money options, where vega is very small, a solution method that does not require a derivative may be prefered. The secant method starts with a bracking interval around the solution, and progressively shrinks it. The algorithm is as follows:

It implementation is straight-forward:

```
ImpliedVolSecant <- function(p, TypeFlag, S, X, Time, r, b, tol, sBounds, maxiter=100) {
sMin <- min(sBounds)
sMax <- max(sBounds)
pMin <- GBSOption(TypeFlag, S, X, Time, r, b, sMin)@price
pMax <- GBSOption(TypeFlag, S, X, Time, r, b, sMax)@price

not_converged <- abs(pMin-pMax) > tol
i <- 1

while(not_converged & (i<maxiter)) {
  sStar <- (sMin + sMax)/2
  pStar <- GBSOption(TypeFlag, S, X, Time, r, b, sStar)@price
  if(pStar < p) {
    pMin <- pStar;
    sMin <- sStar
  } else {
```

```
      pMax <- pStar;
      sMax <- sStar
    }

    not_converged <- (abs(pMin-pMax) > tol)
    i <- i+1
  }

  (sMin+sMax)/2
  }

  t3 <- function(n=100) {
  for (i in 1:n) {
  si <- ImpliedVolSecant(p, TypeFlag, S, X, Time, r, b, tol, c(.1, 1))
  }
  si
  }
```

A timing test shows the value of this algorithm for a deep our of the money call:

```
  S <- 100
  X <- 20
  sigma <- .514
  p <- GBSOption(TypeFlag, S, X, Time, r, b, sigma)@price
  n <- 100

  system.time(t1(n))
```

```
##    user  system elapsed
##   0.359   0.004   0.362
```

```
  system.time(t2(n))
```

```
##    user  system elapsed
##   0.772   0.012   0.783
```

```
  system.time(t3(n))
```

```
##    user  system elapsed
##   0.324   0.008   0.331
```

14.2.4 *Jaeckel's method*

Peter Jackel ((Jaeckel 2006)) proposes a variant of the above method to regularize Newton's method.

Given an option price p, we must solve for σ

$$p = \delta\theta \left[F\Phi\left(\theta\left[\frac{\ln(F/K)}{\sigma} + \frac{\sigma}{2}\right]\right) - K\Phi\left(\theta\left[\frac{\ln(F/K)}{\sigma} - \frac{\sigma}{2}\right]\right)\right]$$

where:

δ. discount factor

θ. 1 for call, -1 for put

F. Forward price: $Se^{(r-d)T}$

σ. $\sigma\sqrt{T}$

Set:

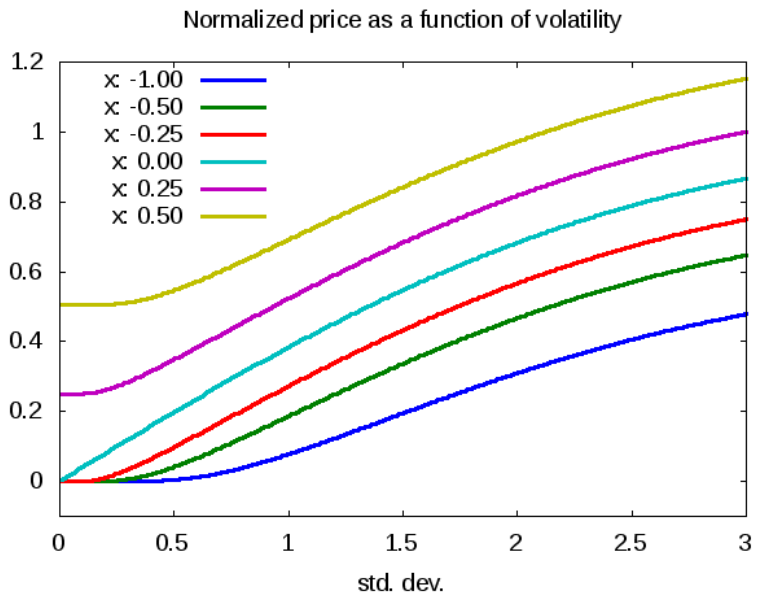
$$\begin{aligned} x = &\quad \ln(F/K) \\ b = &\quad \frac{p}{\delta\sqrt{FK}} \end{aligned}$$

The Black-Scholes equation becomes:

$$b = \theta\left[e^{x/2}\Phi\left(\theta\left[\frac{x}{\sigma} + \frac{\sigma}{2}\right]\right) - e^{-x/2}\Phi\left(\theta\left[\frac{x}{\sigma} - \frac{\sigma}{2}\right]\right)\right]$$

14.2.5 *Normalized Call price as a function of σ*

Figure 14.1: image



As in the previous section, the normalized price function changes convexity at $\sigma_c = \sqrt{|x|}$

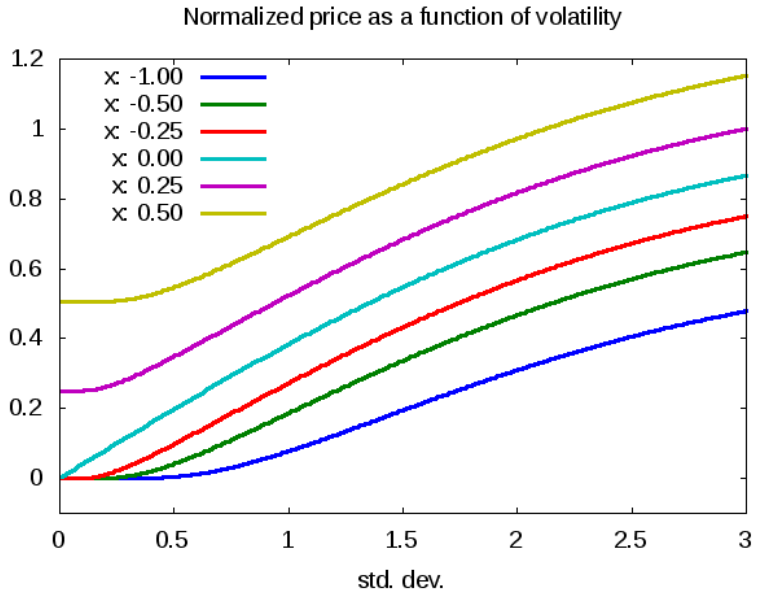


Figure 14.2: image

Jackel further defines

$$f(\sigma) = \begin{cases} \ln(\frac{b-i}{\bar{b}-i}) & \text{if} \bar{b} < b_c \\ b - \bar{b} & \text{otherwise} \end{cases}$$

with:

b_c. $b(x, \sigma_c, \theta)$

i. normalized intrinsic value: $1_{\theta x>0}\theta\left(e^{x/2} - e^{-x/2}\right)$

$\bar{b}$. target normalized price

The function $f(\sigma)$ is now monotonously concave, and the implied volatility problem amounts to solving:

$$f(\sigma) = 0$$

14.2.6 *Transformed objective function*

14.2.7 *Impact of Dividends*

For indices: watch for dividends impact, compute implied dividend rate from C/P parity, then compute implied vol.

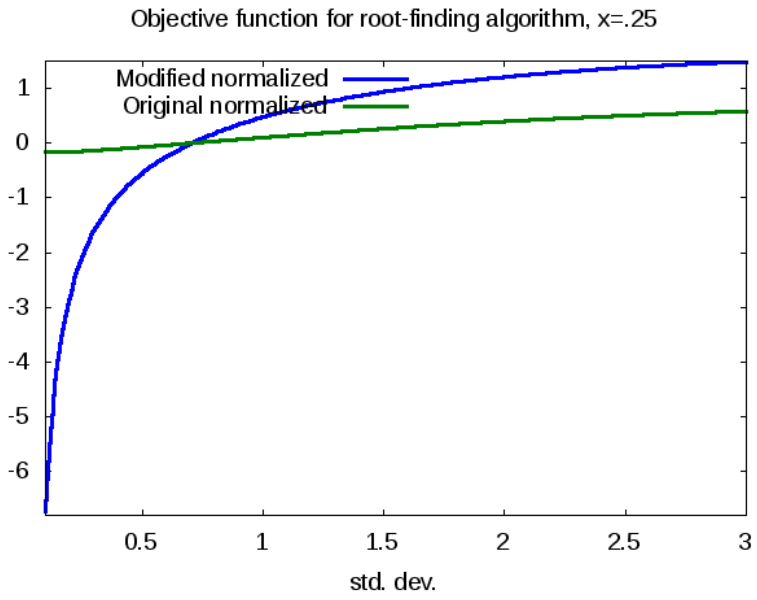


Figure 14.3: image

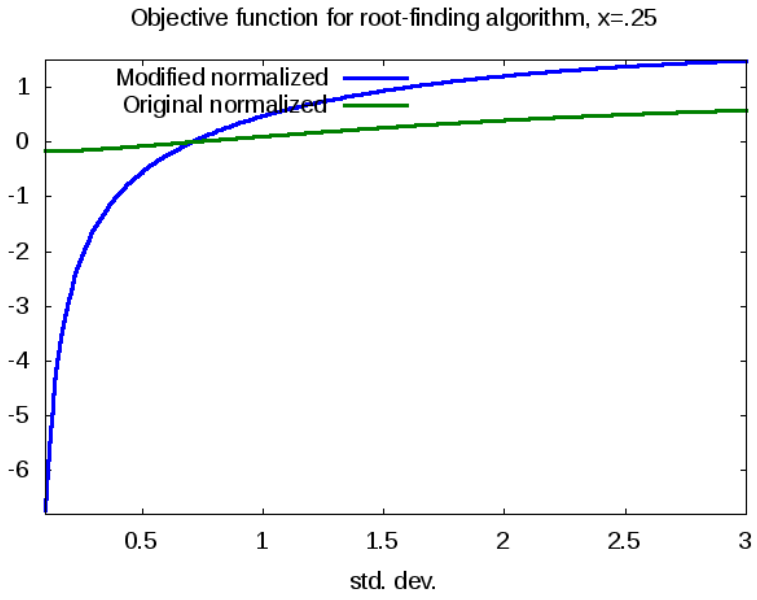


Figure 14.4: image

15 *Vanna-Volga Pricing and Hedging*

```
## Loading required package: timeDate

## Loading required package: timeSeries

## Loading required package: fBasics

## Loading required package: fAsianOptions

## Loading required package: lubridate

## Loading required package: timechange

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

## Loading required package: xtable

##
## Attaching package: 'xtable'
```

```
## The following objects are masked from 'package:Hmisc':
##
##     label, label<-

## The following object is masked from 'package:timeSeries':
##
##     align

## The following object is masked from 'package:timeDate':
##
##     align

## Loading required package: empfin

## Loading required package: fImport

## Loading required package: RCurl
```

If you want to know the value of a security, use the price of another security that's similar to it. All the rest is strategy.
E. Derman

This chapter presents a practical method for pricing and hedging derivatives, taking into account an uncertain volatility. This method is very popular for Foreign Exchange derivatives, but beyond that it illustrates an important principle of asset pricing, which is to relate the price of a complex derivative to the known price of simpler, liquid instruments.

15.1 Principle

Assume a Ito dynamic for volatility, and consider a derivative security $O(t)$, which is now subject to two sources of randomness: the underlying asset S_t and the volatility σ_t. An extended version of Ito's lemma give:

$$\begin{aligned} dO(t,K) = & \frac{\partial O}{\partial t}dt + \frac{\partial O}{\partial S}dS_t + \frac{\partial O}{\partial \sigma}d\sigma_t \\ & + \frac{\partial^2 O}{\partial S^2}(dS_t)^2 + \frac{\partial^2 O}{\partial \sigma^2}(d\sigma_t)^2 + \frac{\partial^2 O}{\partial S \partial \sigma}dS_t d\sigma_t \end{aligned}$$

We now construct a portfolio made of option $O(t,K)$, and a hedge made of Δ_t units of the underlying asset and 3 delta-hedged vanilla options $C_i(t,K_i)$:

$$
\begin{aligned}
dO(t,K) - \Delta_t dS_t - \sum_{i=1}^{3} x_i dC_i(t,K_i) = \\
\left[\frac{\partial O}{\partial t} - \sum_i x_i \frac{\partial C_i}{\partial t}\right] dt \\
+ \left[\frac{\partial O}{\partial S} - \Delta_t - \sum_i x_i \frac{\partial C_i}{\partial S}\right] dS_t \\
+ \left[\frac{\partial O}{\partial \sigma} - \sum_i x_i \frac{\partial C_i}{\partial \sigma}\right] d\sigma_t \\
+ \frac{1}{2}\left[\frac{\partial^2 O}{\partial S^2} - \sum_i x_i \frac{\partial^2 C_i}{\partial S^2}\right] (dS_t)^2 \\
+ \frac{1}{2}\left[\frac{\partial^2 O}{\partial \sigma^2} - \sum_i x_i \frac{\partial^2 C_i}{\partial \sigma^2}\right] (d\sigma_t)^2 \\
+ \left[\frac{\partial^2 O}{\partial S \partial \sigma} - \sum_i x_i \frac{\partial^2 C_i}{\partial S \partial \sigma}\right] dS_t d\sigma_t
\end{aligned}
$$

We can choose Δ_t and x_i to zero out the terms $dS_t, d\sigma_t, (d\sigma_t)^2$ and $dS_t d\sigma_t$. We are left with:

$$
\begin{aligned}
dO(t,K) - \Delta_t dS_t - \sum_{i=1}^{3} x_i dC_i(t,K_i) = \\
\left[\frac{\partial O}{\partial t} - \sum_i x_i \frac{\partial C_i}{\partial t}\right] dt \\
+ \frac{1}{2}\left[\frac{\partial^2 O}{\partial S^2} - \sum_i x_i \frac{\partial^2 C_i}{\partial S^2}\right] (dS_t)^2
\end{aligned}
$$

Using the definition of dS_t and retaining the first-order terms, we have, for all securities function of S_t:

$$
\frac{\partial f}{\partial t} dt + \frac{1}{2}\frac{\partial^2 f}{\partial S^2} = \left(\frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2} S_t^2 \sigma^2\right) dt
$$

Moreover, each asset satisfies the Black-Scholes PDE:

$$
\frac{\partial f}{\partial t} + r S_t \frac{\partial f}{\partial S} + \sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf
$$

For each asset, $\frac{\partial f}{\partial S} = 0$, and we get:

$$
\frac{\partial f}{\partial t} dt + \frac{1}{2}\frac{\partial^2 f}{\partial S^2} = rf dt
$$

The hedged portfolio is thus riskless:

$$dO(t,K) - \Delta_t dS_t - \sum_{i=1}^{3} x_i dC_i(t,K_i) =$$
$$r\left[O(t,K) - \sum_i x_i C_i(t,K_i)\right] dt$$

In summary, we can still have a locally perfect hedge when volatility is stochastic, as long as the prices $O(t,K)$ and $C_I(t,K_i)$ follow the Black-Scholes equation.

Discretize equation

$$eq : vv - 1$$

at time $t = T - \delta t$, where T is the option expiry. We get:

$$O(T,K) - O(t,K) - \Delta_t(S_T - S_t) - \sum_i x_i(t)\left[C_i(T,K_i) - C_i(t,K_i)\right] =$$
$$r\left[O(t,K) - \Delta_t S_t - \sum_i x_i(t) C(t,K_i)\right]\delta t$$

Up to now, we have assumed that all assets are priced with the same volatility σ_t. In practice, we observe market prices $C_i^M(t,K_i)$ that differ from the theoretical Black-Scholes prices $C_i(t,K_i)$. The hedge described above must be implemented with options traded at market price. What are the implications for the price $O(t,K)$? An approximate argument is presented below; see Shkolnikov (2009) for a rigorous treatment.

Consider a hedged position at time $T - \delta t$: long the exotic derivative O with market value $O^M(t)$ and short the hedge portfolio. The value of the hedged position at expiry is:

$$W_T = O(T) - \Delta_t S_T - \sum_i x_i C_i(T) - r\left(O^M(t) - \sum_i x_i C_i^M(t) - \Delta S_t\right)\delta t$$

Set

$$O^M(t,K) = O(t,K) + \sum_i x_i\left[C_i^M(t,K_i) - C_i(t,K_i)\right]$$

to obtain $W_T = 0$.

In summary, if we have a wealth $O^M(t,K)$ defined by

$$eq : vv - 2$$

at time $T - \delta t$, then we can replicate the payoff $O(T,K)$ at expiry, with a hedge at market price. The argument made on the interval $[T - \delta t, T]$ can be applied by backward recursion for each time interval until $t = 0$.

We have both a hedging strategy and a process for adjusting the price of any derivative to account for the smile. let's now consider some implementation details.

15.1.1 *Vanna-Volga Dynamic Hedging*

$$\begin{aligned} O(T,K) = & \quad O^M(t,K) + \Delta_t(S_T - S_t) \\ + & \quad \sum_i x_i \left[C_i(T,K_i) - C_i^M(t,K_i) \right] \\ + & \quad r \left[O(t,K) - \Delta_t S_t - \sum_i x_i C_i^M(t,K_i) \right] \end{aligned}$$

If we have a wealth $O(t,K)$ at time $T - \delta t$, then we can replicate the payoff $O(T,K)$ at expiry, with a hedge at market price.

15.2 *Implementation*

The weights x_i are obtained by solving the system of linear equations:

$$\begin{aligned} \frac{\partial O}{\partial \sigma} &= \sum_i x_i \frac{\partial C_i}{\partial \sigma} \\ \frac{\partial^2 O}{\partial \sigma^2} &= \sum_i x_i \frac{\partial^2 C_i}{\partial \sigma^2} \\ \frac{\partial^2 O}{\partial S \partial \sigma} &= \sum_i x_i \frac{\partial^2 C_i}{\partial S \partial \sigma} \end{aligned}$$

or,

$$b = Ax$$

Since the result of the previous section holds for any derivative that verifies the Black-Scholes equation, we can choose the benchmark securities $C_i(t,K_i)$ as we see fit.

A popular set of benchmark securities, commonly used in the FX market is described next. To simplify notation, we denote $C(K), P(K)$ the call and put of strike K, maturity T:

- An at-the-money straddle:

$$C_1 = C(S) + P(S)$$

- A "risk reversal", traditionally defined as

$$C_2 = P(K_1) - C(K_2)$$

  with K_1 and K_2 chosen so that the options have a Delta of .25 in absolute value.

- A "butterfly", defined as

$$C_3 = \beta(P(K_1) + C(K_2)) - (P(S) + C(S))$$

  with β determined to set the Vega of the butterfly to 0.

This system is popular because the benchmark securities are very liquid, and because the resulting A matrix of (

$$eq : A - matrix$$

) is almost diagonal, which allows an intuitive interpretation of the coefficients x_i.

To summarize, the calculation steps for pricing an option, taking the smile cost into account, are as follows:

1. compute the risk indicators for the option O to be priced:

$$b = \begin{pmatrix} \frac{\partial O}{\partial \sigma} \\ \frac{\partial^2 O}{\partial \sigma^2} \\ \frac{\partial^2 O}{\partial \sigma \partial S} \end{pmatrix}$$

2. compute the A matrix

$$A = \begin{pmatrix} \frac{\partial C_1}{\partial \sigma} & \cdots & \frac{\partial C_3}{\partial \sigma} \\ \frac{\partial^2 C_1}{\partial \sigma^2} & \cdots & \frac{\partial^2 C_3}{\partial \sigma^2} \\ \frac{\partial^2 C_1}{\partial \sigma \partial S} & \cdots & \frac{\partial^2 C_3}{\partial \sigma \partial S} \end{pmatrix}$$

3. solve for x:

$$b = Ax$$

4. the corrected price for O is:

$$O^M(t,K) = O^{BS}(t,K) + \sum_{i=2}^{3} x_i \left(C_i^M(t) - C_i^{BS}(t) \right)$$

   where $C_i^M(t)$ is the market price and $C_i^{BS}(t)$ the Black-Scholes price (i.e. with flat volatility).

The term in x_1 is omitted in (4) since, by definition, the Black-Scholes price and market price of an ATM straddle are identical.

Neglecting the off diagonal terms in A, a simplified procedure is to estimate x_i by:

$$x_2 = \frac{\frac{\partial^2 O}{\partial \sigma^2}}{\frac{\partial^2 C_2}{\partial \sigma^2}}$$

$$x_3 = \frac{\frac{\partial^2 O}{\partial \sigma \partial S}}{\frac{\partial^2 C_3}{\partial \sigma \partial S}}$$

In practice, the weights x_i are scaled to better fit market prices.

15.2.1 Vanna-Volga Dynamic Hedging

Neglecting the off diagonal terms in A, a simplified procedure is to estimate x_i by:

$$x_2 = \frac{\frac{\partial^2 O}{\partial \sigma^2}}{\frac{\partial^2 C_2}{\partial \sigma^2}}$$

$$x_3 = \frac{\frac{\partial^2 O}{\partial \sigma \partial S}}{\frac{\partial^2 C_3}{\partial \sigma \partial S}}$$

In practice, the weights x_i are scaled to better fit market prices.

15.3 Illustrations

15.3.1 Volatility Interpolation

The simplest use of this method is volality interpolation. Given the ATM volatility and at two other strikes, we want to determine the volatility at an arbitrary strike K.

The process is illustrated below, with the market data summarized in Table

$$tab : vv - 1$$

for European options with maturity $T = 1$ year. Interest rate is set to 0 for simplicity.

Volatility data is summarized in Table

$$tab : vv - 1$$

.

Strike	Volatility
80	.32
100	.30
120	.315

```
T <- 1
Spot <- 100
r <- 0
b <- 0
eps <- .001
sigma <- .3

# Benchmark data: (strike, volatility)
VolData <- list(c(80, .32), c(100, .30), c(120, .315))
```

Define an array of pricing functions for the three benchmark instruments:

```
C <- c(
       function(vol=sigma, spot=Spot) GBSOption(TypeFlag='c', S=spot, X=VolData[[1]][1], Time=T, r=r,
       function(vol=sigma, spot=Spot) GBSOption(TypeFlag='c', S=spot, X=VolData[[2]][1], Time=T, r=r,
       function(vol=sigma, spot=Spot) GBSOption(TypeFlag='c', S=spot, X=VolData[[3]][1], Time=T, r=r,
```

Next, define utility functions to compute the risk indicators, all by finite difference:

```
Vega <- function(f, vol, spot=Spot) (f(vol+eps, spot)-f(vol-eps, spot))/(2*eps)

Vanna <- function(f, vol, spot=Spot) {
  (Vega(f, vol, spot+1)-Vega(f, vol, spot-1))/2
}

Volga <- function(f, vol) {
    (Vega(f,vol+eps)-Vega(f,vol-eps))/(eps)
}
```

Finally, the following function computes the Vanna-Volga adjustment to the Black-Scholes price, and the corresponding implied volatility:

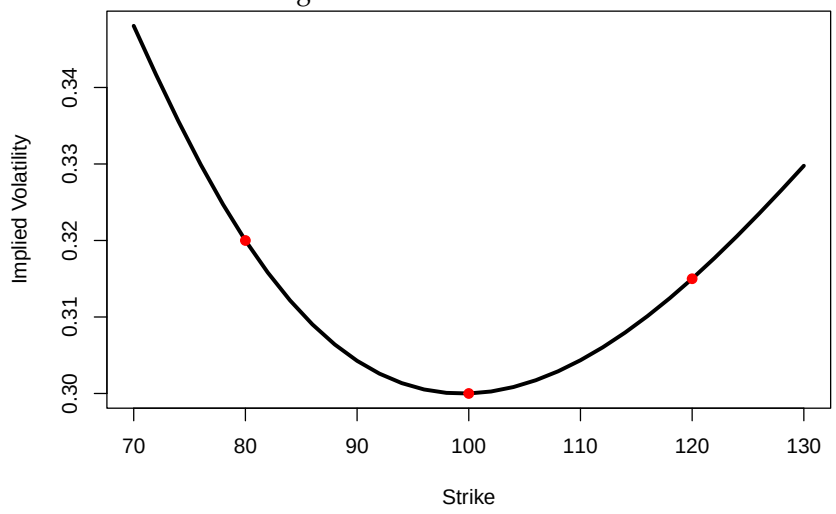
```
VVVol <- function(X) {

O <- function(vol=sigma, spot=Spot) GBSOption(TypeFlag='c', S=spot,
       X=X, Time=T, r=r, b=b, sigma=vol)@price
TV.BS <- O()

# risk indicators for benchmark instruments
B.vega <- sapply(1:3, function(i) Vega(C[[i]], sigma))
B.vanna <- sapply(1:3, function(i) Vanna(C[[i]], sigma))
B.volga <- sapply(1:3, function(i) Volga(C[[i]], sigma))

# risk indicators for new option
O.vega <- Vega(O, sigma)
O.vanna <- Vanna(O, sigma)
O.volga <- Volga(O, sigma)

# Benchmark costs
B.cost <- sapply(1:3, function(i) C[[i]](VolData[[i]][2]) - C[[i]](sigma))

# calculation of price adjustment
A <- t(matrix(c(B.vega, B.vanna, B.volga),  nrow=3))
```

```
x <- matrix(c(0.vega, 0.vanna, 0.volga), nrow=3)
w <- solve(A, x)
CF <- t(w) %*% matrix(B.cost, nrow=3)

# implied volatility
v <- GBSVolatility(TV.BS+CF, 'c', Spot, X, T, r, b, 1.e-5)

v}
```

We finally use the vanna-volga interpolating function to construct the interpolated smile curve.

```
v <- sapply(seq(70, 130, 2), VVVol)
plot(seq(70, 130,2), v, type='l', lwd=3, xlab='Strike', ylab='Implied Volatility')
points(sapply(VolData, function(v) v[1]),
          sapply(VolData, function(v) v[2]), pch=19, col='red')
```

The result is shown in Figure **??**.



15.3.2 *Pricing a Binary Option*

Consider a one-year binary call, struck at the money. Assume that the smile is quadratic. Again, we assume a null interest rate for simplicity.

This time, we use the traditional benchmark instruments of the FX market: straddle, risk-reversal and butterfly, and compute the price of the binary option, adjusted for the smile effect.

```
T <- 1
Spot <- 100
r <- 0
d <- 0
b <- r-d
```

```
sigma <- 30/100
X <- 110.50

# smile function
smile <- function(X) (-(0/20)*(X-Spot) + (1/300)*(X-Spot)^2)/100
```

The strikes corresponding to a 25Δ call and put are computed by inverting the formulae for the Delta of European options. Recall that for a call, the Delta is given by:

$$\Delta = e^{-dT} N(d_1)$$

The strike corresponding to a 25Δ call is therefore:

$$K_{25\Delta} = Se^{-\left(\sigma\sqrt{T}N^{-1}(e^{dT}.25)-(r-d+\frac{\sigma^2}{2})T\right)}$$

```
# strikes at +/- 25 deltas
alpha <- -qnorm(.25*exp(d*T))
Kp <- Spot*exp(-alpha * sigma * sqrt(T)+(r-d+(1/2)*sigma^2)*T)
Kc <- Spot*exp(alpha * sigma * sqrt(T)+(r-d+(1/2)*sigma^2)*T)
```

Define a wrapper function to facilitate calculations on the binary option:

```
O <- function(vol=sigma, spot=Spot) CashOrNothingOption(TypeFlag='c', S=spot,
     X=X, K=100, Time=T, r=r, b=b, sigma=vol)@price
```

The Black-Scholes value, using ATM volatility is:

```
# Theoretical BS value
TV.BS <- O()
print(paste('BS value:', round(TV.BS,2)))
```

```
## [1] "BS value: 31.46"
```

For comparison, we can approximate the binary option with a call spread, giving a value of:

```
# Replication value with call spread
N <- 1000
TV.CS <- N*(GBSOption('c', Spot, X-100/(2*N), T, r, b, sigma+smile(X-100/(2*N)))@price -
            GBSOption('c', Spot, X+100/(2*N), T, r, b, sigma+smile(X+100/(2*N)))@price)

print(paste('Value, approximated by a call spread:', round(TV.BS,2)))
```

```
## [1] "Value, approximated by a call spread: 31.46"
```

We next define the benchmark instruments:

```
# Put
P <- function(vol=sigma, spot=Spot) GBSOption(TypeFlag='p', S=spot, X=Kp,
                           Time=T, r=r, b=b, sigma=vol)@price

# Call
C <- function(vol=sigma, spot=Spot) GBSOption(TypeFlag='c', S=spot, X=Kc,
                           Time=T, r=r, b=b, sigma=vol)@price

# Straddle
S <- function(vol=sigma, spot=Spot) {
  GBSOption(TypeFlag='c', S=spot, X=Spot, Time=T, r=r, b=b, sigma=vol)@price +
  GBSOption(TypeFlag='p', S=spot, X=Spot, Time=T, r=r, b=b, sigma=vol)@price
}

# Risk Reversal
RR <- function(vol, spot=Spot) {
  P(vol, spot)-C(vol, spot)
}

# Butterfly
BF <- function(vol, spot=Spot, beta=1) {
  beta*(P(vol, spot)+C(vol, spot))-S(vol,spot)
}
```

The butterfly must be vega-neutral. This is obtained by solving for β:

```
BF.V <- function(vol, beta) {
  (BF(vol+eps, beta=beta)-BF(vol-eps, beta=beta))/(2*eps)
}

beta <- uniroot(function(b) BF.V(sigma, b), c(1, 1.5))$root
```

Next, we compute the risk indicators for the binary option:

```
O.vega <- Vega(O, sigma)
O.vanna <- Vanna(O, sigma)
O.volga <- Volga(O, sigma)
```

and for the benchmark instruments:

```
S.vega <- Vega(S, sigma)
S.vanna <- Vanna(S, sigma)
S.volga <- Volga(S, sigma)

RR.vega <- Vega(RR, sigma)
```

```
RR.vanna <- Vanna(RR, sigma)
RR.volga <- Volga(RR, sigma)

BF.vega <- 0
BF.vanna <- Vanna(BF, sigma)
BF.volga <- Volga(BF, sigma)
```

By definition the smile cost of the straddle is zero, since it is priced with ATM volatility. For the other two benchmark instruments, the smile cost is the difference between the price with the smile effect and the price at the ATM volatility:

```
# RR and BF cost
RR.cost <- (P(sigma+smile(Kp))-C(sigma+smile(Kc)))-(P(sigma)-C(sigma))
BF.cost <- beta*(P(sigma+smile(Kp))+C(sigma+smile(Kc)))- beta*(P(sigma)+C(sigma))
```

We can now compute the price correction for the binary option. First the approximate method, ignoring the off-diagonal terms in matrix A:

```
# approximate method
CA <- RR.cost * (0.vanna/RR.vanna) + BF.cost*(0.volga/BF.volga)
```

then the more accurate method, solving the 3×3 linear system:

```
# full calculation
A <- matrix(c(S.vega, S.vanna, S.volga,
              RR.vega, RR.vanna, RR.volga,
              BF.vega, BF.vanna, BF.volga), nrow=3)

x <- matrix(c(0.vega, 0.vanna, 0.volga), nrow=3)
w <- solve(A, x)
CF <- t(w) %*% matrix(c(0, RR.cost, BF.cost), nrow=3)
```

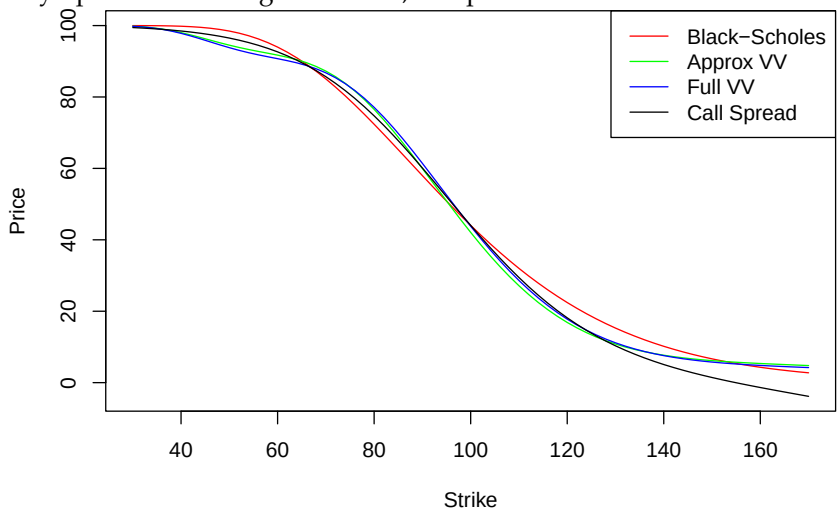
In summary, we get:

- Black-Scholes price: 31.46
- With approximate Vanna-Volga correction: $31.46 + -4.95 = 26.51$
- With acurate Vanna-Volga correction: $31.46 + -3.5 = 27.96$
- the approximation by a call spread is: 28.79

It is worth noting that a naive calculation, where one would plug the ATM volatility plus smile into the binary option pricing model would yield a very inaccurate result:

```
P.smile <- 0(vol=sigma+smile(X))
```

which yields a value of 31.54. Figure **??** compares the values of binary options for a range of strikes, computed with four methods. :



Fixed Income

16 *Elementary Fixed Income Calculations*

```
## Loading required package: fOptions

## Loading required package: timeSeries

## Loading required package: fBasics

## Loading required package: fExoticOptions

## Loading required package: fAsianOptions

## Loading required package: fImport

## Loading required package: RCurl

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

An honest man's word is as good as his bond —
Miguel de Cervantes

This chapter provides a elementary survey of fixed income calculations, with the aim of developing a basic intuition regarding the relationship between the price and the yield of fixed income instruments.

16.1 Bond Pricing

In a simplified framework with even intervals between payments, the present value of a bond with coupon rate c and maturing in n years is given by

$$P(r) = \sum_{i=1}^{n} \frac{cN}{(1+r)^i} + \frac{N}{(1+r)^n}$$

where:

c. coupon rate, in decimal form

N. nominal

n. maturity, measured in years

r. yield, in decimal form

The formula can be implemented in one line:

```
SimpleBondPrice <- function(coupon, n, N=100, yield) {
  N*(sum(coupon/(1 + yield)^(1:n)) + 1/(1+yield)^n)
}
```

As an illustration, consider a bond with 3 years to maturity, a 5% coupon rate. With yield at 4%, price per 100$ nominal amount is

```
P <- SimpleBondPrice(coupon=.05, n=3, yield=.04)
```

or $P = 102.78$.

16.2 The Price-Yield Curves

Figure **??** represents the price-yield relationship for 20 year bonds with coupons ranging from 0 to 15%. In addition to the obvious non-linear nature of the price-yield relationship, it is worth noting that the convexity of the price-yield curve increases with the coupon rate.

```
## plot price-yield as a function of coupon
cp <- c(0, .05, .1, .15)
cl <- c('red', 'green', 'blue', 'black')
y <- seq(0, .20, .01)
n <- 20
p <- matrix(0, length(cp), length(y))
for(i in seq_along(cp)) {
  p[i,] <- sapply(y, function(y) SimpleBondPrice(cp[i], n, yield=y))
}
```

```
plot(y, p[1,], type='l', lwd=2, xlab='Yield', ylab='Price', col=cl[1], ylim=c(min(p), max(p)))
for(i in 2:4) {
lines(y, p[i,], type='l', lwd=2, col=cl[i])
}
legend('topright', paste('cp = ', cp*100, sep=''), cex=.8, lwd=2, bty='n', col=cl)
```

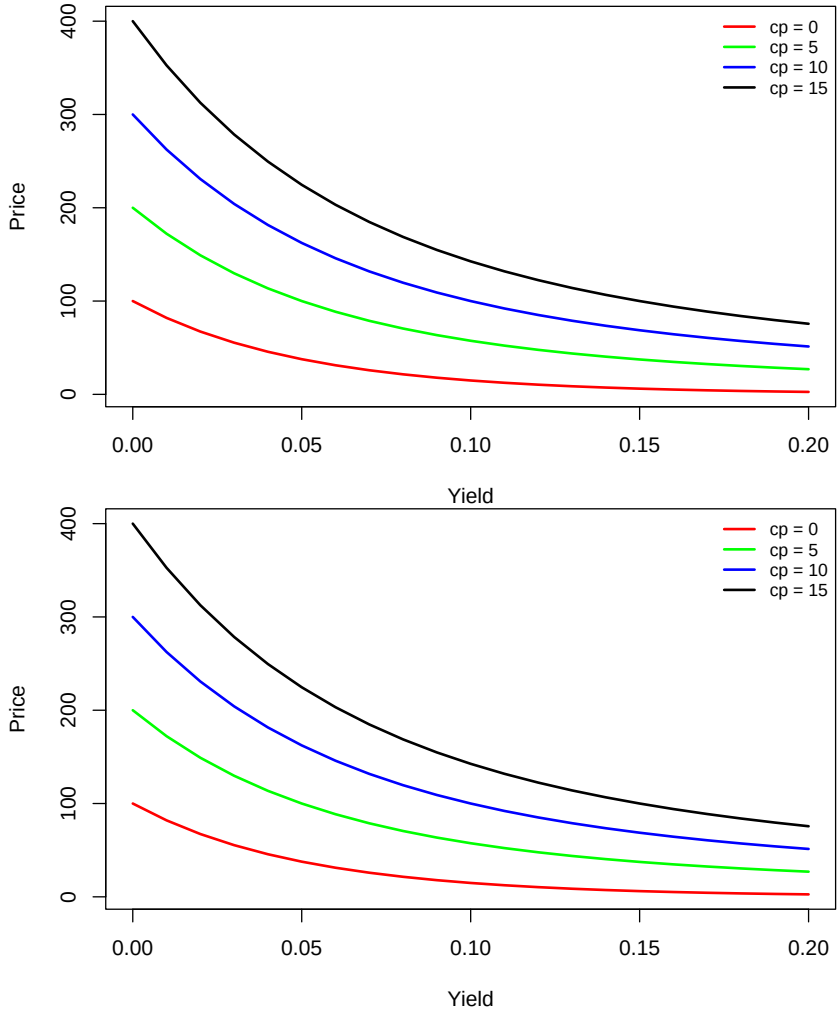


Figure **??** represents the price-yield relationship for 10% bonds with maturities ranging from 3 to 30 years. Regardless of maturity, the bonds are worth 100 when the yield is identical to the coupon. This can be easily established by writing equation (

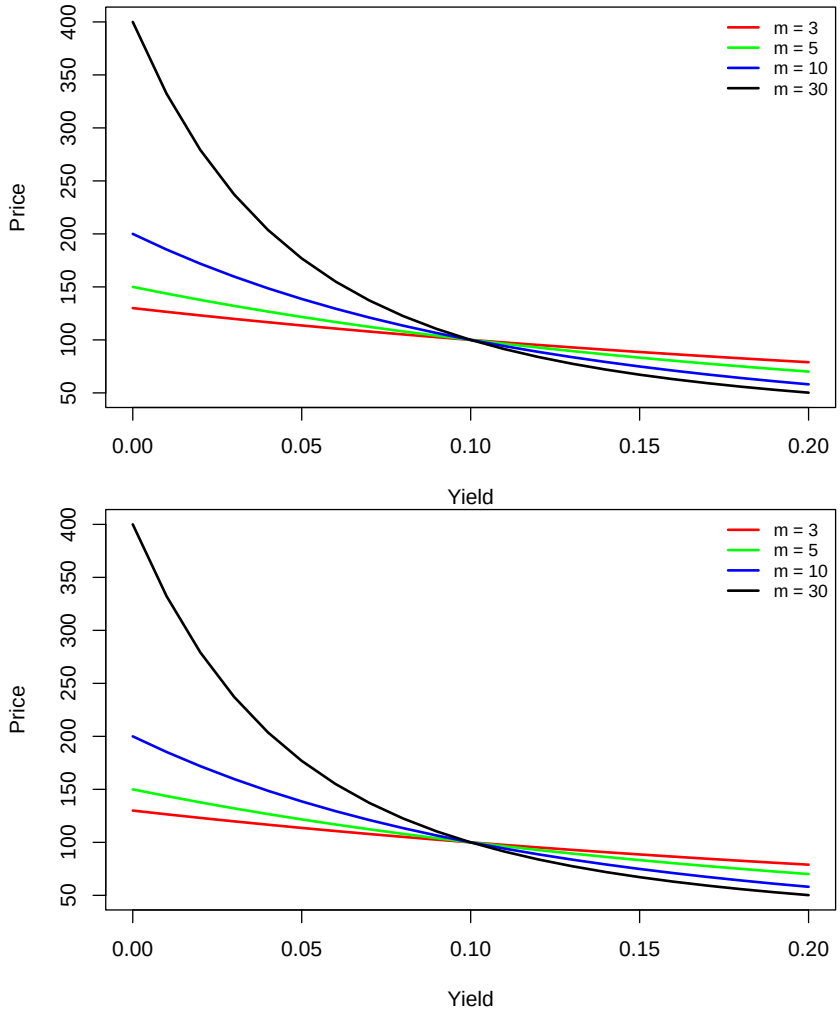
$$eq : fixedinBonds - 1$$

) as:

$$P(r) = N\left[\frac{c}{r}\left(1 - \frac{1}{(1+r)^n}\right) + \frac{1}{(1+r)^n}\right]$$

from which one concludes that $P(r) = N$ when $c = r$.

```
## plot price-yield as a function of maturity
m <- c(3, 5, 10, 30)
cl <- c('red', 'green', 'blue', 'black')
y <- seq(0, .20, .01)
cp <- .1
p <- matrix(0, length(m), length(y))
for(i in seq_along(m)) {
  p[i,] <- sapply(y, function(y) SimpleBondPrice(cp, m[i], yield=y))
}

plot(y, p[1,], type='l', lwd=2, xlab='Yield', ylab='Price', col=cl[1], ylim=c(min(p), max(p)))
for(i in 2:4) {
lines(y, p[i,], type='l', lwd=2, col=cl[i])
}
legend('topright', paste('m = ', m, sep=''), cex=.8, lwd=2, bty='n', col=cl)
```



16.3 Basic Measures of Interest Rate Risk

The interest rate risk of fixed income instruments is traditionally measured by Variation, Sensitivity and Duration.

16.3.1 Variation

Variation V (also known as dollar duration) is the negative of the derivative of price with respect to yield

$$\begin{aligned} V &= -\frac{\partial P(r)}{\partial r} \\ &= \sum_{i=1}^{n} \frac{iF_i}{(1+r)^{i+1}} \end{aligned}$$

```
Variation <- function(coupon, n, N=100, yield) {
  100*sum(sapply(1:n, function(i) ifelse(i<n, i*coupon/(1 + yield)^(i+1),
                                         n*(1+coupon)/(1+yield)^(n+1))))
}
```

Variation is often measured by "PV01", which is the change in bond price for a 1 basis point change in yield:

$$\text{PV01} = V \times 10^{-4}$$

16.3.2 Approximate Portfolio Yield

Variation is a useful tool for computing the approximate yield of a portfolio, given the yield of each bond. Consider two bonds with prices $P_1(r_1)$ and $P_2(r_2)$ and variations $V_1(r_1)$ and $V_2(r_2)$. We look for the common yield r^* such that:

$$P = a_1 P_1(r^*) + a_2 P_2(r^*)$$

Use a first-order Taylor's expansion of price $P_i(r)$ at r_i:

$$P_i(r^*) = P_i(r_i) + (r^* - r_i)V_i(r_i)$$

Replacing $P_i(r^*)$ by this approximation in (

$$eq : fixedinBonds - 2$$

) yields:

$$\sum_{i=1}^{2} a_i(r^* - r_i)V_i(r_i) = 0$$

which gives

$$r^* = \frac{\sum_{i=1}^{2} a_i r_i V_i(r_i)}{\sum_{i=1}^{2} a_i V_i(r_i)}$$

This approximation is valid when the yields r_i are not too far apart, as illustrated by the following calculation.

Consider a portfolio made of two bonds, described in Table

$$tab : fixedinBonds - 1$$

.

Bond	Coupon (%)	Maturity	Quantity	Yield (%)
A	5	5	2	5.20
B	6	7	3	5.60

The approximate portfolio yield is computed as follows:

```
cp <- c(.05, .06); r <- c(.052, .056); T <- c(5,7); a<- c(2,3)
P <- sapply(1:2, function(i) SimpleBondPrice(cp[i], T[i], yield=r[i]))
V <- sapply(1:2, function(i) Variation(cp[i], T[i], yield=r[i]))

r <- sum(r*V*a)/sum(V*a)
```

which gives an approximate yield of $r = 5.47\%$. An exact calculation involves solving for r^* the non-linear equation

$$P = \sum_{i=1}^{2} a_i P_i(r^*)$$

```
fun <- function(r) {
  sum(P) - sum(sapply(1:2, function(i) SimpleBondPrice(cp[i], T[i], yield=r)))}

r <- uniroot(fun, c(.05, .06))$root
```

which gives the exact portfolio yield $r = 5.43\%$.

16.3.3 Sensitivity

The sensitivity S (also called modified duration) is the percent change in price per unit change in yield:

$$S = \frac{1}{P}V = \frac{1}{P(1+r)} \sum_{i=1}^{n} \frac{iF_i}{(1+r)^i}$$

```
Sensitivity <- function(coupon, n, N=100, yield) {
  Variation(coupon,n,N,yield)/SimpleBondPrice(coupon, n, N, yield)
}
```

16.3.4 Modified Duration of a Portfolio

Consider a portfolio made of n bonds, each with nominal amount q_i, price P_i and modified duration $S_i, i = 1, \ldots, n$. Let P be the value of the portfolio, we want to determine the modified duration of the portfolio

$$S = \frac{1}{P}\frac{\partial P}{\partial r}$$

$$\begin{aligned} S &= \frac{1}{P}\sum_i q_i \frac{\partial P_i}{\partial r} \\ &= \frac{1}{P}\sum_i q_i \frac{P_i}{P_i}\frac{\partial P_i}{\partial r} \\ &= \frac{1}{P}\sum_i q_i P_i S_i \\ &= \sum_i \frac{q_i P_i}{P} S_i \end{aligned}$$

The modified duration of a portfolio is the weighted average of the modified durations of the bonds. The weight associated with each bond is the fraction of portfolio value associated with the bond.

16.3.5 Duration

The Macaulay duration D (named after Frederick Macaulay who introduced this risk measure), can be interpreted as the average maturity, weighted by the present value of the cash flow at each payment date.

$$D = \sum_{i=1}^{n} \frac{iF_i}{P(1+r)^i}$$

The Macaulay Duration is closely related to Sensitivity:

$$S = \frac{1}{(1+r)}D$$

```
SimpleDuration <- function(coupon, n, yield) {
  100*sum(sapply(1:n, function(i) ifelse(i<n, i*coupon/(1 + yield)^i,
                      n*(1+coupon)/(1+yield)^n)))/SimpleBondPrice(coupon,n,yield=yield)
}
```

These measures can be used to estimate the approximate impact of yield change on price. Using the previous example, the exact price change caused by a 0.1% yield increase is:

```
d1 <- SimpleBondPrice(coupon=.06, n=10, yield=.051) - SimpleBondPrice(coupon=.06, n=10, yield=.05)
```

or $d_1 = -0.806$. An approximate calculation with Duration yields:

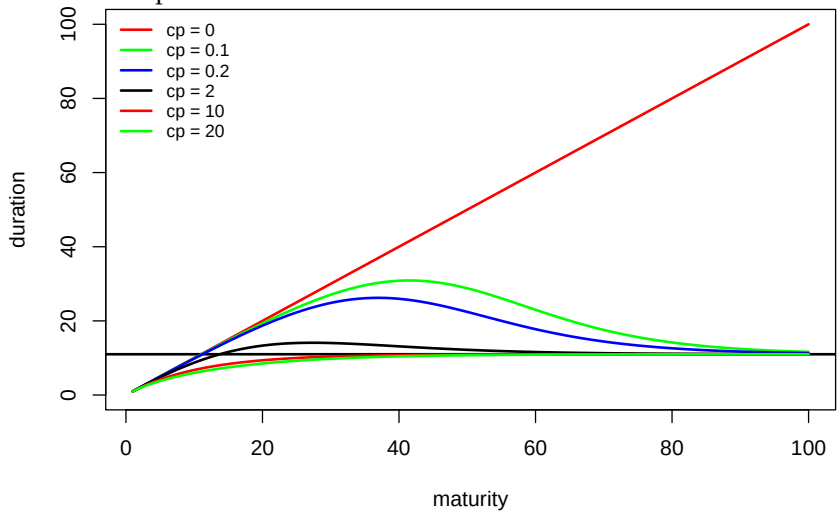
```
P <- SimpleBondPrice(coupon=.06, n=10, yield=.05)
D <- SimpleDuration(coupon=.06, n=10, yield=.05)
d2 <- -P * D/(1+.05) * .001
```

or a change in price of $d_2 = -0.81$ for a .1% increase in yield. Duration is a popular measure of price sensitivity because of its intuitive interpretation: the price sensitivity of a bond with duration D is the same as the one of a zero-coupon bond maturing in D years.

It is a surprising fact that the duration of coupon bonds does not increase indefinitely as maturity increases. The following script computes the duration of bonds of varying maturities and coupon rates.

```
coupon <- c(0, .1, .2, 2, 10, 20)/100
maturity <- seq(from=1, to=100, by=1)
yield <- .1
res <- NULL
for(c in coupon){
  res <- rbind(res, sapply(maturity, function(n) SimpleDuration(c, n, yield)))
  }
```

The resulting values are plotted in Figure **??**. As expected, the duration of the zero-coupon bond is identical to its maturity. For all other bonds, the duration converges to a constant as maturity increases, regardless of coupon rate.



This can be formally shown by first deriving an analytical formula for Duration. Let c be the coupon rate and r the bond yield. The price of a bond with nominal of 1 is:

$$\begin{aligned} P = & \frac{c}{r}(1-(1+r)^{-n})+(1+r)^{-n} \\ = & \frac{1}{r}\left(c(1-(1+r)^{-n})+(1+r)^{-n}\right) \\ = & \frac{1}{r}g(r) \end{aligned}$$

with $g(r) = c(1-(1+r)^{-n})+(1+r)^{-n}$.
Now,

$$\begin{aligned} \frac{\partial \log P}{\partial r} = & -\frac{1}{r}+\frac{g'(r)}{g(r)} \\ = & \frac{1}{P}\frac{\partial P}{\partial r} \end{aligned}$$

But,

$$\begin{aligned} D = & -\frac{1+r}{P}\frac{\partial P}{\partial r} \\ = & (1+r)\left[\frac{1}{r}-\frac{g'(r)}{g(r)}\right] \\ = & \frac{1+r}{r}-(1+r)\frac{g'(r)}{g(r)} \\ = & \frac{1+r}{r}-(1+r)\frac{n(c-r)(1+r)^{-1}+1}{c((1+r)^n-1)+r} \\ = & \frac{1+r}{r}-\frac{n(c-r)+1+r}{c((1+r)^n-1)+r} \end{aligned}$$

The last expression shows that duration converges towards $\frac{1+r}{r}$ as maturity goes to infinity.

16.3.6 Duration of a Portfolio

Consider a portfolio made of n bonds, each with nominal amount q_i, price P_i and duration $D_i, i = 1, \ldots, n$. Let P be the value of the portfolio, what is its Duration? Since Duration is related to sensitivity by the relation

$$D = (1+r)S$$

multiplying both sides of equation

$$eq : mod - duration$$

by $(1+r)$ yields:

$$D = \sum_i \frac{q_i P_i}{P} D_i$$

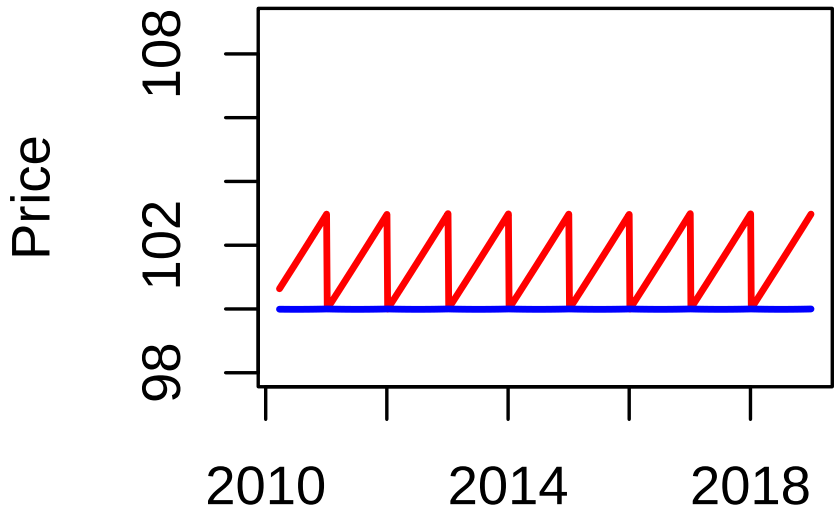
16.4 Bond Clean and Dirty Price

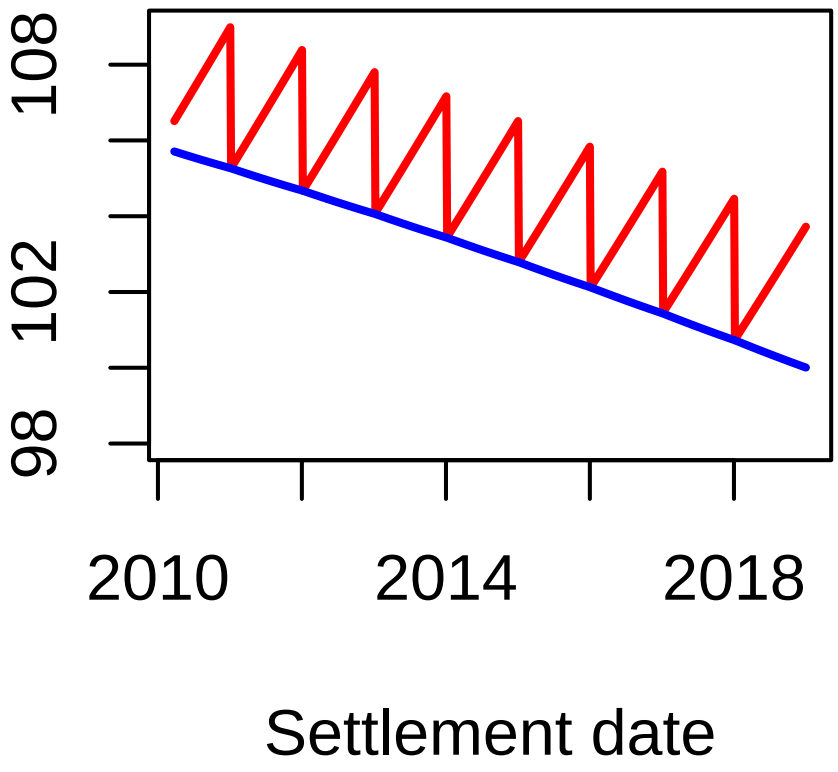
Bonds prices are quoted in "flat" or "clean" price, and not in net present value. The clean price being the net present value less the accrued interest. Therefore, the bond quoted price is not the amount that

the buyer disburses when she buys a bond. The reason for this apparent complication is illustrated in Figure **??** which plots the change over time of a bond clean price and present value, with yield held constant. Quoting a bond by net present value would be impractical, because of the discontinuity around the coupon payment date. The clean price, however, remains steady over time, when yield is held constant.

The plots are obtained as follows, using the function of package :

```
# define a bond
b375 <- Bond('375', myDate('01jan2000'), myDate('04jan2019'), 0.0375, 100, 'a')
b300 <- Bond('300', myDate('01jan2000'), myDate('04jan2019'), 0.0300, 100, 'a')
dtToday <- myDate('25mar2010')
dt <- timeSequence(dtToday, myDate('04jan2019'), by='day')
dt <- dt[seq(1, length(dt),by=4)]

# compute series of clean price, accrued interest and present value
# for given yield
bondcalc <- function(bond, y) {
p <- sapply(dt, function(d) BondYield2Price(bond, d, y))
ac <- rep(NA, length(p))
for(i in seq(length(p))){ac[i] <- BondAC(bond, as.Date(dt[i]))}
cp <- p-ac

list(p=p, ac=ac, cp=cp)
}
bc1 <- bondcalc(b300, .03)
bc2 <- bondcalc(b375, .03)
```




108
102
98
2010
2014
2018
Settlement date

17 *The Term Structure of Interest Rates*

```
  library(fBasics)
  library(empfin)
```

```
## Loading required package: fOptions
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```
## Loading required package: fExoticOptions
```

```
## Loading required package: fAsianOptions
```

```
## Loading required package: fImport
```

```
## Loading required package: RCurl
```

```
## Loading required package: Hmisc
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
  library(tufte)

  data(LiborRates)

  library(YieldCurve)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following object is masked from 'package:timeSeries':
##
##     time<-

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
  data(FedYieldCurve, package='YieldCurve')
  data(ECBYieldCurve, package='YieldCurve')
  data(LiborRates, package='empfin')
```

The relationship between yield and maturity is called the "term structure of interest rate". Since there are many ways to measure yield, there are also many types of term structures. The term structure of interest rate is important because, everything else being equal, maturity is the main determinant of yield. In addition, investors and economists believe that the shape of this curve may reflect the market's expectation of future rates.

17.1 Term Structure Shapes

The term structure of interest rates can assume a variety of shapes, as illustrated by the following 2 examples.

Figure

$$fig : zc - ecb$$

displays a sample of yield curves, published by the European Central Bank (ECB). These curves represent the yield to maturity for AAA rated euro-zone government bonds. We show quarterly data from December 2006 to June 2009. Three patterns are visible:

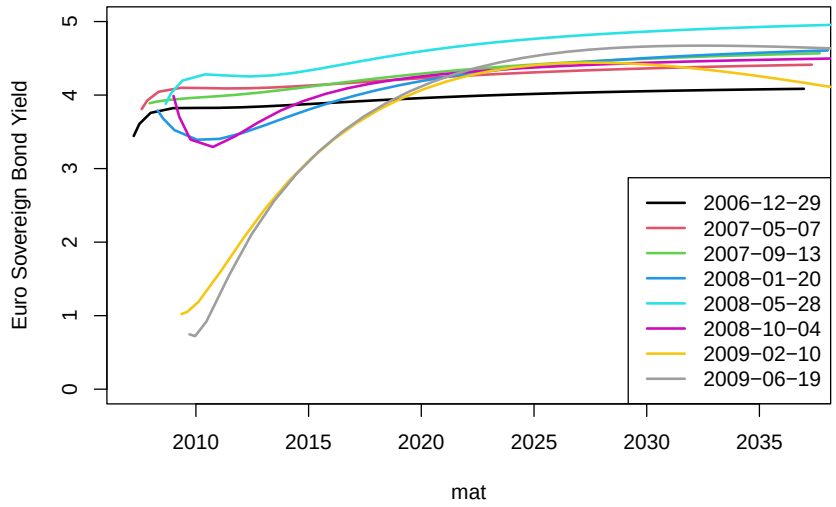
- an almost flat curve (August 2007)
- an inverted curve at the short end, and upward sloping at the long end (January 2008)
- an evenly upward-sloping curve (December 2006, June 2009)

The data is extracted from the package, data set :

```
tau <- c(3/12, 6/12, 1:30)

d1 <- as.Date('2006-12-29')
d2 <- as.Date('2009-07-24')

nbObs <- dim(ECBYieldCurve)[1]
dtObs <- seq(d1, d2, length.out=nbObs)

indx <- seq(1, nbObs, 90)

YC <- ECBYieldCurve[indx,]
dtObs <- dtObs[indx]
nbSample <- length(indx)
```

The plot is generated by:

```
mat = dtObs[1] + (tau *365)
plot(mat, YC[1,],type='l', ylim=c(0,5), ylab='Euro Sovereign Bond Yield', col=1, lwd=2)

for(i in seq(2,nbSample)) {
  mat = dtObs[i] + (tau *365)
  lines(mat, YC[i,],type='l', col=i, lwd=2)
}

legend("bottomright",legend=as.character(dtObs), col=seq(1,nbSample),lty=1, lwd=2)
```

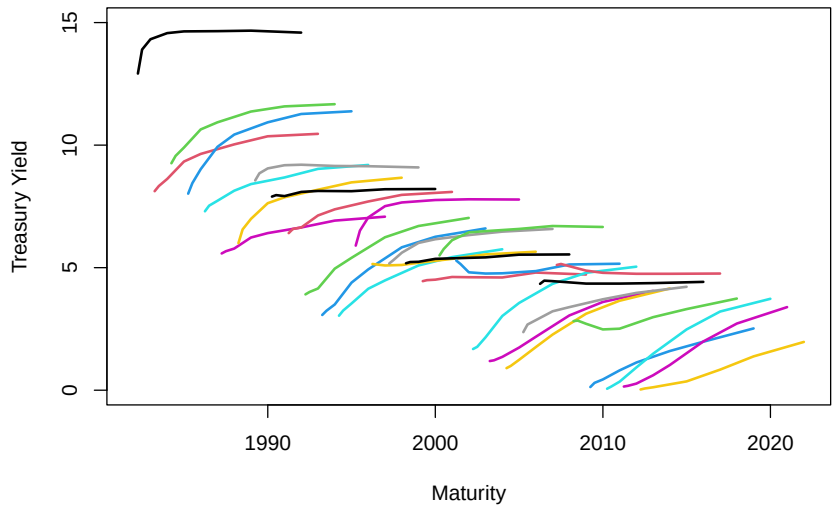


Figure

$$fig : zc - fed$$

presents a similar plot for the US Treasury yield, over the period 1982-2009. This illustrates the wide range of slope that a yield curve can assume over a long period of time. The plot uses the data set , and is obtained as follows:

```
tau <- c(3/12, 6/12, 1, 2, 3, 5, 7, 10)

dtObs <- as.Date(timeSequence(from='1982-01-01', to='2012-11-30', by='month'))

nbObs <- dim(FedYieldCurve)[1]
indx <- seq(1, nbObs, 12)

# monthly sample
YC <- FedYieldCurve[indx,]
dtObs <- dtObs[indx]
nbSample <- length(indx)

mat = dtObs[1] + (tau*365)
plot(mat, YC[1,],type='l', ylim=c(0,15), xlim=c(dtObs[1], dtObs[nbSample]+10*365), col=1, lwd=2, ylab=

for(i in seq(2,nbSample)) {
  mat = dtObs[i] + (tau *365)
  lines(mat, YC[i,],type='l', col=i, lwd=2)
}
```



17.2 *Building a Zero-Coupon Bond Curve*

In a stylized manner, the calculation of a bond zero-coupon yield curve is straight-forward. Assume a set of bonds with cash flows $F_{i,j}$ occurring at regular intervals, where i is the index of the bond and j the index of the cash flow date. We look for an interest rate function $z(t)$ that prices the bonds exactly:

$$P_i = \sum_{j=1}^{n_i} F_{i,j} e^{-z(t_j)t_j}$$

The function $z(t)$ is called the continuously compounded zero-coupon yield curve, or "continuous zero curve" in short.

How to perform the calculation is best explained with an example. Assume that we observe 4 bonds, as summarized in Table

$$tab : zc - bonds$$

.

Maturity	Coupon (%)	Bond Yield (%)
1	9	5.5
2	7	6.0
3	5	7.5
4	4	8.5

Let y_i be the bond yield for bond i maturing in i years, and P_i the corresponding present value.

```
y <- c(5.5, 6.0, 7.5, 8.5)/100
c <- c(9,7,5,4)/100
P <- sapply(seq(4), function(i) SimpleBondPrice(c[i], i, yield=y[i]))
z <- rep(0,4)
r <- rep(0,4)
```

The zero-coupon rates z_i for each maturity are computed recursively, one maturity at a time. The one year zero-rate is obtained by converting the bond yield into a continuously compounded rate:

$$e^{-z_1} = \frac{1}{(1+y_1)}$$

or,

$$z_1 = \ln(1+y_1)$$

which gives $z_1 = 5.35\%$.

The zero coupon rate for year i, with the rates up to year $i-1$ assumed known, is obtained by solving for z_i:

$$P_i = 100\left(\sum_{j=1}^{i-1} ce^{-jz_j} + (1+c)e^{-iz_i}\right)$$

which yields:

$$z_i = -\frac{1}{i}\log\left(\frac{1}{1+c}\left(\frac{P_i}{100} - c\sum_{j=1}^{i-1} e^{-jz_j}\right)\right)$$

This recursive calculation is performed by the function :

```
ZCRate <- function(c, z, i) {
  # compute the ZC rate for maturity i, given the zero-coupon rates
  # for prior maturities
  zc <- -(1/i)*log( 1/(1+c[i]) * ( P[i]/100 - c[i] * sum(exp(-seq(i-1)*z[1:(i-1)]))))
  zc
}

for(i in seq(2,4)) {
  z[i] <- ZCRate(c, z, i)
  }
```

A third term structure of interest is the par yield curve. For a given maturity, the par yield is the coupon rate of a bond priced at par (i.e. priced at 100%). Formally, given a zero-coupon curve, the par yield r_i for maturity i is such that:

$$1 = r_i \sum_{j=1}^{i} e^{-jz_j} + e^{-iz_i}$$

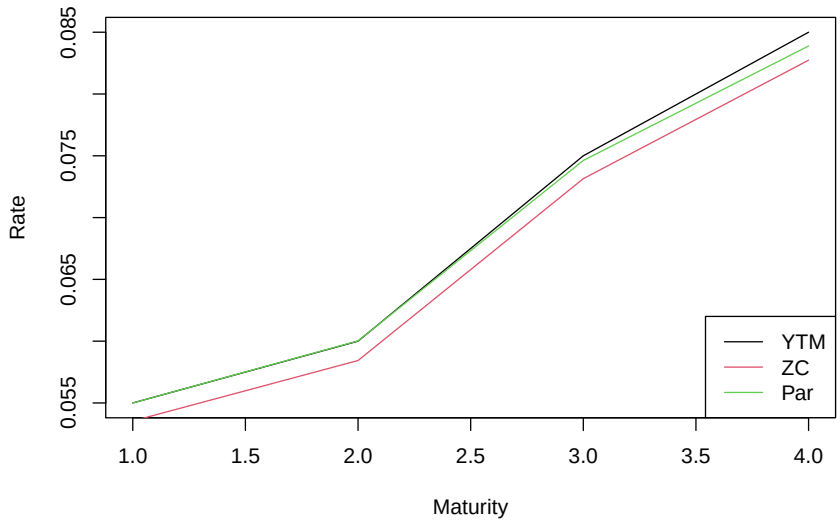
Back to our example, the par yields are computed by the following function, noting that the one year par rate is by definition the corresponding bond yield:

```
ParRate <- function(i) {
  # compute the par rate for maturity i, given the zero-coupon rates for all maturities
  r <- (1-exp(-i*z[i])) / sum(exp(-seq(i)*z[1:i]))
  r
}

r[1] <- y[1]
for(i in seq(2,4)) {
  r[i] <- ParRate(i)
  }
```

Figure

$$fig : zc - 11$$

displays the bond yield curve and the corresponding zero-coupon curve and par curves.



In practice, this simple method is not realistic for several reasons:

- In most markets, we will not find a set a bonds with identical anniversary dates,
- the method outlined above only provides values of $z(t)$ at maturity dates corresponding to the cash flow dates. An interpolation method is needed to obtain $z(t)$ at arbitrary maturities, and finally
- it does not account for the fact that some bonds may trade at a premium or discount: for example, a bond with a small outstanding amount will often trade at a discount, while the current 10 year benchmark bond will trade at a premium.

If we insist on a zero-coupon curve that prices exactly every bond in the sample, we will get a unnatural shape for the zero curve, and even more so for the forward rates.

The alternative is to specify a functional form (sufficiently flexible) for the zero-coupon curve, and to estimate its parameters by minimizing the pricing error over a set of bonds.

Since the term structure of interest rate can equivalently be represented in terms of spot rates, forward rates or discount factors, one has three choices for specifying the functional form of the term structure.

17.3 Functional forms for the zero-coupon curve

A popular method is to model the instantaneous forward rate, $f(t)$. Integrating this function gives the zero-coupon rate. Nelson and Siegel (1987) introduced the following model for the instantaneous forward rate:

$$f(t) = \beta_0 + \beta_1 e^{-\frac{t}{\tau_1}} + \beta_2 \frac{t}{\tau_1} e^{-\frac{t}{\tau_1}}$$

Integrating $f(t)$, the corresponding zero-coupon rates are given by:

$$z(t) = \beta_0 + \beta_1 \frac{1 - e^{-\frac{t}{\tau_1}}}{\frac{t}{\tau_1}} + \beta_2 \left(\frac{1 - e^{-\frac{t}{\tau_1}}}{\frac{t}{\tau_1}} - e^{-\frac{t}{\tau_1}} \right)$$

Svensson (1994) extended this model with 2 additional parameters, defining the instantaneous forward rate as:

$$f(t) = \beta_0 + \beta_1 e^{-\frac{t}{\tau_1}} + \beta_2 \frac{t}{\tau_1} e^{-\frac{t}{\tau_1}} + \beta_3 \frac{t}{\tau_2} e^{-\frac{t}{\tau_2}}$$

The following script, taken from the package, illustrates the fitting of the Nelson-Siegel and Svensson models to US Treasury yields.

```
# maturities in months of FedYieldCurve data set
tau <- c(3, 6, 12, 24, 36, 60, 84, 120)

# parameter estimation
Parameters.NS <- Nelson.Siegel( rate=FedYieldCurve[5,], maturity=tau)
Parameters.S <- Svensson(rate=FedYieldCurve[5,], maturity=tau)

# sample the fitted curves
tau.sim <- seq(1,120,2)
y.NS <- NSrates(Parameters.NS,tau.sim)
y.S <- Srates(Parameters.S, tau.sim, whichRate='Spot')
```

Figure

$$fig : zc - nss$$

shows a comparison of the two fitted spot curves.

18 Simple Interest Rate Derivatives

```
## Loading required package: fOptions

## Loading required package: timeSeries

## Loading required package: fBasics

## Loading required package: fExoticOptions

## Loading required package: fAsianOptions

## Loading required package: fImport

## Loading required package: RCurl

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

In this chapter we cover the pricing of simple interest rate derivatives:

- Forward rate agreements (FRA)
- Euro-Currency Futures
- Short-term (T-Bill) Futures

18.1 Spot and Forward Interest Rates

Definition:

Spot rate. The interest rate for an investment starting today, paid back at a future time T.

Forward rate. The interest rate that applies to an investment to be made in the future.

Forward rates can be infered from spot rates. Let

$B(t,T)$. Discount factor from T to t

$r_{t,T}$. Continuously compounded rate from t to T.

We have:

$$B(t,T) = e^{-r_{t,T}(T-t)}$$

A simplified calculation of forward rates from zero-coupon rates can be performed using the equations:

$$\begin{aligned} B(t_1,t_2) &= \frac{B(0,t_2)}{B(0,t_1)} \\ r_{t_1,t_2} &= -\frac{\ln(B(t_1,t_2)}{t_2-t_1} \end{aligned}$$

The calculation is illustrated by the following example. Table

$$tab : zc - 1$$

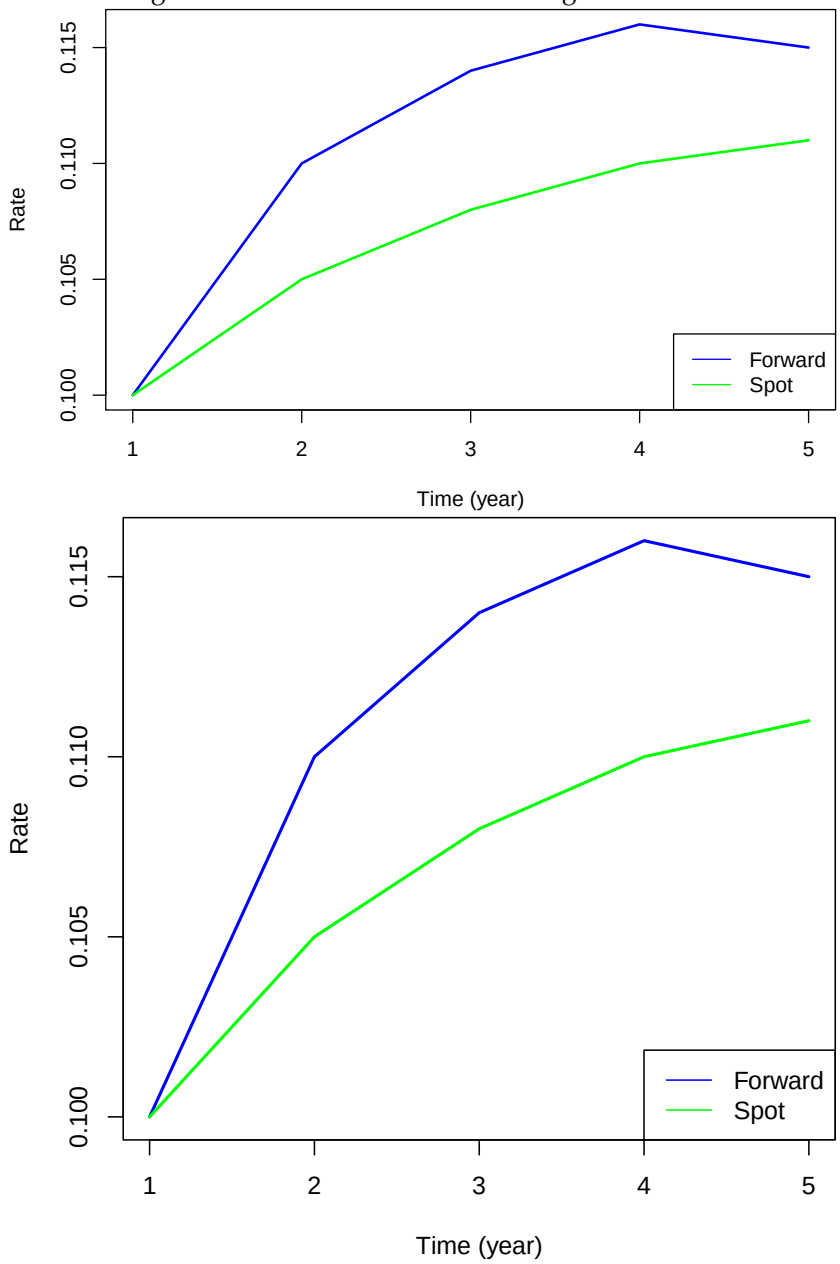
provides a sample term-structure of zero-coupon rates.

Year	Spot Rate
1	10.0
2	10.5
3	10.8
4	11.0
5	11.1

Using these spot rates (continuously compounded), we can compute the 1-year forward rates with the following script:

```
sr <- c(10.0, 10.5, 10.8, 11.0, 11.1)/100
t <- seq(1,5)
B <- exp(-sr*t)
fr <- c(sr[1], -log(B[2:5]/B[1:4]))
```

The resulting forward rate curve is shown in figure **??**.



The basic interest rate derivatives provide a vehicule for locking a short term interest rate a some point in the future. There are many such vehicules:

1. FRA: Forward rate agreement (OTC)
2. T-Bill Futures: Futures on a US 90-day Treasury Bill
3. Euro-currency Futures: Futures on an 3-month interest rate (LIFFE, EUREX)

In addition, Government Bond Futures, to be considered in the next chapter, provide a vehicule for locking a long term future yield.

Currency	Exchange	Future	Contract size	Underlying	Deliverable bonds (TOM in years) *)
EUR	EUREX	Bund-Future	100,000	Bundesanleihe, 10y. 6 %	8.5 – 10.5
EUR	EUREX	BOBL-Future	100,000	Bundesanleihe, 5y., 6 %	3.5 – 5
EUR	EUREX	Schatz-Future	100,000	Bundesanleihe, 2y., 6 %	1,75 – 2,25
EUR	LIFFE	Bund-Future	100,000	Bundesanleihe, 10y., 6 %	8,5 – 10,5
GBP	LIFFE	Long-Gilt Future	100,000	Long Gilt, 7 %	8,75 –13
JPY	TSE	JGB - Future	100 mio	JGB, 20y., 6 %	15 – 21
JPY	TSE	JGB - Future	100 mio	JGB, 10y. , 6	7 – 11
JPY	TSE	JGB - Future	100 mio	JGB, 5 y., 6 %	4 - 5,25
CHF	EUREX	CONF – Future	100,000	Swiss gvt. Bond, 10y., 6 %	8 – 13
USD	CBOT	10-y T-Note	100,000	T-note, 10 y.,6 %	6,5 – 10
USD	CBOT	5-y T-Note	100,000	T-note, 5 y.,6 %	1,75 – 5,25
USD	CBOT	2-y T-Note	200,000	T-note, 2 y., 6 %	4,25 – 5,15
USD	CBOT	T-Bond Future	100,000	T-bond, 30 y., 6 %	mind 15

Figure 18.1: image

@ref(fig:zc-2]

18.2 Forward Rate Agreement (FRA)

Define r_{t_0,t_1,t_2} the interest rate between t_1 and t_2, observed at t_0.

An OTC contract where at time t_0:

1. Buyer to pay fixed rate $r_{t_0,t_1,t_2} = r_f$
2. Seller to pay variable rate $r_{t_1, t_1, t_2} = r_l$

Cash settlement (buyer's perspective) at t_1 is present value of difference between fixed rate and rate observed at $t = t_1$:

$$100 \left[e^{r_l(t_2-t_1)} - e^{r_f(t_2-t_1)} \right] e^{-r_{t_1,t_1,t_2}(t_2-t_1)}$$

Value at time t_0 (from buyer's perspective):

$$100 \left[e^{r_l(t_2-t_1)} - e^{r_f(t_2-t_1)} \right] e^{-r_{t_0,t_0,t_2}(t_2-t_0)}$$

Value is zero if

$$r_f = \frac{r_{t_0,t_2} t_2 - r_{t_0,t_1} t_1}{t_2 - t_1}$$

Recall that the forward rate r_{t,t_1,t_2} is defined by

$$r_{t,t_1,t_2}(t_2 - t_1) + r_{t,t_1}(t_1 - t) = r_{t,t_2}(t_2 - t)$$

Thus, the present value of the FRA can be computed by assuming that the current forward rate will be realized.

As an example, assume today is 1Apr2010, and a firm needs to borrow:

1. 100 M €

2. from 1dec2010 to 1dec2011

Hedge:

1. Buy a "9x12" FRA.

2. $r_f = 3\%$ for 12 months, start in 9 months.

Table

$$tab : FRA$$

summarizes the outcomes under two rate scenarios at horizon. In all cases, the effective borrowing rate is 3%.

	Case 1	Case 2
r_f	.02	.04
Settlement	$100\frac{2\%-3\%}{1.02} =$	$100\frac{4\%-3\%}{1.04} =$
PV of Interest	- 1.96	-3.84
FRA Settlement	-.98	.96
PV of Effective Rate	-2.94	-2.88
Effective Rate	$\frac{2.94}{1.02} = 3.0$	$\frac{-2.88}{1.04} = 3.0$

18.3 T-Bill Futures

A futures contract on a short rate instrument.

1. Underlying asset: 90-days TB

2. Notional amount: USD 1 million

Price of underlying T-Bill:

$$V_t = 100e^{-r_{t_0,t_2}(t_2-t)}$$

Futures price:

$$F_t = V_t e^{r_{t_0,t_1}(t_1-t_0)}$$

or,

$$F_t = 100e^{-r_{t_0,t_1,t_2}(t_2-t_1)}$$

18.3.1 Arbitrage

Simplified example, all rates continuously compounded.

1. 45-day T-Bill rate is 10%
2. 135-day T-Bill rate is 10.5%
3. implied rate from T-Bill Futures: 10.6%

Arbitrage:

1. Sell Futures contract
2. Borrow fund for 45 days at 10%
3. Invest for 135 days at 10.5%

18.3.2 Quotes

Quotation in yield:

$$F_t = 100 - \text{TB yield}$$

Cash price calculation:

$$100 - \frac{90}{360}\text{TB Yield}$$

Price of Futures converges towards price of underlying T-Bill.

Buy a TB futures quoted 96.83. (Buy forward the underlying T-Bill at a yield of

$$y_t = (100 - 96.83) = 3.17\%$$

Delivery price:

$$1,000,000(100 - 3.17\frac{90}{360}) = 992,075$$

P&L at maturity: difference between delivery price and current T-Bill price. if yield is now 4%:

T-Bill price:

$$1,000,000(100 - 4.00\frac{90}{360}) = 990,000$$

P&L = -2075.

18.4 Euro-Currency Futures

Euro-currency Futures are Futures contracts on rates, not on price as the T-Bill contract. The price converges to underlying interest rate.

Three-month Euribor contracts are quoted on EUREX.

1. 1 million € notional
2. Cash settlement, one day after Final Settlement Day
3. Price F_t = 100 - 3-month Euribor rate

The EuroDollar Futures contract quoted on the CME has similar features.

Invoice amount (Euribor)

$$10,000(100 - \frac{1}{4}(100 - F_t))$$

18.5 Eurodollar Rate Versus Forward Rate

We next provide a simple method for calculating the spread between the Eurodollar rate and the forward rate. This spread is a direct consequence of the impact of discounting mentioned earlier.

The PV of a forward swap that receives fixed is:

$$PV = (r_f - r_l)\frac{n}{360}B(0,T)$$

With:

r_f. Fixed rate

r_l. Floating rate

n. tenor in days

$B(0,T)$. discount factor from the swap payment date to today.

The change in PV resulting from a change Δr_l in the floating rate and $\Delta B(0,T)$ in the discount factor is

$$\Delta PV = -\Delta r_l \frac{n}{360}(B(0,T) + \Delta B(0,T))$$

The corresponding change in value of a Eurodollar contract is

$$\frac{90}{360}\Delta r_l$$

The quantity of Eurodollar futures needed to hedge against changes in r_l is

$$H = -\frac{n}{90}B(0,T)$$

To eliminate a riskless profit, the swap position hedged with eurodollar contracts must have a zero expected profit:

$$E\left[\Delta r_l(B(0,T)+\Delta B(0,T))\right] = e\left[B(0,T)(\Delta r_l + s)\right]$$

Solve for the spread s:

$$E\left[s\right] = E\left[\Delta r_l \frac{\Delta B(0,T)}{B(0,T)}\right]$$

Finally,

$$E\left[s\right] = \sigma(r_l)\sigma(\frac{\Delta B(0,T)}{B(0,T)}) < r_l, \frac{\Delta B(0,T)}{B(0,T)} >$$

This adjustment applies to each quarter. The cumulative adjustment can be significant for long-dated forward rate.

18.6 Convexity Adjustment

Calculation of expected forward rate. In a risk-neutral world, the expected price of a bond is its forward price. But the expected rate is not the forward rate, because of the non-linear relationship between rate and prices.

Define:

y_t. Forward bond yield observed at time t for forward contract with maturity T

$B_T(y_T)$. Bond price at time T, function of its yield

σ_T. Volatility of forward bond yield

A Taylor expansion of $B_T(y_T)$ around y_0 gives:

$$B_T(y_T) = B_T(y_0) + (y_T - y_0)G'(y_0) + \frac{1}{2}(y_T - y_0)^2 G''(y_0)$$

Let $E_T()$ be the forward risk-neutral expectation. Applied to both sides, we get:

$$E[B_T(y_T)] = B_T(y_0) + E(y_T - y_0)G'(y_0) + \frac{1}{2}E[(y_T - y_0)^2]G''(y_0)$$

By definition of the expectation,

$$E[B_T(y_T)] = B_T(y_0)$$

Thus:

$$E(y_T - y_0)G'(y_0) + \frac{1}{2}E[(y_T - y_0)^2]G''(y_0) = 0$$

We use the approximation

$$E[(y_T - y_0)^2] = \sigma_T^2 y_0^2 T$$

and obtain after after some algebra:

$$E(y_T) = y_0 - \frac{1}{2}\sigma_T^2 y_0^2 T \frac{G''(y_0)}{G'(y_0)}$$

18.6.1 Special Case of Natural Time Lag

Consider an interest rate derivative where the payoff depends on a τ-period rate, and where the same duration τ occurs between the observation of the rate and the occurence of the payoff. In such case, τ is named a "natural time lag". This is the case for many vanilla interest derivatives such as LIBOR swaps. In this case, the need for a convexity adjustment vanishes, as illustrated by the following example:

Let R_T be the interest rate, maturity $T + \tau$ observed at time T. It determines a cash flow $R\tau$ also to be paid at time $T + \tau$. The present value of this cash flow at time T is:

$$\frac{T\tau}{1 + R\tau} = 1 - \frac{1}{1 + R\tau}$$

Let F be the forward rate between T and $T + \tau$ and B_T the corresponding forward bond price. By definition of the forward rate:

$$B_T = \frac{1}{1 + F\tau}$$

By definition of the risk neutral forward risk measure:

$$B_T = E[\frac{1}{1 + R\tau}]$$

Thus:

$$E[\frac{1}{1 + R\tau}] = \frac{1}{1 + F\tau}$$

Finally,

$$\begin{aligned} E[\frac{R\tau}{1 + R\tau}] = & \quad 1 - \frac{1}{1 + F\tau} \\ = & \quad \frac{F\tau}{1 + F\tau} \end{aligned}$$

Thus, instruments that feature a "natural time lag", such as LIBOR swaps and FRA can be priced assuming that the expected futures rate is the forward rate.

18.6.2 Convexity Adjustment Examples

To Do...

19 *Bond Futures*

```
## Loading required package: fOptions

## Loading required package: timeDate

## Loading required package: timeSeries

## Loading required package: fBasics

## Loading required package: fExoticOptions

## Loading required package: fAsianOptions

## Loading required package: fImport

## Loading required package: RCurl

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

The Euro-Bund Futures contract is an agreement to buy at a fixed price, at a future date, a "notional bond", with a 6% coupon at a 10 year maturity.

19.1 Forward Price

The forward price is a "clean price". Let:

P_t^d. Dirty price

P_t^c. Clean price

AI_t. Accrued interest

To compute the forward clean price at time T:

1. No coupon payments between now and T:

$$P_t^c = P_0^d e^{r_{0,T}T} - AI_T$$

2. There is a coupon payment before T:

   1. Compute I: present value of next coupon
   2. Forward clean price:

$$P_T^d = (P_0^d - I)e^{r_{0,T}T} - AI_T$$

```
b1 <- Bond(id='b1', dtIssue=myDate('01jan2000'), dtMaturity=myDate('15dec2018'),
           couponRate=.05, nominal=100, frequency='a')

# price on coupon date
dtSettlement <- myDate('17dec2010')
p <- BondYield2Price(b1, dtSettlement, .05)
ai <- BondAC(b1, dtSettlement)
print(paste('DtSettlement:', dtSettlement, ' Dirty: ', round(p,2), ' AI: ' , round(ai,2),
            ' Clean: ' , round(p-ai,2)))
```

```
## [1] "DtSettlement: 2010-12-17  Dirty:  100  AI:  0  Clean:  100"
```

```
# price in middle of coupon period
dtSettlement <- myDate('17jun2010')
p <- BondYield2Price(b1, dtSettlement, .05)
ai <- BondAC(b1, dtSettlement)
print(paste('DtSettlement:', dtSettlement, ' Dirty: ', round(p,2), ' AI: ' , round(ai,2),
            ' Clean: ' , round(p-ai,2)))
```

```
## [1] "DtSettlement: 2010-06-17  Dirty:  102.46  AI:  2.49  Clean:  99.97"
```

19.2 Delivery against the Futures contract

A short position on a Futures contract is settled by delivering a certain quantity of one of the eligible bonds in exchange for the payment of

the delivery price. The quantity is defined by a conversion factor to account for differences in maturity and coupon.

Table

$$tab : bund - 1$$

shows the eligible bunds for delivery against the June 2010 Euro-bund contract. Data is as of 25-mar-2010:

Bund	Coupon	Maturity	Factor (k_i)	Yield
374	3.75	4/1/2019	0.852328	3.00
382	3.50	4/7/2019	0.828936	3.05
390	3.25	4/1/2020	0.803710	3.10

The actual delivery cost, also called invoice amount is:

$$IP_T = P_T^c \times CF + AI_T$$

where CF is the conversion factor specific to each bond.

The mechanism is illustrated by the following example. On 25-Mar-2010, the June Euro Bund contract trades at 123.24 on Eurex (www.eurexchange.com). Delivery to take place on 10-Jun-2010.

For Bund 374, the accrued interest on 10-Jun-2010 is approximately:

```
ai = 3.00 * as.numeric(myDate('10jun2010')-myDate('04Jan2010'))/365
ai
```

```
## [1] 1.290411
```

On 25-Mar-2010, you sell one Euro-Bund Futures at 123.24. On delivery day, 10-Jun-2010, you decide to use Bund 374 to settle your short position. You receive the delivery price:

```
Ft <- 123.24
k <- .852328
DelPrice <- Ft * k + ai
DelPrice
```

```
## [1] 106.3313
```

We can compute the cost of delivering Bund 374 under various yield scenarios on 10Jun2010.

19.3 The Conversion Factor

The conversion factor is determined so that, for a flat yield curve at 6% and a Futures trading at par (100), all bonds are equivalent, as far as delivery cost is concerned. This can be easily verified: Table [] compares the delivery price to the value of the delivered security for each eligible bond.

19.4 Determination of the Cheapest to Deliver

When the yield curve is not flat at the notional rate (6% for the Euro-Bund), all bonds are no longer equivalent with respect to delivery cost, and the seller will obviously determine the cheapest to deliver. This is simply done by computing the cost associated with using each bond for delivery.

With data from 25Mar2010, the results are as follows:

Inserttable

which shows that delivering bund 374 is the cheapest way to settle a short position.

It can be shown that when yields are below the notional yield, the cheapest to deliver is the bond with the lowest duration. The inverse holds true when yields are above the par yield.

19.5 Implied Repo Rate

An alternate and more accurate method for determining the cheapest to deliver is to compute the implied repo rate for each bond. The CTD bond is the one that maximizes the total return of an arbitrageur that sells the Futures, buys and hold a deliverable bond, and uses that bond to settle his short Futures position. The bond with the highest IRR is the cheapest to deliver.

Let:

P_t^d. Purchase price of bond at time t

F_t. Futures price

c. Conversion factor

ai_t. Accrued interest at time t

P_T^i. Invoice price, i.e. $F_t fc + ai_T$

The implied repo rate is the return from holding the bond to expiry of the futures contract. The price of the Futures is linked to the IRR by the following relation:

$$F_t c + ai_T = P_t^d \left(1 + IRR\frac{n_1}{n_2}\right)$$

Table [] shows the calculation for each deliverable bond, and confirms that Bund 374 is the cheapest to deliver.

19.6 The Delivery Option

The seller has the option to choose the cheapest to deliver. The approximate theoretical value of the futures is

$$F_t = \frac{P_T^C}{fc}$$

Figure [] shows the theoretical price of the Futures for 2 deliverable bonds, as a function of the yield at settlement. The Futures price must ensure that all deliverable bonds have a positive basis (otherwise, there would be a riskless arbitrage). We see from figure [] that this requirement creates a pattern in the choice of CTD. When yield is above the notional coupon, the long maturity bond is the CTD. The inverse holds true when yield is below the notional coupon.

19.7 IR sensitivity of Bund Futures

From equation

$$eq : irr$$

, one can observe that the Futures price F_t is a function of the short rate (). To correctly measure the IR sensitivity of a IR Futures, one needs to consider the replicating portfolio of a cash-carry trade:

1. Today:
    1. Borrow at the IRR from now until settlement of the Futures
    2. Buy the CTD
2. At delivery:
    1. Deliver the CTD
    2. Receive the Invoice amount
    3. Pay the short term loan

19.8 Cash-Carry Arbitrage

1. Today:
    1. Borrow at the IRR from now until settlement of the Futures
    2. Buy the CTD
2. At delivery:
    1. Deliver the CTD

2. Receive the Invoice amount
3. Pay the short term loan

This identifies two sensitivities:

1. Sensitivity to yield of CTD

$$\frac{\partial F}{\partial y} = \frac{1}{c}\frac{\partial P_t^d}{\partial y}\left(1 + IRR\frac{n_1}{n_2}\right)$$

2. Sensitivity to IRR from today to settlement.

$$\frac{\partial F}{\partial IRR} = \frac{P_t^d}{c}\frac{n_1}{n_2}$$

19.9 Hedging a Bond with a Euro-Bund Contract

How to construct a hedge for a bond, using Euro-Bund contracts and EURIBOR contracts is best explained through an example:

You want to hedge 100 million of notional amount of a 15-year to maturity Bund (B_1) with the June Euro-Bund contract. You first compute the PV01 of your asset and get:

Next, you determine the sensitivity of the Bund futures, assuming that Bund 374 will be the CTD, and get:

1. PV01 / bond yield:
2. PV01 / IRR:

To match the PV01 with respect to bond yield, you must sell:

rounded to x Euro-bund June 2010 futures. This creates an exposure to the 3-month rate (IRR from 25mar2010 to 10Jun2010), which is hedged by y million of q x w FRA.

20 *Libor Swaps*

```
  library(fBasics)
  library(empfin)
```

```
## Loading required package: fOptions
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```
## Loading required package: fExoticOptions
```

```
## Loading required package: fAsianOptions
```

```
## Loading required package: fImport
```

```
## Loading required package: RCurl
```

```
## Loading required package: Hmisc
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

An interest rate swap is a contract to exchange the cash flows of a fixed-coupon bond for the cash flows of a floating rate note of identical maturity. The present value of this contract is therefore the difference between the PV of the fixed leg and the PV of the floating leg.

The pricing formulae use the following notations:

PV_x. PV of the fixed leg

PV_l. PV of the floating leg

r_i. zero-coupon rate for maturity t_i

Q. nominal amount

c. periodic fixed coupon

k. next floating coupon

t_i. payment dates, $i = 1, \ldots, n$

20.1 IR Swap Pricing

The pricing formulae use the following notations:

PV_x. PV of the fixed leg

PV_l. PV of the floating leg

r_i. zero-coupon rate for maturity t_i

Q. nominal amount

c. periodic fixed coupon

k. next floating coupon

t_i. payment dates, $i = 1, \ldots, n$

The PV of the fixed leg is:

$$PV_x = \sum_{i=1}^{n} ce^{-r_i t_i} + Qe^{-r_n t_n}$$

Since the PV of a floating rate note is par at each reset date, the PV of the floating leg is:

$$PV_l = ke^{-r_1 t_1^*} + Qe^{-r_1 t_1^*}$$

At inception, both legs must have equal value.

21 *Fixed Income Risk Management*

```
library(linprog)
library(xtable)
library(empfin)
```

In this chapter, we introduce the two broad categories of Fixed Income risk management strategies: immunization and dedication. The problem being addressed is simple: given a liability stream (anticipated future payments, certain or contingent), and given a initial budget, how should the funds be invested so that the proceeds from the investment portfolio may fund the liability, under all foreseeable interest rate scenarios?

The immunization strategy is a two steps process. The manager first identifies the interest rate risk factors, and then constructs a portfolio such that assets and liabilities have the same exposure to the risk factors, therefore, the global position (assets - liabilities) is immunized against movements of the risk factors.

The dedication or cash flow matching strategy takes a different approach, but is conceptually simple. The goal of this strategy is to avoid interest rate risk by constructing a portfolio that will generate in a timely manner the cash flow needed to meet the liability, thereby evacuating interest risk by construction.

In practice, fund managers often use a mix of both strategies. The two strategies are next described and illustrated.

21.1 Immunization

The method is best described by the following elementary example. A fund manager has a certain liability L_T to be paid at time T. The present value of this liability is $L_T(1+r)^{-T}$. The manager wants to invest that budget in a portfolio that will provide the wealth L_T at the required time, regardless of future interest rates.

She constructs a portfolio that generates cash flows F_i at time T_i. Cash flows paid at time $T_i < T$ will be reinvested till T, and those occurring after T will be sold at T. The value of this portfolio at T is

thus:

$$V_T = \sum_i F_i(1+r)^{T-T_i}$$

and must be equal to L_T. Let's compute the Variation of V_T:

$$\frac{\partial V_T}{\partial r} = \sum_i F_i(T-T_i)(1+r)^{T-T_i-1} \quad (21.1)$$

$$= \left[\sum_i F_i T(1+r)^{-T_i} - \sum_i F_i T_i(1+r)^{T_i}\right](1+r)^{T-1} \quad (21.2)$$

$$= V_T\left[T\frac{\sum_i F_i(1+r)^{-T_i}}{V_T} - \frac{\sum_i F_i T_i(1+r)^{T_i}}{V_T}\right](1+r)^{T-1} \quad (21.3)$$

In the last expression, we recognize the definition of Duration ((16.3.5)), so that the Variation of V_T is finally:

$$\frac{\partial V_T}{\partial r} = V_T\left[T-D\right](1+r)^{T-1}$$

where D is the duration of the stream of cash flows F_i. This equation shows that if the stream of cash flows F_i has the same Duration as the investment horizon T, then, in a first-order approximation, the wealth V_T is insensitive to changes in yield. If yield deceases, the lower value of the reinvested cash flows will be offset by the higher present value of the cash flows maturing after T. The converse is true if yield increases. This investment strategy, which aims at preserving wealth at a given investment horizon is called immunization.

How to construct the assets portfolio is described next. Let the liability $V_T = 100$, $T = 5, V_0 = V_T(1+r)^{-T}$. Current yield is $r = .05$. The investment portfolio will be made of two bonds described below:

Table 21.1: Bonds available for immunization

	Coupon	Maturity
A	0.05	3
B	0.06	7

We next determine the quantities q_A and q_B of each bond in the investment portfolio, so that the present value and the Duration of the portfolio match those of the liability. Remember from section

$$sec : duration - of - a - portfolio$$

that the Duration of a portfolio is a weighted average of the Durations of the bonds in the portfolio.

We express these two conditions by the following equations:

$$V_0 = q_A P_A + q_B P_B \tag{21.4}$$

$$T = q_A \frac{P_A D_A}{V_0} + q_B \frac{P_B D_B}{V_0} \tag{21.5}$$

or, using matrix notation:

$$AQ = b$$

with

$$A = \begin{pmatrix} P_A & P_B \\ \frac{P_A D_A}{V_0} & \frac{P_B D_B}{V_0} \end{pmatrix} Q = \begin{pmatrix} q_A \\ q_B \end{pmatrix} b = \begin{pmatrix} V_0 \\ T \end{pmatrix}$$

This system is solved for q_A and q_B to determine the portfolio composition.

```
r <- 0.05
T <- 5
V_0 <- 100 *(1+r)^(-T)
P_A <- SimpleBondPrice(.05, 3, yield=r)
P_B <- SimpleBondPrice(.06, 7, yield=r)
D_A <- SimpleDuration(.05, 3, yield=r)
D_B <- SimpleDuration(.06, 7, yield=r)

A <- matrix(c(P_A, P_B, D_A*P_A/V_0, D_B*P_B/V_0), nrow=2, byrow=TRUE)
b <- matrix(c(V_0, T), nrow=2)
Q = solve(A, b)
q_A <- Q[1]
q_B <- Q[2]
```

The solution is $q_A = 0.24, q_B = 0.51$. To verify the robustness of the selected portfolio, we next simulate the wealth at horizon T, under various scenarios for r.

Let V_A (resp. V_B) be the wealth at time T generated by 1 unit of bond A (resp. B).

$$V_A = \sum_i F_A(t_i)(1+r)^{(T-T_i)}$$

where $F_A(t)$ is the cash flow generated at time t by one unit of bond A. The wealth at time T is the compounded value of the cash flows occurring before T, and the discounted value of the cash flows occurring after T.

```
FutureValue <- function(coupon, n, yield, T) {
  100*sum(sapply(1:n, function(i) ifelse(i<n, coupon/(1 + yield)^(i-T), (1+coupon)/(1+yield)^(n-T))))
```

```
}

r <- seq(.01, .1, .01)
V_A <- sapply(r, function(x) FutureValue(.05, 3, x, 5))
V_B <- sapply(r, function(x) FutureValue(.06, 7, x, 5))
Hedge <- q_A*V_A + q_B*V_B
```

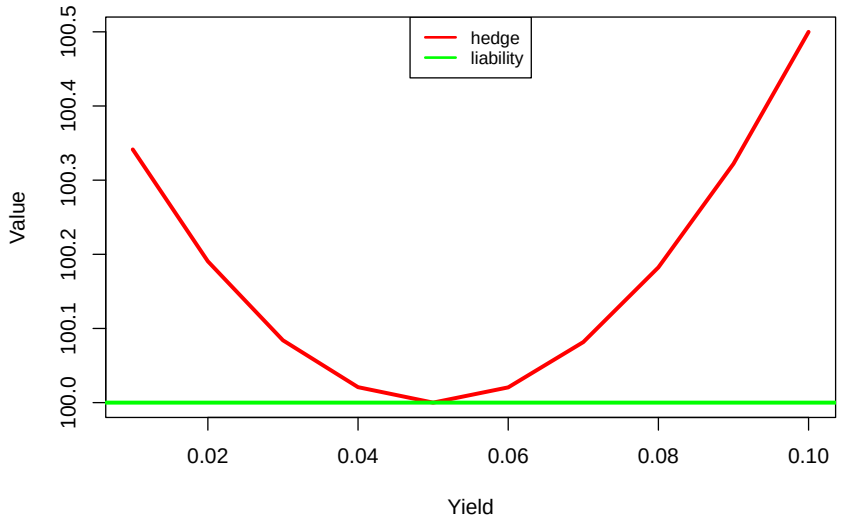


Figure **??** plots the value of the investment portfolio under various yield scenarios. We observe that, under the restrictive conditions of this experiment, the hedge has always a greater value than the liability at horizon T. One should keep in mind, however, that this experiment is done under the assumption that yields for all maturities are identical, and that they all move up or down in parallel. More sophisticated methods address these limitations. The most popular method is to immunize separately various maturity segments. Another approach is to identify several yield curve risk factors, such as a parallel shift, a change in slope and a change in convexity, and immunize the liability against movements along each factor.

A completely different approach is to construct a portfolio that generates the necessary cash flow just in time, so as to minimize the need to re-invest cash that becomes available too early, or sell cash flows that occur too late. This strategy, called Dedication or cash-flow matching, is now described.

21.2 *Dedication*

Assume that, over the next 5 years, you must pay the amounts $L(t), t = 1, \ldots, 5$ summarized in Table **??**. In order to meet this liability, you can invest in a portfolio of 5 bonds described in table 21.3.

Table 21.2: Liability per year

Year	Liability
1	100
2	200
3	150
4	400
5	300

Table 21.3: Bonds available for dedication

Bond	Maturity	Coupon	Yield
1	1	.05	.050
2	2	.07	.075
3	3	.06	.058
4	4	.05	.049
5	5	.08	.081

At every period, you can re-invest excess cash flow at a rate r, chosen conservatively, but cannot borrow. You want, of course, to construct the cheapest portfolio that will match the liability.

To formalize the strategy, we will use the following notation:

q_i. quantity of bond i, purchased at time $t = 0$
$C(t)$. cash balance at time t
$F_i(t)$. cash flow from 1 unit of bond i at time t.
P_i. current price of bond i

The cash balance at each period is the compounded value of the cash balance from the previous period, augmented of the difference between the inflows and the outflows:

$$C(t) = (1 + r)C(t - 1) + \sum_i q_i F_i(t) - L(t)$$

The objective is to determine $q_i, i = 1, \ldots, 5$ and the initial cash balance $C(0)$, so that:

- The cost of the initial portfolio is minimized
- The portfolio is long assets ($q_i >= 0$)
- The cash balance is always positive

The investment strategy can be determined by solving the following linear program:

$$\min \sum_i q_i P_i \tag{21.6}$$

s.t.

$$(1+r)C(t-1) + \sum_i q_i F_i(t) - C(t) = L(t) \tag{21.7}$$

$$q_i >= 0, i = 1, \ldots, n$$

$$C(t) >= 0, t = 1, \ldots, 5$$

There is no need to define an initial cash balance because there is already a one-year bond in the investment portfolio. It is convenient to express the problem in matrix notation:

$$\min P^T x \tag{21.8}$$

s.t.

$$Ax = b$$

$$x >= 0$$

where the column vector x represents the variables of the problem:

$$x = (q_1, \ldots, q_5, c_1, \ldots, c_5)^T$$

and P the vector of asset prices at time $t = 0$:

$$P = (P_1, \ldots, P_5, 0, 0, 0, 0, 0)^T$$

The cash flow definition equations in matrix form are defined with:

$$b = \begin{pmatrix} 100 \\ 200 \\ 150 \\ 400 \\ 300 \end{pmatrix}$$

$$A = \begin{pmatrix} 105 & 7 & 6 & 5 & 8 & -1 & 0 & 0 & 0 & 0 \\ 0 & 107 & 6 & 5 & 8 & 1+r & -1 & 0 & 0 & 0 \\ 0 & 0 & 106 & 5 & 8 & 0 & 1+r & -1 & 0 & 0 \\ 0 & 0 & 0 & 105 & 8 & 0 & 0 & 1+r & -1 & 0 \\ 0 & 0 & 0 & 0 & 108 & 0 & 0 & 0 & 1+r & -1 \end{pmatrix}$$

The program is set up as follows:

```
r <- .01
Amat = rbind(c(105, 7, 6, 5, 8, -1,   0, 0, 0, 0),
             c(0,  107,6, 5, 8,   1+r,  -1, 0, 0, 0),
```

```
              c(0, 0, 106, 5, 8,   0, 1+r, -1, 0, 0),
              c(0, 0, 0, 105, 8,   0,   0, 1+r, -1, 0),
              c(0, 0, 0,   0, 108, 0,   0,   0, 1+r, -1))
```

The right hand side of (21.7):

```
bvec = c(100, 200, 150, 400, 300)
names(bvec) <- c('T1', 'T2', 'T3', 'T4', 'T5');
```

The vector P, the cost of assets at time $t = 0$:

```
y <- .06
cvec = c(SimpleBondPrice(.05, 1, yield=y),
         SimpleBondPrice(.07, 2, yield=y),
         SimpleBondPrice(.06, 3, yield=y),
         SimpleBondPrice(.05, 4, yield=y),
         SimpleBondPrice(.08, 5, yield=y),
         0, 0, 0, 0, 0)
names(cvec) <- c('B1', 'B2', 'B3', 'B4', 'B5', 'C1', 'C2', 'C3', 'C4', 'C5');
```

The program is solved by

```
lp.sol <- solveLP(cvec, bvec, Amat, const.dir=rep('>=', length(bvec)))
```

The optimal portfolio composition is shown in Table 21.4.

Table 21.4: Dedicated portfolio

	Quantity
B1	0.41
B2	1.44
B3	1.04
B4	3.60
B5	2.78

Since we can have as many bonds as time period, and that the A matrix is full row-rank, we have a perfect match between the generated cash flows and the liability stream, at a cost that is therefore exactly equal to the present value of the liability:

```
df <- sapply(1:5, function(i) 1/(1+y)^i)
PV <- sum(df*bvec)
print(paste('dedication cost:', round(lp.sol$opt,2), 'PV of liability:', round(PV,2)))
```

```
## [1] "dedication cost: 939.3 PV of liability: 939.3"
```

Any discrepancy would provide an opportunity for risk-less arbitrage.

Consider now the same problem, but with fewer bonds available to construct the dedicated portfolio (bond 3 is deleted from the set):

```
Amat = rbind(c(105, 7, 5, 8, -1,   0, 0, 0, 0),
             c(0,  107,5, 8,   1+r,  -1, 0, 0, 0),
             c(0, 0, 5, 8,   0, 1+r, -1, 0, 0),
             c(0, 0, 105, 8,   0,   0, 1+r, -1, 0),
             c(0, 0, 0, 108, 0,   0,   0, 1+r, -1))

cvec = c(SimpleBondPrice(.05, 1, yield=y),
         SimpleBondPrice(.07, 2, yield=y),
         SimpleBondPrice(.05, 4, yield=y),
         SimpleBondPrice(.08, 5, yield=y),
         0, 0, 0, 0, 0)
names(cvec) <- c('B1', 'B2', 'B4', 'B5', 'C1', 'C2', 'C3', 'C4', 'C5');

res2 <- solveLP(cvec, bvec, Amat, const.dir=rep('>=', length(bvec)))
```

The optimal portfolio is displayed in table **??**. We no longer have a perfect match, and some cash must be reinvested from period to period, as shown in table 21.5: a cash balance is re-invested from period 2 to period 3.

Table 21.5: Dedicated portfolio

	Period	Cash Balance
C1	1	0.0
C2	2	108.7
C3	3	0.0
C4	4	0.0
C5	5	0.0

The dedication cost ($ 943.86) is naturally higher than the present value of the liability ($ 939.3), since the solution involves reinvesting cash from the second to the third period, at a rate that is lower than the current yield.

Cash-flow matching is a popular strategy for managing short-term liabilities, because it minimizes transactions. It would not be economical to use such strategy for long term liabilities, of for complex cash flows. There is therefore, a natural division of labor between the two strategies that we have sketched.

22 *The Vasicek Model*

```
library(fBasics)
library(empfin)
```

The Vasicek model is interesting on two counts: it is the first equilibrium model of the short rate, and it provides a simple relationship between the dynamic of the short rate and the term-structure of zero-coupon rates. It's limitation is that it cannot exactly fit the observed zero-coupon curve. An extension provided by Hull and White, however, provides that feature.

22.1 Derivatives with Non Traded Underlying Asset

Consider a non traded risk factor x governed by the diffusion:

$$dx = mdt + \sigma dz$$

Consider next two derivatives whose price dynamic is function of the non traded risk factor:

$$\frac{dP_i}{P_i} = \mu_i(x,t)dt + \sigma_i(x,t)dz, i = 1,\ldots,2$$

Consider the portfolio made of $\sigma_2 P_2$ units of asset 1 and $\sigma_1 P_1$ units of asset 2. Its total change in value is:

$$\begin{aligned}
\Delta\Pi = & \quad \sigma_2 P_2 \Delta P_1 + \sigma_1 P_1 \Delta P_2 \\
= & \quad (\sigma_2 P_2)(\mu_1 P_1 \Delta t + \sigma_1 P_1 \Delta z) + \\
& (P_1 + \sigma_1 P_1)(\mu_2 P_2 \Delta t + \sigma_2 P_2 \Delta z) \\
= & \quad (\mu_1 \sigma_2 P_1 P_2 - \mu_2 \sigma_1 P_1 P_2)\Delta t
\end{aligned}$$

Being riskless, the portfolio must earn the riskless rate:

$$\Delta\Pi = r\Pi\Delta t$$

After some algebra, one gets:

$$\frac{\mu_1 - r}{\sigma_1} = \frac{\mu_2 - r}{\sigma_2}$$

Since this is true for two arbitrary securities, it must be true for any security, and therefore,

$$\frac{\mu_i - r}{\sigma_i} = \lambda$$

The constant λ is called the price of risk. With this established, we now go back to derivative security P, function of x. Recall that x is governd by the dynamic:

$$dx = mdt + \sigma_x dz$$

Apply Ito's Lemma to P:

$$dP = \left(\frac{\partial P}{\partial t} + m\frac{\partial P}{\partial x} + \frac{1}{2}\sigma_x^2 \frac{\partial^2 P}{\partial x^2} \right) dt + \frac{\partial P}{\partial x}\sigma_x dz$$

Matching terms in (22.1) and (22.1) we get:

$$\begin{aligned} P\mu = &\quad \frac{\partial P}{\partial t} + m\frac{\partial P}{\partial x} + \frac{1}{2}\sigma_x^2 \frac{\partial^2 P}{\partial x^2} \\ \sigma_P P = &\qquad\qquad\qquad \frac{\partial P}{\partial x}\sigma_x \end{aligned}$$

Now use the price of risk equation:

$$\mu - r = \lambda \sigma_P$$

multiply both sides by P and substitute the terms μP and $\sigma_P P$ to get:

$$\frac{\partial P}{\partial t} + m\frac{\partial P}{\partial x} + \frac{1}{2}\sigma_x^2 \frac{\partial^2 P}{\partial x^2} - rP = \lambda \sigma_x \frac{\partial P}{\partial x}$$

Rearrange terms to get the familiar PDE:

$$\frac{\partial P}{\partial t} + (m - \lambda\sigma_x)\frac{\partial P}{\partial x} + \frac{1}{2}\sigma_x^2 \frac{\partial^2 P}{\partial x^2} = rP$$

This equation is similar to the Black-Scholes PDE for underlying asset that pay a dividend. Set $q = r - (m - \lambda\sigma_x)$ and the equation becomes:

$$\frac{\partial P}{\partial t} + (r - q)\frac{\partial P}{\partial x} + \frac{1}{2}\sigma_x^2 \frac{\partial^2 P}{\partial x^2} = rP$$

In order to price a derivative for which the underlying asset is a non-traded instrument, one needs to know the price of risk corresponding to this risk factor.

22.2 *The Market Price of Interest Rate Risk*

We expect the interest rate market price of risk to be negative. Said otherwise, we expect the forward rate to be greater than the expected future spot rate.

This can be justified by the agents' preference for liquidity: everything else being equal:

1. Investors prefer liquid, short term investments
2. Borrowers prefer fixed rate, long term financing.

Banks provide intermediation between these two classes of agents. By lending at a forward rate that is greater than the expected spot rate, they provide an incentive for borrowers to borrow at a shorter term, and for investors to accept long term comitments.

23 The Model of Hull & White

```
library(fBasics)
```

Hull and White have developed a trinomial model for the short rate, that can be exactly calibrated to the zero-coupon curve. The short rate process is an Ornstein-Uhlenbeck (OU) mean reverting process, and we start this section by a short review of the properties of this process.

23.1 Moments of x_t under a OU Process

To construct a Hull & White model of the short rate, we need to compute the moments of a OU process:

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_T$$

To compute $E(x_t)$, consider the function

$$f(x,t) = xe^{\theta t}$$

Apply Ito's Lemma:

$$\begin{aligned} df &= (\theta x_t e^{\theta t} + \theta(\mu - x_t)e^{\theta t})dt + \sigma e^{\theta t} dW \\ &= \theta\mu e^{\theta t}dt + \sigma e^{\theta t} dW \end{aligned}$$

Integrate:

$$\begin{aligned} x_t e^{\theta t} &= x_0 + \theta\mu \int_0^t e^{\theta u} du + \sigma \int_0^t e^{\theta u} dW_u \\ &= x_0 + \mu(e^{\theta t} - 1) + \sigma \int_0^t e^{\theta u} dW_u \\ x_t &= x_0 e^{-\theta t} + \mu(1 - e^{-\theta t}) + \sigma \int_0^t e^{-\theta(t-u)} dW_u \end{aligned}$$

Thus,

$$E(x_t) = x_0 e^{-\theta t} + \mu(1 - e^{-\theta t})$$

The variance of x_t is now easily computed:

$$\begin{aligned} V(x_t) = & \quad E\left[(x_t - E(x_t))^2\right] \\ = & \quad E\left[\sigma^2\left(\int_0^t e^{-\theta(t-u)}dW_u\right)^2\right] \\ = & \quad \sigma^2 e^{-2\theta t} E\left[\left(\int_0^t e^{\theta u}dW_u\right)^2\right] \\ = & \quad \sigma^2 e^{-2\theta t}\int_0^t e^{2\theta u}du \end{aligned}$$

Finally,

$$V(x_t) = \frac{\sigma^2}{2\theta}(1 - e^{-2\theta t})$$

23.2 *The Hull & White Trinomial Model*

The Hull & White model defines an OU process for the short rate:

$$dr_t = (\theta(t) - ar_t)dt + \sigma dz_t$$

This is an generalization of Vasicek's model, which defined the following process for the short rate:

$$dr_t = a(b - r_t)dt + \sigma dz_t$$

Using results from Vasicek's, one could compute an analytical expression for $\theta(t)$ such that the process is consistent with observed zero-coupon yields. Rather than doing that, however, we will construct a discretized representation of the process with a trinomial tree, and calibrate the parameters of that tree so that the discretized model reprices exactly the zero-coupons.

The tree is constructed in two steps:

1. A trinomial tree is constructed for the process

$$dr_t^* = -ar_t^* dt + \sigma dz_t$$

2. At each time step, the process is shifted to reprice the zero-coupon of that maturity:

$$r_t = r_t^* + \alpha_t$$

Each step is now described.

23.2.1 Step 1

The trinomial tree has time increments Δt. Nodes are evenly spaced with an increment $\Delta r = \sigma\sqrt{3\Delta t}$. Node (i, j) corresponds to time step $i\Delta t$ and rate level $j\Delta r$.

From

$$eq : mean - ou$$

and

$$eq : var - ou$$

,

$$\begin{aligned} E(\Delta r^*) = & \quad r_t^*(e^{-a\Delta t} - 1) \\ = & \quad -ar^*\Delta t \\ V(\Delta r^*) = & \quad \frac{\sigma^2}{2a}(1 - e^{-2a\Delta t}) \\ = & \quad \sigma^2\Delta t \end{aligned}$$

Starting at node (i, j), with rate $r_t^* = j\Delta r$, the rate can move to three values at step $(i + 1) : r_t^* + \Delta r, r_t^*, r_t^* - \Delta r$. We now compute the transition probabilities by matching the first two moments.

$$\begin{aligned} E(\Delta r^*) = & \quad p_u\Delta r - p_d\Delta r \\ = & \quad -ar^*\Delta t \\ = & \quad -aj\Delta r\Delta t \\ V(\Delta r^*) = & \quad p_u\Delta r^2 + p_d\Delta r^2 - E(\Delta r)^2 \\ = & \quad \sigma^2\Delta t \end{aligned}$$

After some algebra, one obtains the following system of linear equations:

$$\begin{aligned} p_u + p_d = & \quad \frac{1}{3} + a^2j^2\Delta t^2 \\ p_u + p_d = & \quad \frac{1}{3} + a^2j^2\Delta t^2 - aj\Delta t \end{aligned}$$

Finally,

$$\begin{aligned} p_u = & \quad \frac{1}{6} + \frac{a^2j^2\Delta t^2 - aj\Delta t}{2} \\ p_u = & \quad \frac{1}{6} + \frac{a^2j^2\Delta t^2 + aj\Delta t}{2} \\ p_m = & \quad 1 - p_u - p_d \end{aligned}$$

This calculation is illustrated below:

```
a <- .1; dt <- 1; s <- .01
```

```
pu <- function(j) {
  (1/6) + (a^2 * j^2 * dt^2 - a*j*dt)/2}

pd <- function(j) {
  (1/6) + (a^2 * j^2 * dt^2 + a*j*dt)/2}

pm <- function(j) {
  1-pu(j) - pd(j)}


j <- 0
print(paste('pu: ', pu(j), ' pm: ', pm(j), ' pd: ', pd(j)))
```

```
## [1] "pu:  0.166666666666667  pm:  0.666666666666667  pd:  0.166666666666667"
```

Sadly, the same calculation at node $(9, -9)$ yields unfortunate results:

```
j <- -9
print(paste('pu: ', pu(j), ' pm: ', pm(j), ' pd: ', pd(j)))
```

```
## [1] "pu:  1.02166666666667  pm:  -0.143333333333333  pd:  0.121666666666667"
```

At this node, we need to change the branching pattern to ensure positive probabilities. We compute now the transition probabilities when the future states of r^* are $r^* + 2\Delta r, r^* + \Delta r, r^*$. The moment-matching equations become:

$$\begin{aligned} E(\Delta r^*) &= p_u 2\Delta r + p_m \Delta r \\ &= -aj\Delta r\Delta t \\ V(\Delta r^*) &= p_u(2\Delta)r^2 + p_m \Delta r^2 - E(\Delta r)^2 \\ &= \sigma^2 \Delta t \end{aligned}$$

After some algebra, one gets:

$$\begin{aligned} p_u &= \frac{1}{6} + \frac{a^2 j^2 \Delta t^2 + aj\Delta t}{2} \\ p_m &= 1 - p_u - p_d \\ p_d &= \frac{7}{6} + \frac{a^2 j^2 \Delta t^2 - 3aj\Delta t}{2} \end{aligned}$$

And we verify that the probabilities are now good:

```
pu2 <- function(j) {
  (1/6) + (a^2 * j^2 * dt^2 + a*j*dt)/2}

pd2 <- function(j) {
  (7/6) + (a^2 * j^2 * dt^2 + 3*a*j*dt)/2}
```

```
pm2 <- function(j) {
  1-pu2(j) - pd2(j)}

j <- -9
print(paste('pu: ', pu2(j), ' pm: ', pm2(j), ' pd: ', pd2(j)))
```

```
## [1] "pu:  0.121666666666667  pm:  0.656666666666667  pd:  0.221666666666667"
```

As a rule, the geometry at each node can be adapted to ensure positive probabilities.

23.2.2 *Step 2*

In the next step, we transform the r^* process into the actual process for the short rate by adding a constant α_i to all nodes at time step $i\Delta t$. The calculation of constants α_i is performed recursively, one time step at a time. We assume that zero-coupon rates are known for each maturity $i\Delta t$. The corresponding zero-coupon prices are:

$$P(0, i\Delta t) = e^{-R(0,i\Delta t)i\Delta t}$$

The short rate at the root of the tree, $r_{0,0}$ is by definition $R(0, \Delta t)$, therefore:

$$\alpha_0 = R(0, \Delta t)$$

Let $Q_{i,j}$ be the price at time o of the Arrow-Debreu security for state (i, j). We now write the price of a zero-coupon maturing at $2\Delta t$.

$$P(0, 2\Delta t) = Q_{1,1}e^{-(\alpha_1+\Delta r)\Delta t} + Q_{1,0}e^{-\alpha_1\Delta t} + Q_{1,-1}e^{-(\alpha_1-\Delta r)\Delta t}$$

which yields:

$$\alpha_1 = \frac{1}{\Delta t}\ln\left[\frac{Q_{1,1}e^{-\Delta r\Delta t} + Q_{1,0} + Q_{1,-1}e^{\Delta r\Delta t}}{P(0, 2\Delta t)}\right]$$

The generic iteration at step i is:

1. Solve for α_i:

$$P(0, (m+1)\Delta t) = \sum_{j=-m}^{m} Q_{m,j}e^{-(\alpha_m+j\Delta r)\Delta t}$$

2. Compute the state prices for time step m:

$$Q_{m+1,j} = \sum_k Q_{m,k}p_{k,j}e^{-(\alpha_m+k\Delta r)\Delta t}$$

23.3 Generalization

The Hull & White model can be generalized to richer short rate processes. For any short rate process of the form

$$df(r_t) = (\theta(t) - af(r_t))dt + \sigma dz_t$$

where $f()$ is invertible, define $x = f(r)$ to get:

$$dx_t = (\theta(t) - ax_t)dt + \sigma dz_t$$

Stage 1 above can be applied to the process dx_t. Stage 2 is slightly modified: let $g(x) = f^{-1}(x)$. The equation to be solved for α_i is now:

$$P(0,(m+1)\Delta t) = \sum_{j=-m}^{m} Q_{m,j} e^{-g(\alpha_m + j\Delta x)\Delta t}$$

Setting $f(x) = \ln(x)$ is particularly interesting, because in does not allow negative interest rates.

24 *Deterministic Time-Varying Volatility*

In some markets, the term structure of volatility has a specific pattern. Many commodities have a seasonal pattern: Natural gas, for example, is in high demand in the winter, but its supply is limited by the capacity of the pipeline network, and this increases price volatility during the heating season. In this chapter, we present and estimate models that are suited to such markets.

24.1 Gabillon's Model

Dynamics of forward prices

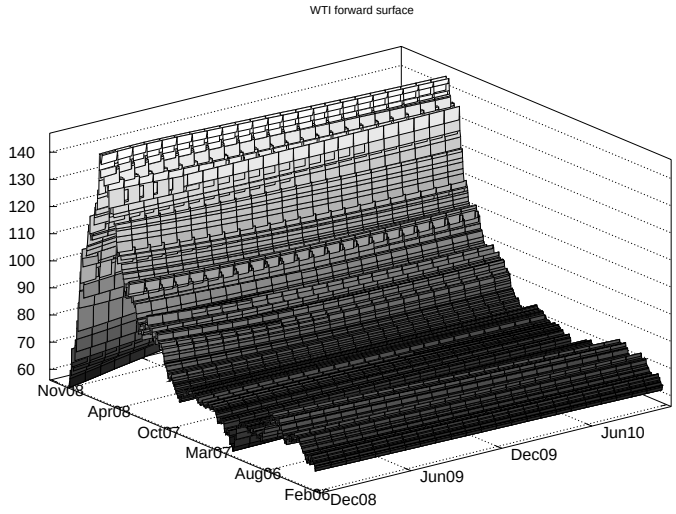


Figure 24.1: image

Let $F(t,T)$ be the value at time t of a futures contract expiring at T. Assume a two factor model for the dynamic of the futures prices:

$$\frac{dF(t,T)}{F(t,T)} = B(t,T)\sigma_S dW_S + (1 - B(t,T))\sigma_L dW_L$$

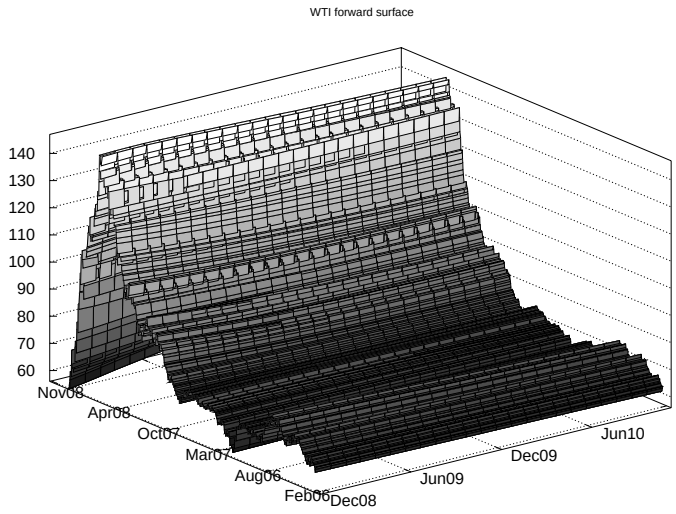


Figure 24.2: image

with

$$\begin{aligned} B(t,T) = &\quad e^{-\beta(T-t)} \\ < dW_S, dW_L > = &\quad \rho \end{aligned}$$

The variance of $\ln(F(t,T))$ is given by:

$$\begin{aligned} \text{Var}[\ln(F(t,T)] = &\quad \sigma_S^2 \int_0^t e^{-2\beta(T-u)} du + \sigma_L^2 \int_0^t (1 - e^{-\beta(T-u)})^2 du + \\ & 2\rho\sigma_S\sigma_L \int_0^t e^{-\beta(T-u)} du \end{aligned}$$

Average variance to expiry is:

$$\begin{aligned} \text{Var}[ln(F(T,T)] &= \tfrac{\sigma_S^2}{2\beta}(1 - e^{-2\beta T}) + \sigma_L^2(T - \tfrac{2}{\beta}(1 - e^{-2\beta T})) + \\ &\quad \tfrac{\rho\sigma_L\sigma_S}{\beta}(1 - 2e^{-\beta T} + e^{-2\beta T}) \\ &= V(T, \sigma_S, \sigma_L, \rho, \beta) \end{aligned}$$

Hybrid Calibration

Estimate ρ and σ_L historically ($\rho = .87$, $\sigma_L = .12$), calibrate σ_S and β to implied ATM volatility.

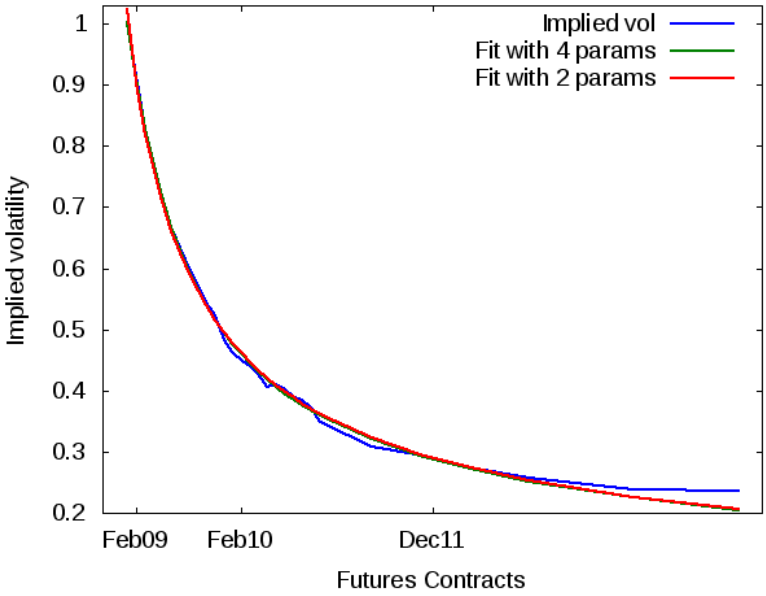
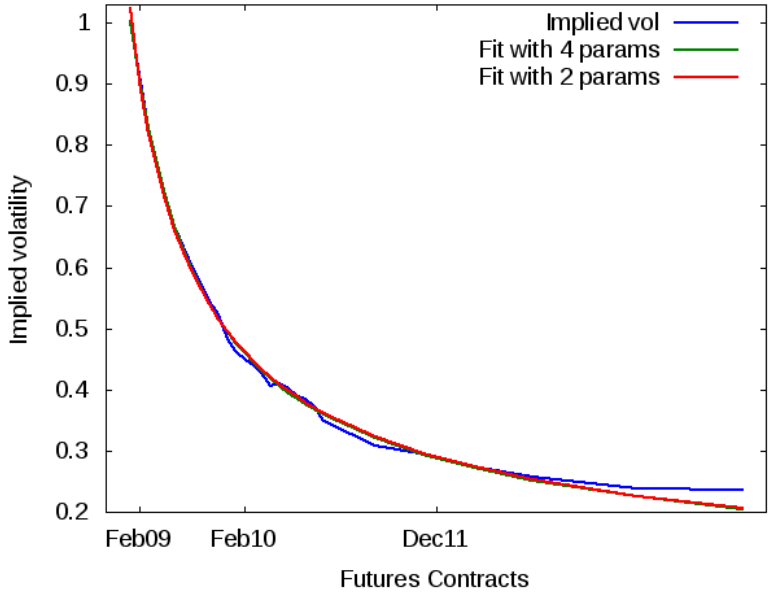


Figure 24.3: image



??

Forward Volatility by Contract

$$\text{Var}[\ln(F(t_1,t_2,T)] = \sigma_S^2 \int_{t_1}^{t_2} e^{-2\beta(T-u)}du + \sigma_L^2 \int_{t_1}^{t_2} (1-e^{-\beta(T-u)})^2 du + 2\rho\sigma_S\sigma_L \int_{t_1}^{t_2} e^{-\beta(T-u)}du$$

24.2 *Deterministic Volatility in a Seasonal Market*

25 *Pricing under an Objective Density*

In this chapter, we consider methods for pricing derivatives under a historical density for the underlying asset.

The goal is to lift the Black-Scholes hypothesis that states that the underlying asset has a log-normal distribution, and use a distribution that better matches the empirical evidences.

Can we transform an empirical density into a density that:

1. is risk-neutral,
2. is consistent with current market observations, and
3. retains the stylized facts of the historical density?

There is a abundant literature on the subject of derivative pricing under empirical densities. See, for example, the review article of Jackwerth (**Jackwerthwinter1999?**), as well as the paper by Derman and Zou (Zou 1999). The principle of Derman's approach is to start with an empirical density for the underlying asset, and to adjust this density in order to make it risk-neutral. Under the risk-neutral density, the expected value of the settlement of a futures contract must be the current forward price. The fair value of a derivative is the discounted expected value of the payoff under the risk-neutral density.

How do we transform an empirical density into a risk-neutral one? The intuition is that we would like to adjust the empirical density in a way that "disturbs" it as little as possible.

Let $P(x)$ be the historical density and $Q(x)$ be the risk-neutral density. The relative entropy $S(P, Q)$ between P and Q measures the increase in information introduced in Q, with respect to P.

The entropy of a random variable X is defined as

$$H(X) = -\sum_i p(x_i) \ln(p(x_i))$$

Therefore, the entropy of a uniform discrete variable with probability $1/N$ associated with each value $x_i, i = 1, \ldots, N$ is $\ln(N)$.

25.1 Conditional Entropy

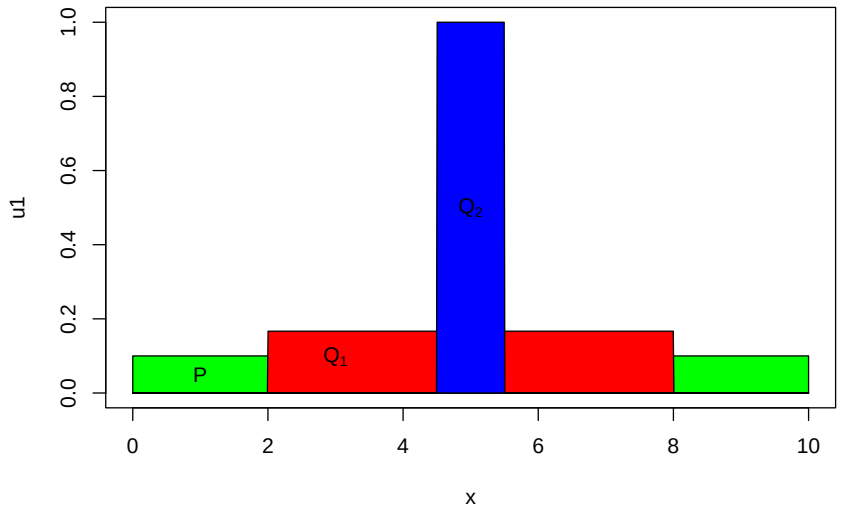
The entropy of a random variable X is defined as

$$H(X) = -\sum_i p(x_i)\ln(p(x_i))$$

Therefore, the entropy of a uniform discrete variable with probability $1/N$ associated with each value $x_i, i = 1, \ldots, N$ is $\ln(N)$.

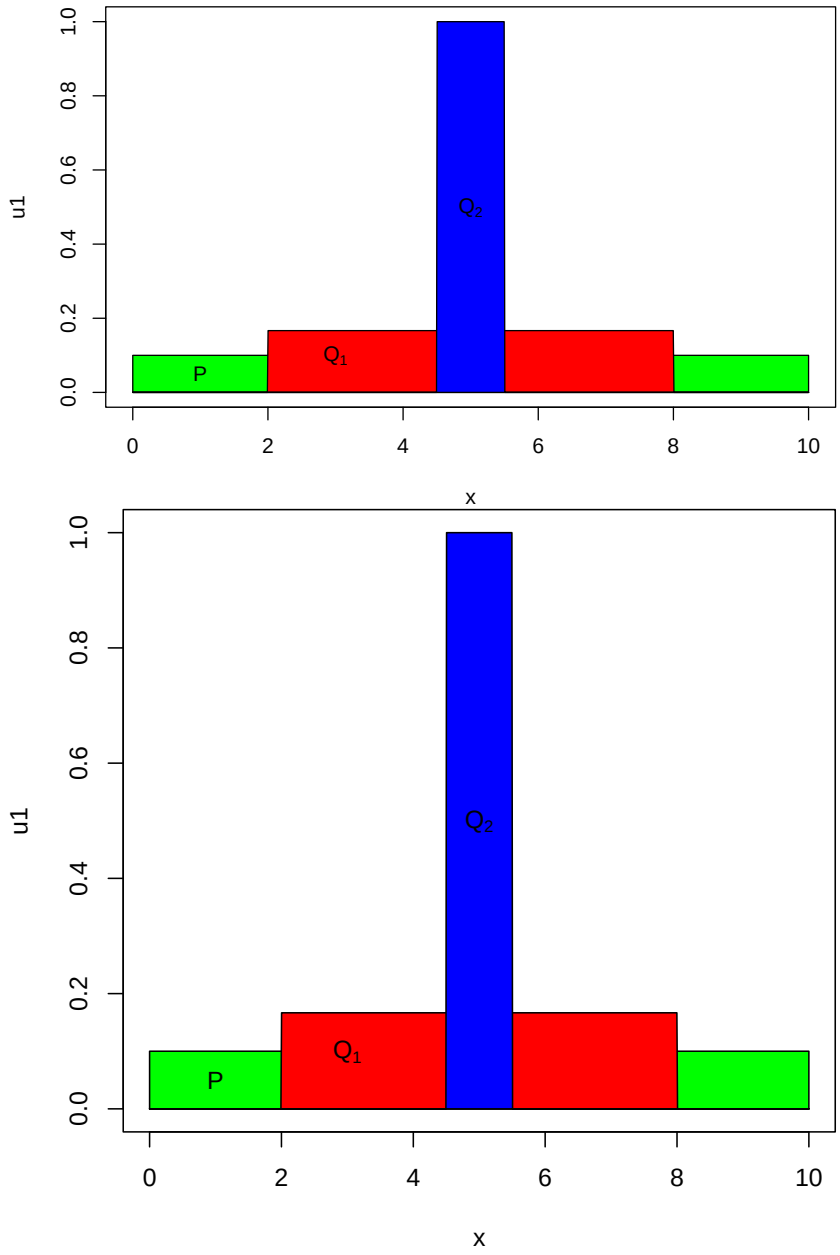
Let P and Q be two random variables. The relative entropy of P and Q measures the closeness between probability distributions over the same set:

$$\begin{aligned} S(P,Q) &= E_Q\{\log Q - \log P\} \\ &= \sum_x Q(x)\log\left(\frac{Q(x)}{P(x)}\right) \end{aligned} \tag{25.1}$$



25.2 Relative Entropy of Uniform Densities

$$M = N^\alpha \quad S(Q,P) = \ln(N)(1-\alpha)$$



Consider for example the distribution P, a uniform distribution over the set $X_N = (x_i, i = 1, \ldots, N)$ and the distribution Q, also uniform over a subset $X_M \in X_N$ of size M:

$$q(x_i) = \begin{cases} \frac{1}{M} & \text{if } x_i \in X_M \\ 0 & \text{otherwise} \end{cases}$$

The distributions are represented in figure **??**. Then

$$S(Q, P) = \ln(N) - \ln(M)$$

Now write $M = N^\alpha$, the relative entropy can be expressed by:

$$S(Q,P) = \ln(N)(1-\alpha)$$

The relative entropy is inversely proportional (in a log scale) to the number of elements that are common to X_M and X_N.

A natural procedure is to compute Q so as to minimize the relative entropy $S(P,Q)$ under the constraint that Q is risk-neutral. We expand upon this idea by computing Q so that Q satisfies an arbitrary number of additional constraints. Specifically, we would like to compute Q such that it exactly prices a number of derivatives. For example, we would like Q to exactly price the at-the-money straddle. Next section describes a method for performing this calculation.

25.3 Computation of a Risk-neutral Density

This formulation is derived from Derman and Zou. We simply expand on their approach in order to calibrate the density Q on prices as well as on implied volatility.

Consider a density P evaluated over n intervals, so that:

$$p_i = P(x_i \leq X \leq x_{i+1})$$

We look for the discrete density Q defined over the sample sampling, which solves the following minimization problem:

$$\begin{array}{ll} \max & -\sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \\ \text{such that} & \\ & \sum_{i=1}^{n} p_i = 1 \\ & \sum_{i=1}^{n} p_i A_j(x_i) = b_j, j = i, \ldots, m \end{array}$$

25.3.1 Calculation of Risk-Neutral Density P

$$\begin{array}{ll} \max & -\sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \\ \text{such that} & \\ & \sum_{i=1}^{n} p_i = 1 \\ & \sum_{i=1}^{n} p_i A_j(x_i) = b_j, j = i, \ldots, m \end{array}$$

The constraints of the second type may be used to calibrate the risk-neutral density to match a variety of market information.

To match a given at-the-money volatility (σ), set:

$$\begin{aligned} A_j(x) &= x^2 \\ b_j &= \sigma^2 + \bar{x}^2 \end{aligned} \tag{25.2}$$

To match the price s of a straddle, set:

$$\begin{aligned} A_j(x) &= \mid x - K \mid \\ b_j &= se^{r(T-t)} \end{aligned} \tag{25.3}$$

25.3.2 *Matching Market Data*

To match a given at-the-money volatility (σ), set:

$$\begin{aligned} A_j(x) &= x^2 \\ b_j &= \sigma^2 + \bar{x}^2 \end{aligned} \tag{25.4}$$

To match the price s of a straddle, set:

$$A_j(x) = \mid x - K \mid \tag{25.5}$$

$$b_j = se^{r(T-t)} \tag{25.6}$$

This formulation may be applied to an arbitrary set of options.
The Lagrangian function is

$$L = -\sum_i p_i \log \frac{p_i}{q_i} + \sum_j \lambda_j (y_j - \sum_i p_i A_j(x_i)) + \mu(1 - \sum_i p_i)$$

The first-order conditions are:

$$\frac{\partial L}{\partial p_i} = -\ln(\frac{p_i}{q_i}) - 1 - \sum_j \lambda_j A_j(x_i) - \mu = 0 \tag{25.7}$$

$$\frac{\partial L}{\partial \lambda_j} = y_j - \sum_i p_i A_j(x_i) = 0 \tag{25.8}$$

$$\frac{\partial L}{\partial \mu} = 1 - \sum_i p_i = 0 \tag{25.9}$$

25.3.3 *Solution*

The Lagrangian function is

$$L = -\sum_i p_i \log \frac{p_i}{q_i} + \sum_j \lambda_j (y_j - \sum_i p_i A_j(x_i)) + \mu(1 - \sum_i p_i)$$

The first-order conditions are:

$$\frac{\partial L}{\partial p_i} = -\ln(\frac{p_i}{q_i}) - 1 - \sum_j \lambda_j A_j(x_i) - \mu = 0 \tag{25.10}$$

$$\frac{\partial L}{\partial \lambda_j} = y_j - \sum_i p_i A_j(x_i) = 0 \tag{25.11}$$

$$\frac{\partial L}{\partial \mu} = 1 - \sum_i p_i = 0 \tag{25.12}$$

Using **??**, we obtain,

$$p_i = q_i e^{(-1-\sum_j \lambda_j A_j(x_i)-\mu)}$$

Divide by $\sum_i p_i$ to eliminate μ and a constant term:

$$p_i = \frac{q_i e^{(-\sum_j \lambda_j A_j(x_i)-\mu)}}{\sum_k q_k e^{(-1-\sum_j \lambda_j A_j(x_k)-\mu)}} \tag{25.13}$$

$$= \frac{q_i e^{(-\sum_j \lambda_j A_j(x_i))}}{\sum_k q_k e^{(-1-\sum_j \lambda_j A_j(x_k))}} \tag{25.14}$$

To calculate λ_j, we use

$$eqn : foc - 2$$

:

$$y_j = \sum_i p_i A_j(x_i)$$

and substitute the expression for p_i:

$$y_j = \frac{\sum_i q_i e^{(-1-\sum_j \lambda_j A_j(x_i))} A_j(x_i)}{\sum_i q_i e^{(-1-\sum_j \lambda_j A_j(x_i))}}$$

To summarize, the values for adjusted probabilities p_i are found to be:

$$p_i = \frac{e^{-\sum_j \lambda_j A_j(x_k)}}{\sum_k e^{-\sum_j \lambda_j A_j(x_k)}}$$

with λ_j solving the system:

$$y_j = \frac{\sum_i q_i e^{-\sum_j \lambda_j A_j(x_i)} A_j(x_i)}{\sum_i q_i e^{-\sum_j \lambda_j A_j(x_i)}}, j = 1, \ldots, m$$

25.3.4 Solution

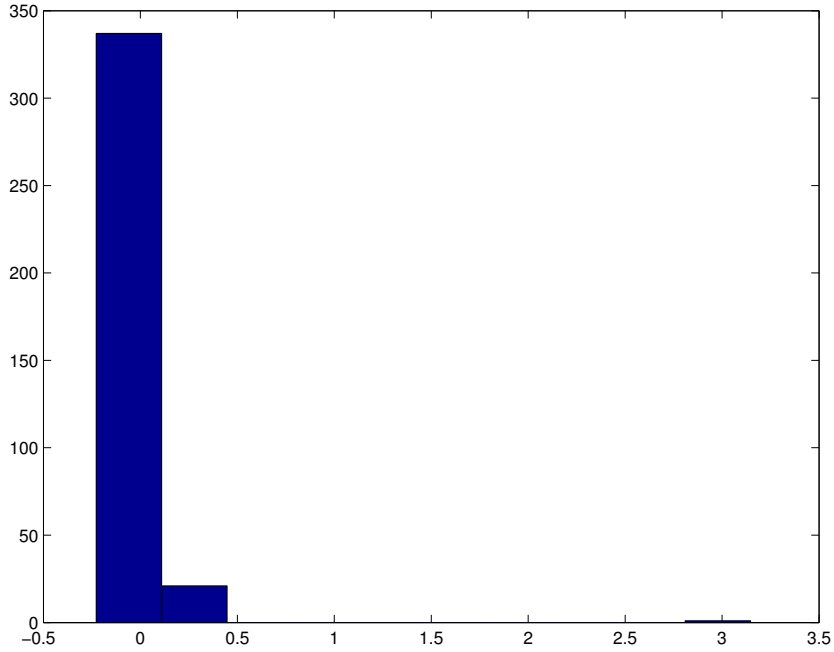
$$p_i = \frac{e^{-\sum_j \lambda_j A_j(x_k)}}{\sum_k e^{-\sum_j \lambda_j A_j(x_k)}}$$

with λ_j solving the system:

$$y_j = \frac{\sum_i q_i e^{-\sum_j \lambda_j A_j(x_i)} A_j(x_i)}{\sum_i q_i e^{-\sum_j \lambda_j A_j(x_i)}}, j = 1, \ldots, m$$

25.3.5 *Illustration*

The method outlined above is sometimes the only practical alternative. Figure **??** represents the distribution of price for the "SoCal basis". This is the difference between the price of natural gas at Henry Hub (the delivery point in Southern Louisiana for the NYMEX Futures contract), and the price in Southern California. In general, the price difference reflects the cost of transporting gas from East to West, say 50 cents per unit of volume. Once in a while, however, there is a disruption in supply and the price differential jumps to $3 or $5. In that context, what is the price of a call option struck at $2?



??

25.3.6 *A Lottery Ticket...*

The density can be further calibrated to available market information. Figure **??** shows the result of the minimum entropy adjustment to the sample distribution to match both the forward price and the value of at-the-money straddles.

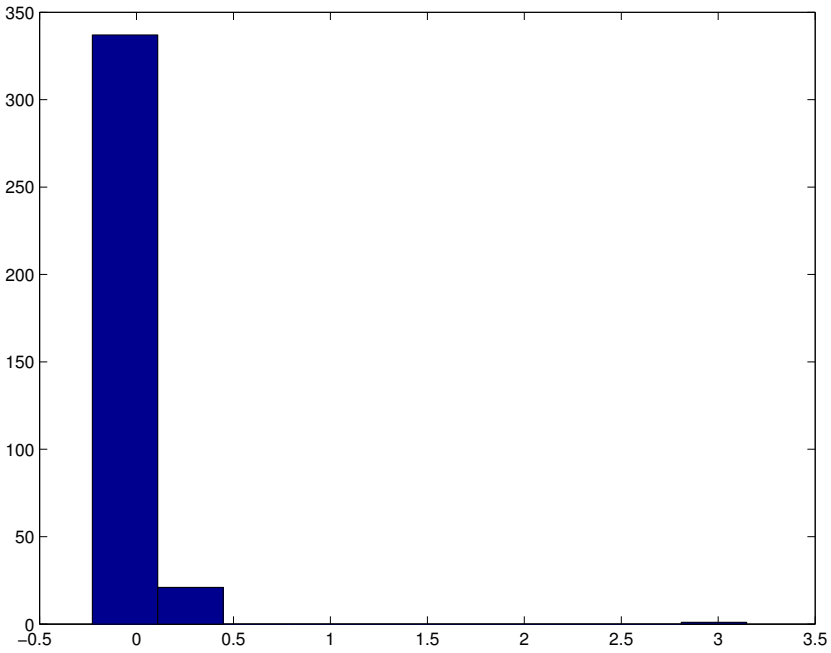
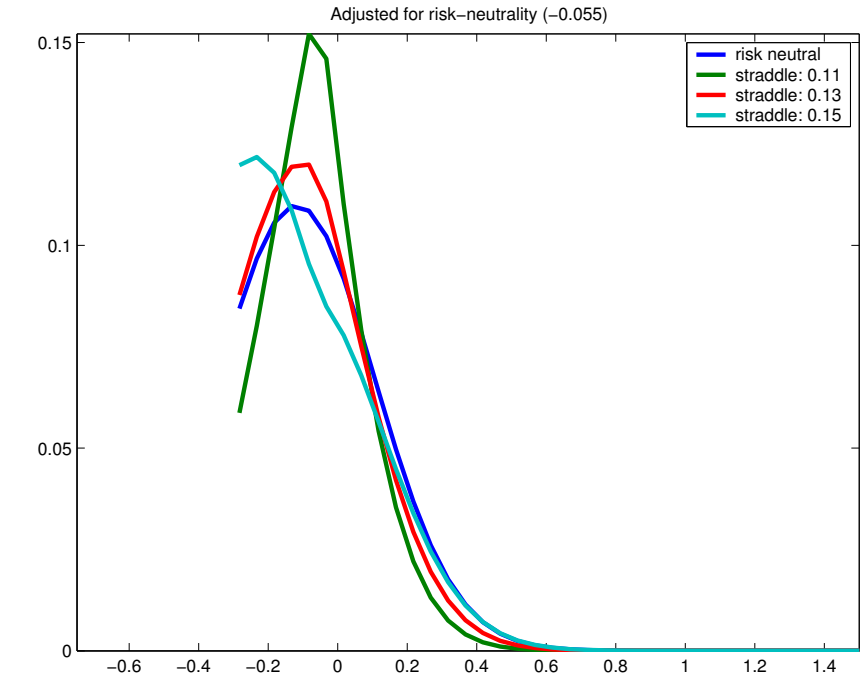


Figure 25.1: image



??

25.4 Monte-Carlo Pricing with Historical Paths

The idea here is to use actual historical paths in a Monte-Carlo simulation. In this section, we summarize the model developed by Potters, Bouchaud and Sestovic (**Marc2001?**).

25.4.1 The Model

Let's first introduce some notation:

x_k. value of underlying asset at step k
$\phi_k(x_k)$. hedge quantity
$C_k(x_k)$. value of derivative

Define the local risk R_k as:

$$E^P\left[(C_{k+1}(x_{k+1}) - C_k(x_k) - \phi_k[x_{k+1} - x_k])^2\right]$$

where $E^P[]$ is the expectation under the objective probability measure.

We look for the pricing function $C_k(x)$ that minimizes the residual hedging risk.

The functions $C(x)$ and $\phi(x)$ are decomposed over an appropriate set of basis functions:

$$C_k(x) = \sum_{a=1}^{M} \gamma_a^k C_a(x) \tag{25.15}$$

$$\phi_k(x) = \sum_{a=1}^{M} \gamma_a^k \frac{\partial C_a(x)}{\partial x} \tag{25.16}$$

Splines provide a convenient set of basis functions. Given a set of knots $t_i, i = 1 \ldots, k$, the spline function for polynomials of degree n is defined by:

$$C(x) = \sum_{j=0}^{n} b_{0,j} x^j + \sum_{i=1}^{k} \sum_{j=0}^{n} b_{i,j} (x - t_i)_+^j$$

Thus, a spline of degree n with k knots is a linear combination of $(k+1)(n+1)$ basis functions. The derivative of $C(x)$ with respect to x is readily computed. To simplify notation, let:

$$C(x) = \sum_{a=1}^{m} \beta_a F_a(x)$$

$$C'(x) = \sum_{a=1}^{m} \beta_a F_a'(x)$$

where $F_a(x)$ defined as above. At each step t in the hedged Monte-Carlo simulation, we obtain the price function by solving for β the following optimization problem (formulation for a call):

where $I(x)$ is the intrinsic value of the derivative being priced. The second constraint enforces the basic non-arbitrage condition that the value of a call must be a monotonically increasing function of the spot.

25.4.2 *Incorporation of bid-ask spread*

A simple modification of the model enables us to account for the bid-ask spread on transactions. We assume that the price paths are mid-market prices, and that transactions are negotiated at the bid or ask price. Let ϵ be the bid-ask spread. The local risk becomes:

$$(C_{k+1}(x_{k+1}) - C_k(x_k) + \phi_k(x_k)(x_k - e^{-\rho}x_{k+1} + \delta\epsilon/2))^2$$

where

$$\delta = \begin{cases} -1 & (x_k - e^{-\rho}x_{k+1}) >= 0 \\ 1 & (x_k - e^{-\rho}x_{k+1}) < 0 \end{cases}$$

25.4.3 *Extension to arbitrary cash flows*

So far, we have only considered contracts with a single payoff at expiry. A simple extension of the model can accommodate arbitrary contingent cash flows at each step of the simulation. Assume that as each time step, the contract generates a cash flow $F(x_k)$. We then define the price function $C(x_k)$ as the contract value ex cash flow. The local risk function is then:

$$(C_{k+1}(x_{k+1}) + F_{k+1}(x_{k+1}) - C_k(x_k) + \phi_k(x_k)(x_k - e^{-\rho}x_{k+1} + \delta\epsilon/2))^2$$

With contingent cash flows at each period, the constraints defined by equations

$$eq : hmc$$

need to be reformulated. Consider first the constraint that the value of the contract must be greater or equal to the intrinsic value of the European option. With multiple cash flows, the equivalent constraint is that the contract value at a given time step and state must be greater than the sum of the expected discounted cash flows, under the conditional probability of this time step and state. The rest of the algorithm is left unchanged.

Cox, J C, Stephen A Ross, and Mark Rubinstein. 1979. "Option pricing: a simplified approach." *Journal of Financial Econometrics* 7: 229–63.

Fowler, Martin. 1996. *Analysis Patterns: Reusable Object Models*. Addison-Wesley.

Genolini, Christophe. 2008. "A (Not So) Short Introduction to S4," 1–68.

Hull, John. 1997. *Options, Futures and Other Derivatives*. Prentice Hall.

Jaeckel, Peter. 2006. "By Implication." *Wilmott Magazine*.

Jarrow, Robert, and Andrew Rudd. 1993. *Option pricing*. Richard D. Irwin.

Joshi, Mark S. 2007. "The Convergence of Binomial Trees for Pricing the American Put." `ssrn.com/abstract=1030143`.

Leisen, D P J, and M Reimer. 1996. "Binomial Models for Option Valuation - Examining and Improving Convergence." *Applied Mathematical Finance* 3: 319–46.

Leland, H E. 1985. "Option Pricing and Replication with Transactions Costs." *The Journal of Finance* 40 (5): 1283–1301.

Nelson, C R, and A F Siegel. 1987. "Parsimonious Modeling of Yield Curves." *Journal of Business* 60: 473–90.

Shkolnikov, Yuriy. 2009. "Generalized Vanna-Volga Method and its Applications." *NumeriX Research Paper*. `http://ssrn.com/abstract=1186383`.

Svensson, Lars E O. 1994. "Estimating and Interpreting Forward Interest Rates: Sweden 1992-1994." *IMF Working Paper* 94/114.

Zou. 1999. "Strike-Adjusted Spread: A New Metric for Estimating the value of Equity Options." *Quantitative Strategies Research Note, Goldman Sachs*.

26 *Index*