Extracting information from FX Options

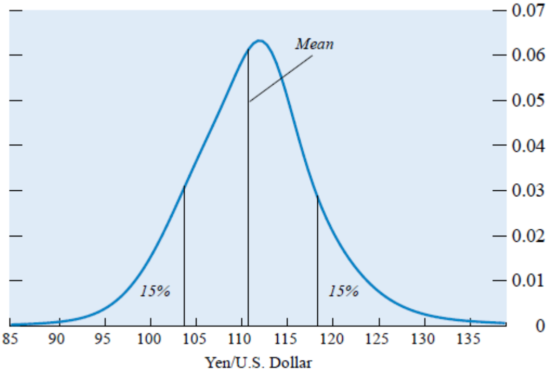
Jorge A. Chan-Lau

February 7, 2018

Objectives

- We want to build this

Figure 50. Distribution for Yen-Dollar Exchange Rate in Early September 1997 Implied by Options Prices on May 20, 1997

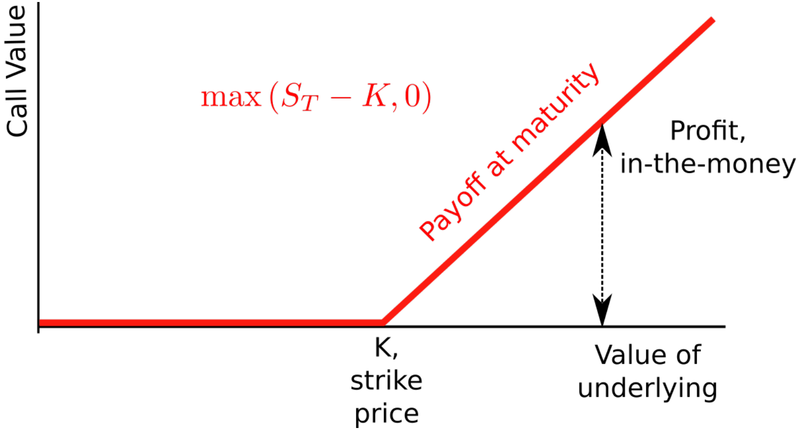


What we will cover

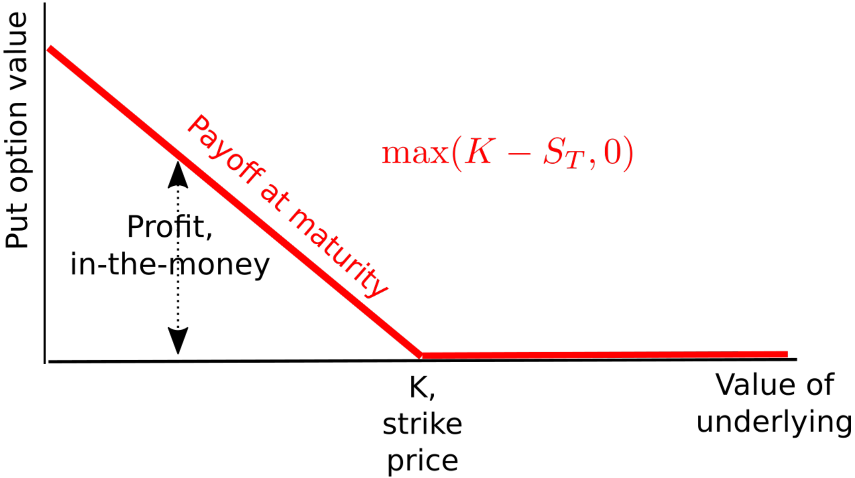
- Learn about basic FX structures
- Develop some intuition about FX option prices
- Get comfortable with a variety of methods
- Side benefit: learn `R` and `RStudio`!
- Lecture notes available at
  https://jchanlauimf.github.io/IMF_FXOptions

Part A:
The Very Basics

Basics: a plain vanilla call option



Basics: a plain vanilla put option



Market reality

- Price quote available for
  - ATM plain options
- Price quotes available for **FX structures**
  - risk reversals
  - butterfly spreads
- Main data sources
  - Bloomberg
  - Reuters
  - Investment banks' data portals

Part B:
Eyeballing the Data

Getting the data in (1)

1. Start RStudio
2. Set the working directory to where we placed the data file
3. Issue this command in the console (replace with your directory)

```
my_wdir = "D:/IMF_FXCourse"  # replace with your directory
setwd(my_wdir)
```

Getting the data in (2)

Clean memory, set up the needed libraries

```
rm(list=ls())                # Clean up memory
library(ggplot2)             # Graphic library
library(lubridate)           # Date manipulation library
library(dplyr)               # Data manipulation library
source("auxFunctions.R")     # Auxiliary functions
```

Getting the data in (3)

```r
filename = "2018_IET_Options_data.csv"
data = read.csv(filename, header=TRUE)
data$Dates = mdy_hm(as.character(data$Dates))
```

2018_ChanLau_Brexit_Case_Study.Rmd × | data ×

Filter

	Dates	Spot	FWD3M	ATM	RR25D3M	BF25D3M	RR10D3M	BF10D3M	Rf	Rd	ImpliedRd
1	2012-12-21	1.61725	1.616769	5.9500	-0.516	0.21	-1.03	0.5750	0.490	0.340	0.4800
2	2012-12-28	1.61705	1.616364	6.4250	-0.539	0.23	-1.03	0.5750	0.530	0.340	0.4135
3	2013-01-04	1.60715	1.606236	6.1000	-0.475	0.21	-1.13	0.5750	0.490	0.330	0.4510
4	2013-01-11	1.61325	1.611847	6.0750	-0.544	0.21	-1.03	0.5690	0.570	0.320	0.4225
5	2013-01-18	1.58680	1.586463	7.0625	-0.680	0.23	-1.28	0.7750	0.560	0.310	0.4410
6	2013-01-25	1.58030	1.579088	6.7800	-0.715	0.23	-1.30	0.8130	0.480	0.405	0.3690
7	2013-02-01	1.56925	1.569613	7.1500	-0.724	0.21	-1.53	0.6000	0.480	0.310	0.3540

The data (1)

FX related

- Spot: the GBPUSD spot exchange rate, i.e. USD per GBP
- FWD3M: the 3-month GBPUSD forward exchange rate
- Rf: the 3-month GBP money market deposit rate, annualized (in percent)
- Rd: the 3-month USD money market deposit rate, annualized (in percent)

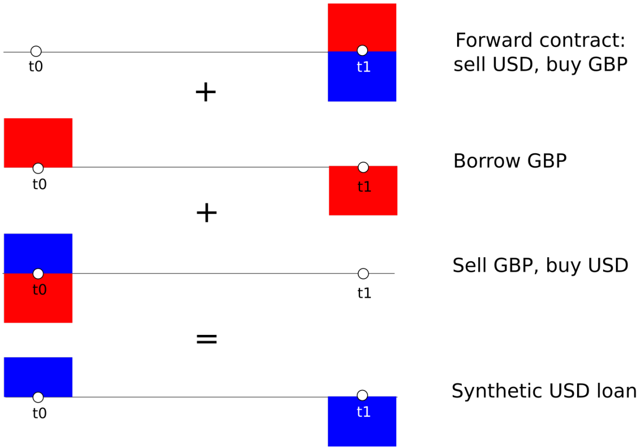
The data (2)

FX option related

- ATM: the at-the-money implied volatility of a GBPUSD option with strike price equal to ATM
- RR25D3M: the price of a 25Δ risk reversal, in annualized volatility units (in percent)
- BF25D3M: the price of a 25Δ butterfly spread, in annualized volatility units (in percent)
- RR10D3M: the price of a 10Δ risk reversal, in annualized volatility units (in percent)
- BF10D3M: the price of a 10Δ butterfly spread, in annualized volatility units (in percent)

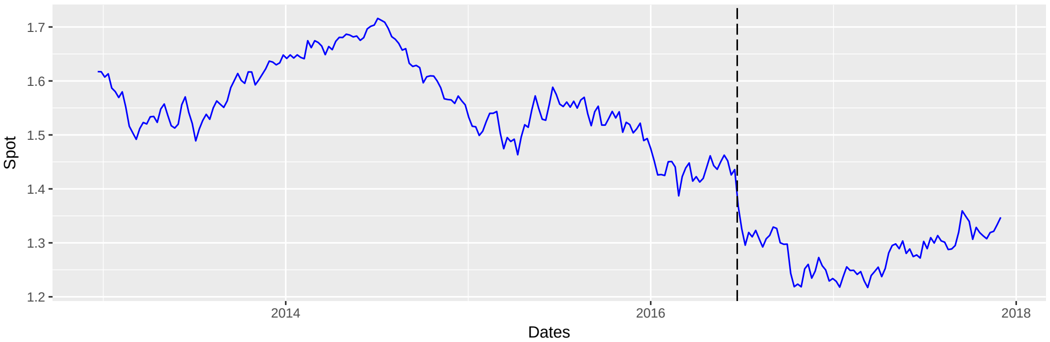
The data (3)

Implied domestic rate from covered interest parity: $F = S\dfrac{\exp\left(\text{Implied } R_d \times T\right)}{\exp\left(R_f \times T\right)}$



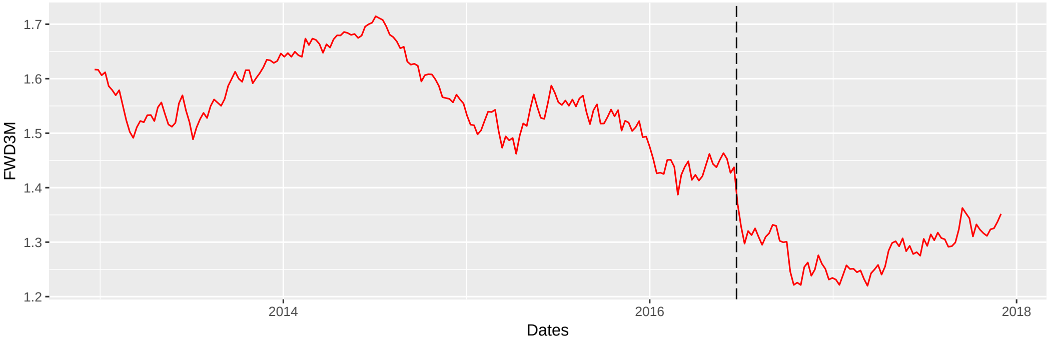
What the data tell us: spot and forward FX rates (1)

```r
# Plot of the spot exchange rate with vertical line at Brexit vote
p1 = ggplot(data, aes(Dates, Spot)) + geom_line(colour="blue")
p1 = p1 + geom_vline(xintercept=as.POSIXct(as.Date(("2016-06-22 UTC"))),
                     linetype="longdash")
p1
```



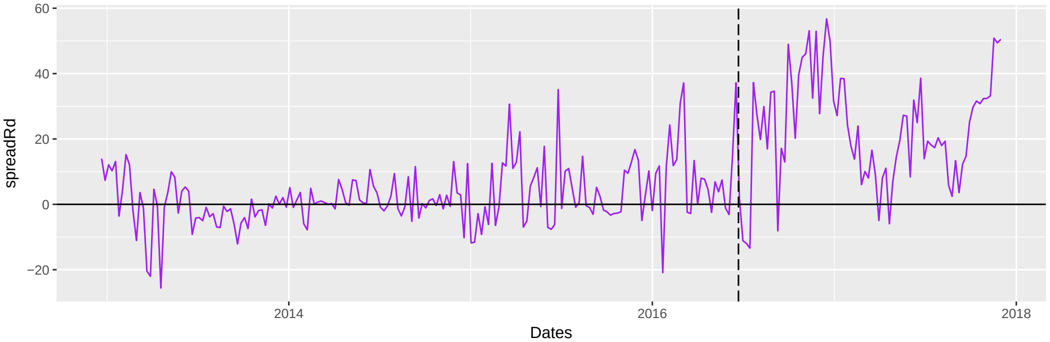
What the data tell us: spot and forward FX rates (2)

```
p2 = ggplot(data, aes(Dates, FWD3M)) + geom_line(colour="red")
p2 = p2 + geom_vline(xintercept=as.POSIXct(as.Date(("2016-06-22 UTC"))),
                     linetype="longdash")
p2
```



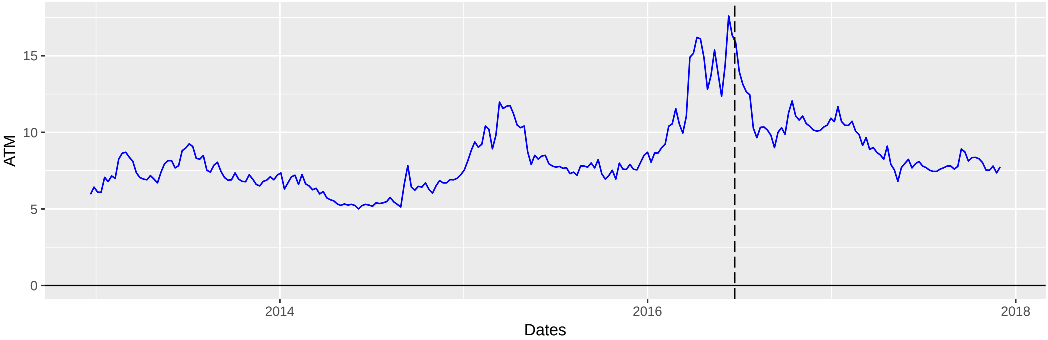
What the data tell us: domestic rate differentials

```
data$spreadRd = (data$ImpliedRd - data$Rd)*100
p3 = ggplot(data, aes(Dates, spreadRd)) + geom_line(colour="purple") +
        geom_hline(yintercept=0)
p3 = p3 + geom_vline(xintercept=as.POSIXct(as.Date(("2016-06-22 UTC"))),
                     linetype="longdash")
p3
```



What the data tell us: ATM volatility

```
p4 = ggplot(data, aes(Dates, ATM)) + geom_line(colour="blue") +
       geom_hline(yintercept=0)
p4 = p4 + geom_vline(xintercept=as.POSIXct(as.Date(("2016-06-22 UTC"))),
                     linetype="longdash")
p4
```



What the data tell us: risk reversals (1)

Risk reversal (measure of vol-skew) [edit]

Risk reversal can refer to the manner in which similar out-of-the-money call and put options, usually foreign exchange options, are quoted by finance dealers. Instead of quoting these options' prices, dealers quote their volatility.

$$R_{25} = \sigma_{call,25} - \sigma_{put,25}$$

In other words, for a given maturity, the 25 risk reversal is the vol of the *25 delta call* **less** the vol of the *25 delta put*. The *25 delta put* is the put whose strike has been chosen such that the delta is -25%.

Figure 3: Risk reversal definition (Wikipedia)

What the data tell us: risk reversals (2)

- Full understanding of the risk reversal requires knowing what Δ is
- But without knowing, we still use risk reversal quotes to assess the prices market participants place on potential exchange rate movements.

$$RR_{25\Delta} = \sigma_{25\Delta C} - \sigma_{25\Delta P}$$

- Pay $\sigma_{25\Delta C}$ for owning the call
- Offset cost somewhat by selling the put at $\sigma_{25\Delta P}$

What the data tell us: risk reversals (3)

- Risk reversal payoff, long a call and short a put, both of them OTM:

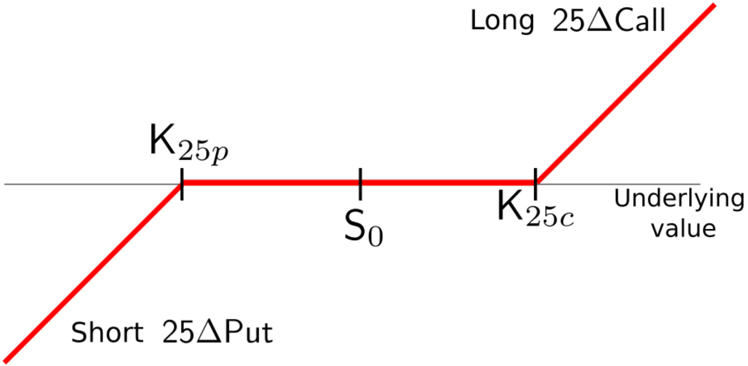
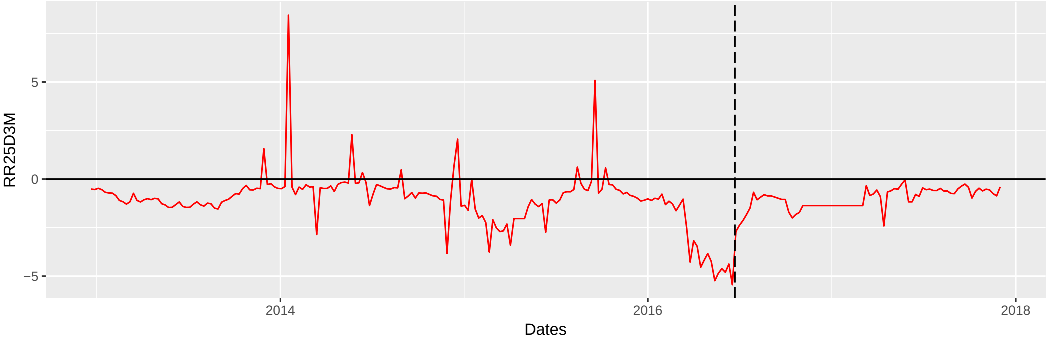


Figure 4: Risk reversal payoff at maturity

- Explain why the price could be positive (or negative)?

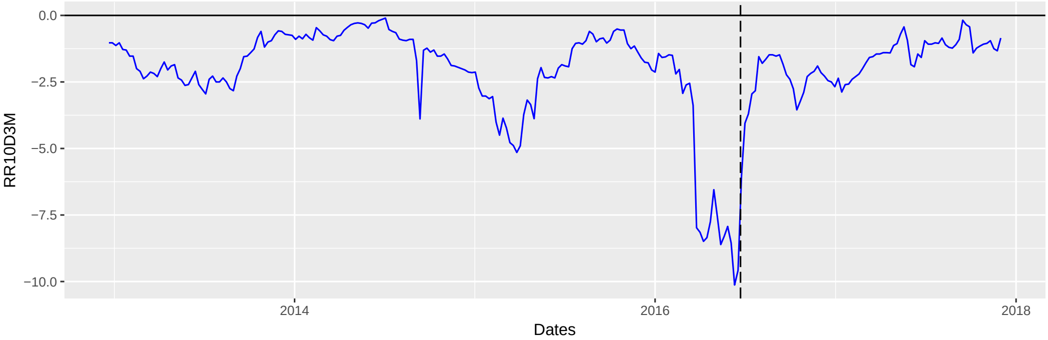
What the data tell us: risk reversals (4)

```
p5 = ggplot(data, aes(Dates, RR25D3M)) + geom_line(colour="red") +
        geom_hline(yintercept=0)
p5 = p5 + geom_vline(xintercept=as.POSIXct(as.Date(("2016-06-22 UTC"))),
                    linetype="longdash")
p5
```



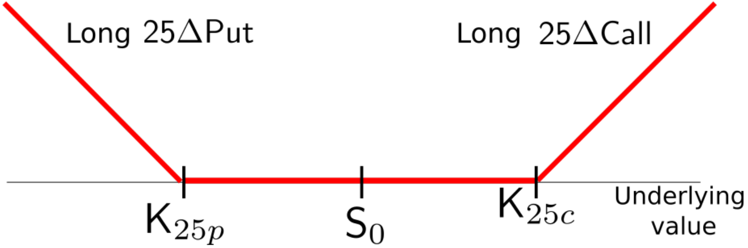
What the data tell us: risk reversals (5)

```
p6 = ggplot(data, aes(Dates, RR10D3M)) + geom_line(colour="blue") +
        geom_hline(yintercept=0)
p6 = p6 + geom_vline(xintercept=as.POSIXct(as.Date(("2016-06-22 UTC"))),
                     linetype="longdash")
p6
```



What the data tells us: butterfly spreads (1)

- Suppose we want to profit from large movements of the exchange rate
- The strangle would make our day !!



What the data tells us: butterfly spreads (2)

- The cost of the strangle is:

$$S_{25\Delta} = \sigma_{25\Delta C} + \sigma_{25\Delta P}$$

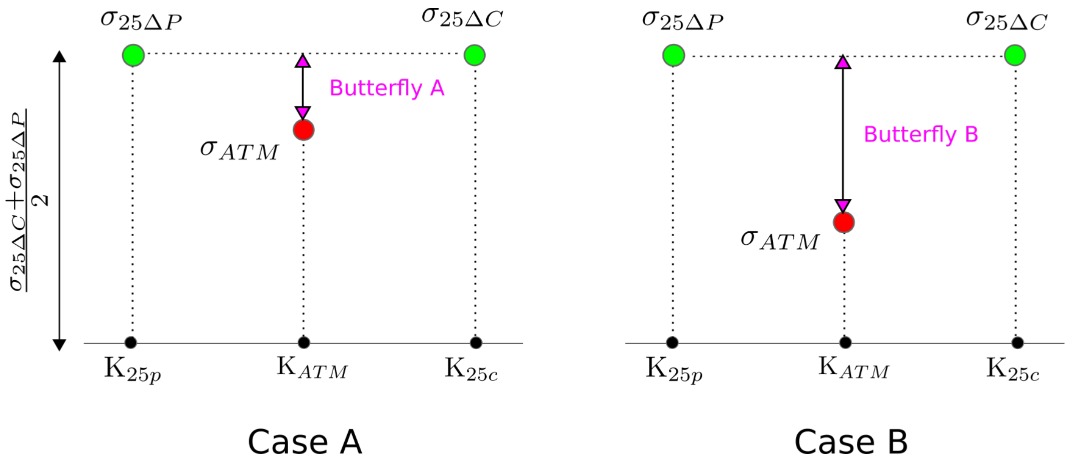
- But markets don't quote strangles

- They quote **Butterfly Spreads**

$$BF_{25\Delta} = \frac{\sigma_{25\Delta C} + \sigma_{25\Delta P}}{2} - \sigma_{ATM}$$

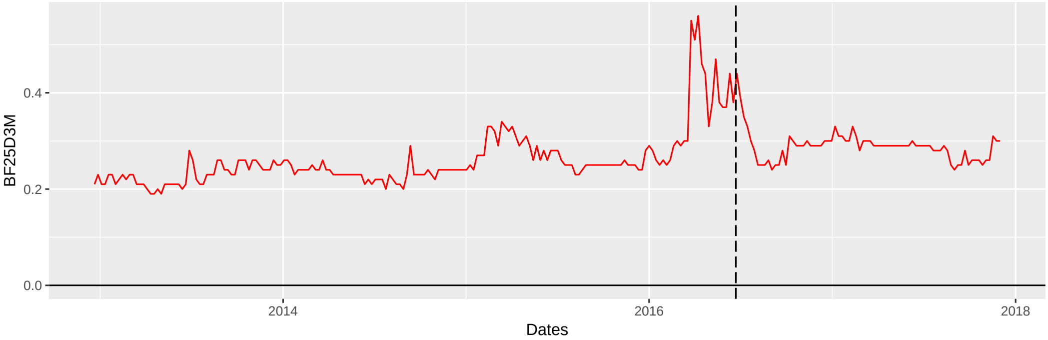
What the data tells us: butterfly spreads (3)

- Butterfly spreads convey more information



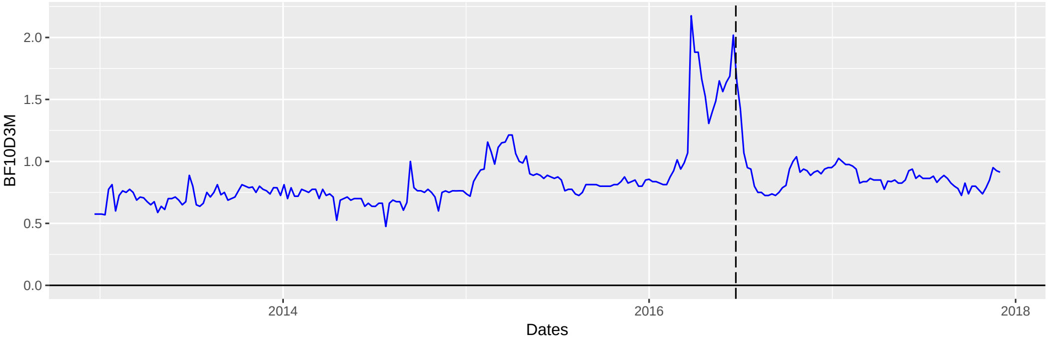
What the data tells us: butterfly spreads (4)

```
p7 = ggplot(data, aes(Dates, BF25D3M)) + geom_line(colour="red") +
        geom_hline(yintercept=0)
p7 = p7 + geom_vline(xintercept=as.POSIXct(as.Date(("2016-06-22 UTC"))),
                     linetype="longdash")
p7
```



What the data tells us: butterfly spreads (5)

```
p8 = ggplot(data, aes(Dates, BF10D3M)) + geom_line(colour="blue") +
        geom_hline(yintercept=0)
p8 = p8 + geom_vline(xintercept=as.POSIXct(as.Date(("2016-06-22 UTC"))),
                     linetype="longdash")
p8
```



Part C:
A Little Bit of Option Pricing

What is Δ (1)

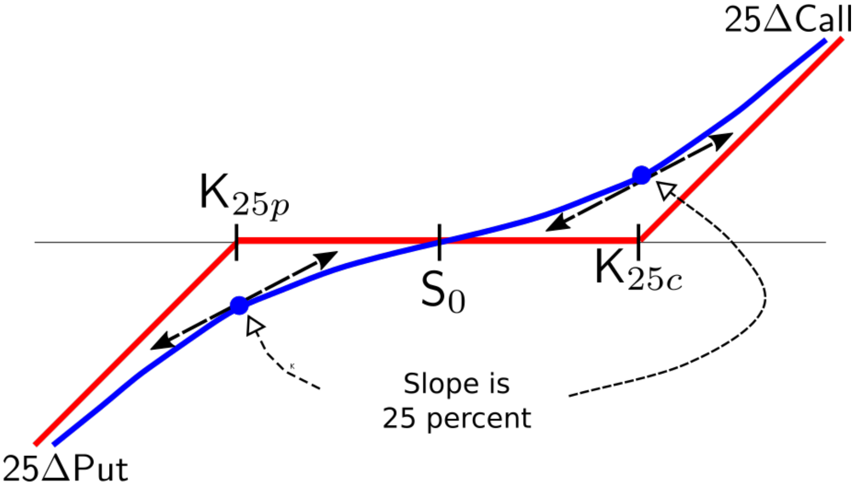
- The Δ of a call

$$\Delta_C = \frac{\partial C}{\partial S} \geq 0$$

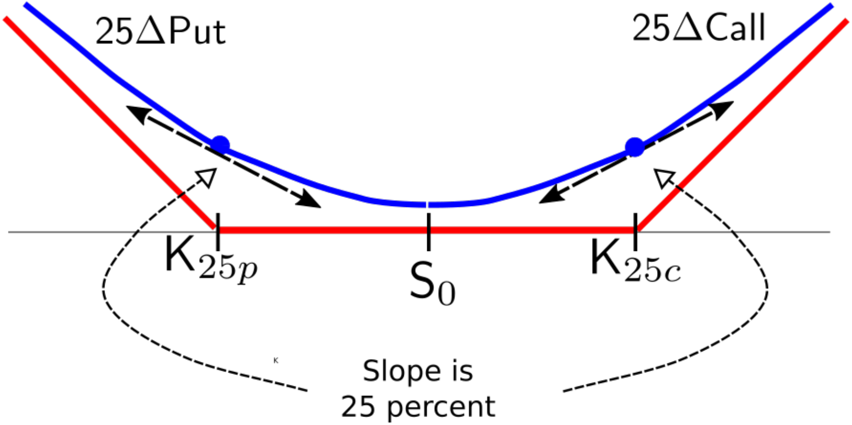
- The Δ of a put

$$\Delta_P = \frac{\partial P}{\partial S} \leq 0$$

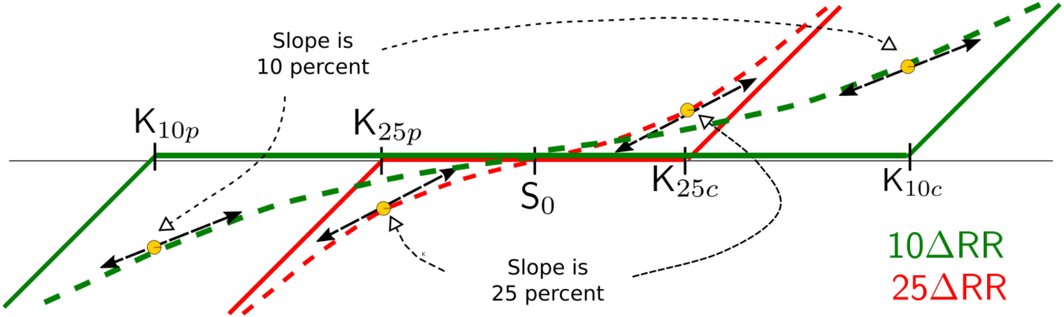
What is Δ (2)



What is Δ (3)



What is Δ (4)



Slope is
10 percent
K_{10p}
K_{25p}
S_0
K_{25c}
K_{10c}
Slope is
25 percent
10ΔRR
25ΔRR

Option prices = replication cost (1)

Dealer costs when buying call option from client

- Funding cost: borrow C at R_d: $R_d C_t \delta t$
- Hedging cost (using delta-hedging)
  - Borrow and sell Δ units of foreign currency
  - Receive ΔS and reinvest at R_d
  - Pay accrued interest on borrowed amount of currency
  - Net gain: $(R_d - R_f)\Delta S_t \delta t$

Option price = replication cost (2)

- Time decay of option: loses value as maturity approaches

$$\frac{\partial C_t}{\partial t}\delta t$$

- Convexity gains since options are non-linear

$$\frac{\partial^2 C_t}{\partial S^2}(\delta S_t)^2$$

Option price = replication cost (3)

- Gains must offset costs, yielding the pricing partial differential equation:

$$\frac{\partial C_t}{\partial t}\delta t + (R_d - R_f)\Delta S_t \delta t + \frac{\partial^2 C}{\partial S_t^2}(\delta S_t)^2 = R_d C_t \delta t$$

Garman-Kohlhagen formula (1)

Assume FX follows a geometric brownian motion, the call option price is:

$$\boxed{C(K, S_t, R_d, R_f, T, \sigma) = S_t \exp(-R_f \times (T - t))N(d_1) - K \exp(-R_d \times (T - t))N(d_2)}$$

and the put option price is:

$$\boxed{P(K, S_t, R_d, R_f, T, \sigma) = K \exp(-R_d \times (T - t))N(-d_2) - S_t \exp(-R_f \times (T - t))N(-d_1)}$$

Garman-Kohlhagen formula (2)

- K is the strike price,
- S_t is the current spot exchange rate,
- R_d is the domestic interest rate,
- R_f is the foreign interest rate,
- $T - t$ is the remaining life of an option maturing at time T,
- σ is the implied volatility of the exchange rate used to price the option,
- $$d_1 = \frac{\ln(S_t/K) + (R_d - R_f + \sigma^2/2)(T - t)}{\sigma \times (T - t)}$$
  $$d_2 = d_1 - \sigma \times (T - t)$$

Implied volatility σ (1)

- All GK formula inputs are observable except implied volatility σ
- Option prices are quoted as **vols**, i.e. volatility
- To obtain price
  - Take quoted vol
  - Use reference values for all other variables
  - Plug into GK formula

Implied volatility σ (2)

Implied volatility

- is different from historical or realized volatility

- is not a forecast of future volatility

- yields an GK option premium reflecting
  - profit margin
  - hedging costs
  - demand and supply in the FX market

Part D:
The Volatility Smile

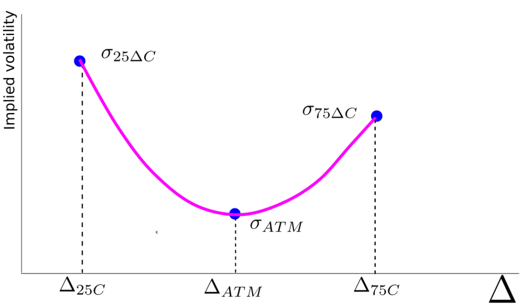
Finding vols for different Δs

- For a given Δ we have prices for:
  - Risk reversal (RR)
  - Butterfly spread (BF)
- The ATM vol, σ_{ATM} is also given
- From the definitions of the RR and the BF

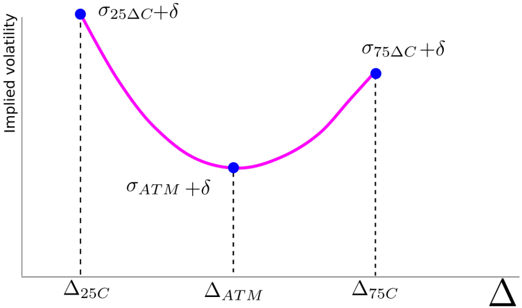
$$\sigma_{25\Delta C} = \sigma_{ATM} + BF_{25\Delta} + \frac{1}{2} RR_{25\Delta}$$
$$\sigma_{25\Delta P} = \sigma_{ATM} + BF_{25\Delta} - \frac{1}{2} RR_{25\Delta}$$
$$\sigma_{75\Delta C} = \sigma_{25\Delta P} \text{ by put-call parity}$$

- Plotting σ against Δ yields the volatility smile

The level of the smile: σ_{ATM}

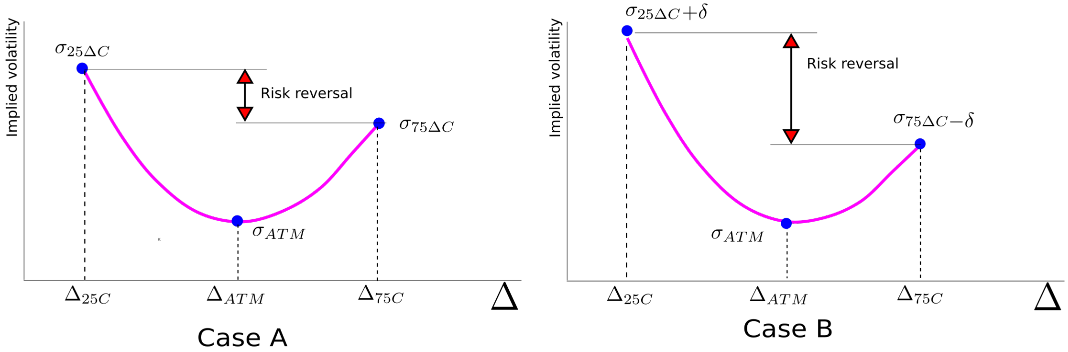


Case A

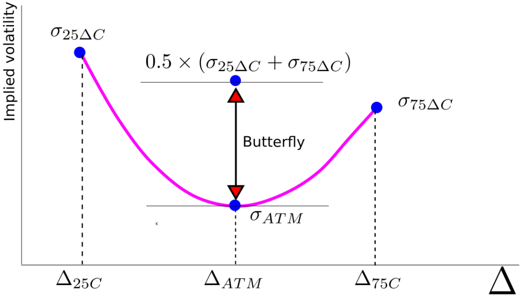


Case B

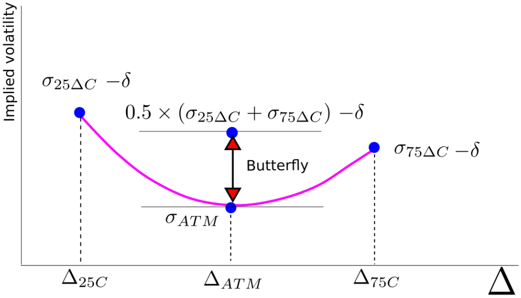
The slope of the smile: the risk reversal



The slope of the smile: the butterfly spread



Case A



Case B

Part E:
Constructing the Volatility Smile: Brexit

Selecting the dates

We are interested in the behavior of the GBPUSD for three dates:

- January 8, 2016 (Pre-Brexit)
- June 24, 2016 (Brexit)
- December 1, 2017 (Post-Brexit)

Getting the data (1)

```
rm(list=ls())                                              # Clean up memory
filename = "2018_IET_Options_data.csv"                     # Name of CSV data file
data = read.csv(filename, header=TRUE)                     # Load datafile
data$Dates = mdy_hm(as.character(data$Dates))              # convert dates to Date 
rownames(data)=NULL                                        # remove row names

# Specify dates for analysis

date01 = as.Date("2016-01-08 UTC")
date02 = as.Date("2016-06-24 UTC")
date03 = as.Date("2016-12-30 UTC")
```

Getting the data (2)

```
# Create data frame this.data

this.data = rbind(
  data[which(data$Dates==date01),],
  data[which(data$Dates==date02),],
  data[which(data$Dates==date03),])

# Delete row names and change the names of the columns

rownames(this.data) = NULL
colnames(this.data) = c("Date","spot","forward","atm", "rr25","bf25",
                        "rr10","bf10","rf","rd","imp_rd")
```

Get additional vols

In addition to the 25Δ and 75Δ vols, obtain the 10Δ and 90Δ vols:

$$\sigma_{10\Delta C} = \sigma_{ATM} + BF_{10\Delta} + \frac{1}{2}RR_{10\Delta}$$
$$\sigma_{10\Delta P} = \sigma_{ATM} + BF_{10\Delta} - \frac{1}{2}RR_{10\Delta}$$
$$\sigma_{90\Delta C} = \sigma_{10\Delta P}$$

Calculating the implied vols (1)

```
# Vols are in percent, expressed them as simple numbers
this.data$atm  = this.data$atm/100
this.data$rr25 = this.data$rr25/100
this.data$bf25 = this.data$bf25/100
this.data$rr10 = this.data$rr10/100
this.data$bf10 = this.data$bf10/100
this.data$rf   = this.data$rf/100
this.data$rd   = this.data$rd/100
this.data$imp_rd=this.data$imp_rd/100

# We will use this.data repeatedly
# Attach it to access its elements

attach(this.data)
```

Calculating the implied vols (2)

```
# Recover vols for different deltas and put them in the data frame

this.data$sigma10c = atm + bf10 + 0.5*rr10
this.data$sigma25c = atm + bf25 + 0.5*rr25
this.data$sigma75c = this.data$sigma25c - rr25
this.data$sigma90c = this.data$sigma10c - rr10
this.data$sigmaatm = atm

Tenor = 3/12   # Maturity of options, 3 months, in years
```

What is the *ATM* strike?

- Retail products

$$K_{ATM} = S$$

$$\Delta_{ATM} = N\left(\frac{\log(F/S) + \frac{1}{2}\sigma^2_{ATM}T}{\sigma_{ATM}\sqrt{T}}\right)$$

- EM currencies, maturities more than one year

$$K_{ATM} = F$$

$$\Delta_{ATM} = \exp(-Rf \times T)N(\frac{1}{2}\sigma_{ATM}\sqrt{T})$$

- Major currencies, maturities of one year or less

$$K_{ATM} = F \times \exp\left(0.5\sigma^2_{ATM} \times T\right)$$

$$\Delta_{ATM} = 0.5 \times exp(-Rf \times T) \simeq 0.5$$

Calculate the Δ_{ATM}

```
# Calculate the strike of the ATM option
K_atm = forward*exp((0.5*(this.data$sigmaatm)^2)*Tenor)
deltaATM = 0.5*exp(-rf*Tenor)
deltaATM
```

```
## [1] 0.4989636 0.4993504 0.4994441
```

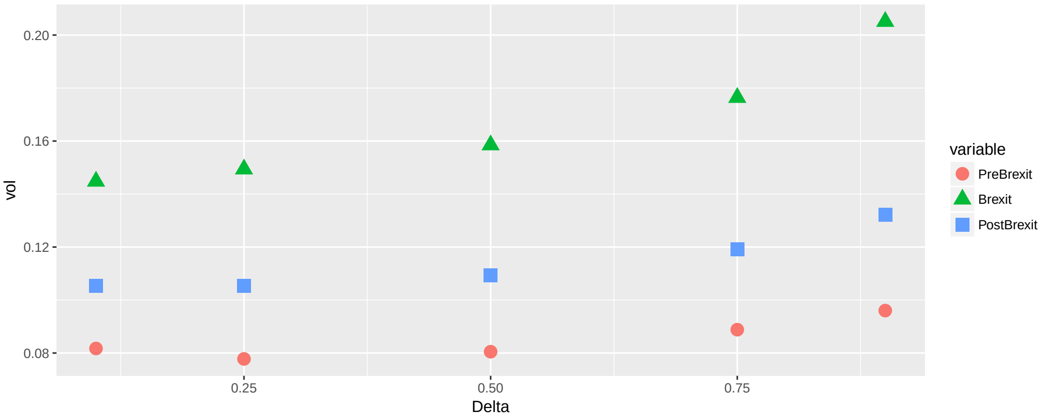
Rough volatility smile (1)

```
# Select only the vols for each delta
list_variables = c("sigma10c", "sigma25c", "sigmaatm",
                   "sigma75c", "sigma90c")

# Read the data as a matrix
vol_data = t(as.matrix(subset(this.data, select=list_variables)))

# Group the deltas in a vector, to be used in the x-axis
delta_vector = c(0.10, 0.25, 0.5, 0.75, 0.9)

# Create the data frame for the chart
vol.smile  = data.frame(delta_vector, vol_data)
rownames(vol.smile) = NULL
colnames(vol.smile) = c("Delta","PreBrexit","Brexit","PostBrexit")
```

Rough volatility smile (2)

```
library(reshape2)
vol.data = melt(vol.smile, id="Delta")
ggplot(data=vol.data, aes(x=Delta, y=value, shape=variable)) +
   geom_point(aes(colour=variable), size=4) +
   labs(y="vol")
```

Rough volatility smile (3)



A more refined volatility smile (1)

```r
# Fit second degree polynomial to smile
fit.Vol = function(data.vol, data.delta,delta.range)
{
  poly.fit = lm(data.vol ~ poly(data.delta, 2, raw=TRUE))

  # Use fitted polynomial to interpolate Delta-Vol Curve
  delta.square = delta.range*delta.range
  delta.interc = rep(1,length(delta.range))

  X = cbind(delta.interc, delta.range, delta.square)
  iVolInterpol = t(t(X)*poly.fit$coefficients)
  iVolInterpol = rowSums(iVolInterpol)

  return(iVolInterpol)
}
```

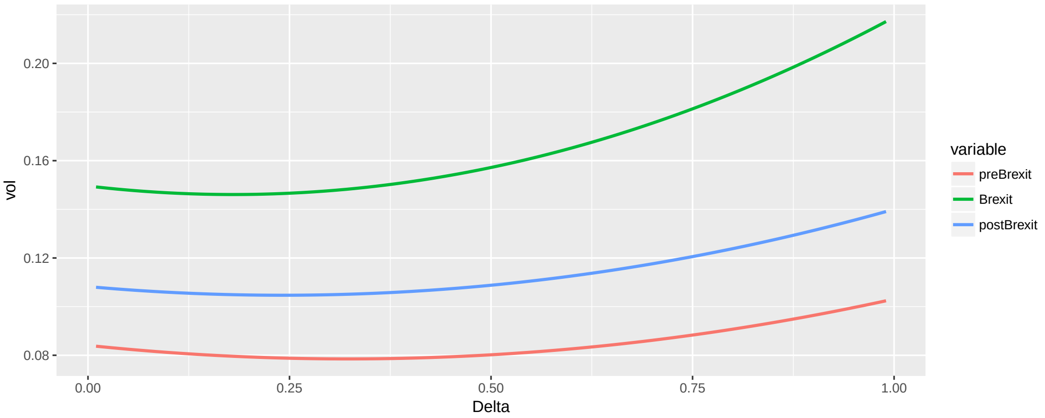
A more refined volatility smile (2)

```
# Get the data points from the vol.smile data frame
data.delta = vol.smile$Delta
data.vol  = vol.smile$PreBrexit

# Interpolation and extrapolation range
delta.range = seq(from=0.01, to = 0.99, by=0.005)

# Obtain the volatility smiles
smile.preBrexit =fit.Vol(vol.smile$PreBrexit, data.delta, delta.range)
smile.Brexit    =fit.Vol(vol.smile$Brexit, data.delta, delta.range)
smile.postBrexit=fit.Vol(vol.smile$PostBrexit, data.delta, delta.range)
```

A more refined volatility smile (3)

```r
# Group the extended volatility smiles in a data frame
smile.df = data.frame(delta.range, smile.preBrexit,
                      smile.Brexit, smile.postBrexit)
rownames(smile.df) = NULL
colnames(smile.df) = c("Delta","preBrexit","Brexit", "postBrexit")

# Create the chart
library(reshape2)
smile.data = melt(smile.df, id="Delta")
ggplot(data = smile.data, aes(x=Delta, y=value, colour=variable)) +
  geom_line(size=1) + labs(y="vol")
```

A more refined volatility smile (4)



Part F:
Extracting Risk-Neutral Densities

Market views are risk-neutral, not objective

Extraction techniques

- Parametric methods
- Semiparametric methods
- Non-parametric methods

Parametric methods

- Specify a parametric distribution function F for the exchange rate
- For the distribution parameters Λ, the price of a call option is

$$C^F(K) = \exp(-R_d \times T) \int_K^\infty (S_T - K)\, dF(S_T; \Lambda),$$

- The parameters that fit the market-implied distribution better solve

$$\arg\min_{\Lambda} \sum_{j=1}^{N} |C^F(K_j) - C^{\text{Market}}(K_j)|^2$$

Semi-parametric methods

- Start with a particular density
- Add expansion terms to distribution
- Each term helps to approximate market-implied distribution

Non-parametric methods

- Plot value of calls against different strike prices, i.e. **market call function**
- Breeden and Litzenberger: risk-neutral distribution proportional to second derivative of the call function:

$$\left.\frac{\partial^2 C}{\partial K^2}\right|_{K=S} = \exp(-R_d \times T)q(S).$$

- Use numerical methods to calculate derivative and find risk neutral distribution

Constructing the market call function (1)

- We need to obtain the market call function
- In other words, move from Δ - σ space to *Option premium* - *Strike* space
- Two useful functions in `auxFunctions.R`
  - `get.strike()`
  - `GKoption.premium()`

Constructing the market call function (2)

```
get.strike = function(vol,delta,S0,fwd,rf,Tenor)
{
  aux = qnorm(delta*exp(rf*Tenor))
  aux = aux*vol*sqrt(Tenor)
  aux = aux - 0.5*vol*vol*Tenor
  K = exp(-aux)*fwd
  return(K)
}
```

Constructing the market call function (3)

```
GKoption.premium = function(K,sigma,S,Tenor,fwd,rf,option_type)
{
  if (option_type =="c") {w=1}
  if (option_type =="p") {w=-1}
  d1 = log(fwd/K)+0.5*sigma*sigma*Tenor
  d1 = d1/(sigma*sqrt(Tenor))
  d2 = d1 - sigma*sqrt(Tenor)
  rd = log(fwd/S)/Tenor + rf
  premium = exp(-rd*Tenor)*(w*fwd*pnorm(w*d1) - w*K*pnorm(w*d2))
  return(premium)
}
```

Constructing the market call function (3)

Strike ranges for the selected dates:

```
K_preBrexit = mapply(get.strike,smile.df$preBrexit, smile.df$Delta,
                     S0=spot[1], fwd=forward[1], rf=rf[1], Tenor=0.25)
K_Brexit    = mapply(get.strike,smile.df$Brexit, smile.df$Delta,
                     S0=spot[2], fwd=forward[2], rf=rf[2], Tenor=0.25)
K_postBrexit= mapply(get.strike,smile.df$postBrexit, smile.df$Delta,
                     S0=spot[3], fwd=forward[3], rf=rf[3], Tenor=0.25)
```

Constructing the market call function (4)

Generate the call premia:

```
Call_preBrexit = mapply(GKoption.premium, K_preBrexit,
                        smile.df$preBrexit, S=spot[1],Tenor=Tenor,
                        fwd=forward[1],rf=rf[1], option_type="c")

Call_Brexit = mapply(GKoption.premium, K_Brexit,
                     smile.df$Brexit, S=spot[2],Tenor=Tenor,
                     fwd=forward[2],rf=rf[2],option_type="c")

Call_postBrexit = mapply(GKoption.premium, K_postBrexit,
                         smile.df$postBrexit,S=spot[3],Tenor=Tenor,
                         fwd=forward[3],rf=rf[3],option_type="c")
```

Constructing the market call function (5)

The data frame `callstrike.df` collects the call premium-strike functions for the selected dates:

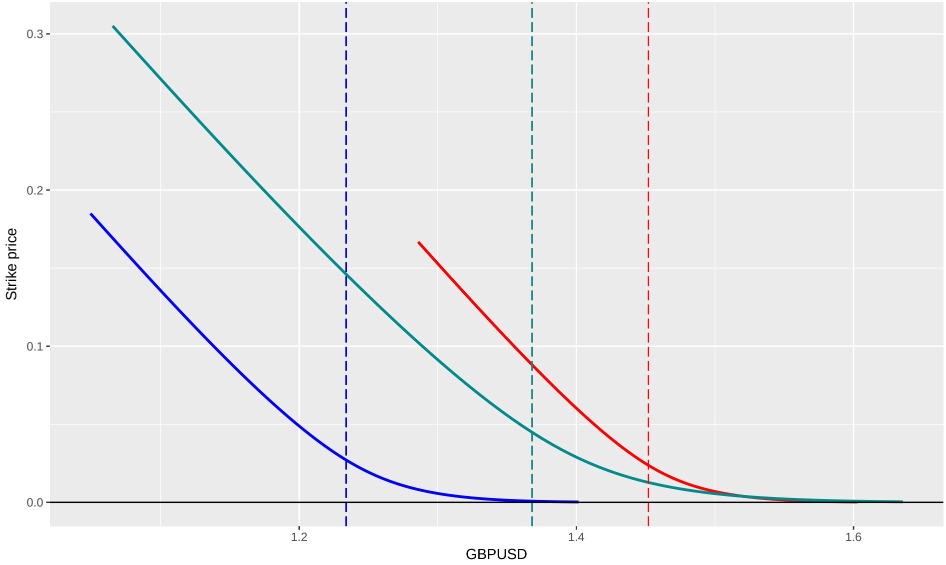
```
callstrike.df = data.frame(K_preBrexit, Call_preBrexit,
                           K_Brexit, Call_Brexit,
                           K_postBrexit, Call_postBrexit)
```

Constructing the market call function (6)

Plot the market call function

```
ggplot(callstrike.df) +
  geom_line(aes(x=K_preBrexit,y=Call_preBrexit), col="red", size=1) +
  geom_line(aes(x=K_Brexit, y=Call_Brexit), col="darkcyan", size=1) +
  geom_line(aes(x=K_postBrexit, y=Call_postBrexit), col="blue", size=1) +
  labs(x="GBPUSD", y="Strike price") +
  geom_vline(xintercept = spot[1], col="red", linetype="longdash") +
  geom_vline(xintercept = spot[2], col="darkcyan", linetype="longdash") +
  geom_vline(xintercept = spot[3], col="blue", linetype="longdash") +
  geom_hline(yintercept = 0, col="black")
```

Constructing the market call function (7)



Constructing the market call function (8)

Question: the call premium-strike functions are downward sloping, i.e. the call premium is higher for lower strike prices. Does this make sense? Explain why.

Constructing the market put function (1)

```
Put_preBrexit = mapply(GKoption.premium, K_preBrexit,
                       smile.df$preBrexit,S=spot[1],Tenor=Tenor,
                       fwd=forward[1],rf=rf[1],option_type="p")

Put_Brexit = mapply(GKoption.premium, K_Brexit, smile.df$Brexit,
                        S=spot[2],Tenor=Tenor,fwd=forward[2],
                        rf=rf[2],option_type="p")

Put_postBrexit = mapply(GKoption.premium, K_postBrexit,
                        smile.df$postBrexit,S=spot[3],Tenor=Tenor,
                        fwd=forward[3],rf=rf[3],option_type="p")
```

Constructing the market put function (2)

```
putstrike.df = data.frame(K_preBrexit, Put_preBrexit,
                          K_Brexit, Put_Brexit,
                          K_postBrexit, Put_postBrexit)
```

Constructing the market put function (3)

```r
ggplot(putstrike.df) +
  geom_line(aes(x=K_preBrexit,y=Put_preBrexit), col="red", size=1) +
  geom_line(aes(x=K_Brexit, y=Put_Brexit), col="darkcyan", size=1) +
  geom_line(aes(x=K_postBrexit, y=Put_postBrexit), col="blue", size=1) +
  labs(x="Strike price", y="Put premium") +
  geom_vline(xintercept = spot[1], col="red", linetype="longdash") +
  geom_vline(xintercept = spot[2], col="darkcyan", linetype="longdash") +
  geom_vline(xintercept = spot[3], col="blue", linetype="longdash") +
  geom_hline(yintercept = 0, col="black")
```

Constructing the market put function (4)