Osäker osäkerhet

Ett kompendium om kvantifiering av volatilitet

ME1316 | KTH

Innehåll

1 Intro

Okej, pysen. Är tentaplugg lite för trist? Har du just avslutat ett filmmaraton av *The Big Short* & *Wolf of Wallstreet* och vill bli nästa Gordon Gekko? Du har ju läst introkurser i ekonomi, och besitter fler avklarade högskolepoäng i matematik än släpandes... Och med en programmeringskurs i bakfickan så måste man ju fråga sig: Varför skulle någon som du *inte* klara av skapa den där algoritmen som hovar in storkovan på aktiemarknaden?

Nå, vare sig det är sant eller inte så kan det vara intressant att börja titta på hur något sådant skulle kunna se ut. Detta kompendium kommer gå igenom hur man kan modellera en aktiekurs, hämta hem data och börja prediktera priset på en typ av finansiell produkt som köps och säljs på marknaden. Lyckas du prediktera priset bättre än marknaden så borde det gå att bli nästa Michael Burry.

Vi ska inte lova för mycket. Men vi vill säga att det som presenteras i detta kompendium är sant, även om inte hela historien alltid berättas. Vi kommer ta en titt på Black-Scholes-modellen och använda den för att skatta priset på *European Call Option* (sv: Europeiska köpoptioner). Detta har möjligtgjorts av lärdomar från SF1918 Sannolikhetsteori och statistik!

1.1 Syftet med kompendiumet

Detta kompendium ska vara ett praktiskt exempel på hur man kan använda en teoretisk modell till att få ut ett resultat med hjälp av verklig data. Vi hoppas att läsaren lär sig begrepp från både finansiella världen samt matematisk sannolikhetsteori. I sig kan kompendiet stå på egna ben, men är skriven för `ME1316 Kvantitativ affärs- och verksamhetsanalys 6.0hp` på Kungliga tekniska högskolan. Kompendiet kan ses som kuriosa för studenter som vill se...

1. Praktisk användning av sannolikhetsteori och statistik
2. Reflekterande över kvantitativa analyser
3. Programmera med data.

Vi ska dock inte dölja det annars latenta syfte: att underlätta i-studenter som går matteinriktningen inför de kurser som väntar.

1.2 Hur man läser Kompendiet

Visst är det intressant att läsa om *hur* man ska läsa det man *redan* läser? Väldigt meta. Eftersom Kompendiet riktar sig till en bredare skara så finns det delar som är mer och mindre intressanta för dig som läsare.

Det finns delar som rör sannolikhetsteori och finansiell matematik, dessa är markerade med ett dollar-tecken, $, i titeln. Dessa delar går att hoppa över utan att tappa någon större förståelse över det som sker senare. Istället finns de där för dig som vill få en kort introduktion till hur vi kan modellera ting matematiskt.

Det har säkert inte undgått dig att lägga märke till att det gömmer sig kod mellan kapitel. Koden är skriven i Python och visar hur vi kan jobba med *big data* utanför en Excel-miljö som vi annars känner oss hemma i. Man kan, likväl som matematiken, hoppa över alla kodstycken utan att det gör något, men det är möjligt att se hur man i praktiken kan arbeta med att analysera data i Python.

1.3 Att köra koden

Kompendiet är skrivet i vad som kallas Notebook-format för programmering. Detta gör att du kan både läsa kompendie-texten och köra koden i samma miljö. Och du kan experimentera genom att redigera koden hit-eller-dit och hipp-som-happ!

Om du vill köra koden rekommenderar vi att du öppnar kompendiet i Google CoLab. Men vill du ha en svårare, men mer lärorik, upplevelse rekommenderar vi Jupyter Notebook. *Superduperbra* att lära sig för framtida stora kodprojekt med datahantering. *Host **KEX:et** host host*

[ ]:
```
# Såhär kan kod se ut!
print("Wow, vad häftigt med kod")
```

```
Wow, vad häftigt med kod
```

För att få en kort introduktion till de bibliotek vi använder oss av i Kompendiet kan bläddra (eller klicka) sig till Appendix 1.

1.4 Förbehåll

Ibland går det fel! Upptäcker du klumpiga formulering, felstavningar eller matematik som du anser är fel, så är det *mycket möjligt* att så är fallet. Detta är ett första utkast till ett kompendium som vi hoppas kan utvecklas med åren som går och att framtida indek-studenter kan fortsätta skriva och utveckla kompendiet i enlighet med hur kurserna på programmet utvecklas med åren som går.

1.5 Matematisk förbehåll $

Och juste, matematiken är skriven för att läsaren ska få en intiutiv förståelse med enkla exempel. Detta gör att mycket av teorin allmängiltighet förvanskas. Men syftet med det är kompendiet är inte att göra en rigorös härledning med alla viktiga definitioner som hörtill, utan att ge insikt i den matematiken som är korrekt. Vi hoppas att det kommer göra att du förstår matten när den presenteras i sin helhet under senare kurser på i-programmet.

2 Bakgrund

[ ]:
```
# Installera Pandas som ger python möjlighet att läsa filer: .csv & .
 ↪xlsx
!pip -q install pandas
# Installera yFinance som kopplar till kursdata från Yahoo Finance
!pip -q install yfinance
# Installera Matplotlib som låter oss rita diagram
!pip -q install matplotlib
# Bibliotek för att räkna med vektorer och liknande i python
!pip -q install numpy
```

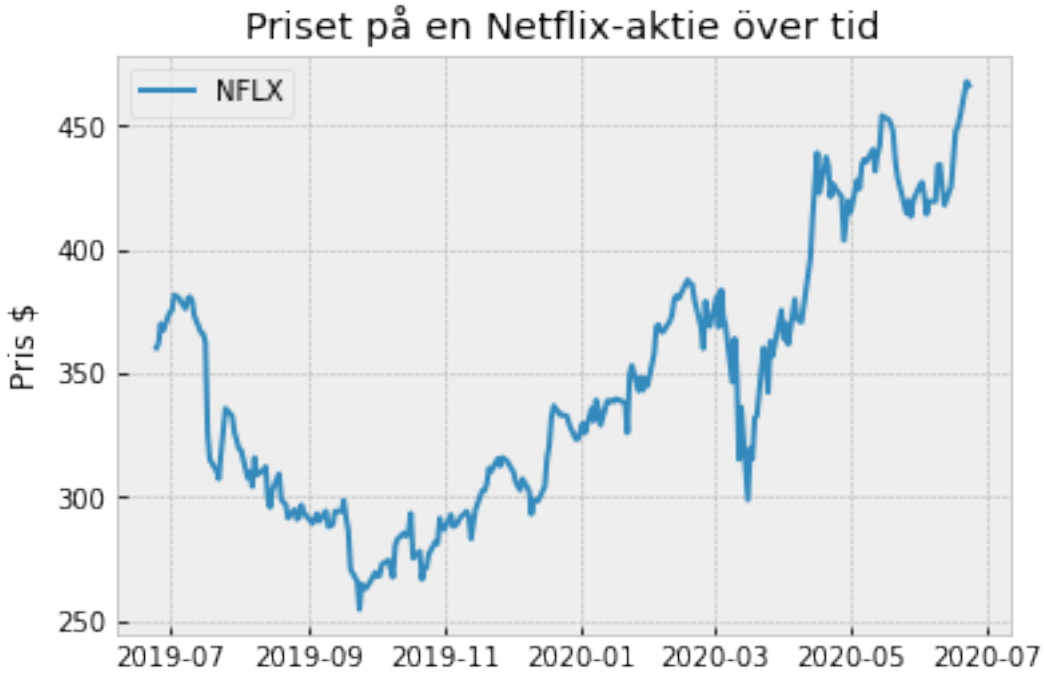
[ ]:
```
# Importera Matplotlib som låter oss rita grafer!
import matplotlib.pyplot as plt
# Så vi får upp diagrammen direkt i vår notebook
%matplotlib inline
from matplotlib import rcParams
#rcParams['figure.dpi']= 100 # ställer in upplösning
plt.style.use('bmh')        # byter förinställt tema
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
```

2.1 Finansiella marknader

Förklaringar må skilja sig åt, men här presenterar vi *finansiella marknader* som platser där kontrakt mellan två parter köps och säljs. Enkla kontrakt speglar tillgångar, exempelvis är en **aktie** en rättighet på andel i ett bolag. Mer komplexa kontrakt bygger på underliggande kontrakt. Rättigheten att få köpa `10 USD` för `100 SEK` i framtiden är ett sådant. Alla kontrakt har ett *marknadspris*. Dock så skiljer sig priset över tid, så varje kontrakt omfattas av en risk för inblandade parter.

[ ]:
```
data = yf.download('NFLX','2019-06-25','2020-06-24')
plt.plot(data['Close'])
plt.title("Priset på en Netflix-aktie över tid")
plt.ylabel("Pris $")
plt.legend(["NFLX"])
plt.show()
```

```
[**********100%**********]  1 of 1 completed
```



Som en följd av risken finns det finansiella tjänster för att aktörer på marknaden ska kunna *hedgea* (skydda) sig mot risk. Ska jag sälja dig mitt bolag i `USD` men föredrar `SEK` kan jag hedga mig genom att planera så att jag köper ett kontrakt som låser fast `USD/SEK`-kurser för framtiden så du inte råkar ut för en käftsmäll när dollarn faller i värde.

I väl fungerande finansiella marknader har parter tillgång till information och låga trösklar till att delta. Detta följer till att prissättningen som blir är lämpliga på sådant sätt att **arbitrage** inte förekommer, ingen part kan tjäna pengar riskfritt. En sådan marknad anses vara en *sensible market*

2.1.1 Efficient market hypothesis

För dig som läser Ayn Rands koserier kommer inte denna teori som någon större överraskning. *Efficient Market Hypothesis* är en teori som säger att marknadspriset speglar det sanna värdet på en tillgång, givet all tillgängliga informationen som finns att tillgå aktörer. I praktiken betyder det att marknadspriset matchar tillgångens riktiga värde. En investerare kommer med andra ord inte kunna "slå" marknaden då marknadskrafterna redan har all information tillgängligt och att tillgångspriset då justeras allteftersom. Med detta antagandet finns det inga "experter" i ordets rätta

bemärkelse utan att slumpen styr hur aktier, och andra tillgångar, utvecklar sig. Om det var förutbestämt att priset skulle stiga skulle efterfrågan öka för att matcha den prisnivå som priset annars hade stigit till.

Exempel Medicinteknikbolaget Fiktiv AB är börsnoterade på en efficient market och meddelar varje kvartal produktresultat för sina medicinska produkter. Detta medföljer att i snitt kommer resultatrapporten inte att höja, eller sänka, aktiepriset då denna information redan är inräknat i priset. I teorin kommer marknaden att underskatta och överskatta ny information i lika stor utsträckning och övertid kommer marknaden sätta ett förväntat pris utifrån den ackumulerade informationen.

Göran,
som är produktchef på Fiktiv AB, sitter ensam på informationen om att företaget nyss har genomfört lyckade Covid-19-vaccinationer. Göran har nu arbritagemöjligheter att tjäna stora pengar på denna unika information. Finansmarknaden där medicinbolaget är noterat är inte längre en efficient market.

Teorins begränsningar och antagande Denna marknadshypotes, och många andra finasiella matematiska modeller, utgår ifrån bland annat utifrån 3 viktiga antagande:

1. Det finns inga arbritagemöjligheter på marknaden (finns inget sätt att tjäna en riskfri vinst).
2. Marknaden värdera förluster/vinster lika (riskneutral).
3. Information är tillänglig för hela marknaden.

Dessa 3 antagande är *knappast* helt sanna i en verklig kontext, men är *tillräckligt* sanna för att låta oss modellerar marknaden.

Ett Emma Frans-fan som har läst boken *Larmrapporten* vet att människor lider av kognitiv bias vid beslutfattande vilket får dem att agera irrationellt, till exempel **bekräftelsebias**. Detta är en del av den kritiken som har förts kring teorin om den effektiva marknaden.

Reflektionsfrågor

1. Hur påverkar aktörers kognitiva bias marknaden och efficient market theoreom?

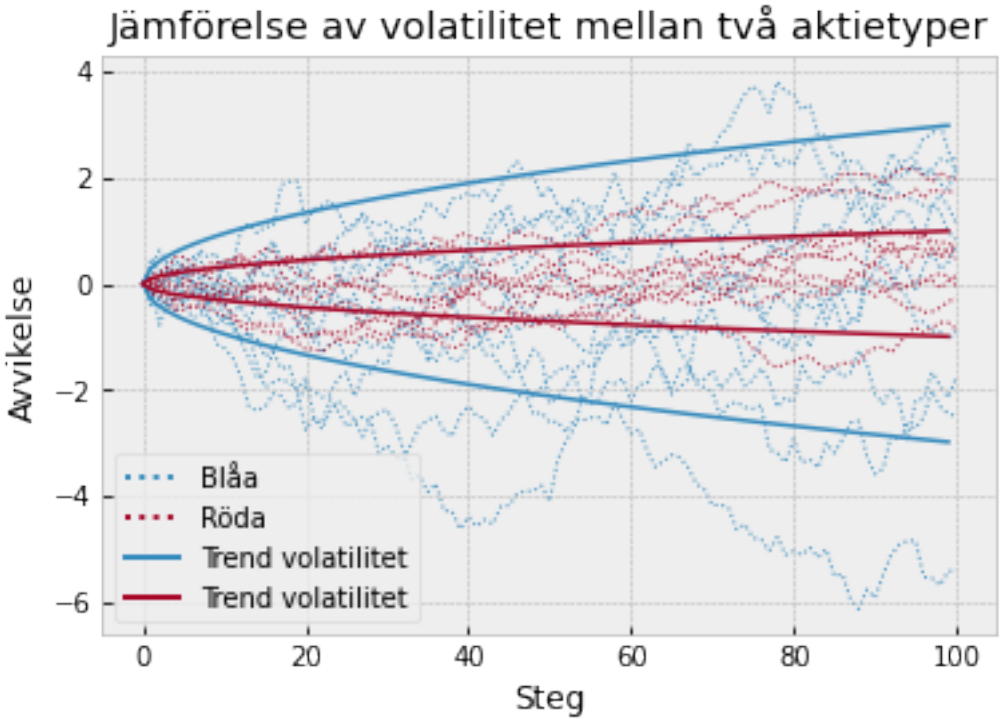
2. Om nu market efficient hypothesis stämmer och att aktier styrs helt av slump och följer matematiken, varför har man då inte enbart automatiserade algoritmer som handlar?

2.1.2 Volatilitet

Volatiliteten är ett mått på kortsiktiga fluktuationer i priset på en finansiell tillgång. Med hög volatilitet fluktuerar priset mer och riskerar således att över tid drifta bort från det ursprungliga priset. Låg volatilitet gör att priset är stabilare, men aldrig helt riskfritt.

[ ]:
```
# Standardavvikelser för risk/volatilitet
sig_x = 0.3 # hög för aktie X
sig_y = 0.1 # låg för aktie Y
# Slumpa 10 kurser i 100 steg
for i in range(10):
  X,Y = [0], [0] # aktiekurser från start
  for j in range(100):
      X.append(np.random.normal(X[-1], sig_x))
      Y.append(np.random.normal(Y[-1], sig_y))
  plt.plot(X,':',color='#348ABD',linewidth=1.0,zorder=1,alpha=0.9)
  plt.plot(Y,':',color='#A60628',linewidth=1.0,zorder=2,alpha=0.9)
# Trendlinje
x_trend = [sig_x*np.sqrt(x) for x in range(100)]
y_trend = [sig_y*np.sqrt(y) for y in range(100)]
# Plots
plt.plot(x_trend,color=u'#348ABD',linewidth=1.5)
plt.plot([-x for x in x_trend],color=u'#348ABD',linewidth=1.5,)
plt.plot(y_trend,color=u'#A60628',linewidth=1.5)
plt.plot([-y for y in y_trend],color=u'#A60628',linewidth=1.5)
# Diagram
from  matplotlib.lines import Line2D

plt.legend(handles = [Line2D([],[], color='#348ABD',linestyle=':
 ↪',label='Blåa'),
                      Line2D([],[], color='#A60628',linestyle=':
 ↪',label='Röda'),
                      Line2D([],[], color='#348ABD',label='Trend␣
 ↪volatilitet'),
                      Line2D([],[], color='#A60628',label='Trend␣
 ↪volatilitet')])
plt.ylabel("Avvikelse")
plt.xlabel("Steg")
plt.title("Jämförelse av volatilitet mellan två aktietyper")
plt.show()
```



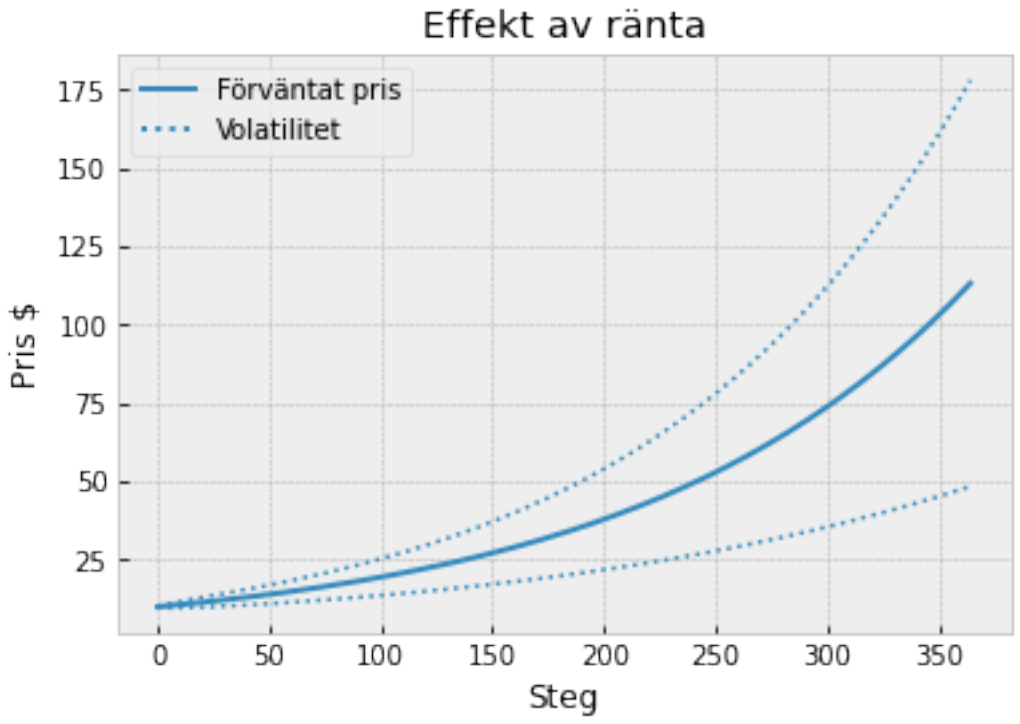
`Blåa aktier`, som har en hög volatilitet fluktuerar mycket mer i pris, medan `Röda aktier` har lägre volatilitet och således lägre risk över tid.

2.1.3 Riskfri ränta

Det finns också sätt att förvara sina tillgångar utan risk. För dig och mig innebär det att stoppa madrassen full, för en storbank handlar blir det svårare. Istället säljer Sveriges riksbank statsobligationer som ses som det enda riktigt riskfria alternativet. Dessa har en avkastningsränta r, som beror på tidshorisont man binder sig till. Detta kallas ibland för reporänta.

Alla länders riksbanker har någon typ av statsobligation med en viss riskfri avkastning, men de kan skilja sig i hur obligationen är strukturerad.

[ ]:
```
sig_x = 0.3 # volatiliteten per steg
r = 0.2     # riskfriränta per 30:e steg
# Trendlinje (från känd formel)
plt.plot([10*np.exp(r*x/30) for x in np.arange(365)])
# Volatilitet
x_trend = [sig_x*np.sqrt(x) for x in range(365)]
plt.plot([(10+x)*np.exp(r*i/30) for i,x in enumerate(x_trend)],':',
         color=u'#348ABD',linewidth=1.5)
plt.plot([(10-x)*np.exp(r*i/30) for i,x in enumerate(x_trend)],':',
         color=u'#348ABD',linewidth=1.5)
# Diagram
plt.legend(handles = [Line2D([], [], color='#348ABD', marker='',
                             markersize=10, label='Förväntat pris'),
                      Line2D([], [], color='#348ABD', marker='',␣
 ↪linestyle=':',
                             markersize=10, label='Volatilitet')])
plt.xlabel("Steg")
plt.ylabel("Pris $")
plt.title("Effekt av ränta")
plt.show()
```



Ränteeffekten påverkar en finansiell tillgångs förväntade prisutveckling över tid, samtidigt som volatiliteten sätts i drift.

2.1.4 Aktier

Vi förstår att du säkert redan placerar hela ditt studielån oavkortat på ett ISK-konto hos Nordnet varje månad och lägger allt på det senaste du hör om i *Unga Aktiespararnas podcast* - så vi tänker inte förklara aktier mer än vad vi redan sagt: att det är ett kontrakt som säljs på en marknad.

Mer nämnvärt är dock en kort matematisk modellering av aktier som en stokastisk variabel S. Aktien har en förväntad avkastning r som växer med tiden, T, samt en kortsiktig risk som mäts med varians σ som också växer i med tiden.

$$E[S_T] = S_0 e^{rT}$$

$$Var[S] = \sigma^2 T.$$

Känner du dig inte nöjd med med den här modelleringen? Bra, för som alla modeller så stämmer den inte! Men läs nästa kapitel om Matematisk modellering för att bli varse om en mer ingående matematisk modell, som har lite mindre fel.

2.1.5 Derivat

Någonting det är värt att skriva några längre meningar om är **derivat** (en: *derivatives*) som är ett samlingsord för de komplexa kontrakt som köps och säljs på marknader som bygger på en underliggande aktie. Ett derivat är ett avtal mellan två parter som förbinder sig till någonting som väntas ske i framtiden. Vanliga derivat är *optioner* och *terminer* som förbinder parter till att handla finansiella produkter för ett bestämt pris en tid framöver.

Den, troligtvist, vanligaste optionen är **European Call Option**. Optionen bygger på en underliggande aktie S, och ger köparen rätten att ett bestämt datum T i framtiden köpa aktien för priset K. Det spännande är att man inte måste köpa aktien, utan bara gör en betalning idag för att få erbjudandet att köpa den i framtiden för ett förutbestämt

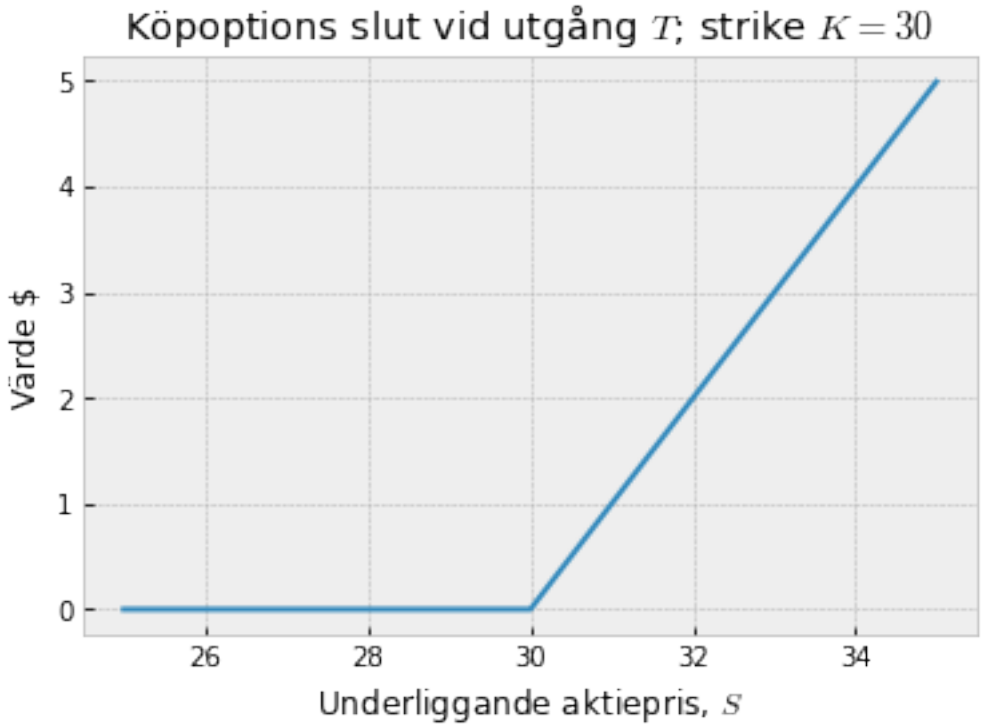
pris. Vinsten för köparen är om priset på den underliggande aktien är mer värt än bestämt pris + premie då köparen får aktien för under marknadsvärdet. Om kostnaden överskrider aktiekursen så finns det ingen mening att *utöva* kontraktet (eng: exercise) varpå derivatet är värdelöst.

Matematiskt kan vi beskriva European Call Option:

$$C(S_T, K) = \max\{S_T - K, 0\}$$

där max-funktionen gör att c aldrig är negativ, och S_T är aktiens pris vid slutdagen.

[ ]:
```
S = 30.0             # Underliggande aktiepris
K = 30.0             # Strike
t = 0.5              # Löptid (år)
S_ = np.arange(25.0, 35.0, 0.01)    # vektor för x-axeln
# European Call Option-funktion
call_payoff = lambda S, K: np.maximum(S_ - K, 0.0)
# Diagram
plt.title('Köpoptions slut vid utgång $T$; strike $K = 30$')
plt.xlabel('Underliggande aktiepris, $S$')
plt.ylabel('Värde $')
plt.plot(S_, call_payoff(S_, K)) # plot(x,y)
plt.show()
```



Priset på på en European Call Option sätts av marknaden, men goda investerare har internat modeller bedömer vad ett rimligt marknadspris må vara.

2.1.6 Black-Scholes

Black-scholes är en modell som använder sig av statistisk prediktion och använder sig av sannolikhetsteori baserat på antaganden om fördelningen och aktiens uppträdande för att få fram ett pris på *European call & put options* (sv: *köp-och säljoptioner*). Formeln är baserad på en matematisk modellering av aktier som stokastiska processer som har en drift r och en volatilitet σ. Modellen består av 5 variabler, varav 3 är varierande och 2 är okända. Dessa är:

- S, dagens aktiepris
- K strike price (sv: lösenpris),
- T time to maturity (sv: löptid)
- r risk-free interest rate (Riskfriränta)
- σ volatility (sv: volatilitet, standardavvikelsen av avkastningen).

Vi ser Black-Scholes-modell som en funktion där aktie, strike och tid kan variera

beroende på hur kontraktet läggs upp. Det okända är volatilitet och ränta. Funktionen ger det rättfärdiga priset på kontraktet *European Call Option*. Modellen utgår ifrån att market efficent theorem gäller.

$$BS_{S,K,T} : \sigma \times r \rightarrow \text{pris}$$

2.2 Matematisk modellering $

Let's get mathecal!

2.2.1 Probability Space $

För att kunna greppa sannolikhet och förutspå framtiden är det viktigt att introducera en modellering som är flexibel och användbar. I detta kapitel kommer vi introducera $(\Omega, \mathcal{F}, P)$ kallat *Probability Space*, även känt som *Probability Triplet* som är en matematisk konstruktion för slumpmässiga processer.

1. Ω *sample space*, samling av alla möjliga utfall w_i.

$$\Omega = \{w_1, w_2, w_3, ..., \}$$

2. $\mathcal{F}$ *event space*, mängden möjliga kombinationer av möjliga utfall i sample space. Nämnt som power set.

$$F = 2^{\Omega}$$

3. P *probability function*, är den funktion som beskriver sannolikheten för olika, från 0 till 1, att event sker.

$$P : \mathcal{F} \rightarrow [0, 1]$$

Exempel: Under helgen sitter Börshajen Hasse och gruvar sig över hans aktieinnehav av Fiktiv AB. När marknaden öppnar på måndag finns det två möjligheter: uppgång eller nedgång. Hans beskriver utfallsrummet som sådant att $\Omega = \{\text{Upp}, \text{Ned}\}$. Event space $\mathcal{F} = \{\{\}, \{\text{Upp}\}, \{\text{Ned}\}, \{\text{Upp}, \text{Ned}\}\}$

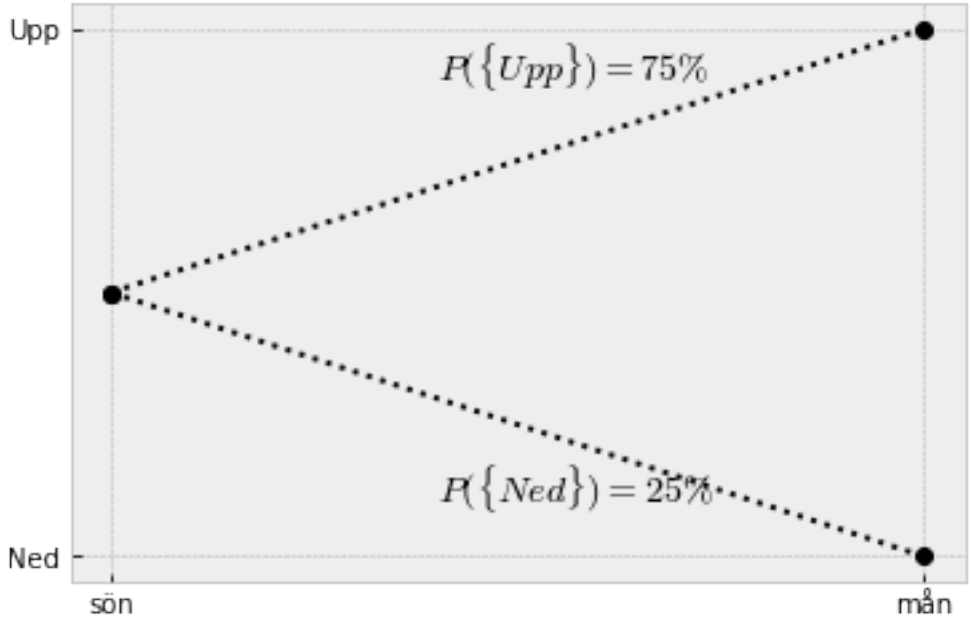
Hasse är lite osäker på sannolikhetsfördelning, men är godtrogen om sin tur och tillskriver till sin modell:

$$P(\text{Upp}) = 75\%$$

$$P(\text{Ned}) = 25\%$$

.

[ ]:
```
tid = ['sön', 'mån']
plt.plot(tid,[0,1], ':',marker='o',color="black")
plt.plot(tid,[0,-1], ':',marker='o',color="black")
plt.yticks([-1,1],['Ned', 'Upp'])  # Set label locations.
plt.text(0.4, 0.8, "$P(\{Upp\})=75\%$", fontsize=14)
plt.text(0.4, -0.8, "$P(\{Ned\})=25\%$", fontsize=14)
plt.show()
```



2.2.2 Stochastic Process $

En *Stochastic Process* (sv: *stokastisk process*) beskriver värdet av en stokastisk variabel när denna övergår i olika tillstånd över diskret tid t, från $t = 0$ upp till ett $t = T$. Det kan beskrivas matematiskt $\{S_n\}_{n=0}^{T}$ där en stokastisk variabel S övergår mellan flera tillstånd $S_0, S_1, S_2 ...$ över tid.

$$\{S_n\}_{n=0}^{T} = \{S_0, S_1, S_2, \ldots, S_T\}$$

För en aktie kan vi se slutpriset varje dag som stokastisk process. Ω blir alla möjliga utfall för hela kedjan. Det finns en drift r, samt en slumpmässighet: volatilitet σ

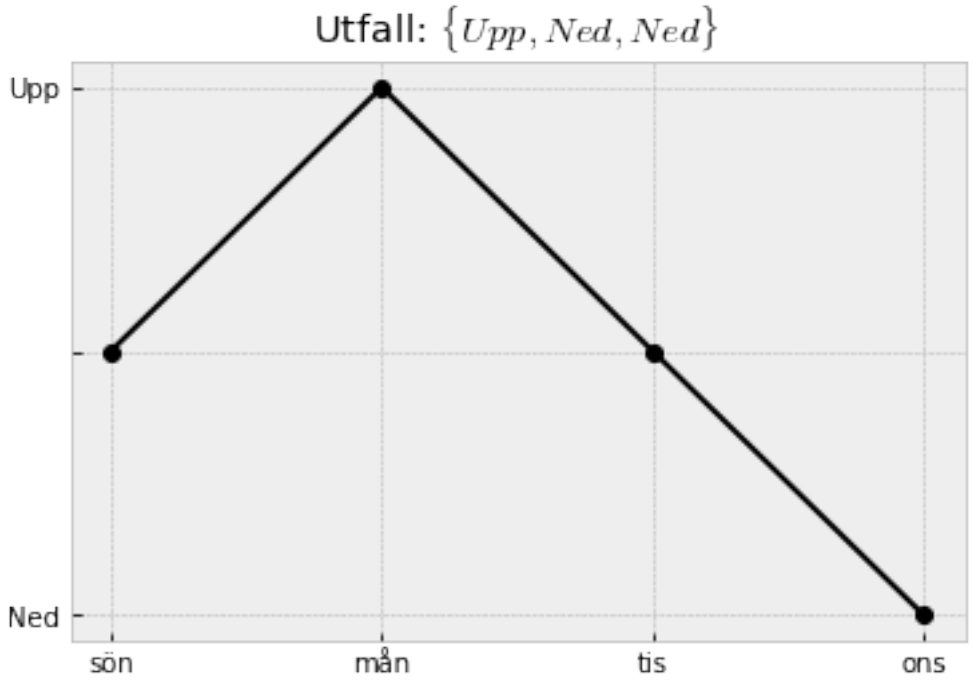
Exempel För Börshajen Hans ser han halva veckans aktiepris som en stokastisk process av upp- och nedgångar.

$$\{S_n\}_{n=\text{mån}}^{\text{fre}} = \{S_{\text{mån}}, S_{\text{tis}}, S_{\text{ons}}\}$$

Probability Space blir genast mycket mer komplex. Möjliga kombinationer $\Omega = \{\{U,U,U\},\{U,U,N\},\{U,N,U\},...\{N,N,N\}\}$ totalt 2^3 (!).

Vidare får vi ett ännu större eventrum $\mathcal{F}$ av power set 2^{Ω}. En mängd med totalt $2^{|2^{\Omega}|}$ kombinationer, alla med olika sannolikheter. Viktigt att notera är att $\{U,U,N\} \neq \{U,N,U\}$ även om de i slutändan faller på samma pris - de ska ses som två olika utfall.

[ ]:
```
plt.plot(['sön','mån','tis','ons'],[0,1,0,-1], marker='o',color="black")
plt.yticks([-1,0,1],['Ned','', 'Upp'])  # Set label locations.
plt.title("Utfall: $\{Upp,Ned,Ned\}$")
plt.show()
```



2.2.3 Filtration $

En filtration över en Probability Space $(\Omega, \mathcal{F}, P)$ är en grupp av mängder $\mathcal{F}_t : t \geq 0$ som är delmängder till $\mathcal{F}$ på så vis att $\mathcal{F}_0 \in \mathcal{F}_1 \in \mathcal{F}_2 \in \cdots \in \mathcal{F}$.

Vinsten är att vi kan modellera all känd information upp till en tidpunkt t. $\mathcal{F}_t$ blir en delmängd av alla möjliga utfall från den tidpunkten. Vi modellerar information genom att filtrera bort det som inte längre är möjligt. Genom att skapa en filtrationsmängd $\mathcal{F}_t$ som sammanställer all känd kunskap upp till en tidpunkt t. Vi säger då att den stokastiska processen är $\mathcal{F}$-measurable. Den stokastiska processen är $\mathcal{F}$-measurable om den mängd som inte längre är möjligt fram till tiden t är tillräcklig för att bestämma den fortsatta processen.

Exempel För Börshajen Hans blir aktie blir detta information om hur aktiepriset har utvecklat sig över tid, alltså historisk data. Innan marknaden öppnar, under söndagen, har en ingen information, $\mathcal{F}_0 = \{\}$. När priset väl har satts av marknaden kan han utfall $X|\mathcal{F}_{\text{mån}} = \text{Upp}$. Där blir en samling av alla utfall som inte längre är möjliga $\mathcal{F}_{\text{mån}} = \{N, ...\}$.

Filtrationen gör det möjligt att sätta betingelser till sina modeller $X|_{\mathcal{F}_t}$. Väntevärdet för

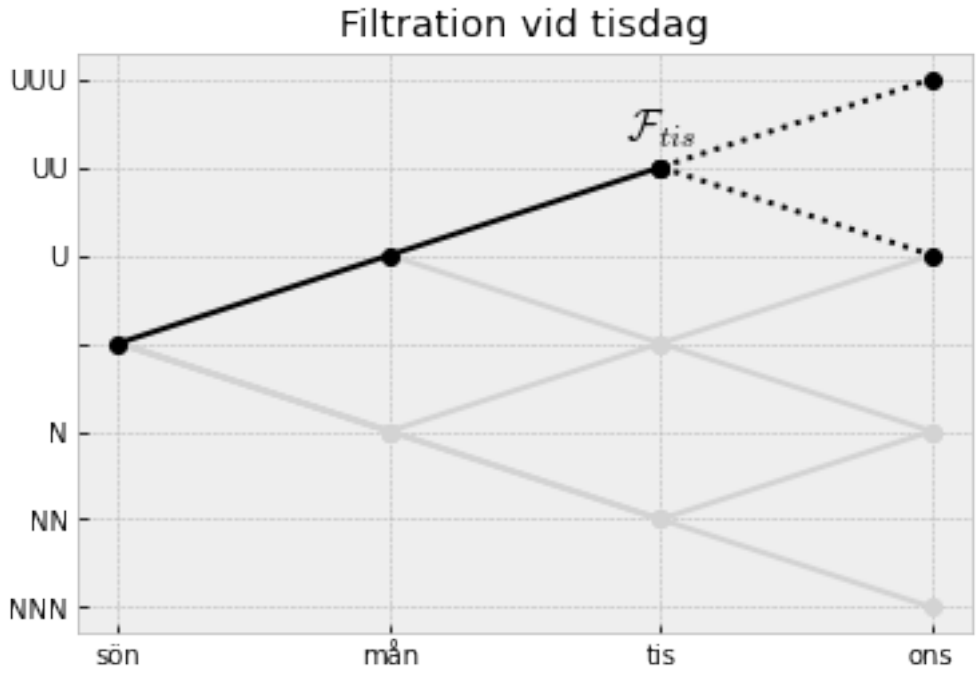
ett derivat, alltså priset, givet dagens information kan således uttrycka sig som

$$E[X|\mathcal{F}_t]$$

[ ]:
```
dagar = ['sön','mån','tis','ons']

plt.plot(dagar,[0,1,0,-1], marker='o', color = 'lightgrey')
plt.plot(dagar,[0,-1,0,1], marker='o', color = 'lightgrey')
plt.plot(dagar,[0,-1,-2,-1], marker='o', color = 'lightgrey')
plt.plot(dagar,[0,-1,-2,-3], marker='o', color = 'lightgrey')

# F_tisdag
plt.plot(dagar[:3],[0,1,2], marker='o', color = 'black')
plt.plot(dagar[2:4],[2,1],':', marker='o', color = 'black')
plt.plot(dagar[2:4],[2,3],':', marker='o', color = 'black')
plt.yticks([-3,-2,-1,0,1,2,3],['NNN','NN','N','', 'U','UU','UUU'])
plt.title("Filtration vid tisdag")
plt.text(2, 2.3, r"$\mathcal{F}_{tis}$",
         fontsize=16, horizontalalignment='center')
plt.show()
```



$$\mathcal{F}_{\text{tis}} = 2^{\Omega} \backslash \{\{Upp, Upp, Upp\}, \{Upp, Upp, Ned\}\}$$

2.2.4 Martingales $

Givet vår probability space $(\Omega, \mathcal{F}, P)$, så är det viktigt att förstå att vår probability function P i praktiken blir okänd. Men vi kan bygga upp den med antaganden om hur vi ser saker fungera. För att urskilja, och följa praxis, kommer vi studera riskneutrala Q som vår probability function.

En Q martingale är probability function på en stokastisk process som följer att väntevärdet under Q är processen är densamma som det senast kända värdet:

$$E^{Q}[X_t|X_s] = X_s : s \leq t$$

Minns du efficient market theorem? Var det inte ungefär det vi sa? Dagens värde på aktien en aktie är bästa gissningen för morgondagens värde. Med det så är bästa skattningen på sannolikhetsfunktionen P, att den sammanfaller med $= Q$! När det endast finns ett unikt sådant Q på en arbitragefri marknad så betecknar man det som en complete market.

Notera att betinga med X_s är i vår modell densamma som att betinga med $\mathcal{F}_s$.

2.2.5 Random Walk $

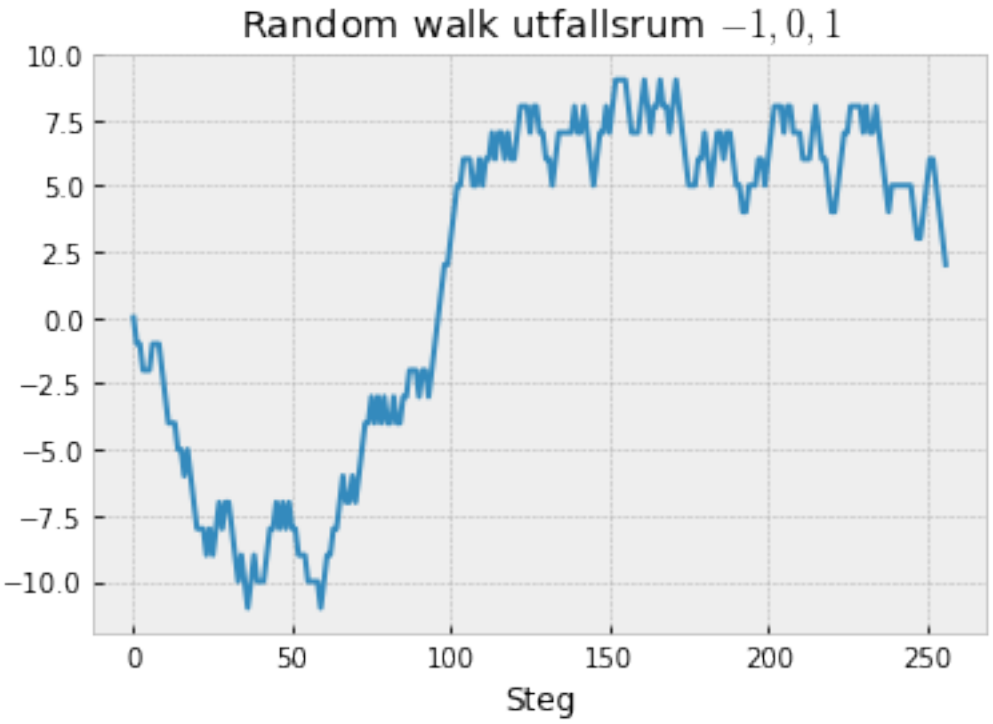
Vi Random walk som bygger på stokastisk process $\{S_n\}_{n=0}^{T}$ och ser processen som en kedja av steg med markovegenskaper.

För en aktiekurs modellerar vi stokastiska processen som en Q Martingale med en viss volatilitet.

$$E^{Q}[X_t|\mathcal{F}_s] = X_n : s \leq t.$$

$$Var^{Q}[X_t|\mathcal{F}_s] = \sigma^2(t - s)$$

```
[ ]: X = [0] # start
for i in range(256):
    X.append(int(X[-1]+np.random.randint(-1,2))) # slump -1, 0 eller +1
plt.plot(X)
plt.title("Random walk utfallsrum $-1, 0, 1$")
plt.xlabel("Steg")
plt.show()
```



Random walks kan formuleras i kontinuerligt tid och i högre dimensioner, vilket då kallas *Wiener Processes*, eller *Brownian Motion*. Visst låter det kul? Går att läsa mer om här.

2.2.6 Black-Scholes $

Black-Scholes prissätter European Call Option från en analytisk härledning från en modell av derivatet som $\max\{S - K, 0\}$ där S följer den stokastiska process som beskrivits ovan. Brasklappen som ska läggas till är att vi har byggt vår modell av marknaden i diskret tid, medan BS antar kontinuerlig t. Skillnaden är att Random walk då kan beskrivas som en differentialekvation som underlättar den analytiska

härledningen.

En differentialekvation som 1997 ledde till att nobelpris i ekonomi (jaja, i Alfreds minne om man man ska vara precis). Här är den i sin helhet:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

Där V är värdet på köpoptionen i sig. Som i sin tur kan lösas analytiskt till:

$$BS(S_0, K, T, \sigma, r) = \overbrace{\Phi(d_1)S_0}^{\text{Förväntad avkastning}} - \underbrace{\Phi(d_2) \cdot e^{-rT}K}_{\text{Förväntad kostnad med diskontenring (PV)}}$$

där Φ är normalfördelningsfunktion $\sim N(0,1)$. d_1 och d_2 ges av:

$$d_1 = \frac{\overbrace{ln\frac{S_0}{K}}^{\text{Värde}} + \overbrace{(r + \frac{\sigma^2}{2})T}^{\text{Väntevärde}}}{\underbrace{\sigma\sqrt{T}}_{\text{Standardavvikelse}}} ; d_2 = \frac{ln\frac{S_0}{K} + (r - \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$$

Beviset lämnas åt läsaren.

Okej okej, se det inte som att vi sopar bevisen under mattan, utan att vi sparar det härliga till SF2701 Financial Mathematics då allt detta gås igenom på riktigt.

Utseendet på d_1 och d_2 härstämmar från $z = \frac{x-\mu}{\sigma} \sim N(0,1)$. Black scholes-modellen antar att de historiska och framtida priserna kommer följa en normalfördelning. d_1 tar fram z-värdet som ger en sannolikhetsvikt på väntevärdet på priset efter löptiden T. d_2 viktar sannolikheten att det framtida priset kommer att överstiga K och den "kostnaden" som uppstår av den riskfria räntan - sannolikheten att köptoptionen generar en vinst efter tiden T med andra ord . Väntevärdet i formeln är väntevärdet för log-normalfördelningen och kommer ifrån att aktiens utveckling och volatilitet är log-normalfördelat.

3 Prissättning

Det fina med Black-Scholes modell är att den i sig är en analytisk lösning. Det fula är att den bygger på två variabler vi vet mindre om σ och r. Vi behöver finna en skattning på dessa, när vi talar om skattade värden markerar vi dem med en liten hat för att kunna särskilja från det sanna värdet: $\hat{\sigma}, \hat{r}$.

Det finns i huvudsak två sätt att finna värdet på σ. Antingen **historisk**: $\hat{\sigma}_{\text{hist}}$, eller **implicerad**: $\hat{\sigma}_{\text{imp}}$.

Historisk skattning bygger på antagandet att historisk volatilitet säger någonting om framtiden. Detta visar sig inte vara helt osant då aktier under kortare perioder har en stabil volatilitet.

Implicerad volatilitet bygger på att köpoptioner redan har ett satt pris av marknaden. Det är så enkelt som att surfa in på Avanza och kolla vad marknadsvärdet på en köpoptioner är i dagsläget. Den implicerade volatiliteten är det $\hat{\sigma}_{imp}$ då Black-Scholes matchar priset, och den står ofta vid information om optionen när man handlar den.

Det är viktigt att nämna en filosofisk poäng att $\hat{\sigma}$ egentligen inte är rätt. Men det är vårt mål att sätta turen på vår sida och få den så rätt som möjligt.

- Validiteten: $E[\hat{\sigma}] = \sigma$.
- Reliabilitet: $Var[\hat{\sigma}] = 0$

3.1 Historisk marknadsdata

Vi kommer hämta hem marknadsdata från Yahoo Finance, vilket vi tvivlar är ett kontroversiellt tillvägagångssätt. Vårt huvudargument är att det är den lättaste datakällan att nyttja då det finns färdigskrivna bibliotek till Python.

Det senaste årets blir dagsslutpriser blir ett **stickprov**. När vi väljer ut vår stickprov måste vi ha så hög validitet och realibilitet som möjligt. Som vi vet ifrån föreläsningarna i ME1316 blir prognoser alltid fel, *men* genom en god datahantering som genererar hög validitet och realibilitet kan skattningen komma nära det "rätta" värdet. Första steget är att förstå och hitta eventuella mönster i vår data. Eftersom vi gör en tidsserieanalys försöker vi hitta samband som vi tror kommer upprepa sig. Tidshorisonten kommer därför vara av stor betydelse när vi gör vårt urval. Genom att välja ett tidshorisont som

inte ger oss en representerade volatilitet fås det som Emma Frans benmämner som **urvalsbias**. Andra möjliga urvalsfel är extrema datapunkter, felaktigt klustrad data. Fundera därför över följande frågor:

3.1.1 Reflektionsfrågor

1. Hur långt i tiden bör vår historiska data gå tillbaka? Är den beroende på vår prognotiserade tidshorisont?
2. Vilken säsongsvariation kan finnas i vår data?
3. Nu tittar vi på *metric data*, vilken typ av *non-metric data* kan man plocka in?

3.1.2 Bibliotek

Vi börjar med att ladda hem de Python-bibliotek som kommer användas i projektet.

[ ]:
```
# Installera Pandas som ger python möjlighet att läsa filer: .csv & .
↪xlsx
!pip -q install pandas
# Installera yFinance som kopplar till kursdata från Yahoo Finance
!pip -q install yfinance
# Installera Matplotlib som låter oss rita diagram
!pip -q install matplotlib
# Bibliotek för att räkna med vektorer och liknande i python
!pip -q install numpy
```

[ ]:
```
# Importera Matplotlib som låter oss rita grafer!
import matplotlib.pyplot as plt # För att rita grafer
import yfinance as yf           # För Yahoo Finance
import numpy as np              # För matematik-funktioner
import pandas as pd             # För att hantera dataframes␣
 ↪(excel-blad)
# Redigerar standardinställningar för matplotlib
%matplotlib inline
import matplotlib as mpl
mpl.rcParams['figure.dpi']= 100
plt.style.use('bmh')
```

3.1.3 Ladda ned data

Vi använder biblioteket `yfinance` för att ladda hem Netflixs marknadsdata från `2020-07-01` och ett år framåt. Aktien handlas på Nasdaq och har symbolen `NFLX`.

[ ]:
```
# Ladda hem data ('aktie',från, till) från Yahoo Finance
data = yf.download('NFLX','2019-07-01','2020-07-02')
print(data.head()) # variabeln 'data' är av objektstypen 'dataframe' i␣
 ↪Pandas
```

```
[**********************100%**********************]  1 of 1 completed
                 Open        High         Low       Close   Adj Close  ␣
 ↪Volume
Date
2019-07-01  373.500000  376.660004  372.000000  374.600006  374.600006 ␣
 ↪4992600
2019-07-02  374.890015  376.000000  370.309998  375.429993  375.429993 ␣
 ↪3625000
2019-07-03  376.690002  381.989990  375.839996  381.720001  381.720001 ␣
 ↪3799000
2019-07-05  378.290009  381.399994  375.559998  380.549988  380.549988 ␣
 ↪3732200
```

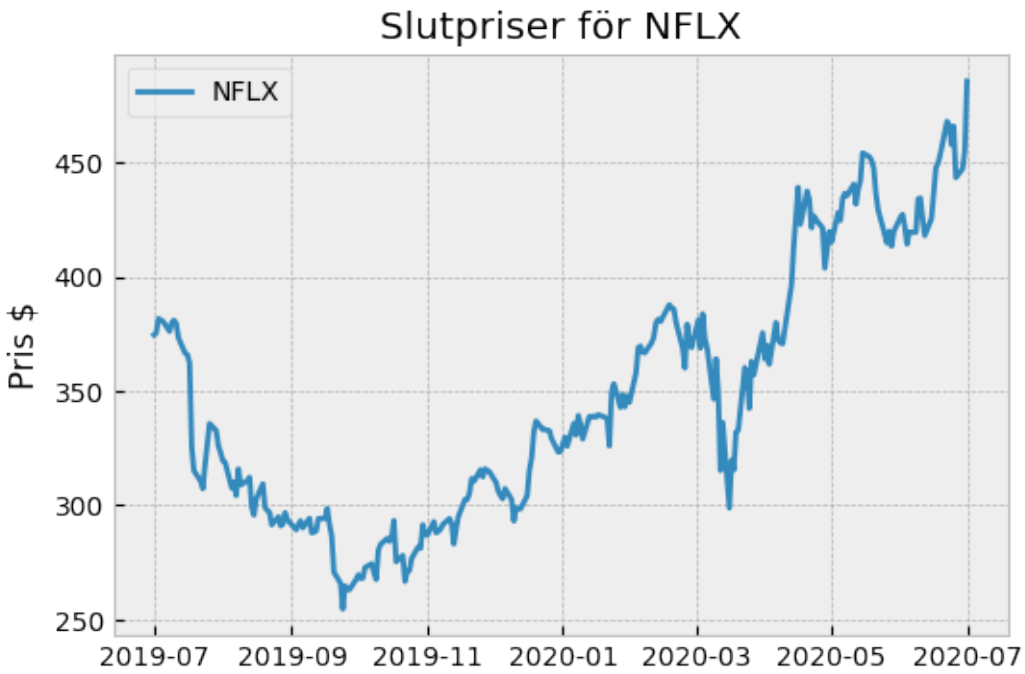
```
2019-07-08  378.190002  378.250000  375.359985  376.160004  376.160004 ␣
↪3113400
```

Yahoo Finance ger oss tillgång till varje dag marknaden är öppen, så har vi allting ifrån dagens första pris, högsta, lägsta, stängning.

3.1.4 Deskriptiv analys

Väljer aktiens sista pris varje dag, `Close`.

[ ]:
```
plt.plot(data['Close'])              # Plotta kolumn 'Close' i data
plt.title("Slutpriser för NFLX")     # Diagramstitel
plt.ylabel("Pris $")                 # Y-axelns text
plt.legend(['NFLX'])
plt.show()
```



[ ]:
```
print(data['Close'].describe())
```

```
count    254.000000
mean     347.376023
std       56.660978
min      254.589996
```

```
25%       298.455002
50%       335.804993
75%       381.037491
max       485.640015
Name: Close, dtype: float64
```

3.2 Skattning för modell

3.2.1 σ och r

Mitt sparkonto på danske-bank har 0% avkastning, så kanske vi borde använda det. Men (!) eftersom vi arbetar med en amerikansk aktie tittar vi på amerikanska statsobligationer. I skrivandets stund har dessa en reporänta på $r = 1,43\%$ om man låser fast sig på 30 år.

[ ]:
```
r = 0.0143 # procent
```

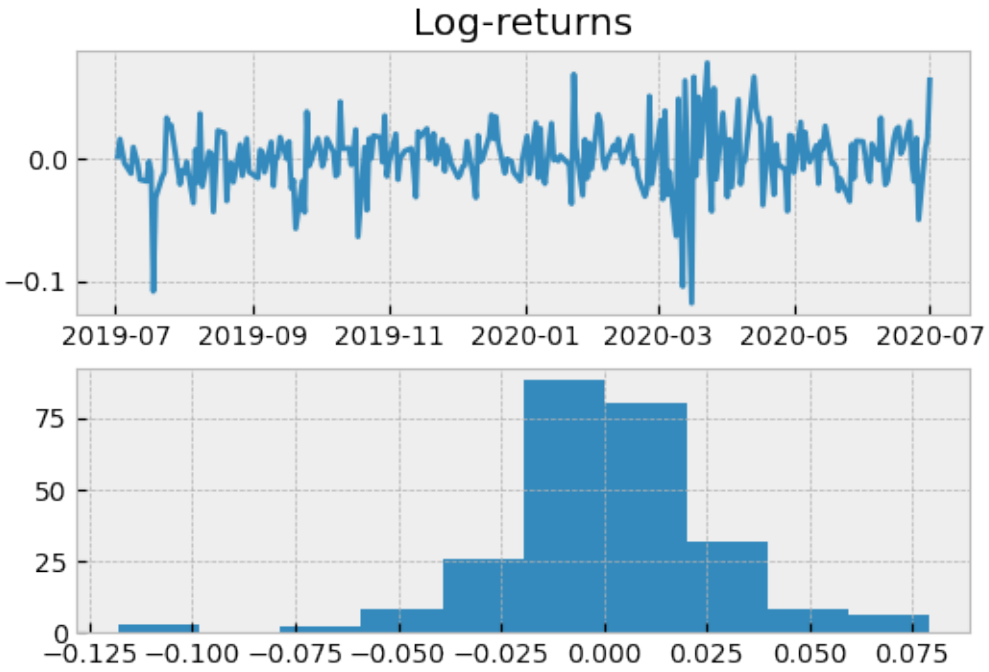
För att få ut σ behöver vi titta på av den procentuella förändringen av priset varje dag. Vi antar att den är log-normalfördelad (vilket ofta stämmer). Volatiliteten blir standardavvikelsen på den samplad log-returns, ofta betäcknat som u_i.

$$\text{log-returns} = u_i = \ln(\frac{S_i}{S_{i-1}})$$

[ ]:
```
# Skapar en ny kolumn i datasetet som heter 'log_returns'
data['log_returns'] = np.log(data['Close'].pct_change().dropna()+1)
# Diagram
plt.subplot(2, 1, 1)
plt.plot((data['log_returns']))  # Titta på procentuell förändring
plt.title("Log-returns")
plt.subplot(2,1,2)
plt.hist(data['log_returns']) # Titta på fördelningen
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:839:
RuntimeWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
```

```
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:840:
RuntimeWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```



Vet att $\sigma_{\text{yr}} = \sigma_{\text{dag}}\sqrt{T}$ som härstammar från random walk. Då amerikanska börsen har öppet 252 dagar om året använder vi det som vårt T.

[ ]:
```
# Räkna ut volitaliteten från formel för skattad standardavvikelse
stickprov = data.loc['2020-01-01':'2020-07-02']['log_returns']
sigma_dy = np.std(stickprov)
sigma_yr = sigma_dy*np.sqrt(252)
print("Daglig volitalitet: ", sigma_dy,"\nÅrlig volatilitet:",␣
 ↪sigma_yr)
```

```
Daglig volitalitet:  0.03073711567446288
Årlig volatilitet: 0.48793658456432487
```

Snyggt! En punktskattning på σ, som vi betecknar med en hatt: $\hat{\sigma}$. Detta för att visa att vi syftar på vår numeriska volitalitet.

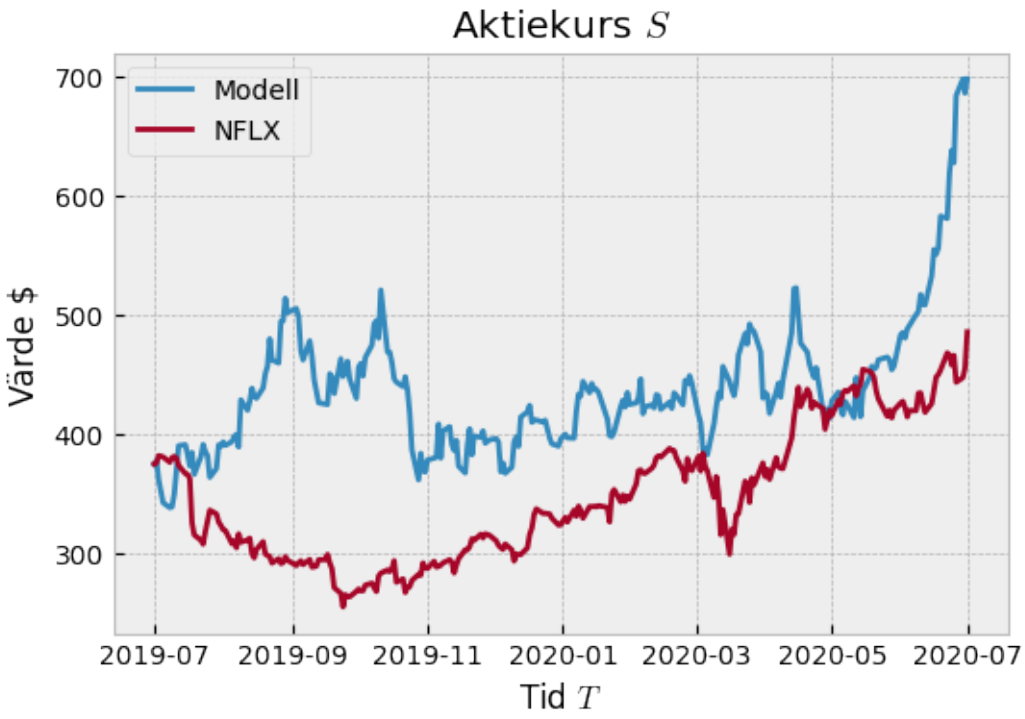
Reflektionsfrågor

4. Bör vi ta med extrema och avvikande datapunkter i vår skattning? Varför/varför inte?
5. Bör vi hantera datan för att få fram en bättre skattning på σ?
6. Vilken bias kommer det få på vårt punktsattning $\hat{\sigma}$?

3.2.2 Modellering av aktiekurs

Nu när vi har fått $\hat{\sigma}$ kan vi modellera aktiens utfall som en random walk.

[ ]:
```
# X är vår modellerade aktiekurs
X = [data['Close'][0]]        # matcha från start
# gå lika många steg som dagar i andra aktien
for i in range(len(data.index)-1):
    X.append(X[-1]*np.exp((np.random.normal(0,sigma_dy)))) # gå steg i␣
 ↪walk
# Diagram
plt.plot(data.index, X) # plot vår modellering
plt.plot(data['Close']) # plot NFLX
plt.title("Aktiekurs $S$")    # Titel
plt.xlabel("Tid $T$")         # namn x-axel
plt.ylabel("Värde $")         # namn y-axel
plt.legend(['Modell','NFLX']) # sätter in skylten (legend) med namn
plt.show()
```



3.2.3 Modellera bias $

För att försöka förstå hur bias och slumpen påverkar vår skattning kan vi göra en enkel modell på $\hat{\sigma}$.

$$\hat{\sigma} = \sigma + \text{bias} + \text{slump}$$

Sanningen är väl att det inte är en addativ modell som är den underliggande sanningen som spökar. Men om det vore så *enkelt* kan vi se:

$$E[\hat{\sigma}] = E[\sigma + \text{bias} + \text{slump}] = \sigma + E[\text{bias}] + E[\text{slump}]$$

Så vårt mål är att få $E[\text{bias}] = E[\text{slump}] = 0$ för att vår punktsakttning av σ ska vara väntevärdesriktig.

Reflektionsfrågor

6. Kan du tänka dig en bättre modell av $\hat{\sigma}$?

3.3 Beräkna BS-priset

Vi minns att Black-Sholes kan ses som en funktion:

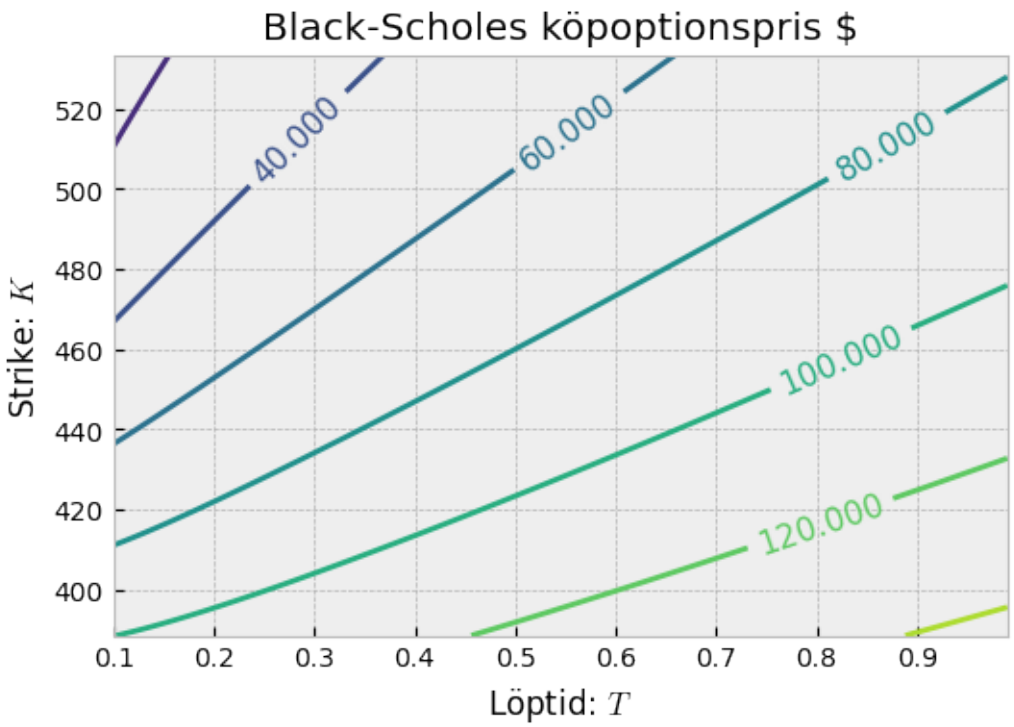
$$BS_{S,K,T}(\sigma, r) = \text{pris}$$

[ ]:
```
def black_scholes(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.
↪sqrt(T))
    d2 = (np.log(S / K) + (r - 0.5 * sigma ** 2) * T) / (sigma * np.
↪sqrt(T))
    from scipy.stats import norm
    return (S*norm.cdf(d1, 0.0, 1.0)-K*np.exp(-r*T)*norm.cdf(d2, 0.0,␣
↪1.0))
```

σ och r punktskattades till $\hat{\sigma}$, $\hat{r}$. Senaste aktiepriset är givet till det senaste priset på marknaden. Men... hur ska vi bestämma strike K och löptid T? I sig kan de vara vad som helst.

Vi kan variera T och K för att se hur priset ändrar sig.

```
[ ]: S_ = data['Close'][-1]              # Dagens aktiepris
     k = np.arange(0.8*S_, 1.1*S_, 1)   # x-axel
     t = np.arange(0.1, 1, 0.01)        # y-axel
     T_,K_  = np.meshgrid(t,k)          # skapa matris
     Z = np.empty(K_.shape)             # skapa z-axel
     # gå igenom matris och sätt z-axel till bs-pris
     for i in range(Z.shape[0]):
       for j in range(Z.shape[1]):
         Z[i,j] = black_scholes(S_,K_[i,j],T_[i,j],r,sigma_yr)
     # Lite mer avancerat plotningsteknik med matplotlib (men snygg graf,␣
      ↪va?)
     fig, ax = plt.subplots()
     CS = ax.contour(T_,K_, Z)
     ax.clabel(CS, inline=1, fontsize=12)
     ax.set_title("Black-Scholes köpoptionspris $")
     ax.set_ylabel("Strike: $K$")
     ax.set_xlabel("Löptid: $T$")
     plt.show()
```



Eftersom köpoptioner ger rätten att köpa optioner för priset K om tiden T så är det väldigt rimligt att det inte finns något värde att sätta K till högre än dagens aktiepris S_0.

3.3.1 Marknadspriser

Det går att ladda hem marknadspriset för optioner med hjälp av `yfinance`, men det blir nog lättast om du går in på Yahoo Finance NFLX Options för att se hur de köps och säljs.

Avanza har också en bra optionslista för svenska aktier.

Tänk på att sortera efter volym för att få upp de optioner som handlats mest, och således bör reflektera ett *riktigt* marknadspris bäst.

3 juli 2020 handlades `NFLX` för `476.89 USD`. Vi hitta en köpoptionen `NFLX Aug 2020 470.000 call` som 3 juli 2020 låg på `33.05 USD`. Optioner på Nasdaq går nästan alltid ut tredje fredagen i månaden som står i betäckningen, alltså 35 börsdagar från 3 juli.

[ ]:
```
S_NFLX = 476.89
K_NFLX = 470
T_NFLX = 35/252
print("Vårt beräknade pris:",
      black_scholes(S_NFLX, K_NFLX, T_NFLX, r, sigma_yr), "USD")
```

```
Vårt beräknade pris: 38.31751335341335 USD
```

Eh... det var ju inte samma som marknadspriset? Nä, och det kanske inte var helt väntat heller. Vi sa ju det innan: Modeller är fel! Något vi behöver är att titta på konfidensintervall på vårate $\hat{\sigma}$, men innan det vill vi besvara en vettig fråga som har dykt upp. Nu när vi bevisligen har fått ett annat σ: Vad är marknadens skattning av σ?

3.3.2 Implicerad Volatilitet

Vi kan räkna ut den implicerade volatiliteten till det σ som matchar marknadspriset i vår Black-Scholes-formel. Vi nyttjar Netwon-Rhapson som kan hittar rötter!

[ ]:
```
from scipy.optimize import newton
# Skriv om BS så vi letar efter ett nollställe
objFunction = lambda x : black_scholes(S_NFLX, K_NFLX, T_NFLX, r,␣
↪x)-33.05
sigma_imp = newton(objFunction, sigma_yr)
print("Marknadens implicerade volatitlitet: ", sigma_imp)
```

```
Marknadens implicerade volatitlitet:  0.4123939711920358
```

Marknaden tror alltså på en högre risk i framtiden än den som var utfallen från förra året.

Reflektionsfrågor:

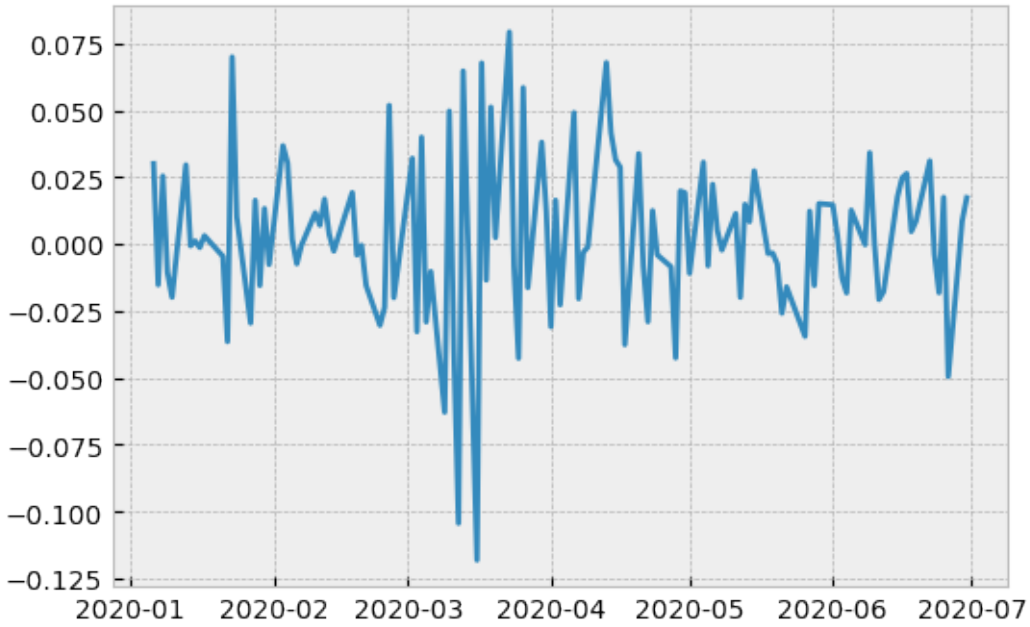
8. Varför tror du $\hat{\sigma}_{imp} < \hat{\sigma}_{hist}$? Gäller det alltid?
9. Hur kan vi bestämma ett konfidensintervall på $\hat{\sigma}_{imp}$?
10. Varför behövs en kvanitativ analys av $\hat{\sigma}$ när vi kan ta den implicerade volatiliteten rakt av?

3.4 Säkerhet på $\hat{\sigma}$

Jaja, ta det lugnt! Vi hör ditt skrik och panik. Hur säkra är vi på $\hat{\sigma}_{\text{hist}}$? Vi saknar ju helt konfidensintervall! Är det någonting vi är säkra på är det att saker är osäkra, och det bör vi kvantifiera.

Hittills har vi punktskattat σ genom att anta att den är värdet på dagligavkastningen är normalfördeled: $\ln \frac{S_t}{S_{t-1}} \sim N(0, \sigma)$

[ ]:
```
stickprov = data.loc['2020-01-04':'2020-06-30']['log_returns']
plt.plot(stickprov)
plt.show()
```



3.4.1 Shapiro-Wilks test

För att se om en population följer en normalfördelning kan vi använda oss av Shapiro-Wilks test.

$$H_0 : \text{log-returns är normalfördelad}$$

Testet finns förprogrammerat i Python-biblioteket Scipy: dokumentation.

```
[ ]: from scipy import stats
     test_statistic, p_value = stats.shapiro(stickprov)
     print("P-value: ",p_value)
```

```
P-value:  0.0019483135547488928
```

Vi kan alltså inte förkasta nollhyptosesen! Vi håller fast vid vårt antagande att datan är normalfördelad.

3.4.2 Konfidensintervall på $\hat{\sigma}$

Från *Sannolikhetsteori och statistikteori med tillämpning*, kursboken i *SF1918 Sannolikhet och statistik*, finner vi konfidensintervallet för standardavvikelsen i **Sats 12.2**

$$I_\sigma = (\sqrt{f/\chi_{\alpha/2}(f)} \cdot \hat{\sigma}, \sqrt{f/\chi_{1-\alpha/2}(f)} \cdot \hat{\sigma})$$

[ ]:
```
df = len(stickprov)-1
alpha = 0.05
sig_yr = np.std(stickprov)*np.sqrt(252)
I = [np.sqrt(df/stats.chi2.isf(alpha/2,df))*sig_yr,
     np.sqrt(df/stats.chi2.isf(1-alpha/2,df))*sig_yr]
print("Sigma:\t", sig_yr, "\nI-05%:\t", I[0], "\nI-95%:\t",I[1])
```

```
Sigma:    0.4847650114718144
I-05%:    0.4308219707249858
I-95%:    0.554272129734337
```

3.4.3 Effekt på BS

[ ]:
```
print("Undre 95%:\t",black_scholes(S_NFLX, K_NFLX, T_NFLX, r, I[0]),␣
 ↪"USD")
print("Estimerad:\t", black_scholes(S_NFLX, K_NFLX, T_NFLX, r,␣
 ↪sig_yr), "USD")
print("Övre 95%:\t", black_scholes(S_NFLX, K_NFLX, T_NFLX, r, I[1]),␣
 ↪"USD")
```

```
Undre 95%:        34.334698273569586 USD
Estimerad:        38.096329340557276 USD
Övre 95%:         42.94329688809168 USD
```

Nämen se så fint!

4 Alternativa modeller $

4.1 För volatilitet $

I det här kompendiumet har vi tagit fram σ genom en statisk modell där varje datapunkt är viktat jämnt. Grundantagandet som vi har gjort hela vår analys bygger på att σ är ett konstant värde. Detta är ett antagande som inte håller i en verklig kontext och något som tar hänsyn till det är adaptiva modeller. Vi kommer därför i detta kapitel gå igenom några sådana i syfte att läsaren själv kan estimera σ med modeller som är mer anpassade för en mer trendbaserad data.

Formeln som vi har utgått ifrån när vi har viktat datapunkterna jämnt ser ut som följande:

$$\hat{\sigma}_n^2 = \frac{1}{m}\sum_{i=1}^{m} u_{n-i}^2 : u_i = \ln\frac{S_i}{S_{i-1}}$$

Om vi vill vikta datapunker behöver vi introducera en variabel α som lägger större vikt på de senare datapunkerna så att formeln ser ut som följande:

$$\sigma_n^2 = \sum_{i=1}^{m} u_{n-i}^2 \cdot \alpha_i$$

där α sätter en större procentuell andel på de senare datapunkerna och därför har vi att:

$$\sum_{i=1}^{m} \alpha_i = 1$$

En ytterligare förfining av den adaptiva modellen för σ_n^2 är att lägga till en långvarig varians, V_L, med viktningen γ så att

$$\sigma_n^2 = V_L \cdot \gamma + \sum_{i=1}^{m} u_{n-i}^2 \cdot \alpha_i \text{ där } \gamma + \sum_{i=1}^{m} \alpha_i = 1$$

Den långvariga variansen, V_L, bygger på en tro att variansen bör ligga kring något medel, men dra ifrån det om närliggande historik talar för det.

4.1.1 EWMA-modellen

Denna modellen härleds ifrån den viktande modellen i det föregående avsnittet och tar större hänsyn till de senaste datapunkterna där α minskas exponentiellt ju längre bak i tiden vi går. Formeln lyder:

$$\sigma^2 = \lambda\sigma_{n-1}^2 + (1-\lambda)u_{n-1}^2$$

där n är varje datapunkt. Ett högt värde på $\lambda < 1$ gör att vi får en mer långsam minskning i vår seriedata. I praxis brukar denna vara runt 0.94.

Notera att formeln är rekursiv!

Fördelarna med denna modell är:

- behövs relativt lite data
- tar stor hänsyn till de senaste förändringarna i vårt dataset.

```
[ ]: def EwmaModelRekursiv(log_returns, lam):
       """
       rekursiva delen av EWMA-modellen
       """
       if len(log_returns)==1:
         return log_returns[0]**2
       else:
         return lam*EwmaModelRekursiv(log_returns[1:
     ↪],lam)+(1-lam)*(log_returns[0]**2)

     def EwmaModel(stickprov, lam):
       """
       EMWA-modellen utifrån en pandas-dataframe "stickprov" som har en␣
     ↪kolumn
       'log_returns' redo.
       """
       log_returns = stickprov['log_returns'].to_numpy()
       sig_dy = np.sqrt(EwmaModelRekursiv(log_returns, lam))
       sig_yr = sig_dy*np.sqrt(252)
       return sig_yr
```

4.1.2 GARCH-modellen

Till skillnad ifrån EMWA-modellen tar denna modell hänsyn till den långvariga medelvariansen, V_L. Formeln lyder:

$$\sigma_n^2 = V_L\gamma + \alpha u_{n-1}^2 + \beta\sigma_{n-1}^2$$

där γ lägger en viktning på den långvariga medelvariansen, α lägger viktning på den kvadrerade förgående dagens avkastningen och β på gårdagens varians. Och eftersom dessa variabler är viktade gäller:

$$\gamma + \alpha + \beta = 1$$

Om du kollar riktigt noggrant är EMWA-modellen bara ett specialfall av GARCH.

Fördelen med denna modell är att den också tar hänsyn till *mean reversion*, att saker det på sikt håller sig stabilt kring en viss volatilitet.

[ ]:
```
def GarchModelRekursiv(log_returns, V_L, gamma, alpha, beta):
  """
  Den rekursiva delen av formeln
  """
  if len(log_returns)==1:
    return log_returns[0]**2
  else:
    return␣
↪V_L*gamma+alpha*(log_returns[0]**2)+beta*GarchModelRekursiv(log_returns[1:
↪],V_L, gamma, alpha, beta)

def GarchModel(stickprov, V_L, gamma, alpha, beta):
  """
  Garch Model utifrån en pandas-dataframe "stickprov" som har en␣
↪kolumn
  'log_returns' redo. Komihåg gamma+alpha+beta=1
  """
  log_returns = stickprov['log_returns'].to_numpy()        # skapa en␣
↪vektor
  sig_dy = np.sqrt(GarchModelRekursiv(log_returns, V_L, gamma, alpha,␣
↪beta))
  sig_yr = sig_dy*np.sqrt(252)
  return sig_yr
```

4.2 Reflektionsfrågor

1. Vilka antaganden bygger våra nya modeller på?
2. Hur hanterar man det prognosfel som uppstår?
3. Finns det omständigheter när den adaptiva modellen är att föredra framför den statiska och vice versa?

4.3 För optioner

Det tog några sidor att gå igenom, men förhoppningsvis gav de föregående bladen någon form av insikt! För det är inte helt oanvändbar kunskap, köpoptioner fyller en viktig funktion. Investerare använder dessa, och liknande, rättigheter som en försäkring mot potentiella risker framtiden skymmer. Optioner är fyller en nyttig funktion, inte bara för investerare, utan också för industriverksamheter. En väldigt användbar option är något som kallas *valutaoptioner*.

En valutaoption ger dig rättighet till att köpa en bestämd mängd av en valuta i framtiden för en förutbestämt växelkurs. Prissättningen av en valutaoption görs på ett liknande sätt som på ett derivat med en aktie som underlag, och med viss ändring i formeln. Black scholes-modellen modifieras till följande:

$$BS_{\text{forex}}(S_0, K, T, \sigma, r) = \Phi(d_1)S_0 \cdot e^{r_f T} - \Phi(d_2) \cdot e^{-r_d T} K$$

där d_1 och d_2 ges av

$$d_1 = \frac{\ln S_0/K + (r_d - r_f + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

variablerna som skiljer från den tidigare modellen är: * S *spot exchange rate*, dagens växelkurs. * r_f *foreign risk-free interest rate*, den utländska valutans riskfria ränta. * r_d *domestic risk-free interest rate* den inhemska valutans riskfria ränta.

[ ]:
```
def black_scholes_forex(S, K, T, r_d, r_f, sigma):
  """
  Lämnas åt läsaren... Behövs i Kapitel 5!
  """
  return bs_pris
```

5 Valutaoptioner med Streamingaffären

[ ]:
```
# Installera Pandas som ger python möjlighet att läsa filer: .csv & .
 ↪xlsx
!pip -q install pandas
# Installera yFinance som kopplar till kursdata från Yahoo Finance
!pip -q install yfinance
# Installera Matplotlib som låter oss rita diagram
!pip -q install matplotlib
# Bibliotek för att räkna med vektorer och liknande i python
!pip -q install numpy
```

[ ]:
```
# Importera Matplotlib som låter oss rita grafer!
import matplotlib.pyplot as plt # För att rita grafer
import yfinance as yf           # För Yahoo Finance
import numpy as np              # För matematik-funktioner
import pandas as pd             # För att hantera dataframes␣
 ↪(excel-blad)
# Redigerar standardinställningar för matplotlib
%matplotlib inline
import matplotlib as mpl
mpl.rcParams['figure.dpi']= 100
plt.style.use('bmh')
```

Efter ditt lyckade projekt med Robyn Green på Streamingaffären har ledningen beslutat att en ny roll på utlandsavdelningen där du innehar en chefsbefattning. Som väl upplyst om kundernas efterfrågan och har du sökt med ljus och lykta efter nästa stora inköp. Ett litet mexikanskt produktionsbolag har väckt ditt intresse med sin kommande tv-serie *Knarkos Suecia*. Serien stämmer, enligt er dataanalys, in på alla indikatorer som era utländska kunder fattar tycke för: spänning, mord och drama.

Du bestämmer genast med med att skicka iväg en förfrågan om ett licensköp. Svaret anländer några dagar senare och kostnaden för licensen är på 2.5 miljoner peso för samtliga 10 avsnitt i första säsongen för fem år framåt med sändningsrättighet i *hela* Europa. Vilket kap! Men ruskigt nära det pris ledningsgruppen har satt sin

inköpsbudget för det kommande året.

Du ryser. Kontraktet måste vara klart så fort som möjligt för att hindra era konkurrenter till att noppa åt sig serien. Något avslappnade är faktumet att betalningen inte behöver ske förrän nästa sommar. *Men vänta nu,* 2,5 miljoner pesos, vad kommer det kosta oss om 12 månader?

Licensköpet är otroligt viktigt för den övergripande strategin: att fånga och behålla en stor kundkrets. Men är å andra sidan en potentiell stötesten för en redan ansträngd ekonomisk situation på företaget. Du har fått klara instruktioner om en inköpsbudget på 1 050 000 kr för hela det kommande räkneskapsåret, allt som allt. En eventuellt överskridande summa måste finansieras med dyra lån. Samtidigt är varenda sparad krona en krona tillbaka företaget som kan hålla er över ytan lite längre. Kanske är det bästa svaret en valutaoption som botar dina huvudbryn.

Du lägger upp de tre alternativet du har på bordet:

1. Köp 2,5 miljoner pesos idag. *Risklöst, men dyrt då det låser budgeten tidigt på året.*
2. Vänta 12 månader och hoppas på att kronan blir mer värd mot peson. *Riskfullt men billigt.*

Du inser nu att du ännu en gång behöver göra en kvantitativanalys som ska ligga till grund för beslutfattande. Det som behövs göra är att få till en prognos kring volatiliteten, och således risken, mellan svenska kronan `SEK`, och mexikanska pesos `MXN`.

Försa steget är att hämta hem historisk data om valutakursen.

Yahoo Finance saknar data från mexikanska pesos `MXN` till svenska kronan `SEK`, så vi blir tvungna att gå från amerikanska dollarn `USD` i ett mellansteg.

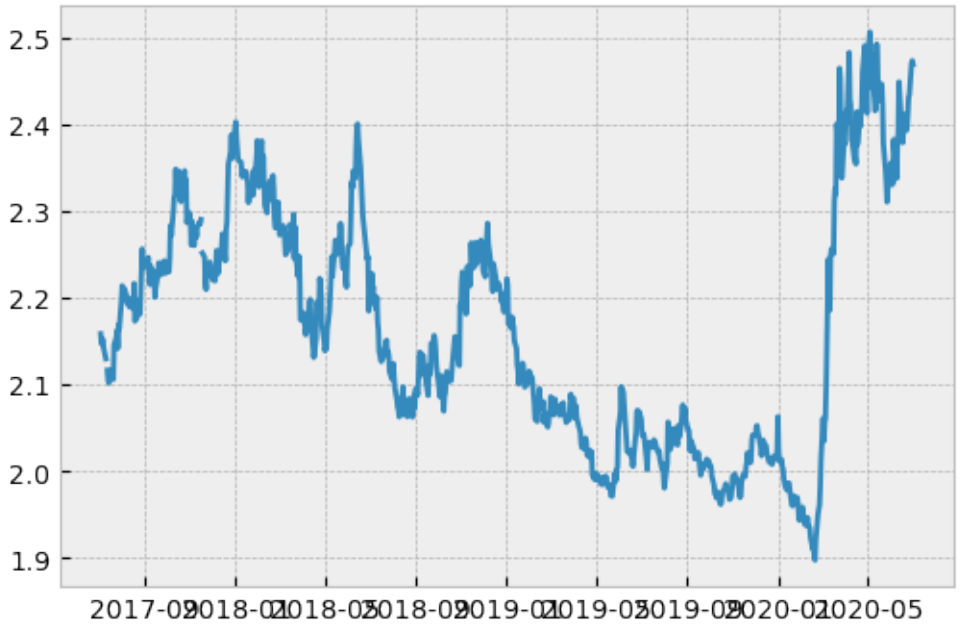
[ ]:
```
# Ladda hem historisk data från Yahoo Finance
USDSEK = yf.download('SEK=X','2017-07-01','2020-07-01')
USDMXN = yf.download('MXN=X','2017-07-01','2020-07-01')


SEKMXN = pd.DataFrame()
SEKMXN['Close'] = USDMXN['Close']/USDSEK['Close']
print(SEKMXN.head())
```

```
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
                Close
Date
2017-07-03  2.158303
2017-07-04  2.146869
2017-07-05  2.151575
2017-07-06  2.151765
2017-07-07  2.141837
```

[ ]:
```
print(SEKMXN.describe()) # En bild av datan du hämtat hem
plt.plot(SEKMXN)
plt.show()
```

```
            Close
count  780.000000
mean     2.163152
std      0.139319
min      1.897551
25%      2.041057
50%      2.150136
75%      2.262372
max      2.506115
```



3 år är en väldigt lång mycket data. Så vi kan plocka ut ett stickprov!

[ ]:
```
stickprov = SEKMXN['2019-01-01':'2020-07-01']
```

För att hitta rätt i denna djungel behövs en kvantitativ analys kring volatiliteten. Hur du bestämer den är upp till dig

- Ska en statisk eller adaptiv modell användas i detta läge?
- Vilka antaganden bygger jag min prognos kring? kan jag testa dessa antagande?
- Hur ser min data ut? Vilken fördelning ser det ut att vara?
- Finns det ytterligare värde med att komplettera med en kausal prognos?

Estimera σ utifrån valfri modell.

[ ]:
```
#sigma = #modell(stickprov, parametrar)
```

Om växelkursen om ett år S_T följer:

$$S_T \sim N(S_0, \sigma\sqrt{T_1})$$

Vad är risken att du kommer överskrida budgeten?

$$\text{Risk} = P[S_T \geq S_{krit}]$$

Där S_{krit} är den kritiska nivå då budgeten spricker!

Vilken tankeställare, tänker du samtidigt som du rör dig mot dagens tredje kopp kaffe. Väl vid kaffemaskinen står finansnissen Hans, och inte kan du inte låta att bikta dina bekymmer.

– Men! Det finns ju ett tredje alternativ, skriker Hans ut:

3. Köp en valutaoption som skyddar mot dyr peso , men samtidigt behåller vinsten om billig.

– Jag ringer banken genast och kollar vad det kan kosta, fortsätter han samtidigt som han hasar sig iväg.

En halvtimma senare glider han in på ditt kontor med ett klart bekymmersamt ansiktsuttryck.

– Femtiotusen kronor för en säkran på $2,25$ `SEK/MXN`, kastar han ur sig. Över budget, men det kanske är värt ändå?

Förberedd sedan ett handlingsbeslut som ska redovisa inför ledningsgruppen huruvida en köpoption ska inhandlas eller inte. Och om inte, vilken risk utsätter sig företaget för?

Det vi vet nu är att:

- $S_0 : 2.47$ `SEK/MXN 1 juli 2020`
- $K : 2,25$ `SEK/MXN`
- $T : 1$
- $r_d : 0\%$
- $r_f : 5\%$

Vilken volatilitet räknar banken med? Kan du föreslå ett annat K som är mer fördelaktig för din situation, men som banken fortfarande borde gå med på?

6 Appendix

6.1 Att använda Python-biblioteken

[ ]:
```
!pip install -q yfinance
!pip install -q pandas
!pip install -q matplotlib
!pip install -q numpy
```

6.1.1 Numpy

Räkna matte är ju roligt. Python är vanligtvis ingen höjdare på matte, utan ofta kommer språk som *Matlab* och *R* och stjäl rampljuset. Som tur är det några som har tagit det roliga till Python och byggt biblioteket *Numpy* som underlättar vektorberäkningar och annat skoj.

[ ]:
```
import numpy as np
X = np.arange(5,24,2) # skapar en vektor
print(X)              # Skriv ut X
print(type(X))        # Vilken datatyp är X?
```

```
[ 5  7  9 11 13 15 17 19 21 23]
<class 'numpy.ndarray'>
```

Även om `X` ser ut som en lista så är den av typen array!

Med hjälp av funktioner från Numpy kan vi utföra räkneoperationer, både snabbt och enkelt.

[ ]:
```
Y = np.round(np.sqrt(X),1)  # elementvis sqrt, sedan avrunda till 1␣
 ↪decimal
print(Y)
print("Y Medelvärde:", np.mean(Y))
print("Y Standardavvikelse:", np.std(Y))
```

```
[2.2 2.6 3.  3.3 3.6 3.9 4.1 4.4 4.6 4.8]
Y Medelvärde: 3.65
Y Standardavvikelse: 0.8249242389456136
```

Vad som annars hade tagit en `for`-loop kan Numpy göra på en rad.

Det går även bra att generera datamängder på mer avancerade vis.

$$Z \sim N(5, 1)$$

[ ]:
```
Z = np.random.normal(5,1,size=10) # normalfördelad population med 10 element
print(Z)
```

```
[4.55918751 5.398011   4.77576504 4.52453627 5.97237526 5.65866942
 4.48479617 5.98023531 5.24800603 6.0364279 ]
```

För vidare läsning om Numpy och alla dess dolda möjligheter rekommenderar vi:

- Stack Absure Numpy Intro
- *Diverse guider på Youtube...*

6.1.2 1.2 Pandas

Python har inga förinstallerade verktyg för att kunna läsa in Excel-filer - men som tur kan vi förlita oss på tredjepartsverktyg som löser biffen!

Pandas är ett bibliotek som ger en ny datatyp till python: DataFrame. När vi läser in `.csv`-filer så konverteras dessa till objekt av dataframe-typen som sedan kan hanteras i Python. Pandas använder sig av Numpy för att hålla ordning på data i kolumner och rader, så det är väldigt behändigt att nyttja funktioner från det ena och använda hos det andra.

Vi hämtar hem `Ericsson aktie`-data m.h.a yfinance-biblioteket.

[ ]:
```
import pandas as pd
import yfinance as yf
ericsson_df = yf.download('ERIC','2020-01-01','2020-07-01') #Ericsson-aktiedata
ericsson_df.head()
```

```
[*********************100%***********************]  1 of 1 completed
```

[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	8.96	9.03	8.94	9.01	8.923130	5368000
2020-01-03	8.83	8.89	8.79	8.85	8.764673	6355200
2020-01-06	8.80	8.95	8.78	8.91	8.824094	3953500
2020-01-07	8.83	8.87	8.80	8.82	8.734962	4259900
2020-01-08	8.83	8.96	8.82	8.90	8.814190	4294600

`Ericsson_df`är en Pandas-dataframe som är indexerad efter datum.

`.head()` är en metod som gör att bara de fem första raderna i dataframen skrivs ut.

[ ]:
```
ericsson_df['Open'].head() # välj kolumn 'Open', titta på .head()
```

[ ]:
```
Date
2020-01-02    8.96
2020-01-03    8.83
2020-01-06    8.80
2020-01-07    8.83
2020-01-08    8.83
Name: Open, dtype: float64
```

Vi kan även filtrera våran `dataframe`.

[ ]:
```
ericsson_df[ericsson_df['Open']>9.5] # välj de rader i df där 'open'␣
↪är större än 9.5
```

[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-06-03	9.57	9.88	9.57	9.84	9.84	19572200
2020-06-04	9.65	9.77	9.64	9.67	9.67	12431900
2020-06-05	9.64	9.73	9.59	9.62	9.62	8579100

[ ]:
```
# Skapa en ny kolumn som heter Medel = (high+low)/2
ericsson_df['Medel'] = 1/2*(ericsson_df['High'] + ericsson_df['Low'])
ericsson_df.head()
```

[ ]:

Date	Open	High	Low	Close	Adj Close	Volume	Medel
2020-01-02	8.96	9.03	8.94	9.01	8.923130	5368000	8.985
2020-01-03	8.83	8.89	8.79	8.85	8.764673	6355200	8.840
2020-01-06	8.80	8.95	8.78	8.91	8.824094	3953500	8.865
2020-01-07	8.83	8.87	8.80	8.82	8.734962	4259900	8.835
2020-01-08	8.83	8.96	8.82	8.90	8.814190	4294600	8.890

Vi kan göra mer komplexa beräknar mellan raderna. Exempel kolla den procentuella skillnadner i `Medel` över tid.

[ ]:
```
ericsson_df['pct_change'] = ericsson_df['Medel'].pct_change()
ericsson_df['pct_change'].head()
```

[ ]:
```
Date
2020-01-02         NaN
2020-01-03   -0.016138
2020-01-06    0.002828
2020-01-07   -0.003384
2020-01-08    0.006225
Name: pct_change, dtype: float64
```

Nämnen! Se att Pandas ansätter odefinierade siffror till `NaN`. Det är inte så konstigt då det inte finns någon tidigare dag att kolla den procentuella förändringen på.

`NaN` kan dock orsaka problem om vi beräknar medelvärden och liknande. Som tur går de att filtrera bort med metoden `.dropna()`.

[ ]:
```
ericsson_df['pct_change'].dropna().head()
```

[ ]:
```
Date
2020-01-03   -0.016138
2020-01-06    0.002828
2020-01-07   -0.003384
2020-01-08    0.006225
2020-01-09    0.002250
Name: pct_change, dtype: float64
```

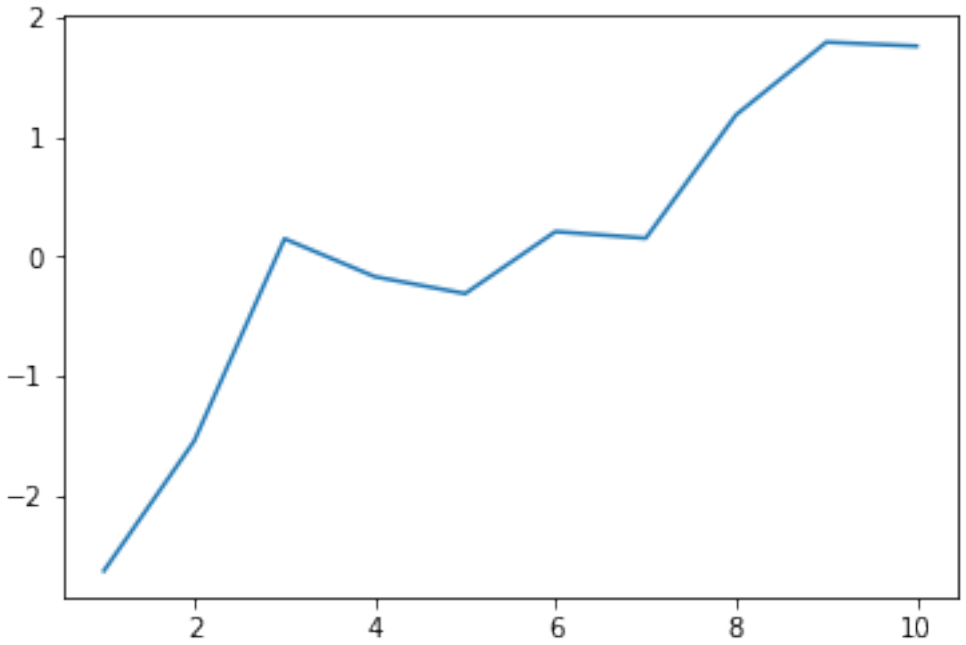
För vidare läsning hänvisar vi till: * 10 minutes to pandas * Kaggle's Guide to Pandas * Stack Overflow för Pandas

6.1.3 Matplotlib

Matplotlib är varken det bästa, det enklaste, eller det snyggaste biblioteket för att visualisera data i Python. Men det går enkelt att plotta data med bara någon rad kod, och man har väldigt mycket frihet! Dessutom fungerar det likt Matlabs plottningsfunktioner vilket gör det väldigt lättlärt för dig som har arbetat med det språket innan.

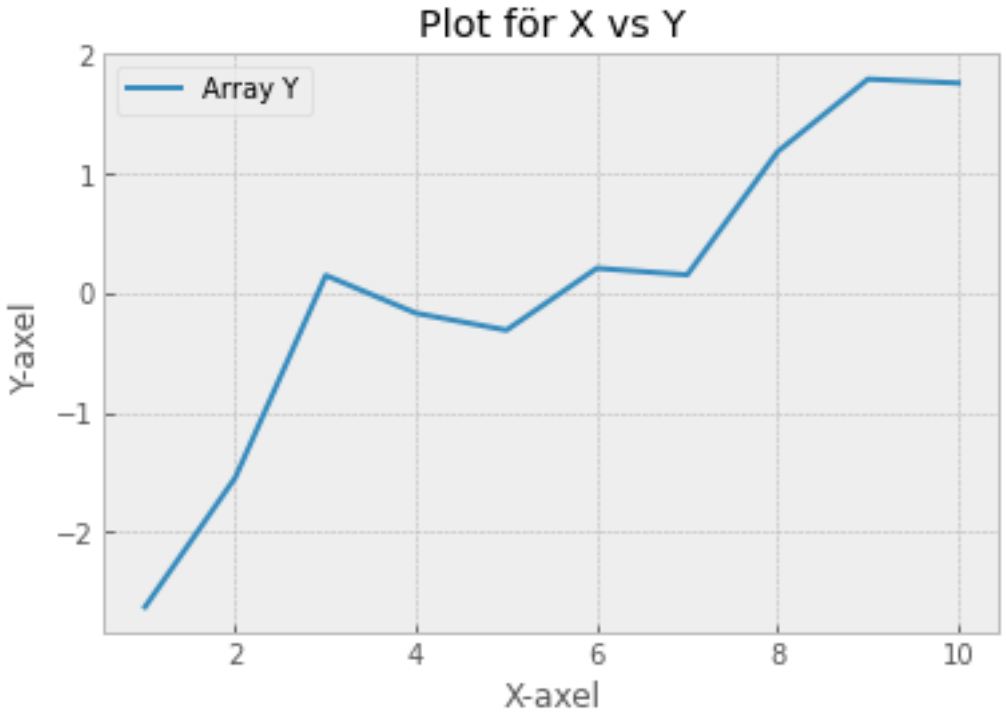
[ ]:
```
import matplotlib.pyplot as plt


X = np.arange(1,11) # en nump
Y = np.random.normal(0,1,size=10)
plt.plot(X,Y)
plt.show() # visar plotten! obs, detta rensar .plt
```



Mm, inte jättesnyggt kanske. Men vi enkelt förbättra plotten!

```
[ ]: plt.style.use('bmh') # byter tema
     plt.plot(X,Y)
     plt.xlabel("X-axel")
     plt.ylabel("Y-axel")
     plt.legend(['Array Y'])
     plt.title("Plot för X vs Y")
     plt.show()
```



Matplotlib erbjuder hög flexibilitet för att rita grafer, men tyvärr går det och blir komplext väldigt fort för att få dessa möjligheter. *Men men*, biblioteket är väldigt användbart och eftersom många andra bibliotek för att visualisera data i Python är byggda från Matplotlib blir det ett måste.

Det är väldigt viktigt att veta att Matplotlib har två sätt att rita diagram, antingen via `.plt` eller någonting som heter `.fig` & `.ax`. Väldigt viktigt att känna till då många guider på internet använder det ena och vissa det andra sättet för att rita sina diagram. Detta utan att nämna att det finns två olika metoder, vilket blir förvirrande för en nybörjare.

Vi rekommenderar följande källa för att få ett grepp om hur Matplotlib fungerar:

- Giles McMullen-Klein guide till Matplotlib

6.2 Terminologi

European call option - ett derivat som ger rättigheten att köpa en aktie för ett förutbestämt pris vid ett bestämt datum.

Hedging - gardera sig mot ogynnsamma pris/kursutveckling genom att göra olika investeringar som helt eller delvis tar ut varandra vid vinst eller förlust.

Arbritage - en process där tillgångar säljs och köps för en riskfri vinst.

Kognitivt bias - Snedvridningeffekt i hjärning som får oss att tänka irrationellt.

Bekräftelsebias - Kognitiv snedvridning som gör att vi noterar det som bekräftar det vi redan tror på och bortser från det som talar emot det vi tror på.

Volatilitet - Ett mått på fluktuation hos en kurs. Ibland sedd som kortsiktigt risk.

Derivat - Ett finansiellt instrument vars värde beror på en underliggande tillgången

Strike price - Det förutbestämda priset för en tillgång i en option.

Time to maturity - Tiden till slutdatum för en option.

Riskneutral - Att vikta vinster och förlust jämt. Människor är inte riskneutrala då vi värderar förluster mycket högre än vinst.

log-normalfördelning - En statistisk fördelning som är lämplig för att modellera tillgångspriser då dessa aldrig kan vara negativa men som samtidigt tar hänsyn till att dessa tillgångar kan stiga i all oändlighet.

Validitet - *se föreläsningsanteckningar ME1316.*

Reliabilitet - *se föreläsningsanteckningar ME1316.*

Urvalsbias - När informationen vi studerar inte är representativt för informationen som helhet.

Stickprov - Ett mindre urval data ur ett stort dataset .

6.3 “Svar” till Reflektionsfrågor

Se inte följande text som det absoluta, och entydiga, svaret till reflektionsfrågorna i kompendiet. Se det som exempel på tankegångar man skulle kunna ha när man svarar på frågorna.

6.3.1 Kapitel 2

1. Hur påverkar aktörers kognitiva bias marknaden och efficient market theoreom?

Kognitiv bias gör att en investerare inte tar helt rationella beslut trots att all information är tillgänglig. Den kognitiva bias som uppstår är förenligt med teori så till vida grad att den enskilda investeraren inte kommer att slå marknaden men oförenlig på det sättet att det då uppstår förtjänster/arbritagemöjligheter för andra investerare.

2. Om nu market efficient hypothesis stämmer och att aktier styrs helt av slump och följer matematiken, varför har man då inte enbart automatiserade algoritmer som handlar?

Automatiserade algoritmer kan endast ta hänsyn till information vi låter dem tillgå, mycket kvalitativ information om omständigheter kan vara svår att kodifiera.

6.3.2 Kapitel 3

1. Hur långt i tiden bör vår historiska data gå tillbaka? Är den beroende på vår prognotiserade tidshorisont?

*… lagom? Går man inte tillräckligt långt tillbaka i tiden riskerar man att ha för få datapunkter. Går man för långt kan volatiliten ha ändrats av annat.*En bra tumregel är att ta med så mycket data som man ska prognostisera på. Med andra ord, om vi ska försöka uppskatta vad volatiliteten är om 2 år så ska historisk data på 2 år användas. Detta med anledning av att fånga in eventuella “systematiska” komponenter. *

2. Vilken säsongsvariation kan finnas i vår data?

Volatiliteten kan ändras när ny information släpps och eftersom företagens rapporter vanligtvis publiceras års- och kvartalsvis är det möjligt att volatiliteten ändras kring dessa punkter.

3. Nu tittar vi på *metric data*, vilken typ av *non-metric data* kan man plocka in?

Man kan söka upp mer information som kan ge mer insikt i datan. Har den underliggande aktien publicerat en kvartalsrapporten som vi bör ta hänsyn till?

4. Bör vi ta med extrema och avvikande datapunkter i vår skattning? varför/varför inte?

Extrema datapunker kan säga något om framtiden eftersom black swan-event, som coronapandemin, bidrar till den osäkerheten om framtiden och vad den väntar. Ett rimligt antagande är därför att volatiliten kommer att vara hög.

5. Bör vi bearbeta datan för att få fram en bättre skattning på σ?

Ja, det behöver vi. Våra modeller är inte bättre än datan som vi stoppar in i dem. Glöm inte; skit in, skit ut.

6. Vilken bias kommer det få på vårt punktsattning $\hat{\sigma}$?

Se 3.2.3 Modellering av bias.

8. Varför tror du $\hat{\sigma}_{imp} < \hat{\sigma}_{hist}$? Gäller det alltid?

Pågrund av det "säkerhetslager" som bankerna lägger på priset för optionen . Nej, om marknaden tror på en mer stabil kurs framöver där den historiska kursen har varit hög så gäller inte sambandet

9. Hur kan vi bestämma ett konfidensintervall på $\hat{\sigma}_{imp}$?

Ofta finns det information om var aktörer har placerat bud på optionen, men inte hittar någon köpare. Dessa kan bli gränser för intervallet.

10. Varför behövs en kvantitativ analys av $\hat{\sigma}$ när vi kan ta den implicerade volatiliteten rakt av?

Den kvantitativa analysen behövs för att kunna se om historien kan säga något om framtiden.

6.3.3 Kapitel 4

1. Vilka antaganden bygger våra nya modeller på?

Att historien säger någonting om framtiden, och att den närmaste tiden har större betydelse.

2. Hur hanterar man det prognosfel som uppstår?

Med en säkerhetsmarginal (konfidensintervall) och alltid utvärdera våra prognosverktyg.

3. Finns det omständigheter när den adaptiva modellen är att föredra framför den statiska och vice versa?

Med en löptid som är kort så bör man använda en adaptiv modell för bättre precision. Det är också en modell som fungerar bra när vi inte har någon säsong eller tydlig trend. En prognos på en längre tidshorisont, där mönstret i datan ser någorlunda konstant ut och med en data som följer en statistisk fördelning så fungerar den statiska bra.