Inlämningsuppgift 3:
Simulering av en swingtradingstrategi för aktiehandel

Bakgrund

Att utnyttja den naturliga pendlingen upp och ner i priset på en aktie, genom att köpa och äga aktier endast några dagar i taget, kallas i tradingsammanhang för *swingtrading*. Det är välkänt att aktier har en tendens att uppvisa överdriven volatilitet och att en nedgång oftare följs av en uppgång än ytterligare en nedgång. I boken ”Framgångsrik aktiehandel – 10 vinnande strategier” presenterar Peter Nilsson, Johnny Torssell och Johan Hellström bland annat en swingtradingstrategi vid namn *Rikoschett-strategin*. Den går, enkelt förklarat, ut på att köpa aktier som gått dåligt under dagen strax innan börsen stänger, för att sedan sälja igen när aktien går upp, dock senast inom fem börsdagar efter köp.

I denna inlämningsuppgift ska du skriva **ett modulärt program** som **läser in historiska data** för en aktie och **simulerar swingtrading med Rikoschett-strategin**. Syftet är att fastställa om strategin har en *edge* (d v s om den leder till fler/större vinstaffärer än förlustaffärer och därmed har potential att öka en investerares kapital). Din kod ska **beräkna avkastningen och längden** (antal dagar) för en genomsnittlig affär. Den ska även **rita upp två figurfönster**, ett som visar fördelningen av vinster/förluster och ett som visar hur ett kapital som använts för denna typ av swingtrading vuxit/minskat under perioden. Den text som **skrivs ut till Command Window** ska även **skrivas till en fil**.

Mål

Inlämningsuppgiften avser att testa:

1. Algoritmer, variabelhantering, loopar, logiska satser
2. Skapa och anropa funktioner
3. Inläsning av data från csv-fil
4. Formaterad utskrift till fil
5. Hantering av strukturer och cellmatriser
6. Att utifrån en uppgiftsbeskrivning kunna skriva ett mindre program

Algoritm

I denna uppgift ska vi skriva ett program som simulerar Rikoschett-strategin för aktiehandel[1]. Reglerna för denna strategi är enkla:

> Regler för Rikoschett
>
> Köp aktien strax innan stängning om aktiekursen ligger i Rikoschett-zonen.
>
> Om aktien ligger på plus dag 1 till 4 efter köpet så sälj positionen strax innan stängning.
>
> Sälj aktien strax innan stängning på femte dagen efter köp oavsett om positionen visar vinst eller förlust.

Rikoschett-zonen innebär en stängning i den nedre tiondelen av dagens *range*, där rangen beräknas som:

$$\text{Range} = \frac{\text{Stängningskurs - Lägstakurs}}{\text{Högstakurs - Lägstakurs}}$$

Högstakurs och Lägstakurs avser här den högsta respektive lägsta kurs som aktien handlats för under innevarande dag. Vi vill med andra ord köpa aktier om:

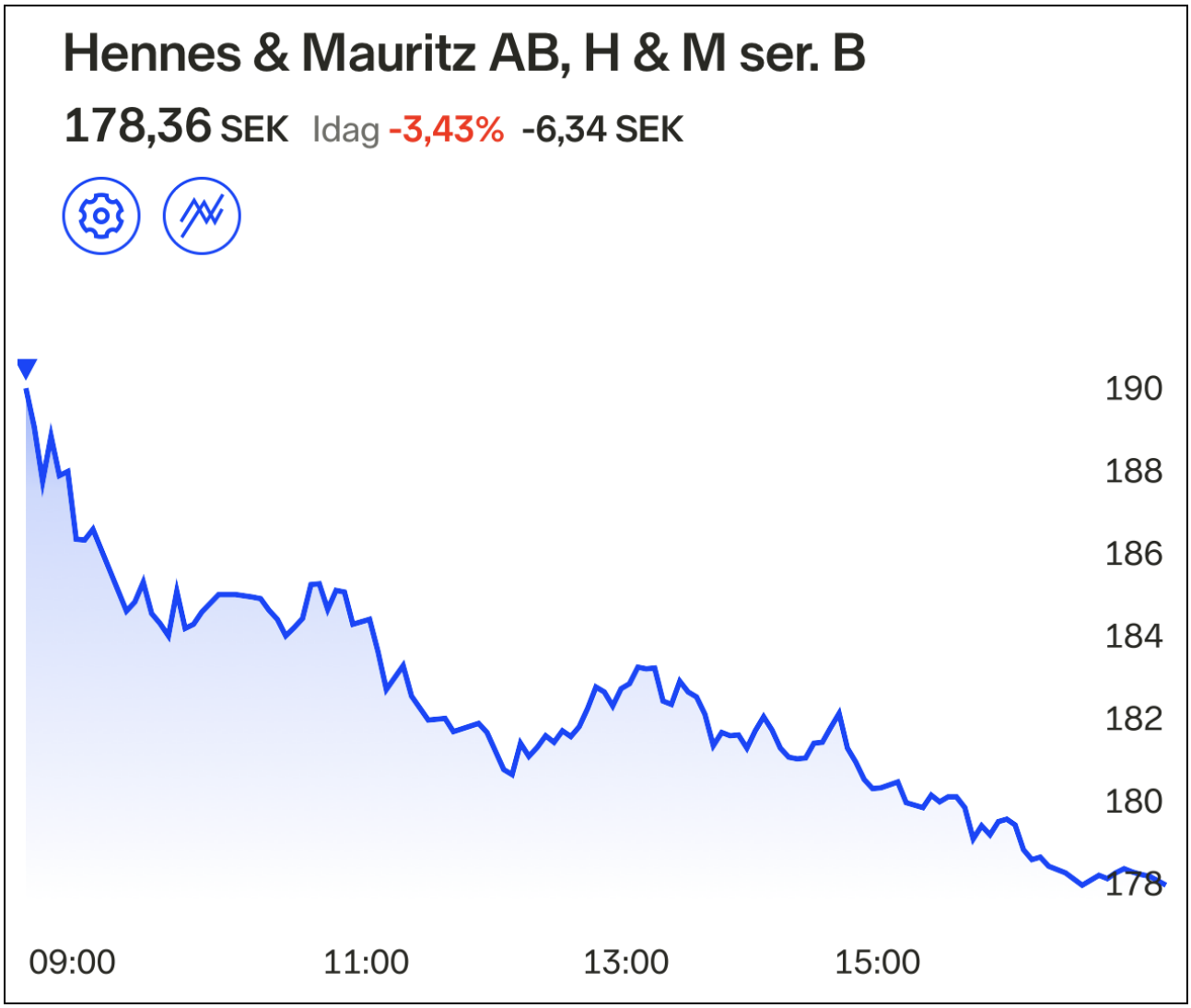
$$\text{Stängningskurs} \leq 0.1*(\text{Högstakurs-Lägstakurs}) + \text{Lägstakurs}$$

och sälja så snart en efterföljande dags stängningskurs överstiger vad vi köpte aktierna för, dock

[1] Källa: Kapitel 4 i ”Framgångsrik aktiehandel – 10 vinnande strategier” av Nilsson, Torssell & Hellström (Aktiespararna Kunskap, ISBN 978-91-89212-43-5).

senast vid stängning på den femte dagen efter köp (detta för att slippa bli sittande med aktier som sjunkit i värde).

Figur 1 nedan illustrerar hur en aktiekurs kan variera under en handlingsdag på Stockholmsbörsen, från öppning 09:00 till stängning 17:25 (i detta exempel Hennes & Mauritz B från 30/9-2021).



Figur 1: Illustration över variationerna i pris för en Hennes & Mauritz B-aktie på Stockholmsbörsen den 30/9-2021. Aktien öppnade på 190,90 kr (vilket visade sig bli dagens högsta) och stängde på 178,36 kr, med en lägstanotering på 178,20 kr. Detta innebär att den stängde i den så kallade rikoschett-zonen. Bilden är ett screenshot från nätmäklaren Nordnets app.

Det kan mycket väl hända att en aktie stänger i Rikoschett-zonen flera dagar efter varandra. I vår simulering kommer vi då att anta att vi går in i en ny affär varje gång så sker, d v s vi kan ha flera samtidigt pågående affärer, med olika köppris och olika villkor för när försäljning ska ske.

För att inte komplicera uppgiften onödigt mycket kommer vi att anta att vi inte har några omkostnader för aktiehandeln[2]. Vi kan därmed ”räkna på 1 aktie”, d v s anta att för varje affär vi initierar så köper vi en enstaka aktie. Vi är helt enkelt bara intresserade av att identifiera den procentuella utvecklingen av vårt kapital som vi kan erhålla med denna strategi.

Uppgift

Börja med att ladda ner de fyra datafilerna (`ERIC_B-2015-10-01-2021-09-29.csv`, `HM_B-2015-10-01-2021-09-29.csv`, `TELIA-2015-10-01-2021-09-29.csv` och `VOLV_B-2015-10-01-2021-09-29.csv`)[3]. Du kan använda dig av en enskild datafil när du utvecklar ditt program, men testa därefter att ditt program fungerar även med de andra datafilerna (detta är ett enkelt test för att se till att du inte hårdkodat in någon information). Filerna finns att hämta i mappen ”Inlämningsuppgift 3” bland kursens dokument i Canvas.

Skriv ett program, i form av en skriptfil `rikoschett.m` i Matlab, som:

- **Läser in data från en av datafilerna** och sparar den data vi behöver i en *struct*. Detta ska åstadkommas av en funktion, `readStockData`, som du också ska skriva (mer information om denna senare).

- Efter inläsningen av data behöver programmet **simulera swingtrading enligt Rikoschett-strategin**. Detta ska åstadkommas av en funktion, `simulateStrategy`, som du också ska skriva (mer information om denna senare). Denna funktion ska använda den *struct* som den första funktionen returnerade som invariabel, och ska svara med en *struct* som innehåller en sammanställning över samtliga affärer som genomförts och en vektor som innehåller hur kapitalet utvecklats över dessa affärer.

- Programmet ska därefter **skriva ut resultatet av simuleringen till skärmen** och **rita upp de efterfrågade graferna**. Även detta ska göras av en funktion, `printAndPlotResult`, som du också ska skriva (mer information om denna senare). Med andra ord kommer själva huvudprogrammet `rikoschett.m` att vara kort och se ut ungefär på följande sätt:

[2] I verkligheten behöver man ofta betala någon form av *courtage*, d v s en avgift för själva köpet eller försäljningen av aktier.

[3] Du kan själv hämta data för andra aktier och/eller perioder här: http://www.nasdaqomxnordic.com/aktier

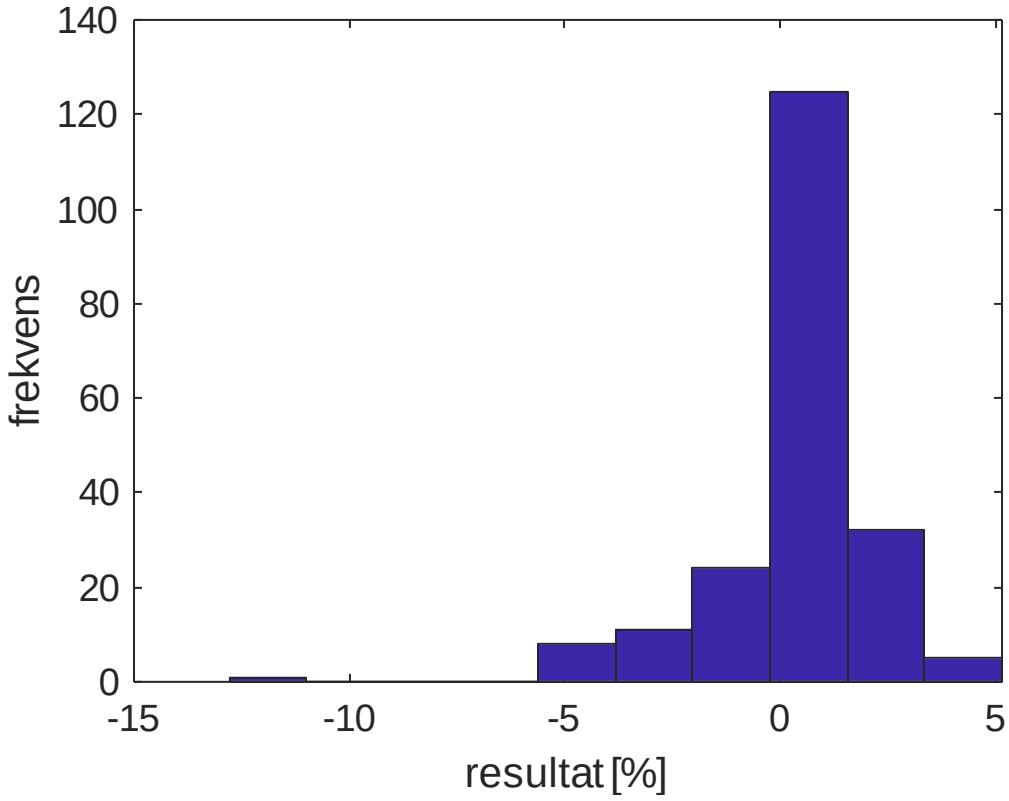
```
clear all
close all
clc

% Read data
filename = 'TELIA-2015-10-01-2021-09-29.csv';
stock = readStockData(filename);

% Simulate strategy
[transaction, money] = simulateStrategy(stock);

% Print and plot results
printAndPlotResult(stock, transaction, money);
```

- Utskriften till kommandofönstret ska se ut på följande sätt (exempel med `TELIA-2015-10-01-2021-09-29.csv` som indatafil):

```
Totalt antal dagar i data: 1504
Antal rikoschett-lägen: 206 (13.70%)
Genomsnittligt utfall: 0.30%
Genomsnittlig träffsäkerhet: 78%
Genomsnittligt antal dagar i affär: 2.25
```

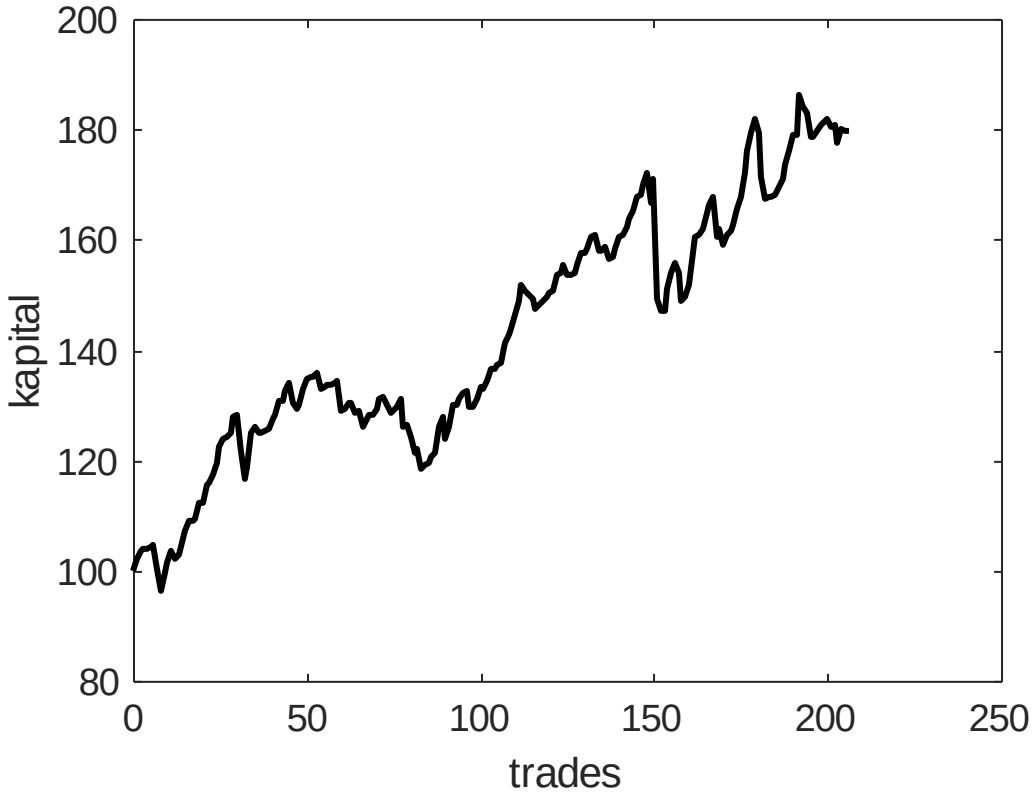
  Exakt samma text ska också skrivas till en textfil, `myResult.txt`.

- Den första av de två figurerna som ska ritas upp är ett histogram över utfallet i de affärer som har genomförts:



Det exakta utseendet på grafen är inte viktigt denna gång, men det ska vara ett histogram som visar hur många affärer (frekvens) som faller inom ett givet spann i termer av utfall (resultat) i procent.

- Den andra figuren ska visa hur ett kapital utvecklas över tid om vi tänker oss att vi kan göra alla affärerna vi har identifierat, en efter en. Med andra ord utgår vi från 100% av vår förmögenhet, och låter den sedan öka/minska lika mycket som utfallet av varje affär, *som om vi hade köpt aktier för hela vårt kapital i varje enskild affär i en lång sekvens*. Exemplet nedan visar hur det ser ut med `TELIA-2015-10-01-2021-09-29.csv` som indatafil:



Återigen är det exakta utseendet på grafen inte viktigt denna gång, men det ska vara en linjegraf som illustrerar utvecklingen av kapitalet (på *y*-axeln) som funktion av antal trades (affärer) (på *x*-axeln).

Rent generellt gäller för denna uppgift att det är *hur programmet fungerar* som är det viktiga, inte exakt hur du löst uppgiften eller exakt hur graferna ser ut (bara informationen är den som efterfrågas och i övrigt korrekt). I denna anda följer nu information om hur de tre funktionerna kan konstrueras. *Om du löser uppgiften på ett annat sätt så är det helt OK, så länge din lösning fungerar och inte bygger på någon form av hårdkodning!*

Funktioner

Funktionen `readStockData` ska ha ett inargument och ett utargument.

Inargument: `filename` en textsträng (char-vektor eller string) som anger filnamnet på den datafil som ska användas

Utargument: `stock` en structure som innehåller fyra fält: `high`, `low`, `close` och `range`

Funktionen ska läsa in data från filen på valfritt sätt (använd t ex `importdata`, *Import Data*-guiden med *Generate script/function*, eller vad du nu känner dig mest bekväm med). Bilden nedan visar hur början på en datafil ser ut om vi kikar på den i Excel:

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Bid	Ask	Opening price	High price	Low price	Closing price	Average price	Total volume	Turnover	Trades
2	2019-07-02	42,5	42,53	42,25	42,74	42,12	42,64	42,538	10719375	455975365,3	6959
3	2019-07-01	42,19	42,2	41,65	42,32	41,59	42,21	42,086	13346907	561691322,2	8339
4	2019-06-28	41,32	41,33	41,23	41,51	41,07	41,24	41,284	10681011	440954465,5	5473
5	2019-06-27	41,07	41,1	41,38	41,4	41,05	41,16	41,163	7209163	296754887,4	4230
6	2019-06-26	41,3	41,32	41,51	41,52	41,26	41,43	41,39	6047530	250308858,3	3747
7	2019-06-25	41,53	41,55	41,32	41,6	41,12	41,47	41,457	6546448	271383110,6	3877
8	2019-06-24	41,24	41,26	41,44	41,76	41,24	41,32	41,419	10807744	447606234	6176
9	2019-06-20	41,63	41,65	41,92	42,04	41,59	41,68	41,705	14829120	618445462,1	5740
10	2019-06-19	41,64	41,67	42,07	42,07	41,58	41,58	41,692	10459539	436092724,4	4745
11	2019-06-18	42,03	42,04	41,32	42,1	41,18	42,01	41,812	12695128	530855575,1	5270
12	2019-06-17	41,24	41,26	41,35	41,53	41,21	41,21	41,319	5829753	240874365	3694

Vi är intresserade av att plocka ut High price (högstakurs, till fältet `high`), Low price (lägstakurs, till fältet `low`) och Closing price (stängningskurs, till fältet `close`). Vi vill också fylla fältet `range` med en beräkning av (Closing price – Low price)/(High price – Low price).

Att datafilerna innehåller ,-tecken istället för . som decimalavskiljare kan, beroende på hur du läser in datan, ställa till problem. Du kan t ex använda `regexp` för att byta ’`,`’ mot ’`.`’ i en teckensträng.

Datafilerna har hämtats direkt från Stockholmsbörsen och är sorterade i fallande datumordning. För att grafen som visar kapitalets utveckling över tid ska bli rättvisande behöver du därför byta ordning (”flippa”) vektorerna som sparats i de olika fälten. Detta kan du göra t ex med `fliplr`.

Funktionen `simulateStrategy` ska ha ett inargument och två utargument.

Inargument: `stock` utargumentet från `readStockData`

Utargument: `transaction` en vektor av structures där varje element i struct-vektorn representerar en affär; structen har fyra fält: `buy`, `sell`, `days`, `gain`

`money`	en vektor som visar kapitalets utveckling om alla affärer i `transaction` utförs i en följd efter varandra

Funktionen behöver loopa över alla dagar som ingår i datan i `stock` och för varje dag avgöra huruvida `range <= 0.1`. Om detta är fallet ska en ny affär initieras och ett nytt struct-element ska initialiseras i `transaction`. `buy`-fältet ska spara stängningskursen (från `close`-fältet i `stock`), vilket är den kurs som man köper aktier för i just denna affär.

För varje affär måste funktionen stega framåt i tiden i `stock` och avgöra när affären avslutas. Detta sker när `stock.close` är större än `transaction.buy` för den aktuella affären ELLER vid stängning fem dagar efter köp, vilket som nu inträffar först.

När affären avslutas sparas stängningskursen i `sell`-fältet (det är denna kurs vi säljer vid). Antalet dagar affären pågått (1, 2, 3, 4 eller 5) sparas i `days`-fältet. Avkastningen (den procentuella vinsten) sparas i `gain`-fältet och beräknas som:

```
transaction(i).gain = (transaction(i).sell - transaction(i).buy)/transaction(i).buy;
```

där `i` är den aktuella affären.

Vektorn `money` uppdateras genom att nästa värde tilldelas som en funktion av föregående värde multiplicerat med en faktor (1 + avkastningen):

```
money(i+1) = money(i)*(1 + transaction(i).gain);
```

Första elementet i `money` måste således innan dess initialiseras till 100. Med andra ord blir `money` ett element längre än `transaction`.

Notera att om ett rikoschett-läge inträffar när det är färre än 5 dagar kvar i data kommer vi inte att kunna utvärdera detta läge på ett korrekt sätt. Ta därför inte med de sista 5 dagarna när du letar rikoschett-lägen. Du kan t ex välja att endast leta efter dagar med rätt range i `stock.range(1:end-5)`.

Funktionen `printAndPlotResult` ska ha tre inargument och inga utargument.

Inargument:		
	`stock`	utargumentet från `readStockData`
	`transaction`	utargumentet från `simulateStrategy`
	`money`	utargumentet från `simulateStrategy`

Funktionen ska skriva ut ett textmeddelande likt det som visas i exemplet på sidan 5, dels till Command Window och dels till en fil, `myResult.txt`. Utskriften görs lämpligen med `fprintf` och kan då skrivas på samma sätt till skärm och fil (m h a `fopen` och `fclose`). Histogrammet ritas med `hist` och kapitalutvecklingen med `plot`. Det är informationen i vektorn `[transaction.gain]` som ska illustreras i histogrammet och informationen i vektorn `money` som ska illustreras i linjegrafen.

Totalt antal dagar i datan kan erhållas ur dimensionerna på `stock`. Antalet rikoschett-lägen utläses från längden på `transaction`, och kan sättas i förhållande till storleken på `stock`. Det genomsnittliga utfallet är ett enkelt medelvärde av `[transaction.gain]` och den genomsnittliga träffsäkerheten avgörs av antalet affärer med `transaction.gain > 0` i förhållande till det totala antalet affärer. Det genomsnittliga antalet dagar i affär är ett enkelt medelvärde av `[transaction.days]`.

Ledning

Det är som vanligt **starkt rekommenderat att dela in uppgiften i mindre delar** för att varje del för sig ska bli mer överskådlig. Vi ser då att uppgiften består av sex huvudsakliga delmoment:

1. Läsa in en given .csv-fil med data
2. Identifiera den information som ska sparas i en struct
3. Simulera Rikoschett-strategin baserat på inläst data
4. Rita histogram över utfallet och linjegraf över kapitalutvecklingen
5. Skriva formaterad data till Command Window och till fil
6. Skapa ett modulbaserat program där steg 1-2 utförs i en funktion, 3 i en annan och 4-5 i en tredje.

Ett förslag är att göra uppgiften i ordningen 1-2-3-4-5-6, d v s man kan börja skriva ett långt script och först på slutet dela upp det i tre funktioner, om man vill fokusera på själva huvudproblemet i början och inte fastna i funktionsdefinitionerna. **Testa programmet efter varje delsteg**. För att enklare kunna verifiera att mellanresultaten är riktiga, kan du t ex tillfälligt ta bort semikolonet på slutet av en rad.

Följande avsnitt i kursboken/Matlabhjälpen är extra viktiga för just denna inlämningsuppgift:

- Inläsning av data från .csv-fil (Kapitel 9.2-3)
- Hantering av strukturer och cellmatriser (Kapitel 8.1 & 8.2)
- Funktionsfiler (Kapitel 3.7 & 10.1)
- Skriva formaterad text till fil (Kapitel 3.3.2 & 9.3)
- `if`-satser, `for`-loopar, `while`-loopar (Kapitel 4.1-2 & 5.1-4)
- Histogram (Kapitel 12.1.2)

Det är **starkt rekommenderat** att ha gjort åtminstone några övningar ur boken och övrigt material på varje område innan du kastar dig över uppgiften!

Formalia

Du *ska* använda **kommentarsatser** när du programmerar! Det är mycket viktigt! **Ditt huvudprogram såväl som dina funktionsfiler ska innehålla programhuvud** med **ditt namn**

och personnummer, samt **en kortare förklaring** av vad de gör. Programhuvudet i funktionerna ska innehålla information om vad de olika in- och utargumenten representerar. Koden skall i övrigt vara kommenterad på ett sådant sätt att det går att följa vad den gör.

Elektronisk inlämning

Inlämningsuppgiften ska rapporteras genom att de nödvändiga filerna lämnas in elektroniskt via Canvas **senast fredagen den 15:e oktober klockan 23:59**. **Följande fyra filer ska bifogas:** `rikoschett.m`, `readStockData.m`, `simulateStrategy.m`, och `printAndPlotResult.m`. m-filerna ska vara **väl kommenterade samt innehålla en beskrivning** av vad de gör.

Glöm inte att fylla i **namn** i programhuvud i alla filer ordentligt (så att vi är säkra på vem som gjort uppgiften)! Vid misstanke om plagiat skickas ärendet direkt till disciplinnämnden.

Du behöver **inte** skriva någon rapport. Ladda upp filerna en och en utan att packa ihop dem till en ZIP. Ladda inte upp de utdelade csv-filerna (vi har dem redan). Vid rättning kommer en femte ”hemlig” datafil användas, så det finns ingen mening i att hårdkoda information som rör filnamnen.

Lycka till med uppgiften!