Datorlaboration 1, Statistisk översiktskurs (SÖK)

I den här första datorlabben kommer ni bekanta er med programmeringsspråket R och dess arbetsmiljö RStudio, använda R som miniräknare, göra enkla beräkningar och plottar och ladda ned data, mm.

Ta det lugnt

Det är många nya saker som presenteras, var beredd på att allt inte kommer att sjunka in direkt. Försök att ta dig igenom hela texten, och övningarna. Du kan sen återkomma till denna labb på andra datorlaborationer för att repetera de delar du behöver.

*Under kursens gång, titta också på de olika videos som finns under **R-hjälpen**, i Athena.*

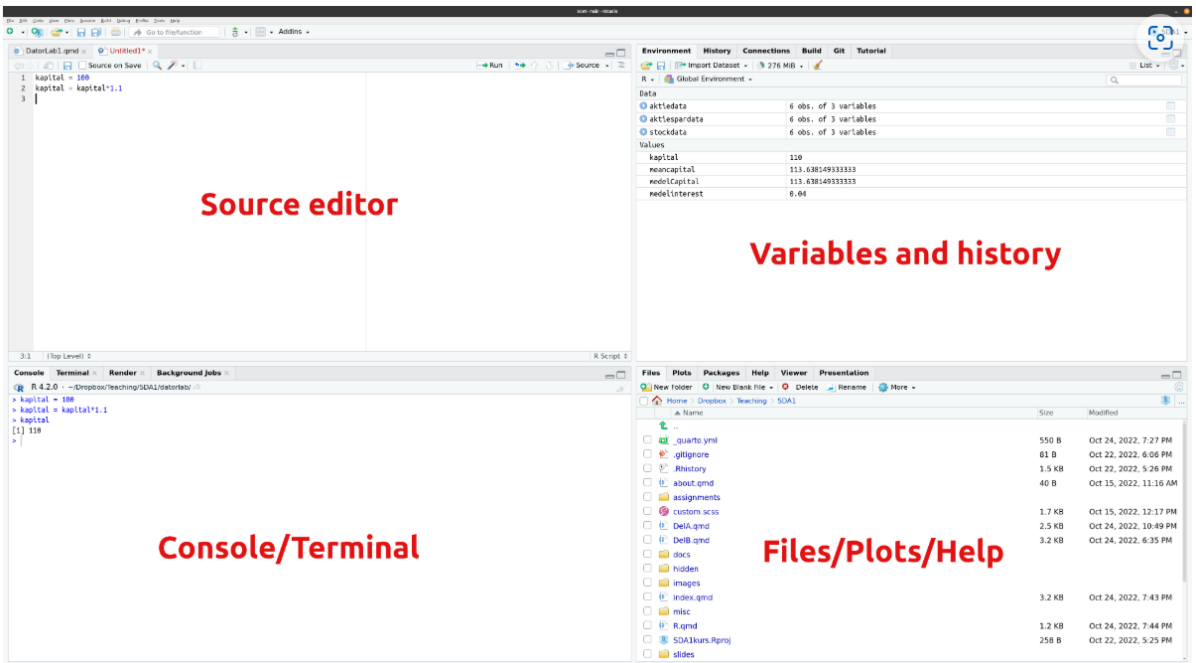
Installera R på egen dator

Om du inte sitter i en datorsal på Campus måste du först installera R och sen RStudio.

Gå till denna länk från Statistiska institutionen, gå till rubriken ”Ladda ner R och Rstudio”, där finns instruktioner steg för steg. Versionsnumret på R kan skilja sig från det nummer som står i instruktionen, eftersom mjukvaran hela tiden uppdateras.

När du är klar med installationen, gå tillbaka till denna labbinstruktion.

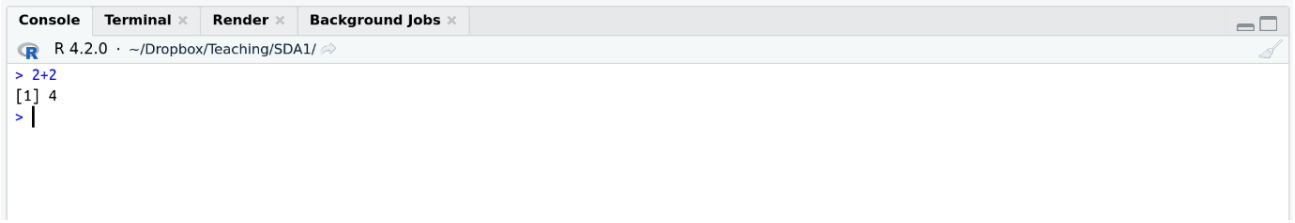
För att starta RStudio letar du upp programmet på din dator och klickar på startikonen. När RStudio startas upp kommer det att se ut så här (kan se olika ut på olika datorer):



De olika delarna av RStudio kallas ofta för **Panes**. Vi kommer gå igenom dessa delar vartefter, men vi börjar med att utforska **Console**, även kallad **terminalen**.

1. Använda R som en miniräknare

Ett bra sätt att vänja sig vid R är att använda R som en slags miniräknare. I fönstret **Console** i RStudio (vanligtvis nere till vänster) kan man skriva olika typer av **kommandon** som skickas till R för beräkning:



Tecknet > kallas för **kommandoprompt** (eller bara **prompt**) och är R's sätt att tala om att det väntar på att ett nytt kommando ska skrivas in (vid det blinkande strecket). Vi säger att vi "skriver in något på prompten".

Multiplikation och division

R använder * för multiplikation och / för division. Potenser (upphöjt till) skrivs med tecknet ^, 3^2 är mao. 9. Du skriver tecknet ^ genom att trycka Shift + tangent till vänster om Enter. Sen måste du trycka Mellanslag/Space innan du skriver exponenten, annars blir det 3^2 och det fungerar inte i R.

Uppgift 1.1

Prova att skriva 2+2 efter det nedersta > tecknet i **Console** och sedan trycka på Enter-tangenten. R bör svara (**returnera**) med talet 4.

Uppgift 1.2

Beräkna 3^2 (3 upphöjt till 2) i Console. [läs informationen ovan]

Uppgift 1.3

Skriv in talet (2+3)/(2+5) i Console och se att R svarar med 0.7142857. (se till att få med alla parenteser)

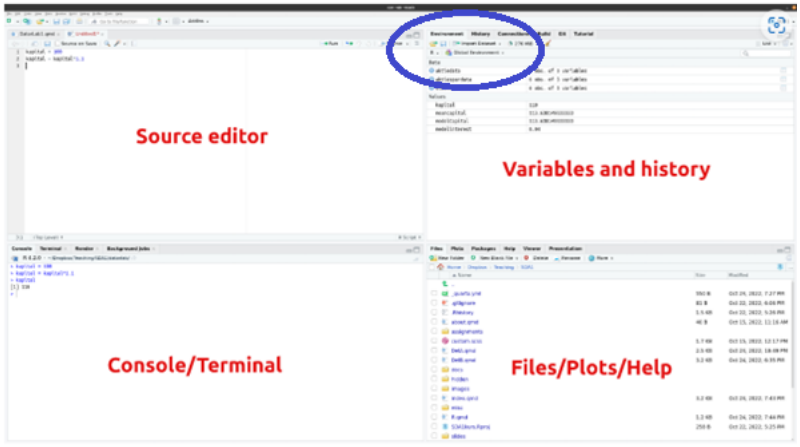
Uppgift 1.4

Du köper aktier för 100 kr. Avkastningen första året är 10%. Använd R för att beräkna värdet på ditt aktiekapital efter ditt första år som aktiesparare, dvs skriv in 100*1.1, tryck Enter, och se R returnera 110.

När du har "kört" ett kommando i R (skrivit in kommandot och tryckt Enter) kan du trycka uppåtpilen på tangentbordet, du kommer då till ditt senaste kommando (ibland vill man bara ändra någon detalj i ett kommando som man redan använt sig av). Om man trycker uppåtpil igen kommer man till näst senaste kommandot, osv.

---

*Uppe till höger i Rstudio finns också en flik som heter **History**, där kan du se dina senaste kommandon. Om du dubbelklickar på ett kommando i listan dyker det kommandot upp i Console. När nya kommandon körs kommer kommandon från längre upp i historiken att försvinna från History.*

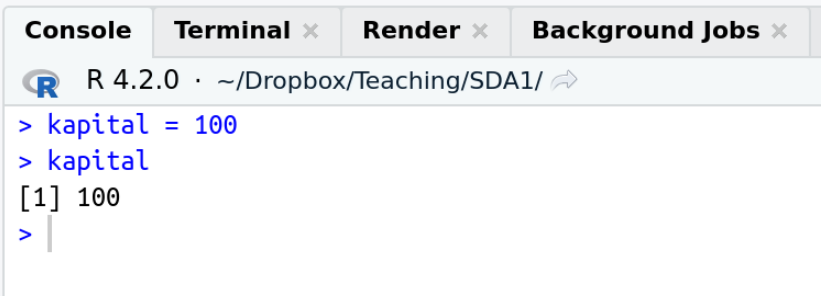


...

2. Använd variabler och definiera en vektor i R

Du är nu inne på ditt andra år som aktiesparare. Avkastningen år 2 är 6%. Hur mycket aktiekapital har du efter år 2? Vi kan beräkna detta genom 100*1.1*1.06 i Console och få svaret 116.6 kr. Men finns det något sätt att återanvända vår tidigare beräkning 100*1.1 = 110 kr så att vi bara behöver multiplicera detta tal med ökningen för år 2, dvs. 1.06?

Vi kan lösa detta genom att spara undan vår första beräkning i en **variabel**. Vi kan ge denna variabel (nästan) vilket namn vi vill. Vi kommer kalla den för **kapital** och börjar med att sätta värdet på variabeln kapital till 100, det ursprungliga kapitalet. Vi skriver kapital = 100 i Console. Vi kan sen testa att R nu faktiskt minns att kapitalet är 100 genom att bara skriva variabelns namn, dvs. kapital, följt av Enter i kommandoprompten i Console:



R skriver ut det värde (100) som vi tilldelade variabeln kapital.

Språkbruk

*vi säger att vi **tilldelar variabeln** kapital **värdet** 100. Lite mer slarvigt säger vi: ”vi sätter kapital till 100”.*

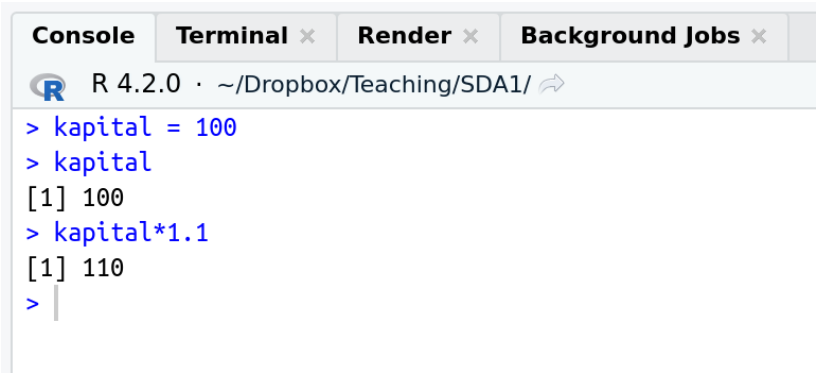
Se upp!

*Om du stänger ner RStudio och sen startar om programmet (eller om RStudio låser sig) minns inte R längre värdet på kapital och andra variabler. R minns faktiskt inte ens att det fanns något som hette kapital och kommer att klaga om du skriver kapital på prompten följt av Enter. R och RStudio minns bara variabeln inom en **session**, dvs tills du avslutar RStudio. Om man vill spara data mellan olika sessioner måste man spara ner variablerna på datorns disk (eller på någon lagring på internet). Mer om detta senare.*

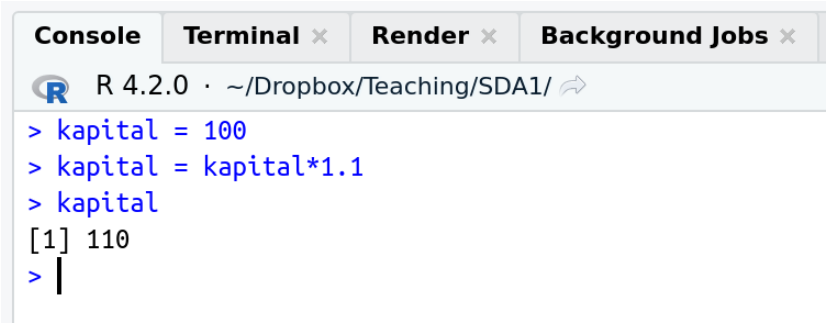
Variabeltilldelningar

Istället för att skriva **kapital = 100** kan vi lika gärna skriva **kapital <- 100** , där symbolen **<-** skrivs med de två tecknen < (mindre än) som finns långt ner till vänster på tangentbordet och - (bindestreck). Att skriva variabeltilldelningar med **<-** är egentligen den rekommenderade varianten i R men både varianterna fungerar.

Vi kan nu beräkna kapitalet efter år 1 (i början av år 2) genom att multiplicera variabeln kapital med talet 1.1:



Vi kan också **skriva över** värdet i variabeln kapital med ett nytt värde. Vi kanske vill att kapital alltid ska innehålla värdet på det aktiekapital som jag har just nu. Låt oss först ändra värdet på kapital till värdet efter år 1:

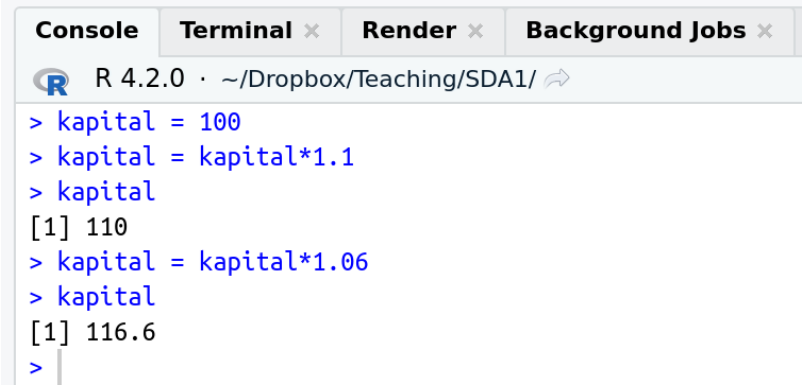


Notera speciellt raden kapital = kapital*1.1. R läser detta som:

”Jag (R) ska plocka fram värdet 100 ur variabeln kapital. Sen ska jag multiplicera det med 1.1 för att få talet 110. Det talet 110 stoppar jag sen tillbaka i variabeln kapital.”

Värdet på variabeln kapital är nu alltså 110.

Det fina med det här att vi nu kan fortsätta att ändra variabeln kapital efter att ytterligare ett år har gått, dvs värdet på ditt aktiekapital efter år 2:



Variabeln kapital är nu 116.6 och du är redo för din kommande avkastning under år 3.

Uppgift 2.1

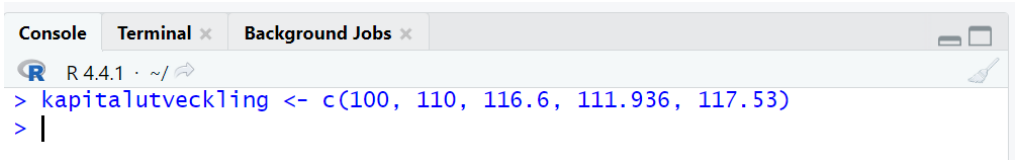
Om du inte redan gjort det börja med kapital = 100 och gå igenom stegen ovan.

Avkastningen år 3 blev tyvärr minus 4%. Uppdatera variabeln kapital så att den visar att värdet på kapitalet efter år 3 är 111.936 kr. Notera att en minskning med 4% innebär att vi måste multiplicera med (1-0.04), dvs med 0.96. Multiplikation med tal mindre än 1 leder till en minskning av kapitalet.

Uppgift 2.2

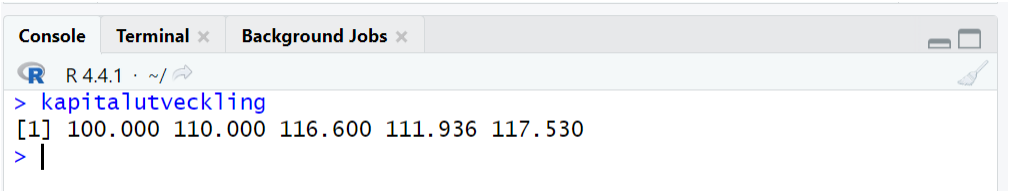
Du började år 1 med 100 i kapital, i början av år 2 hade du 110, sedan 116.6 i början av år 3 och 111.936 i början av år 4. Vi antar att kapitalet växer så att det är 117.53 i början av år 5.

Skriv in följande i R i Console, och tryck Enter:

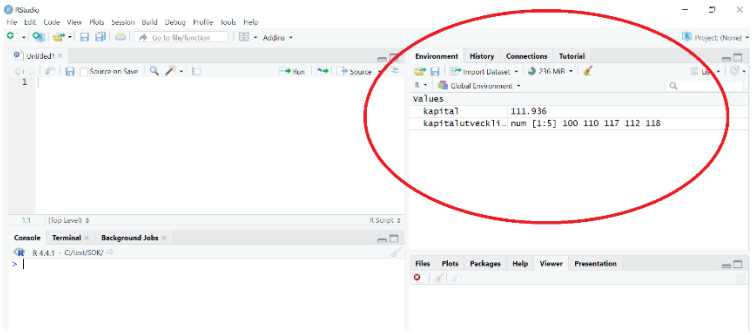


Denna kodrad definierar en **vektor**, dvs. en sekvens av tal, i detta fall fem tal. Var noggrann med att ha kommatecken mellan varje tal och att använda punkt för decimaltal. Kommanot **c()** är vad som definierar vektorn.

Om du vill se hur vektorn ser ut, skriv kapitalutveckling i Console och tryck Enter:

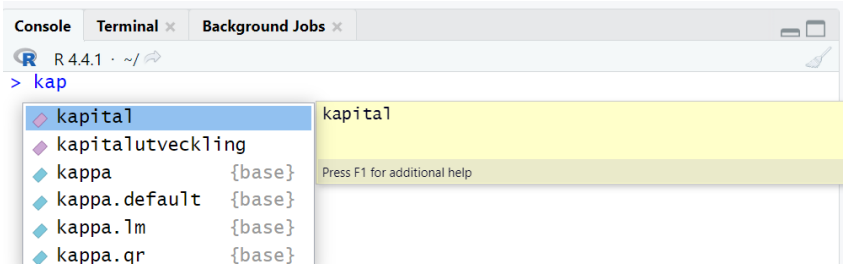


Din vektor ska också ha dykt upp i övre högra delen av Rstudio. Om du inte kan se din vektor kapitalutveckling uppe i högra delen av Rstudio, klicka på fliken **Environment**, du ska se kapitalutveckling i listan:



---

*När du är i Console och börjar skriva ett kommando, kommer R att **efter tre bokstäver** ge dig förslag (om det finns något kommando, någon variabel, eller liknande, som börjar på just dessa tre bokstäver). Om du ovan har definierat kapital och kapitalutveckling ovan, bör R ge dig förslag om dessa och andra kap-ord när du skrivit de tre bokstäverna kap:*



*Du kan gå med tangentbordets piltangenter i listan. Om du står vid ett av ordförslagen och trycker datortangenten **Tab** (till vänster på tangentbordet) skrivs (i fallet ovan) just ”kapital” ut i Console.*

---

3. Gör en graf över kapitalutvecklingen

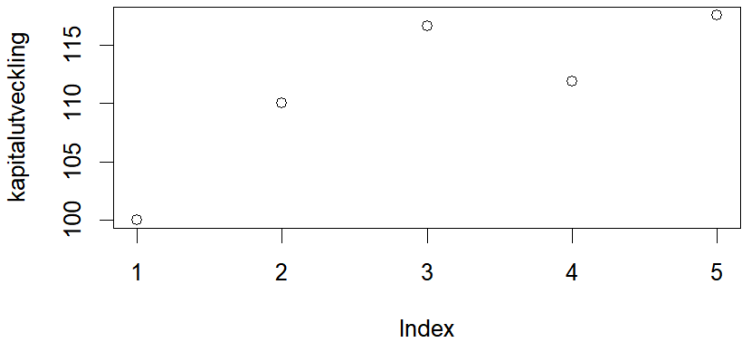
Uppgift 3.1

Gör nu följande kommandon, för att göra grafer över din kapitalutveckling.

```
Console  Terminal  Background Jobs
R 4.4.1 · ~/
> plot(kapitalutveckling)
>
```

R gör nu en graf över utvecklingen. Vektorn kapitalutveckling har fem värden och R plottar helt enkelt dessa värden som y-värden (vertikala axeln) och väljer i detta fall själv en x-variabel och x-axel som är ordningsnumret för varje observation (ett index). I vårt fall överensstämmer detta index med åren 1-5.

Du bör få en figur som denna:

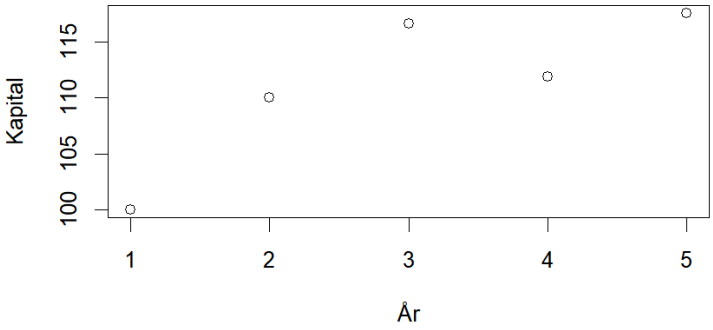


Uppgift 3.2

Testa att gå till det senaste kommandot, i Console, genom att trycka uppåtpil på tangentbordet. Om du sen stegar bakåt i kommandot med vänsterpil kan du ändra i kommandot.

Byt namn på axlarna i grafen, till ”Kapital” och ”År”, genom att använda **argumenten ylab** och **xlab**:

```
Console  Terminal  Background Jobs
R 4.4.1 · ~/
> plot(kapitalutveckling, ylab="Kapital",xlab="År")
>
```

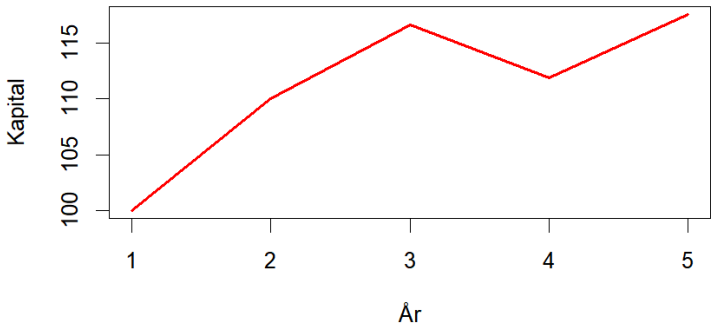


Uppgift 3.3

Testa uppåtpil igen för att komma tillbaka till senaste kommandot.

Gör nu så att, istället för punkterna i figuren ovan, visas en linje som binder samman punkterna (argumentet **type=”l”**) och välj röd färg och tjockleken 2 (argumentet linewidth, **lwd**):

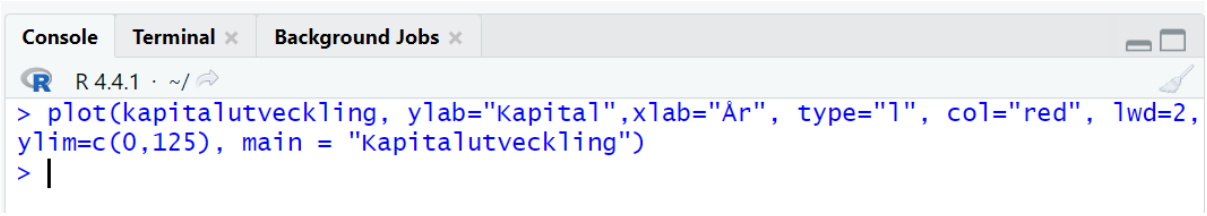
```
Console  Terminal  Background Jobs
R 4.4.1 · ~/
> plot(kapitalutveckling, ylab="Kapital",xlab="År", type="l", col="red", lwd=2)
>
```

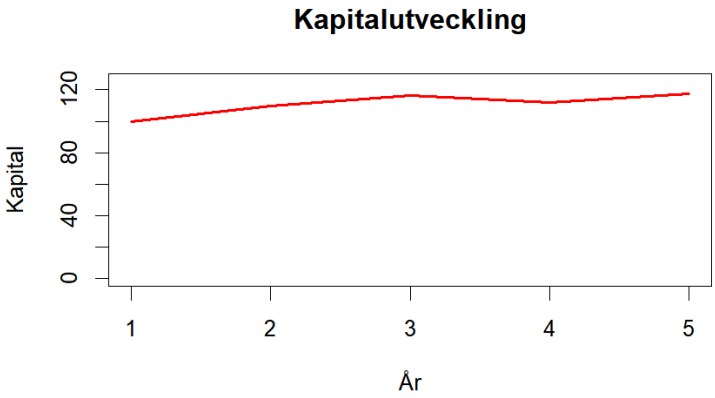


Uppgift 3.4

Figuren ovan kan ge sken av att kapitalet växt jättemycket. Y-axeln börjar på 100.

Ändra y-axeln till att gå mellan 0 och 125 (argumentet **ylim**) och sätt rubriken ”Kapitalutveckling” (argumentet **main**).





Nu har vi gjort några enkla grafer baserade på en vektor med fem observationer och vi kommer återkomma till andra grafer och figurer, i denna och framtida laborationer.

Uppgift 3.5* (vid intresse)

Skapa vektorn kapitalbildning igen, men nu med sex eller fler värden, och gör sen någon av plottarna ovan igen, i en annan färg och med en annan linjetjocklek.

Objekt

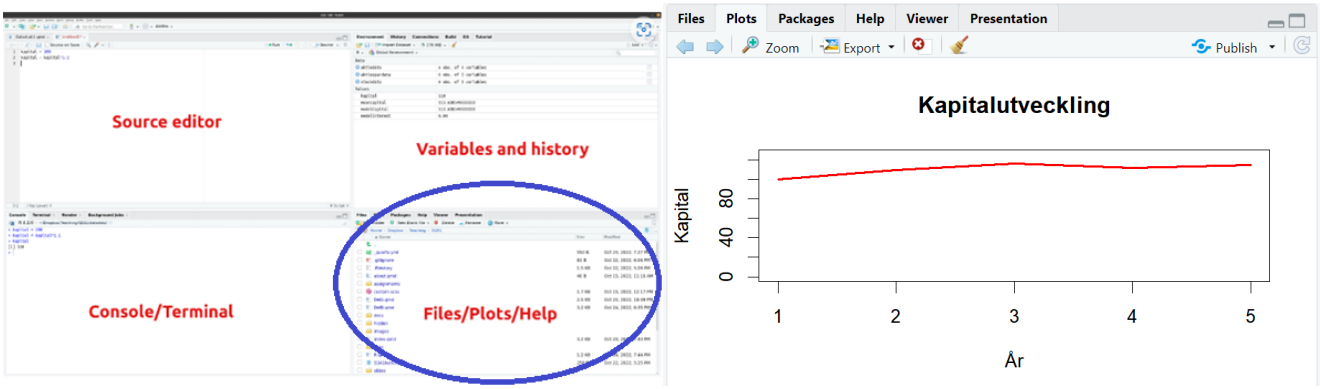
*Variabler, vektorer, data frames (som vi kommer till) kallas också för **objekt**. Det finns många olika typer av objekt och vi kommer att se flera olika typer längre fram i laborationerna.*

Funktioner och argument

*På samma sätt som i matematik, där f(x) betyder att vi har en funktion f av ett argument x (exv. f(x) = x^2 – funktionen kvadrerar x) är olika R-kommandon **funktioner** som utför något baserat på olika input – olika **argument**. Exempelvis kan kommandon som läser in filer från datorns hårddisk (se kapitel 9 nedan) benämnas funktioner och bland funktionens argument finns just filen som ska läsas in.*

4. Bläddra bland plottar och kopiera (exportera) en av de gjorda graferna

I kursen kommer ni göra en inlämningsuppgift som bland annat ska innehålla figurer och beskrivning av dessa. I nedre högra delen i Rstudio finns fliken ”Plots” (där dina grafer redan ska ha dykt upp).



Klicka på pilarna under ”Plots” och bläddra mellan de grafer du precis gjort.

Klicka på ”Export” och sedan, när grafen dyker upp i ett eget fönster, klicka på ”Copy Plot”.

Testa sen att klistra in bilden i det ordbehandlingsprogram du brukar använda (Microsoft Word, Google Docs, etc).

5. Organisera dig med filmappar

Det är viktigt att ha ordning på sina filer på datorn och kunna tala om för R var dina filer finns så R kan läsa dem. Vi rekommenderar att du skapar en mapp/folder/katalog för varje kurs du läser. Spara inte alla filer på datorns skrivbord eller i mappen **Downloads** eller liknande. Gå till filhanteringsprogrammet på datorn och skapa mappen **SOK**. Så här:

- Windows: Starta programmet **File Explorer** och navigera (klicka) till den mapp där du vill skapa din kursmapp. Skapa en ny mapp med namnet **SOK**.
- Mac: Starta programmet **Finder** och navigera (klicka) till den mapp där du vill skapa din kursmapp. Skapa en ny mapp med namnet **SOK**.
- Linux: Starta programmet **Nautilus** (om du använder Ubuntu, annars kan du prova att söka på ordet files om din Linux-distribution använder en annan filhanterare). Navigera (klicka) till den mapp där du vill skapa din kursmapp. Skapa en ny mapp med namnet **SOK**.

Undvik svenska bokstäver mm., ha logiska namn: *Använd inte åäö i namn på mappar eller filer. Det är stor risk att du får problem. Använd inte heller svenska bokstäver i variabelnamn, starta inte variabelnamn med siffror och använd inte mellanrum i variabelnamn. R skiljer också på stora och små bokstäver. Det finns också reserverade ord, använd inte dessa till variabelnamn. Använd logiska (och korta) variabelnamn som beskriver det du arbetar med.*

Uppgift 5.1

Skapa mappen **SOK** på din dator.

Att tänka på om du jobbar på en universitetsdator i datorsalen: Dina filer kommer försvinna

Om du skapar en mapp på en dator i datorsalen kommer den tyvärr inte att finnas kvar nästa gång du loggar in på annan dator, eller kanske inte ens på samma dator. Vi har föreslagit till IT-avdelningen på SU att ordna studentkonton på datorerna med individuella lagringsutrymmen som automatiskt dyker upp som en mapp när ni loggar in i

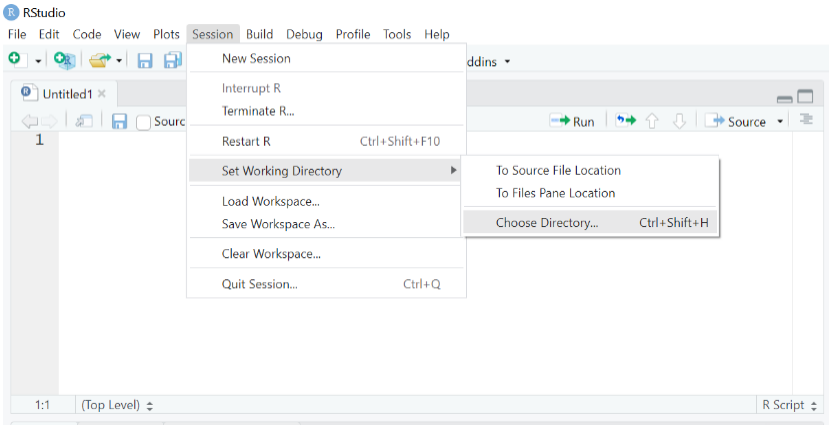
datorsalarna. Men i dagsläget finns tyvärr inte detta. Man får själv spara ner sina filer på nätet (t ex med en tjänst som Dropbox eller OneDrive). Som student vid SU har man tillgång till 1TB lagring i OneDrive, se denna sida för mer info. Ett annat sätt är att skapa SOK-mappen på ett USB-minne som man alltid bär med sig.

6. Sätt arbetskatalog (working directory) inne i R

De filer du har i din SOK-mapp vill du kunna komma åt inifrån R och du kommer också vilja spara arbete som du gör i R i samma mapp. Det enklaste sättet är att inne i R välja din **arbetskatalog** (**working directory**) till just din SOK-mapp. (**i praktiken är mapp och katalog samma sak**).

Du talar alltså om för R var arbete ska sparas och var R ska leta efter filer, exv. data som vi vill importera till R.

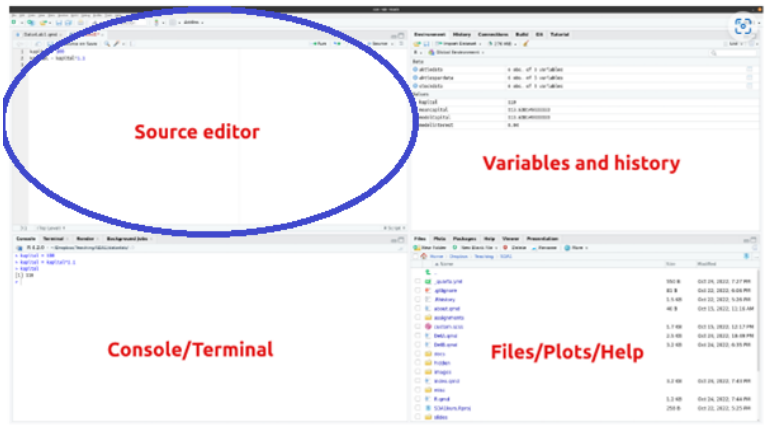
Ändra **working directory** i R till **SOK**-mappen genom att längst upp i R välja menyn **Session** och sen **Set Working Directory** och sen **Choose Directory…** och klicka dig sedan fram till mappen **SOK**.



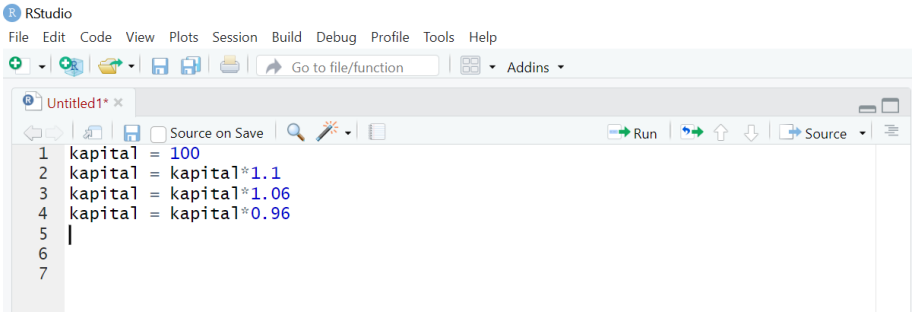
7. Använda Editorn (Source) för att spara kod

Att skriva kommandon direkt i Console har en nackdel: R minns inte kommandona vi har skrivit i en tidigare session (innan vi stängde ner RStudio). Varje gång vi startar upp RStudio måste vi skriva om våra kommandon om vi vill fortsätta våra beräkningar där vi slutade senast. (Det är inte riktigt sant, fliken **History** i övre högra delen av RStudio minns ju faktiskt gamla kommandon, men det är opraktiskt att förlita sig på **History**).

Vi skulle vilja skriva alla våra kommandon i en textfil som vi kan spara på datorns hårddisk (i **SOK-mappen**) och sen bara köra om alla kommandon i en senare session. **Source editorn** i övre vänstra delen av RStudio används för detta.



Om vi klickar på menyn **File** och sen under menyn **New File** och slutligen på **R Script** öppnas en tom textfil i **Editorn** som heter Untitled1 eller liknande. Här kan vi skriva in kommandon som vi vill spara för framtida sessioner. Vi kan exempelvis skriva in våra beräkningar av kapital:

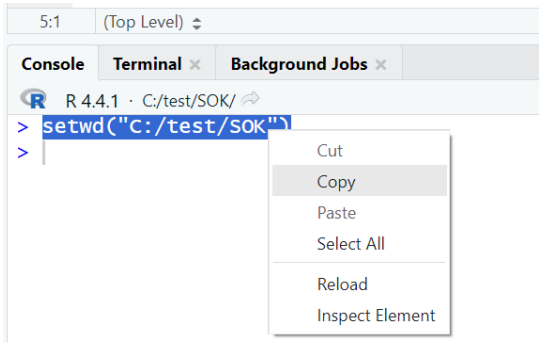


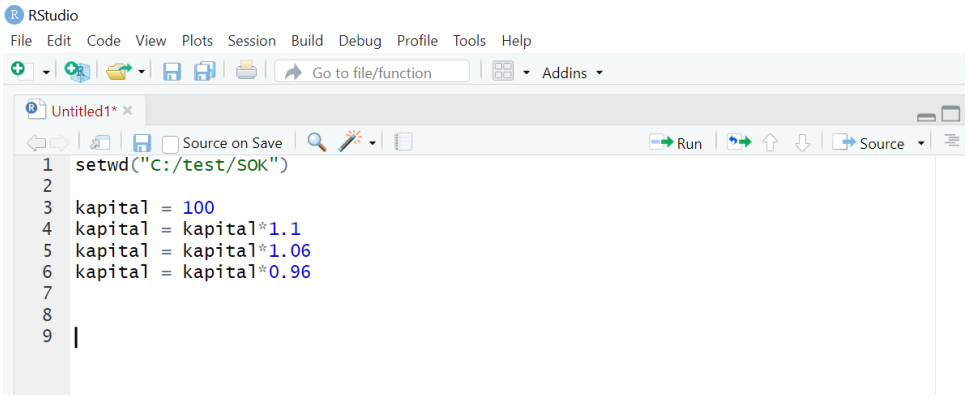
Innan vi går vidare:

- Lägg till ett par tomma rader längst upp i textfilen du håller på att skapa.

- När du ovan (i kapitel 6) bestämde arbetskatalog skrevs ett kommando ut i Console (**setwd(…..)**). Vad som står i parentesen i kommandot beror på vilken mapp just du valde. Markera kommandot (exempelvis i Console, se bild nedan), kopiera, och klistra in kommandot **längst upp** i textfilen.

- Nu har vi ett kommando längst upp i textfilen som kommer se till att vi hamnar i rätt katalog när textfilen används (vi skulle kunna ha ändrat, frivilligt eller ofrivilligt, arbetskatalog av något skäl). Om du undrar vilken arbetskatalog du jobbar mot inne i R kan du köra kommandot **getwd()**.





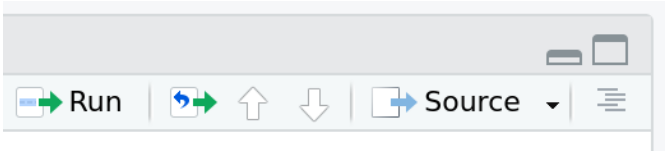
*I appendix finns lite mer om hur kommandot **setwd()** används för Windows, Mac och Linux,*

Vi kan spara textfilen genom att klicka på menyn **File** och sen på **Save** och sen navigera till mappen **SOK** genom att klicka i rutan som kommer upp. Döp filen till **stock** eller något annat namn som talar om vad filen innehåller (aktie på engelska). Klicka på Save/Spara. Filen kommer automatiskt att få filändelsen .R, dvs filen kommer alltså heta **stock.R** och RStudio vet då att det är en fil med R-kommandon.

SPARA ALLTID det arbete du gör – det du vill spara - i en fil som denna!

Ett samling kommandon kallas också för **kod** och vi säger att vi arbetar med en **kodfil** i **editorn**. Vi kan också säga att vi har skrivit ett **program**.

Vi kan köra alla kommandon i filen stock.R genom att klicka på Source - knappen uppe i högra hörnet av editorn:



När vi säger att vi ”**kör kommandon**” menar vi att datorn utför (**exekverar**) våra kommandon, rad för rad. Man kan se det som att Source-knappen skickar kommandona till Console så vi slipper skriva in dem där.

Om vi kör filen ovan (tryck på Source) och sen skriver kapital i Console ska vi få svaret 111.936.

När man arbetar med kod vill man ofta köra ett kommando i taget. Det kan man göra genom att placera markören (det blinkande strecket) på den rad man vill köra (**exekvera**) och sen klicka på **Run**-knappen i editorn (se bilden ovan). Du behöver inte markera koden på raden, det räcker att bara placera markören någonstans på raden. Kortkommandot Ctrl + Enter gör samma sak (Command + Enter på Mac).

Ofta ställer man markören på den första raden i koden och klickar på Run-knappen om och om igen för att köra varje rad, en efter en. Om man har kört ett kommando i filen och vill se om kapitalvariabeln verkligen har ändrats kan man gå till Console, skriva kapital och trycka Enter. Fliken **Environment** i övre högra delen av RStudio ska också visa den uppdaterade variabeln.

Man kan också köra flera rader kod på en gång genom att markera raderna och trycka på Run-knappen.

Uppgift 7.1

Skapa filen stock.R med instruktionerna ovan och kör sedan hela filen genom att använda Source-knappen. Undersök vilket värde variabeln kapital har efter att filen körts genom att (i Console) skriva kapital, följt av Enter.

Uppgift 7.2

Inget händer i Console när du klickar på Source, koden körs ”i bakgrunden”. Om vi exempelvis vill se slutvärdet på kapital kan vi lägga till en rad i slutet av koden med kommandot **print** och variabeln du är intresserad av, dvs. print(kapital). Klicka på Source - knappen igen och verifiera att värdet 111.936 skrivs ut i Console.

Uppgift 7.3

Kör kommandot på rad 1 i stock.R genom att använda Run-knappen. Undersök vilket värde variabeln kapital har, i Console. Upprepa detta för de resterande raderna i stock.R.

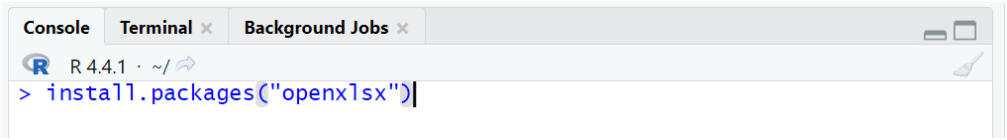
Uppgift 7.4 – arbetskatalogen

Kör kommandot **getwd()** i Console och verifiera att R skriver ut samma katalog (samma sökväg) som du bestämt genom det första kommandot, setwd(), i textfilen.

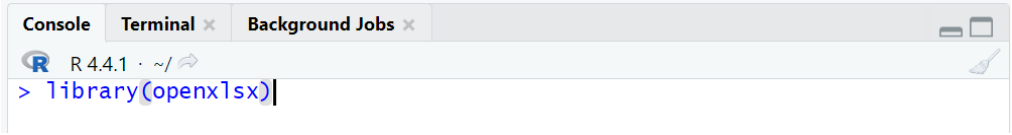
8. Installera och ladda paket

R har en ”grundinstallation” som vi kan kalla **base R**. Ofta vill vi använda annan funktionalitet än vad som finns i base R. För att göra detta installerar och laddar vi s.k. **paket**. Det finns ett stort antal paket. Vi ska nu installera och ladda paketet **openxlsx**, som innehåller kommandon för att läsa in Excelfiler.

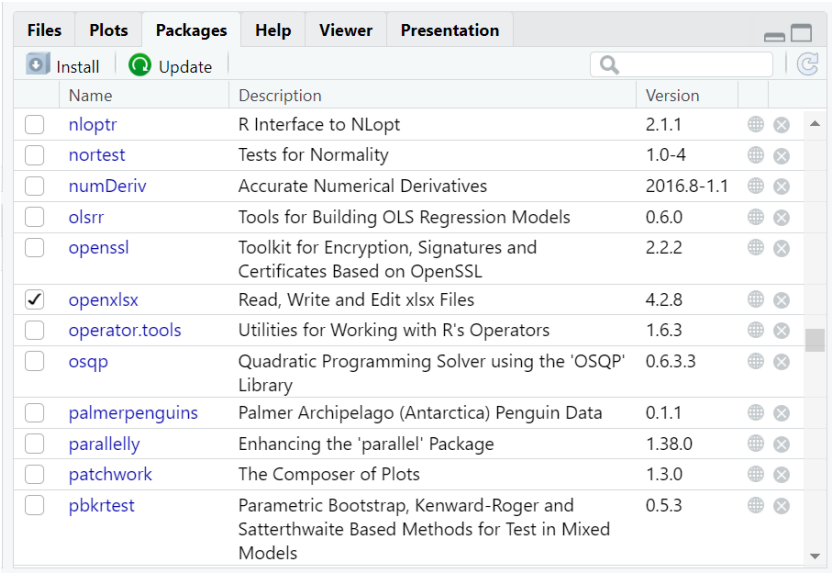
Kör följande två kommandon i R, först



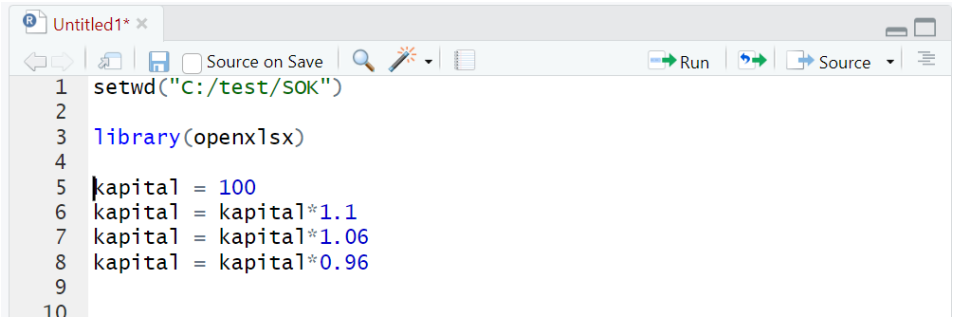
Notera att paketets namn står inom citattecken. När kommandot körs kommer typiskt viss utskrift i rött på skärmen (i Console). Så länge kommandot har kört som det ska (det kan exv. stå ”the downloaded packages are in...”, eller liknande, på skärmen) kan du ignorera utskrifterna. Kör sen



Notera att paketets namn **INTE** står inom citattecken. När kommandot körs kommer typiskt viss utskrift i rött på skärmen. Så länge kommandot har kört som det ska kan du ignorera dessa utskrifter. Du kan gå till fliken Packages i nedre högra hörnet av Rstudio, scrolla ner till bokstaven o och se om paketet finns med i listan och om rutan är iklickad.



Varje gång du startar RStudio behöver du ladda (men inte installera) de paket du ska använda. Ett sätt att säkerställa att man får med de paket man vill använda är att **i början av sin textfil** ha library-kommandon för de paket man vill använda:



Uppgift 8.1

Installera och ladda paketet **openxlsx**.

Uppgift 8.2

Installera och ladda paketet **mosaic**, som vi ska använda i laboration 2. Följ ovanstående procedur, men byt ut paketnamnet.

På laboration 2 kommer du behöva ladda (men inte installera) paketet mosaic igen.

I R kan du använda antingen apostrof eller citattecken, båda kommer att ge samma resultat. Om du exempelvis ska installera mosaic-paketet kommer install.packages(”mosaic”) och install.packages('mosaic') göra samma sak.

---

9. Läsa in data från en fil

Hittills har vi skrivit in alla kommandon och tal (t ex kapital = 100), dvs vi har matat in **data** själva. Ofta läser man istället in datamaterial från en fil eller från andra källor.

Här är en Excel-fil med data från sex år:

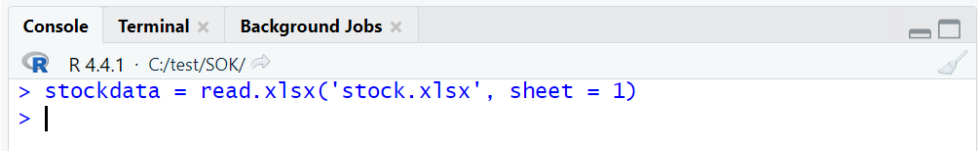
	A	B	C
1	year	capital	return
2	2018	100.00	0%
3	2019	110.00	10%
4	2020	116.60	6%
5	2021	111.94	-4%
6	2022	117.53	5%
7	2023	125.76	7%

Excel-filen är döpt till *stock.xlsx*, men den skulle kunna ha ett annat namn.

Vi vill nu läsa in data från stock.xlsx. Du kan ladda ner filen här. Beroende på din dator kommer en av två saker hända:

1. filen stock.xlsx hamnar automatiskt i mappen **Downloads** (eller liknande) på din dator. Du får då kopiera eller flytta filen till din arbetskatalog (din SOK-mapp) genom att använda datorns filhanterare.
2. du får välja var du vill spara ner filen. Klicka dig fram till din arbetskatalog (din SOK-mapp) och spara filen där.

Läs nu in data från Excel-filen genom kommandot **read.xlsx**. Här är hela kommandot (**funktionsanropet**):



Det är några saker att reda ut här. Först, read.xlsx() är en **funktion** vilket betyder att det är ett kommando som **gör något** baserat på funktionens **input-argument**:

- Det första **argumentet** 'stock.xlsx' är en **textsträng** (den har apostrofer) och talar om för read.xlsx() *vilken* Excelfil som ska läsas in. Vi behöver inte säga mer eftersom filen ligger i vår arbetskatalog, som ju R känner till.

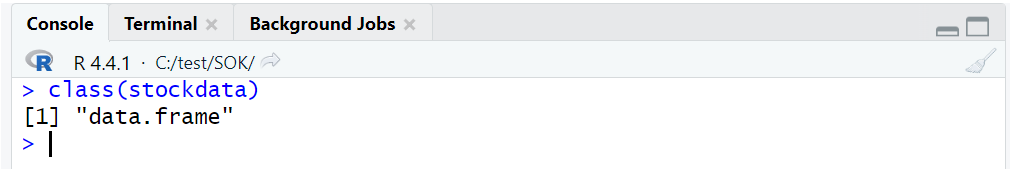
- det andra input-argumentet är sheet = 1, vilket säger åt read.xlsx() att data ligger i det första kalkylbladet i filen stock.xlsx. En Excelfil kan ju ha flera kalkylblad, och vill man läsa det andra bladet ändrar man 1 till 2. (men just stock.xlsx har bara ett blad så prova inte detta, R kommer att klaga).

Men vad betyder **stockdata =** i början av kommandot? Jo, om R ska läsa in data till arbetsminnet måste den ge dessa data ett namn, vilket här är valt till stockdata. Du kan använda ett annat namn om du vill.

stockdata är en typ av **variabel**, på samma sätt som vi tidigare hade både en ”vanlig variabel” (kapital, ett tal) och en vektor (kapitalutveckling).

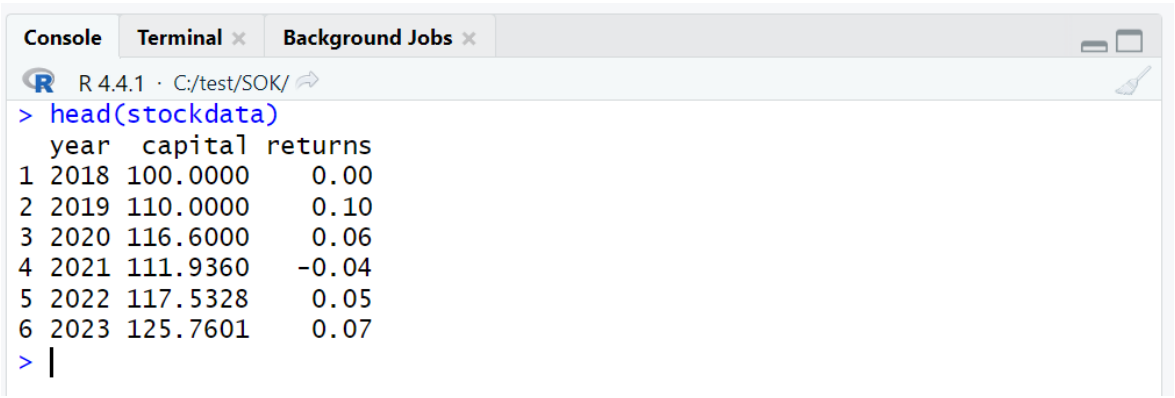
Variabeln stockdata är mer komplex. Den innehåller en hel tabell med värden.

R har olika varianter av tabellvariabler. Den viktigaste för oss är en så kallad **data frame**. Du kan se att tabellvariabeln stockdata faktiskt är en data frame genom att använda kommandot **class()**, skriv class(stockdata) i Console:



Du ska också kunna se stockdata uppe i högra delen av Rstudio, under fliken Environment.

När man har läst in ett datamaterial är det bra att ta en snabbtitt på den inlästa tabellen för att se att den blev korrekt inläst. Precis som vi gjorde med variabeln kapital för att se dess värde (t ex 100) kan vi skriva stockdata i Console för att se dess värde (som ju är en tabell). Om man har en tabell med många rader blir det fort jobbigt att skriva ut alla rader. Kommandot **head()** är då praktiskt, som bara skriver ut det första sex raderna av tabellen.



Data på formatet ovan kallas **tidy** vilket kan översättas med **välstrukturerade**. **Varje variabel finns i en kolumn och varje observation finns på en rad.**

Uppgift 9.1

Läs in stockdata enligt ovan.

Uppgift 9.2* (vid intresse)

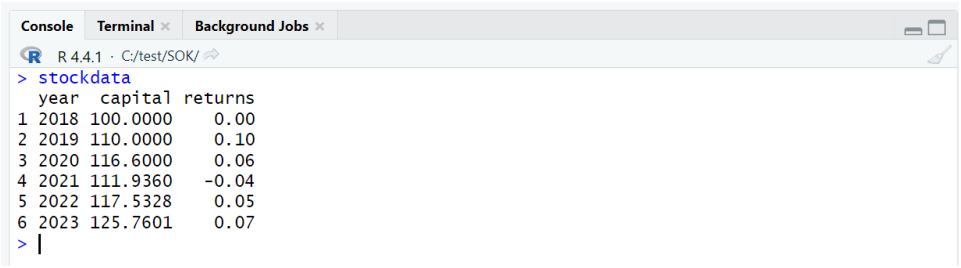
Läs mer om hur man läser in data här:
https://statisticssu.github.io/SDA1/tutorials/ReadingDataFromFile/ReadingDataFromFile.html

10 Mer om datatypen data frame och om indexering etc.

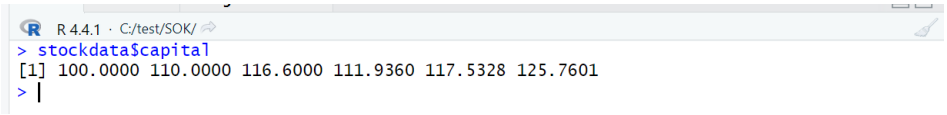
Data frame är den vanligaste datatypen vi jobbar med när vi läser in data som vi är intresserade av (man säger ofta att man läser in ett dataset). En data frame är en tabell med information om vad kolumnerna heter (och ibland även

vad raderna heter). Hur gör du för att exempelvis välja ut en viss rad eller en kolumn, eller några värden ur en data frame? Här kommer några mycket användbara kommandon som du kan använda, för små såväl som stora data frames, med tusentals eller miljontals observationer.

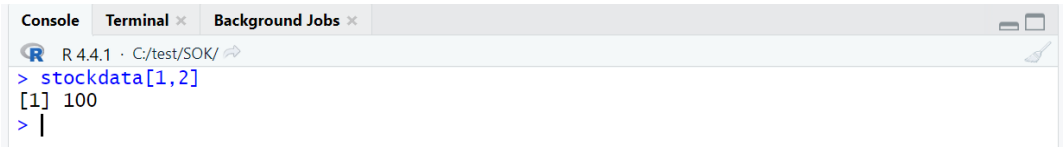
Titta först på stockdata. Normalt skriver vi inte ut alla data i Console, men eftersom vi har få rader skriver vi ut allt:



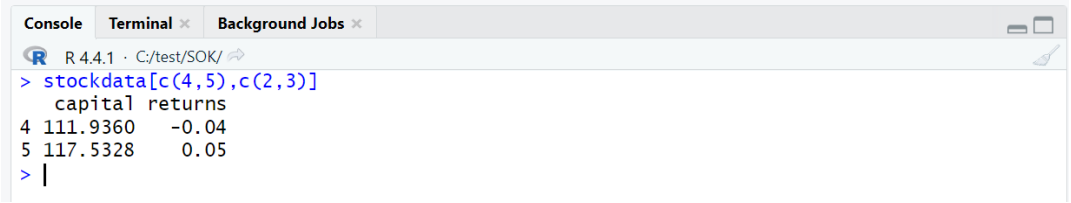
Använd dollartecken och sen variabelnamn för att komma åt en enskild variabel (kolumn) i en data frame:



Om du istället använder hakparenteser och sen rad- och kolumnnummer (separerade med komma) kan du plocka ut enskilda element ur vår data frame, exempelvis första raden, andra kolumnen:

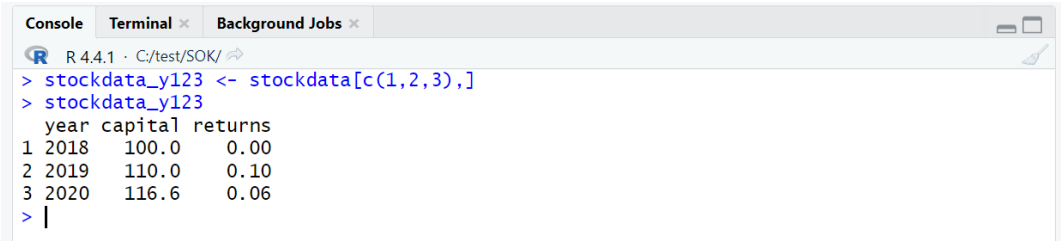


Talet före kommatecknet, inne i hakparentesen, är alltså radnummer, och efter kommatecknet har vi kolumnnummer. Om du vill ta fram exempelvis elementen på rad 4-5 och i kolumnerna 2-3, ska du använda dig av vektorn c(4,5) för radnumren och vektorn c(2,3) för kolumnnumren, som följer:



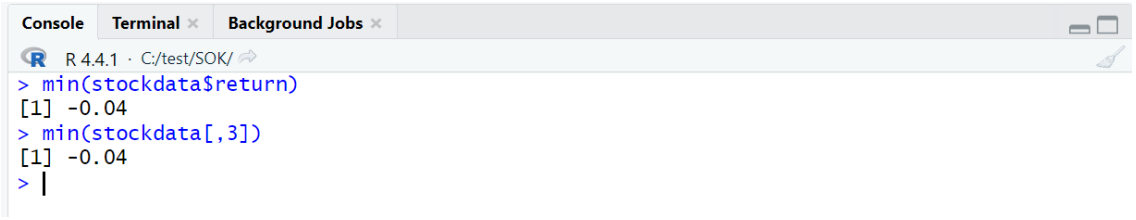
Dubbelkolla att utskriften är korrekt (med vår första utskrift, av alla data, ovan).

Du kan också lägga in resultaten i en egen ny data frame, säg att vi bara år intresserade av de första tre åren av stockdata:



I detta fall ville vi ha med alla kolumner, vi specificerade då inte kolumnnummer (det är tomt efter kommatecknet), vilket R tolkar som att vi vill ha alla kolumner.

Om du vill ha minimumvärdet av en variabel, kan du (exempelvis) använda **min()** och göra något av följande:

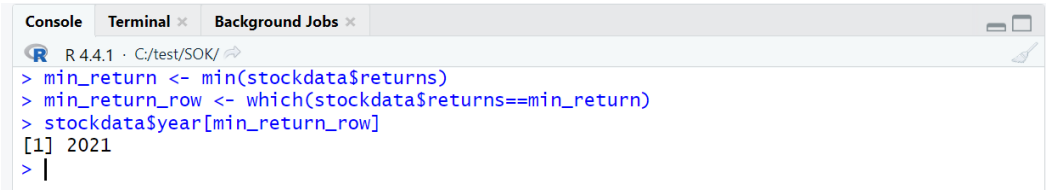


Det första kommandot letar upp det minsta värdet i variabeln returns, det andra kommandot letar upp det minsta värdet i kolumn 3 (dvs i variabeln returns). Kommandona ger samma resultat.

Om du vill veta vilken rad som har lägst värde på returns kan du exempelvis först spara minsta värdet från ovan och sen använda kommandot **which()**, som följer

.

Det minsta värdet sparas först i variabeln min_return. Sen frågar vi, med which(), vilken rad i variabeln returns som innehåller minvärdet, vilket är rad 4. Vi gjorde också en **logisk jämförelse**, och använde därför två likamedtecken i rad, **==**. Vi kan sen ta ett steg till och spara detta värde och sen ta fram vilket år som hade denna avkastning:



Vi gjorde här samma sak som ovan, men sparade också radnumret (4) i variabeln min_return_row. Sista raden plockar sedan fram det år som finns på rad 4 i stockdata (vi tar fram det fjärde värdet i kolumnen year i vår data frame).

Uppgift 10.1

Gå igenom kommandona ovan.

Uppgift 10.2

Skriv ett uttryck med R-kod som räknar ut skillnaden mellan högsta värdet på returns och lägsta värdet på returns.

Uppgift 10.3

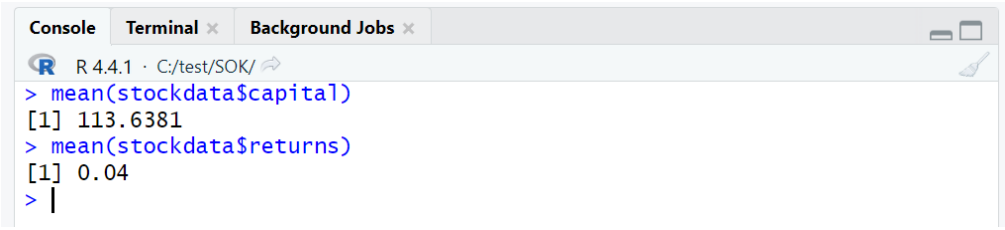
Skriv en rad R-kod som tar fram vilket år som har högst kapital (eller flera rader kod, dvs gör uppgiften i steg).

Uppgift 10.4

Testa att ändra valfritt värde i stockdata, exempelvis genom att skriva stockdata[radnr, kolumnnr] = nytt värde, där ”radnr”, ”kolumnr” och ”nytt värde” byts ut mot lämpliga värden. Spara ändringen i ett nytt objekt, exv. stockdata_uppdaterad. Dubbelkolla att ändringen genomfördes.

Uppgift 10.5

På laboration 2 ska vi bland annat skapa tabeller, figurer och räkna ut statistiska mått, såsom medelvärde och standardavvikelse. Testa här att köra följande kommandon för att räkna ut medelvärde (funktioen **mean()**) av de två variablerna capital och returns.



Kapitel 11. Plotta (igen)

I kapitel 3 ovan gjorde vi en graf med en variabel, och R valde själv x-axel. Nu använde vi istället vår data frame stockdata och två variabler.

Uppgift 11.1

Testa följande två kommandon. Blir det någon skillnad?

plot(stockdata$capital~stockdata$year)

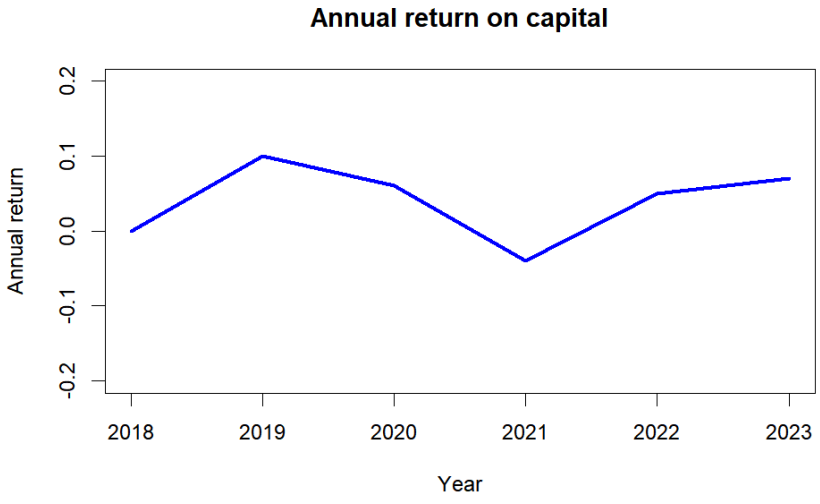
plot(capital~year, data=stockdata)

Båda kommandon är av typen ”plotta y mot x”, där y kommer före **tildetecknet** (**~**) och x efter.

På näst sista sidan i detta dokument finns en lista med olika kortkommandon för hur du får fram tilde och andra tecken och kommandon på din dator.

Uppgift 11.2

Med din data frame stockdata, gör en graf som denna. Du behöver alltså använda någon av de två kommandona från 11.1 (den variant du föredrar) och lägga till argument i plot-funktionen. Vid behov gå också tillbaka till kapitel 3.



Till sist, för laboration 1

Ta det lugnt, använd R-hjälpen och testa dig fram

Det är många nya saker som presenteras, var beredd på att allt inte kommer att sjunka in direkt. Försök att ta dig igenom hela texten, och övningarna. Du kan sen återkomma till denna labb på andra datorlaborationer för att repetera de delar du behöver.

---

*Under kursens gång, titta också på de olika videos som finns under **R-hjälpen**, i Athena. R-hjälpen är en uppsättning videos som spelats in a statistiska institutionen. Det finns också bland annat ”Cheat sheets”, med vanliga kommandon.*

När du gör labbarna i kursen, testa gärna egna saker, exempelvis genom att modifiera någon variabel eller liknande. Sätt exempelvis något av kapitalvärdena i vektorn kapitalutveckling ovan till ett negativt värde, eller lägg till fler år i vektorn, och se hur grafen över kapitalutveckling ändras.

---

*Det finns en inbyggd hjälpfunktion i R. För att få hjälp med exempelvis kommandot **openxlsx** kan du skriva **help(openxlsx)** eller **?openxlsx** i Console. Just för detta kommando, som tillhör ett paket vi installerat, behöver paketet vara installerat och laddat för att hjälp ska dyka upp. Ibland är hjälpen inte jättelätt att använda, men du kan börja använda hjälpfunktionen gradvis under kursen. Genom att exempelvis googla kan man också få fram mycket information.*

Appendix (* - vid intresse)

Om att sätta arbetskatalog manuellt

För att slippa att klicka sig fram via menyer hela tiden är det praktiskt att ändra working directory i början av den kodfil man jobbar med. Då kan man bara köra den kodfilen och working directory ställs in automatiskt. Kommandot setwd() gör samma sak som vi gjorde i kapitel 6 via menyerna. Här måste vi dock veta **sökvägen** (**path**) till mappen, dvs datorns sätt att hitta till mappen **SOK**. Sättet att skriva sökvägar på skiljer sig åt mellan Windows/Mac/Linux. I dessa exempel antar jag att din SOK-mapp är placerad i mappen Documents.

- På en Mac skriver vi kommandot: setwd('/Users/username/Documents/SOK'), där username ska ersättas med ditt användarnamn på din Mac (namnet som kommer upp när du loggar in på datorn). Notera apostroferna kring filvägen i setwd-kommandot.
- På en Windows-dator skriver vi kommandot setwd('C:/Documents/SOK'). Notera apostroferna kring filvägen i setwd-kommandot. [Windows-sökvägar skrivs vanligtvis med backslash \. I RStudio skriver man ändå / för att matcha med Mac och Linux].
- På en Linux-dator skriver vi kommandot setwd('/home/username/Documents/SOK'), där username ska ersättas med ditt användarnamn som du använder när du loggar in.

Kortkommandon etc.

Description of Shortcuts and Special Characters in R

Description	Windows/Linux	Mac
Run code	Ctrl + Enter	⌘ + Enter
Run code and stay on current line	Alt + Enter	Option + Enter
Select All	Ctrl + A	⌘ + A
Copy	Ctrl + C	⌘ + C
Paste	Ctrl + V	⌘ + V
Cut selected text	Ctrl + X	⌘ + X
Delete line	Ctrl + D	⌘ + D
Undo	Ctrl + Z	⌘ + Z
Redo	Ctrl + Shift + Z	⌘ + Shift + Z
Select multiple lines/area	Shift + arrow key	Shift + arrow key
Duplicate line/area	Ctrl + Shift + D	⌘ + Shift + D
Scroll up/down	Ctrl + up/down arrow key	⌘ + up/down arrow key
Go to the top	Ctrl + Home	⌘ + Home
Go to the bottom	Ctrl + End	⌘ + End
Go to line	Shift + Alt + G	⌘ + Shift + Option + G
Save	Ctrl + S	⌘ + S
Save all documents	Ctrl + Alt + S	⌘ + Option + S
Open document	Ctrl + O	⌘ + O
$ symbol	Ctrl + Alt + 4	Option + 4
Vertical bar/pipe (logical OR): |	Ctrl + Alt + < or Alt gr + <	Option + 7
Assignment (<-)	Alt + -	Option + -
Zoom out/in	Ctrl + -/+	⌘ + -/+
Interrupt running code	Esc	Esc
Clear Console	Ctrl + L	⌘ + Option + L
Left square bracket [	Ctrl + Alt + 8 or Alt gr + 8	Option + [or Option + 8
Right square bracket]	Ctrl + Alt + 9 or Alt gr + 9	Option +] or Option + 9
Left curly brace {	Ctrl + Alt + 7 or Alt gr + 7	Option + Shift + 8
Right curly brace }	Ctrl + Alt + 0 or Alt gr + 0	Option + Shift + 9
Slash /	Shift + 7	Shift + 7
Backslash \	Alt gr + + or Ctrl + Alt + +	Option + Shift + 7
Tilde symbol ~	Alt gr + ~(key near Å)	Option + ~(key near Å)
Search expression	Ctrl + F	⌘ + F
Arrange indentation	Ctrl + I	⌘ + I
Show list of common shortcuts	Shift + Alt + K	Option + Shift + K

Denna version av dokumentet: 2026-03-27

Materialet i Statistisk översiktskurs har tagits fram av Ulf Högnäs och Anders Fredriksson, med inspiration och ibland direkt användande av material från andra kurser och personer, bland annat kurserna Statistik och dataanalys 1-3, med material av Michael Carlson, Ellinor Fackle Fornius, Jessica Franzén, Oskar Gustafsson, Oscar Oelrich, Mona Sfaxi, Karl Sigfrid, Mattias Villani, Valentin Zulj, med flera.