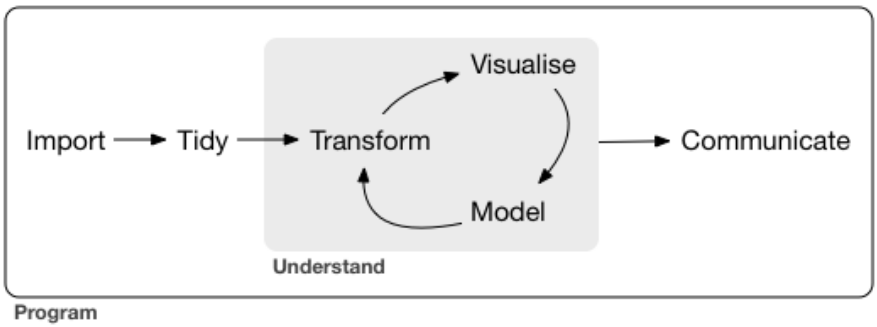
Kurskompendium - introduktion till R

Contents

1 Introduktion

Denna introduktionskurs syftar till att få den ovana R-användaren att bli hyggligt bekväm med att hantera de viktigaste verktygen i R och i det modernare gränssnittet *Rstudio*.

Kursen bygger på Hadley Wickham´s bok *R for Data Science* som också finns tillgänglig på internet (https://r4ds.had.co.nz/index.html). I själva verket följer innehållet i kursen ganska slaviskt Wickham´s bok. Motivet är att HW har varit en drivkraft bakom att utveckla det från början tämligen svåröverskådliga och inkonsistenta programspråket till ett mer konsistent och därmed sänkt inlärningströskeln. Det har han gjort genom att utveckla och modifiera programmeringsspråket samt utvecklat en rad packages för att hantera data på ett effektivare och möjligen modernare sätt än tidigare.



Man kan beskriva datahantering och -analys som en process som omfattar

1. Importera data
2. Rensa data
3. Transformera data till ändamålsenliga arbets-data - reproducerbarhet!
4. Visualisera, utforska data för överblick och förståelse

5. Modellera data
6. Och något om hjälpmedel för att kommunicera resultaten

Del 1-3 handlar om att importera och manipulera data så att det blir möjligt att analysera dem. Del 3-5 syftar till att få en överblick och förstå datamängden och hur olika komponenter hänger samman med varandra. Del 6 handlar om att kommunicera resultaten från analysen.

Vi kommer att beröra samtliga delar mer och mindre. I ett första avsnitt dyker vi in i del 3-5 för att snabbt komma in i programmeringsspråket och känna på olika verktyg för visualisering, modellering och transformering av data. Det andra avsnittet behandlar del 1-3 och brottas då med verktyg för att anpassa rådata till arbets-data och i det tredje avsnittet behandlas verktyg för att kommunicera resultaten.

Men först lite basal R/RStudio-hantering.

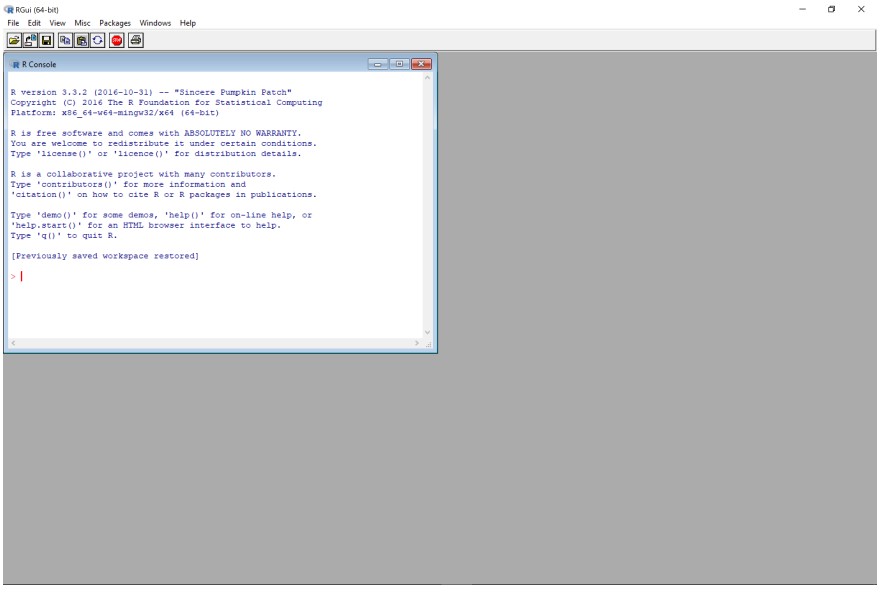
1.1 Basics

I detta avsnitt ska vi väldigt översiktligt nosa på R-konsolen som självständigt program och använda det som en kalkylator. Därefter ska vi bekanta oss med RStudio och börja uppskatta finessen med att ha ett separat gränsnitt. Vi ska även titta på några olika sätt att få hjälp när du kör fast – för det kommer du att göra.

1.2 R-konsolen

R fungerar som ett självständigt program och man kan i princip göra alla beräkningar och bearbetningar direkt i programmets eget gränssnitt – konsolen. I det här avsnittet ska vi bekanta oss översiktligt med konsolen.

Starta upp R. nedanstående skärmbild kommer upp. I konsolen ses en standardtext som visar vilken version av R som körs, en deklaration om R samt några tips om hjälp-funktioner.



Menyraden innehåller sedvanliga funktioner och länkar. Vi ska inte gå igenom dessa utan bara känna på hur det är att arbeta i detta gränssnitt. Låt oss börja med att titta på några inbyggda R-demon. Skriv

```
demo()
```

vid prompten (>) och tryck ENTER. Då kommer en ”output”-skärm fram vilken visar vilka tillgängliga demon som finns. Låt oss titta på några exempel på R:s grafiska möjligheter. Det gör man genom att

ange demo(graphics). Det kan man förstås skriva ut men ett smartare sätt är att unyttja att R håller reda på tidigare kommandon och genom att istället trycka ”uppåt"-pilen på tangentbordet kommer det senaste kommandot att synas på skärmen. Med hjälp av vänsterpilen flyttar du markören innanför parentesen och skriver *graphics* så att kommandot nu är

```
demo(graphics)
```

Tryck ENTER. I konsolen ses nu en bekräftelse på att R har laddat grafik-demot. Tryck ENTER en gång till för att köra igång demot. Då förbereds R genom att ladda in nödvändiga packeages och data-set. Längst ner står ”Waiting to confirm page change...”. R är redo att visa olika grafiska outputs. Högst upp på output-skärmen uppmanas vi att trycka ENTER för att bekräfta fortsättningen. Då vi gör det visas ett nytt diagram för varje gång ENTER trycks ned. I konsolen visas den syntax R använt för att framställa diagrammen. Efter ett antal ENTER händer inget mer. Vi har nått slutet på demon och det markeras på konsolen genom att prompten (>) nu blivit röd. R är nu redo för nya kommandon.

Låt oss använda R som en kalkylator. Man skriver då in de beräkningar mnan vill utföra och trycker ENTER. Till exempel:

```
2+3
4*7
9/4
log(4)
3^2
sqrt(6)
```

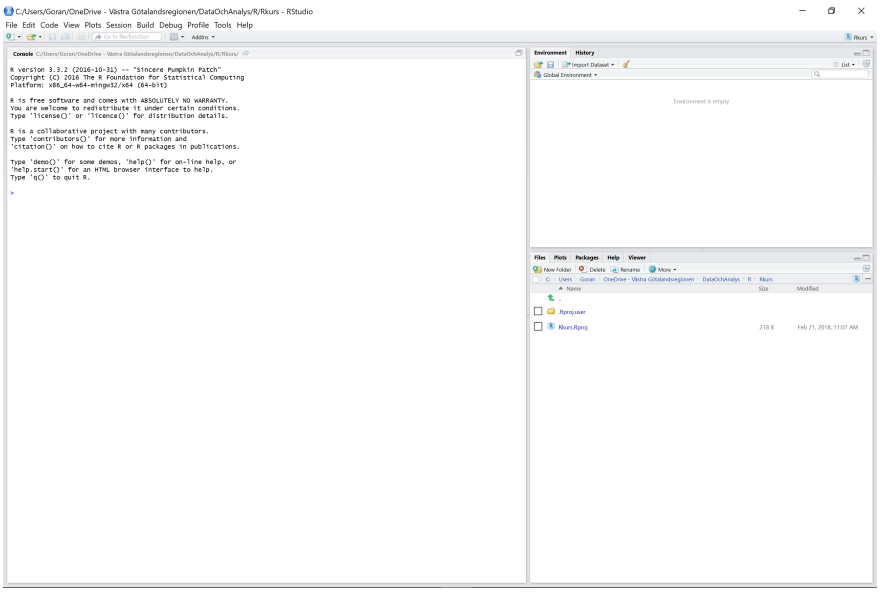
även om R-konsolen är fullt funktionell är användningen en mödosam process, speciellt då man använder mer komplexa script för sina sessioner. Det blir mycket tangent-tryckningar och ofta omständligt att felsöka scripten man jobbar med. Därför bör man använda något av de modernare gränssnitt som utvecklats. Det kanske mest använda är RStudio, vilket vi nu går över till. Skriv

```
q()
```

i konsolen, tryck ENTER och välj att inte spara. Tryck OK.

1.3 RStudio

RStudio är en ingegrerad utvecklingsmiljö (IDE, integrated development environment) för R programmering. När du öppnar programmet möts du av två dominerande ytor, konsolen och resultat-ytan nere till höger (*console* och *output*).



Konsolen är identisk med gränssnittet i det nativa R-gränssnittet som vi nosade på ovan.

Innan vi börjar laborera med RStudio behöver vi emellertid installera en modul, eller ett paket (package), som innehåller funktioner, data och dokumentation nödvändiga för att genomföra kursen. Sådana paket (packages) utgör en viktig del av den breda funktionaliteten i R och vi ska komma tillbaka till dessa lite senare. Men för nu nöjer vi oss med att installera en modul som heter `tidyverse` och är egentligen en samling av andra moduler för att förenkla och effektivisera datahanteringen.

Det finns ett par olika sätt att installera en modul som *tidyverse*. För tillfället ska vi installera *tidyverse* med en enkel rad med kod. I konsolen skriv

```
install.packages("tidyverse")
```

och tryck ENTER.

Tidyverse är en samling moduler vilka bygger på samma konsistenta programlogik och utnyttjar kapaciteten i R optimalt för att hantera data. För kursen täcker tidyverse behovet av moduler som inte finns med i basversionen av R. Nästan. Vi ska ladda ned ytterligare tre moduler som innehåller data till övningsexemplen längre fram. Så på samma sätt som tidigare, skriv

```
install.packages(c("nycflights13", "gapminder", "Lahman"))
```

Och tryck ENTER.

Dessa moduler innhåller data över flygtrafik, global utveckling och baseball-data.

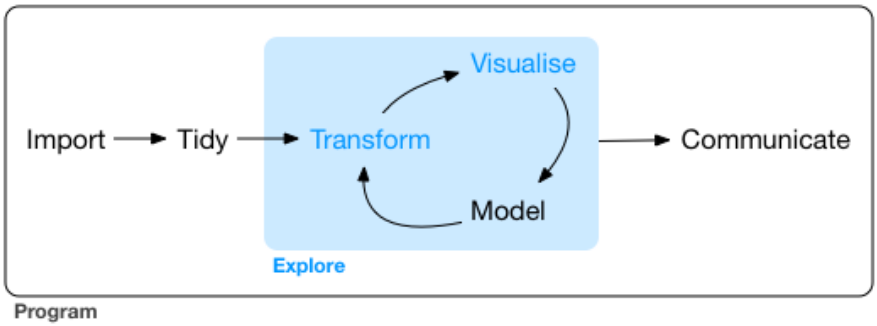
1.4 Hjälpfunktioner

R och RStudio innehåller flera inbyggda möjligheter att få hjälp när man kör fast, vilket inte sällan blir fallet. Förutom dessa måste nämnas möjligheten att googla fram en lösning på problemet. Det kan gälla alltifrån tämligen ospecifika frågor om t.ex. en viss funktion till att förstå vad ett felmeddelande betyder. Att kombinera sökfrasen med ”R” räcker i allmänhet för att resultatet ska vara tillräckligt R-specifikt. Google är förvånansvärt effektivt för att få klarhet i just felmeddelanden vilka ofta är ganska kryptiska.

Om Google inte ger svaret bör man försöka med Stackoverflow (http://stackoverflow.com/) och i sökfrasen inkludera [R] för att begränsa sökningen till R-specifika svar.

2 Utforska data

Målet med detta avsnitt är att du så snabbt som möjligt ska komma på banan med några grundläggande verktyg/funktioner för att utforska (explore) en datamängd, dvs. att få en översikt över vilka data som finns, pröva preliminära hypoteser, testa dem och allteftersom lära känna data ordentligt. Det handlar alltså om den mellersta fasen i nedanstående bild - fr.a att Visualisera och transformera data för överblick och förståelse.



För detta ändamål ska vi använda ett antal verktyg. Det är verktyg för att *visualisera* datastrukturer, att *transformera* data för att kunna undersöka associationer och (preliminärt) *formulera hypoteser* för vidare analys.

I avsnittet finns även ett par stycken som handlar om arbetsflöden i R - "good practice" för att skriva och organisera R-kod.

3 Visualisering

Placeholder

Placeholder

4.1 Skriva kod

4.2 Anropa funktioner

4.3 Övningar

5 Transformering av data

Placeholder

5.1 Introduktion

5.2 Förberedelser

5.3 `dplyr` grunder

5.3.1 Filtrera rader/poster med `filter()`

5.3.1.1 Jämförelser

5.3.1.2 logiska operatorer

5.4 Övningar

5.5 Arrangera poster med `arrange()`

5.6 Övningar

5.7 Välj kolumner med `select()`

5.8 Övningar

5.9 Lägg till flera variabler med hjälp av `mutate()`

5.9.1 Funktioner och operatorer att användas med `mutate()`

5.10 Övningar

5.11 Kombinera multipla operationer med the pipe

6 Arbetsflöde: scripts

Hittills har vi använt fr.a. konsolen för att skriva och köra kod. Detta kan fungera ygligt med relativt enkla script men mer komplexa script skriver man effekjtivare i Editorn, uppe till vänster. Öppna upp den genom att Klicka på File/New file/R script, eller Använd kortkommandot Ctrl+Shift+N. Då bör skärmen se ut ungefär så här:

Editorn är ett utmärkt ställe att placera kod som man vill behålla. Experimentera i konsolen och när du skrivit ett kodavsnitt, flytta upp det till konsolen. Rstudio sparar innehållet i Editorn då du avslutar Rstudio och öppnar det aautomatiskt nöösta gång du startar upp.

##Att köra kod Editorn är en utmärkt plats at bygga upp komplexa ggplot2 diagram eller längre sekvenser av dplyr-kod. Kom ihåg snabbkommandot Ctrl+Enter vilket exekverar den kod som amrkören finns på. Det innebär att ävebn en längre selkvens som sträcker sig över flera rader körs så länge sekvensen är sammanhängande, t.ex. med hjälp av "the pipe" %>% eller i en ggplot med "+". Istället för att köra uttryck för uttryck kan du även exekvera hela scriptet i ett steg - kortkommandot Ctrl+Shift+S. genom att använda det regelbundet förvissar du dig om att scriptet fungerar som tänkt. Det är en god vana att börja scriptet med att ladda in samtliga de moduler/packages du beöver. Det är en god hjälp för minnet då man återvänder till scriptet efter ett tag och om du delar ditt script med andra är det lätt för dem att se vilka moiduler som behövs för att köra scriptet.

##Diagnostics Editorn har ett antal inbyggda verktyg för att identifiera syntaxfel. Skriv X y <- 10 Och hovra över det röda krysset, notera popup-skylten

Rstudio upmärksammar dig också opå potentiella problem. Skriv 3 == NA

Notera vad som händer då du hovrar över utropstecknet i marginalen.

##Praktik 1. Det finns en mängd tips och goda råd out there. Ett sådant ställe är Rstudio tips Twitterkonto, https://twitter.com/rstudiotips. Pröva med att gå dit och leta upp något du finner intressant. Pröva det. 2. Vilka andra misstag kan Rstudio markera? Gå till Rstudios support-sida och kolla in https://support.rstudio.com/hc/en-us/articles/205753617-Code-Diagnostics

7 Explorerande analys av data

Placeholder

8 Workflow: Projects

Placeholder

8.1 Vad är viktigt?

8.2 Vart lever analysen? Arbetsbiblioteket

8.3 Sökvägar och bibliotek/mappar

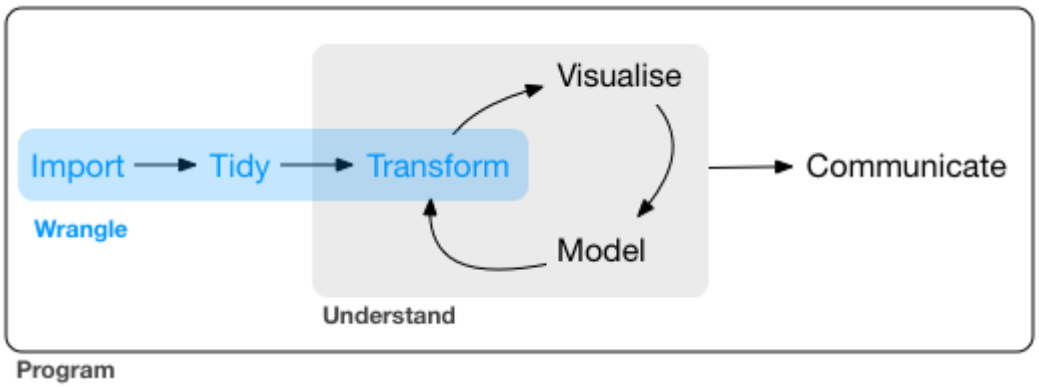
8.4 RStudio projects VIKTIGT!

8.5 Sammanfattningsvis

9 Wrangle - att bråka med data

9.1 Introduktion

Nu ska det handla om att få rådata att bli användbara för analyser och visualisering, det som Wickham kallar *data wrangling* och som är en förutsättning för att kunna arbeta med sina data. Man kan urskilja tre delar i *data wrangling*:



Vi ska gå igenom föjande:

- Först *tibbles*, den variant av dataram (data frame) som används i denna kurs, om vad som skiljer den från en traditiionell data fram och hur de kan skapaas "för hand".
- Sedan *data import*, hur man importerar data från andra källor in till R. Tyngdpunkten ligger på rektangulära text-format men vi ska nämna några andra verktyg för att hantera andra typer av data-format.
- Därefter *tidy data*, som handlar om att städa, rensa rådata, att göra dem "tidy". Mer specifikt om ett konsistent sätt att lagra eller forma data för att underlätta transformering, visualisering och modellering.

Till sist ska vi gå igenom några modernare sätt att hantera olika data-format i R:

- Avsnittet om *Relational data* handlar om verktyg för att hantera relationsdata-mängder.
- I *Strings* introduceras regular expressions, ett kraftfullt verktyg för att manipulera text.
- *Factors* handlar om hur R hanterar kategoriska data.
- *Dates and times* handlar om de viktigaste verktygen för att hantera datum och tid.

10 Tibbles

Placeholder

10.1 Skapa tibbles

10.1.1 Tibbles vs data frames

10.2 Utskrift (Printing)

10.3 Urval

10.4 Använda äldre kod

11 Importera data

Placeholder

11.1 Jämförelse med base R

11.2 Övningar

11.3 Omvandla vektorer

11.3.1 Numeriska variabler

11.3.2 Textsträngar

11.3.3 Kategoriska data

11.3.4 Datum och tid

11.3.5 Övningar

11.4 Hur `readr` "plockar isär" (parse) en fil

11.5 Problem

11.6 Skriva till en fil

11.7 Andra datatyper

12 Städa data (tidy data)

Placeholder

12.1 Introduktion

12.2 Spreading and gathering

12.2.1 Gathering

12.2.2 Spreading

12.3 Övningar

12.4 Separera och förena

12.4.1 Separate

12.4.2 Unite

12.5 Övningar

12.6 Missing values

12.7 Övningar

12.8 En case study

12.9 Övningar

12.10 Non-tidy data

13 Relationer mellan datamängder

Placeholder

13.1 Introduktion

13.2 `nycflights13`

13.3 Keys

13.4 *Mutating joins*

13.4.1 Att förstå *joins*

13.4.2 Inner join

13.4.3 Outer joins

13.4.4 Duplicate keys

13.4.5 Definiera nyckel-kolumner (*key columns*)

13.4.6 Övningar

13.5 *Filtering joins*

13.6 Problem med *joins*

13.7 Set operations

14 Textsträngar (strings)

Placeholder

14.1 Introduktion

14.2 Basics

14.2.1 `stringr`-funktioner

14.2.2 Kombinera strängar

14.2.3 Extrahera delar av textsträngar (subsetting)

14.2.4 Övningar

14.3 Matcha textmönster med hjälp av *regular expressions*

14.3.1 Basic matches

14.3.2 Ankare (Anchors)

14.3.2.1 Övningar

14.3.3 Tecken-klasser

14.3.3.1 Övningar

14.3.4 Repeterande tecken

14.4 Verktyg

14.4.1 Upptäck matchningar

14.4.2 Övningar

14.4.3 Extrahera matchningar

14.4.4 Gruppereade matchningar

14.4.5 Ersätt matchningar

14.4.6 Dela upp textsträngar

14.4.7 Hitta matchningar

14.5 Andra typer av mönster

14.6 Andra typer av regexps

14.7 stringi

15 Kategoriska variabler (Factors)

Placeholder

15.1 Introduktion

15.2 Skapa factors

15.3 General Social Survey

15.4 Modifiera ordningen av factors

15.5 Modifiera factor levels

16 Datum och tidsformat

Placeholder

16.1 Introduktion

16.2 Skapa datum och tid

16.2.1 Från textsträngar

16.2.2 Från enskilda kompnenter

16.2.3 Från andra datum/tid-format

16.2.4 Övningar

16.3 Datum/tid-komponenter

16.3.1 Extrahera komponenter

16.3.2 Avrundning

16.3.3 Ange komponenter

16.4 Tidsintervall

16.4.1 Varaktighet

16.5 Hur länge har VGR funnits?

16.5.1 Perioder

16.6 Ett kalenderår

16.7 Ett exakt år senare

16.7.1 Intervall

16.8 Summering

17 Programmering i R

17.1 Introduktion

Nu är det dags för programmering i R/Rstudio. I denna del ska vi gå igenom fyra avsnitt som vart och ett ger lite olika aspekter på att programmera i R och som underlättar för dig att skriva bra kod.

1. Första avsnittet handlar om **pipes**, `%>%`, som ger större möjligheter att skriva kod på ett mer överskådligt sätt och när man bör respektive inte bör använda det
2. Sedan handlar det om att skriva **funktioner** vilket såväl underlättar skrivningen som läsbarheten i koden
3. Därefter blir det **data-strukturer** i R. Det handlar om vektorer, de fyra vanligaste samt tre *S3 klasser* byggda ovanpå dem och dessutom vad listor vs data-frames är och hur de skiljer sig.
4. Till sist några verktyg som underlättar *iterationer* i programkod - loopar och funktionsprogrammering.

Vi kommer att beröra det nödvändigaste för att hantera R/Rstudio på ett effektivt sätt men det finns en närmast överväldigande mängd programmerings-kunskaper att tillgå. Några tips:

- En hel del finns i Wickhams bok, lätt tillgängligt. http://r4ds.had.co.nz/program-intro.html
- *Hands on Programming with R*, av Garrett Grolemund. En introduktion till R som ett programmeringsspråk. Påminner om innehållet i Wickhams bok men har andra exempel och lite annat angreppsätt.
- *Advanced R* av Hadley Wickham. En fortsättning/fördjupning av Wickhams grundbok, dvs den som ligger till grund för denna kurs. Finns online på http://adv-r.had.co.nz/

18 Pipes

En *pipe* är ett kraftfullt verktyg för att skriva en sekvens av multipla operationer. Pipes (`%>%`) kommer från modulen `magrittr` och laddas autmatiskt med `tidyverse`.

Det finns flera fördelar med att använda *pipes* då man skriver kod, fr.a då man skriver kod som innehåller sekventiella operationer men det finns också tilllfällen då pipes är mindre användbara. t.ex. om

- Pipes är längre än typ 10 steg. Då bör man istället skapa intermediära objekt med meningsfulla namn vilket ger överskådlighet och bättre underlättar debugging
- Du har multipla inputs eller outputs
- Du arbetar med DAGs och komplexa strukturer/relationer mellan objekt. Pipes är i grunden ett linjärt verktyg med vilket det lätt blir svåröverskådlig kod om man vill uttrycka komplexa relationer.

Det finns mer att lära sig om pipes i Wickhams bok http://r4ds.had.co.nz/pipes.html för den som vill veta mer.

19 Funktioner

Placeholder

19.1 När bör du skapa en funktion?

19.2 Funktioner är till för människor och datorer

19.3 Villkorlig exekvering

19.3.1 Villkor

19.3.2 Multipla villkor

19.4 Funktionsargument

19.5 Kolla värden

19.6 punkt-punkt-punkt

19.7 Returnera värden

19.7.1 Explicita `return`-uttryck

19.8 Skriva funktioner med hjälp av pipes

19.9 Environment (svårt hitta ett svenskt uttryck)

20 Vektorer

Placeholder

20.1 Introduktion

20.2 Atomic vectors

20.2.1 Logical vectors

20.2.2 Numeriska vektorer

20.2.3 Character

20.2.4 Använda atomic vectors

20.2.4.1 Konvertering (*Coercion*)

20.2.4.2 Testfunktioner

20.2.5 Scalars och recycling

20.2.5.1 Benämna vektorer

20.2.5.2 Subsetting

20.2.6 Rekursiva vektorer (lists)

20.2.6.1 Visualisera lists

20.2.6.2 Subsetting lists

20.2.7 Attribut

20.2.8 Förstärkta vektorer (Augmented vectors)

20.2.8.1 Factors

20.2.8.2 Dates och date-times

20.2.8.3 Tibbles

21 Itereringar

Placeholder

21.1 For-loops

21.2 Variationer av for-loopar

21.2.1 Modifiera ett existerande obkjekt

21.2.2 När längden på output inte är känd

21.2.3 När sekvensens längd är okänd

21.3 for-loops vs. Funktioner

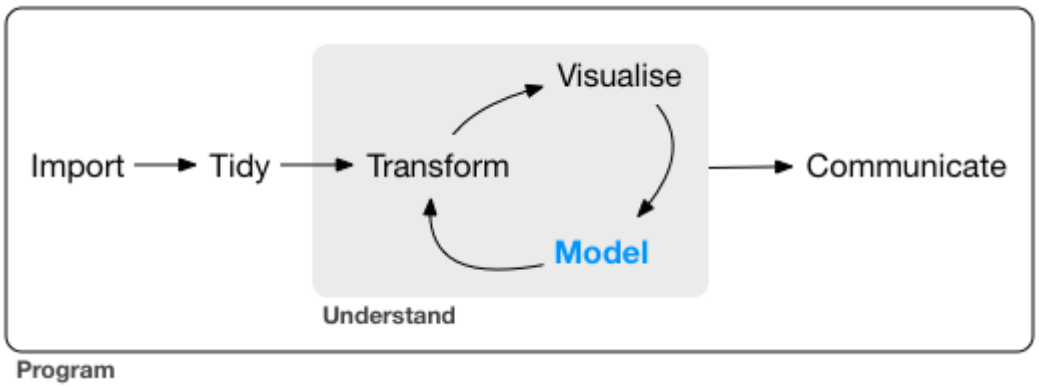
21.4 Mappningsfunktioner

21.4.1 Genvägar

21.5 Hantera errors

22 Introduction

Detta avsnitt ska handla om modellering av data. Syftet i detta skede är snarare *explorativt* (att lära känna data) än analytiskt. Tanken är att vi ska kika på ett antal verktyg för att bättre förstå *variation* i datamängder. Det kommer inte att behandla statsitsisk teori utan fpokus ligger på hur R kan användas för att undersöka datamängder.



- I nästa del, *Modellering. Basics*, ska vi titta närmare på "mekaniken" bakom fr.a. linjära modeller.
- I *Bygga modeller i R* ska vi göra just det, och
- I *Flera modeller* ska vi nosa på hur man kan använda multiipla modeller och därigenom få kombinera modellerings- och programmerings-verktyg.

23 Modellering: basics

Placeholder

23.1 En enkel modell

23.2 Visualisera modeller

23.2.1 Prediktioner

23.2.2 Residualer

23.3 Formler och modell-familjer

23.3.1 Kategoriska variabler

23.3.2 Interaktioner

23.3.3 Interaktioner (två kontinuerliga variabler)

23.3.4 Transformationer

23.4 Missing values

23.4.1 Andra modell-familjer

24 Bygga modeller i R

Placeholder

24.1 Varför är diamanter av låg kvalitet dyrare?

24.1.1 Sambandet pris och karat

24.1.2 En mer komplicerad modell

24.2 Vad är det som påverkar antalet dagliga flighter?

24.2.1 Veckodag

24.2.2 Lördags-effekten

24.3 Lära mer om modellering

25 Grafik för att kommunicera

Placeholder

25.1 Etiketter (labels)

25.2 Annoteringar

25.3 Scales

25.3.1 Axis ticks och teckenförklaringar

25.3.2 Teckenförklaringens (legend) layout

25.3.3 Byta ut en scale

25.4 Zooming

25.4.1 Teman (themes)

25.5 Spara graferna