

Generell informasjon om R-kurset

Det vil bli holdt 2 sesjoner på ca 3 timer hver. Under kursene vil det bli gjennomgang av konsepter, eksempler og litt praktisk oppgaveløsning i R.

Sesjon 1 vil være et “krasjkurs i R”, og handle om import og vasking av data, datatransformasjon og visualisering i R. Sesjon 2 vil ta utgangspunkt i data fra Bulder og handle om datamodellering / maskinlæring.

Etter hver sesjon vil dere få et sett med frivillige hjemmeoppgaver. For de som ønsker å løse disse oppgavene så anbefaler vi at dere leverer inn individuelt, men det er selvsagt lov å samarbeide også. Vi har estimert at oppgavene vil ta rundt 8 timer, men dette vil variere basert på hvilket nivå dere er på i dag.

Alle som leverer hjemmeoppgavene vil få tilbakemeldinger, tips og forbedringsforslag.

Introduksjon til R

PwC Consulting



The background is a blurred photograph of a group of people in a room, likely a classroom or a meeting. Several individuals have their hands raised, suggesting an interactive session or a Q&A period. The room has a whiteboard with some diagrams, a clock on the wall, and various papers or posters pinned to it. The overall atmosphere is one of active participation.

Hvilke forventninger har dere til kurset?

Hvilke programmeringsspråk har dere vært
borti tidligere?

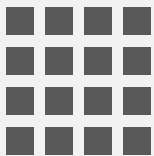
Dagens agenda

1. Introduksjon
2. Praktiske eksempler på anvendelse av R
3. Generelt om R
4. Bruk av pakker
5. Import og vasking av data
6. Datatransformasjon
7. Visualisering

01

Introduksjon

Data og analyse blir en stadig viktigere del av dagens samfunn



Mer data tilgjengelig
enn tidligere



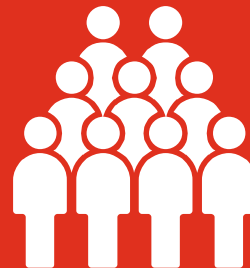
Mer prosessorkraft
tilgjengelig



Demokratisering av
algoritmer og biblioteker



Bedre tilgjengelighet
til kunnskap og verktøy



Ingen barrierer for
å utvikle analytisk
kompetanse

Virksomheters evne til å innhente og tilgjengeliggjøre data er typisk større enn evnen til å **bruke data for å skape verdi**



Datatilgang

Hvor er dataene?



Rapportering

Hva skjedde?



Årsaksanalyser

Hvorfor skjedde det?



Prediktiv

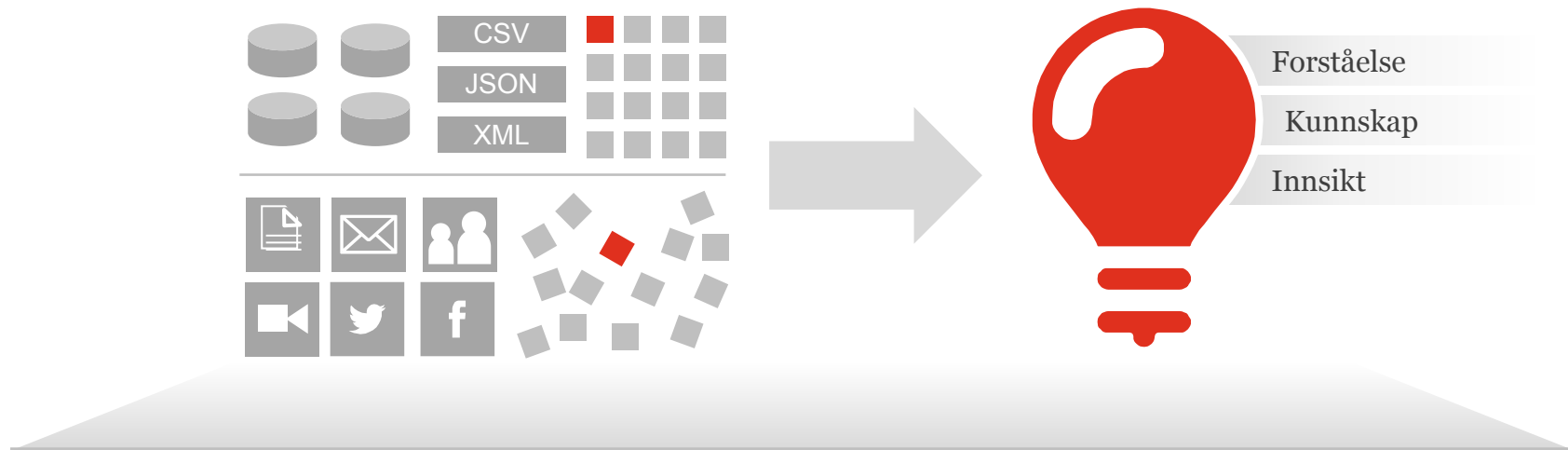
Hva vil skje?







Preskriptiv

Hva bør skje?

Data analytics innebærer å generere innsikt fra data



- 1 ? Problem-definisjon
- 2  Datafangst
- 3  Datavask
- 4  Modellering
- 5  Kommunikasjon

Ved å lære seg R kan man få analyser som er



Faktadrevne



Dynamiske
og interaktive



Reproduserbare



Basert på nye data

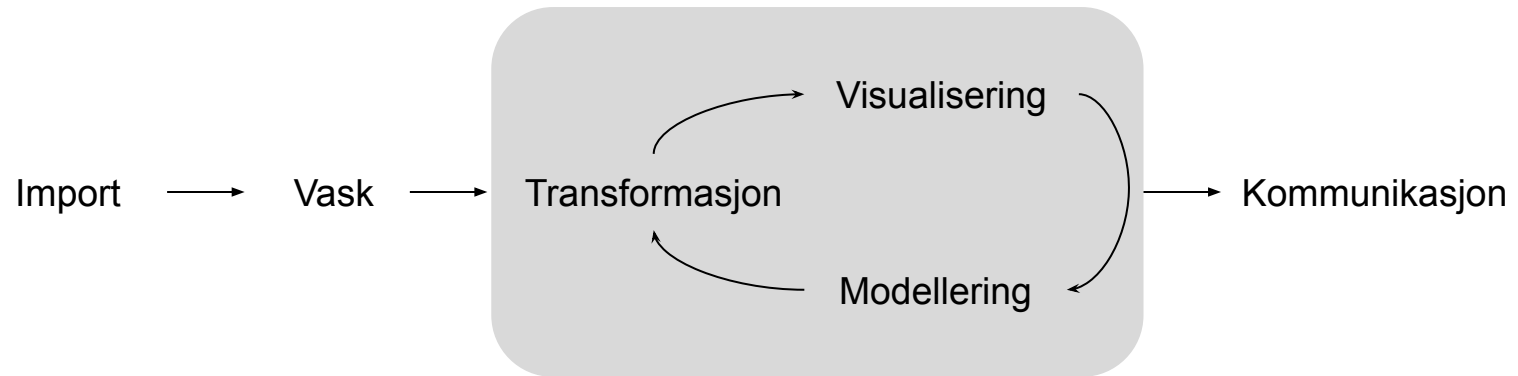


Visualisert på
nye måter



«Produkter» - i
tillegg til analyse

R kan med fordel brukes i de fleste steg hvor man jobber med data



Hva er R?

Open-source statistisk programmeringsspråk

- Laget av Robert Gentleman og Ross Ihaka i 1993
- Originalt basert på språket S, blant annet tilgjengelig gjennom den kommersielle S-Plus pakken

Utvikles kontinuerlig av iherdige brukere

- Over 16 000 tilleggspakker på CRAN (Comprehensive **R** Archive Network)

R vokser ekstremt raskt

- Tidligere et sært språk kun for statistikk-nerder
- Nå et av de mest brukte programmeringsspråkene globalt.



RStudio er et populært IDE som nesten alle R-brukere foretrekker

02

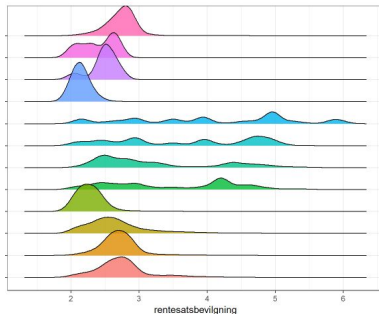
Praktiske eksempler på anvendelse

Noen eksempler på bruk av R



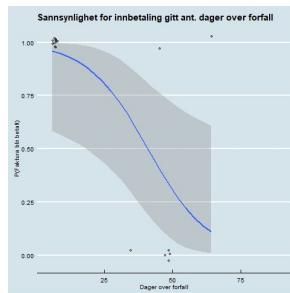
Reprising av låneportefølje

Modellen finner optimalt rentenivå basert på blant annet risikoprofil, akseprate og prissensitivitet.



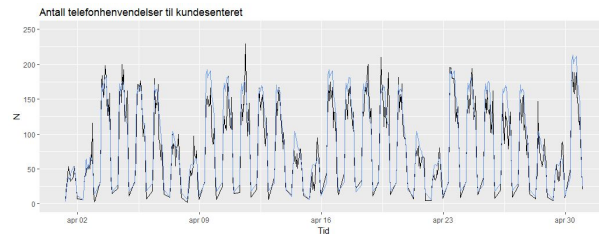
Beslutningsmotor nye utlån

Modellen vurderer kundene basert på store mengder data.



Prognosemodell kundesenter

Prognosene settes time-for-time ca. to måneder i forkant.



03

Basics i R

Assignment og navngivning av objekter

- Alt som eksisterer i R må ha et unikt navn. Hvis du skriver en linje med kode uten å gi det et navn, blir det bare printet, ingenting lagres (vanligvis). Hvis du lager et nytt objekt som har samme navn som et gammelt objekt, blir det gamle overskrevet
- For eksempel, skriver du `1 + 2` vil du få printet “3”, mens dersom du skriver:
`x <- 1 + 2`, lagrer du en ny variabel med navn x
- For assignment i R bruker vi “<-”. Det er også mulig å bruke “=”, men dette regnes ikke som best practice
- Prøv å gå objekter meningsfulle navn ved å bruke **snake_case** konvensjonen, f.eks. “numeric_vector”.

Datatyper i R

Datatype	Eksempel	R-kode
Numeric	42.5	<code>my_numeric <- 42.5</code>
Integer	4	<code>my_integer <- 4</code>
Character	"some text"	<code>my_character <- "some text"</code>
Logical	True	<code>my_logical <- TRUE</code>
Factor	Ukedag - mandag/tirsdag...	<code>my_factor <- factor(...)</code>
Date	"2018-01-01"	<code>my_date <- as.Date("2018-01-01")</code>

De viktigste datastrukturene i R er vektorer og dataframes. En dataframe består av et valgfritt antall like lange vektorer

Vektor



22	52	31	25
----	----	----	----

```
Alder = c(22, 52, 31, 25, 67)
```

Merk: En vektor kan bare inneholde én type data, her integer

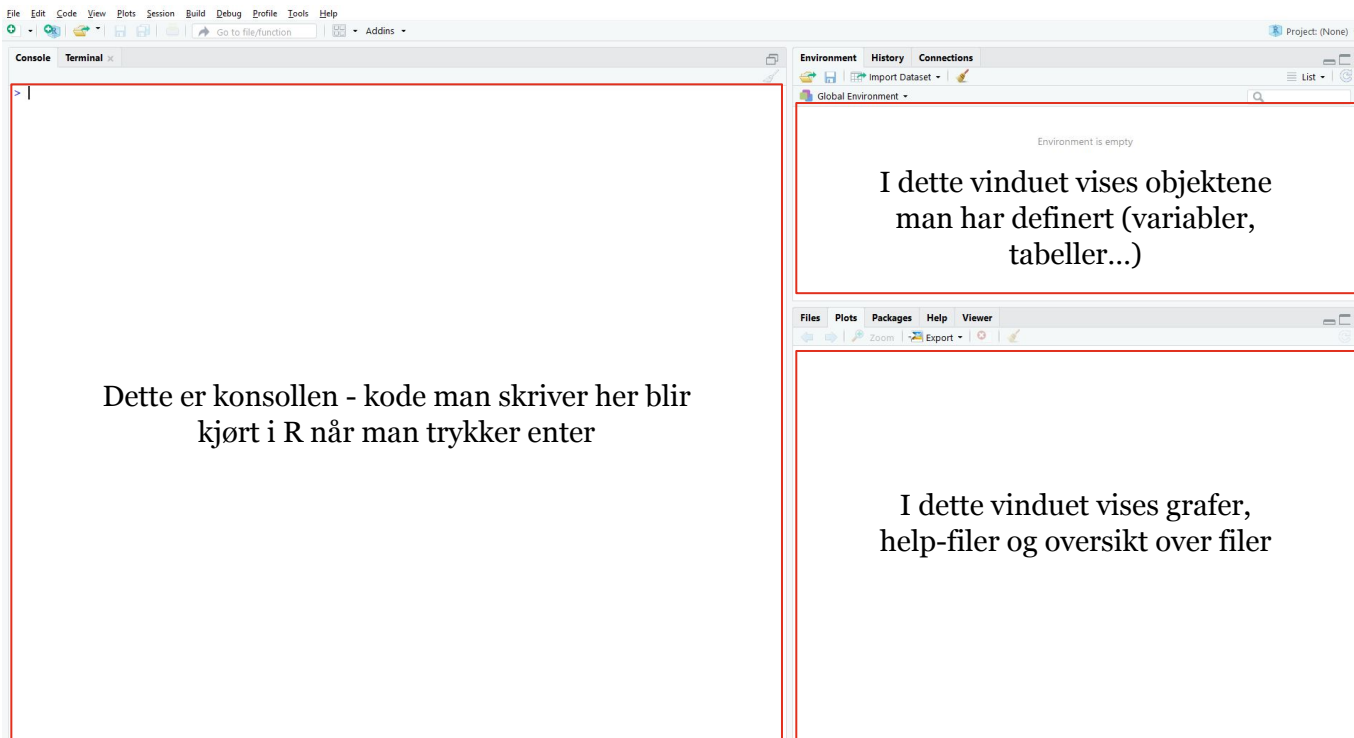
Dataframe

Alder	Kjønn	Vekt
22		
52		
31		
25		

```
df <- data.frame(Alder = c(22, 52, 31, 25, 67),  
                 Kjønn = c(),  
                 Vekt = c())
```

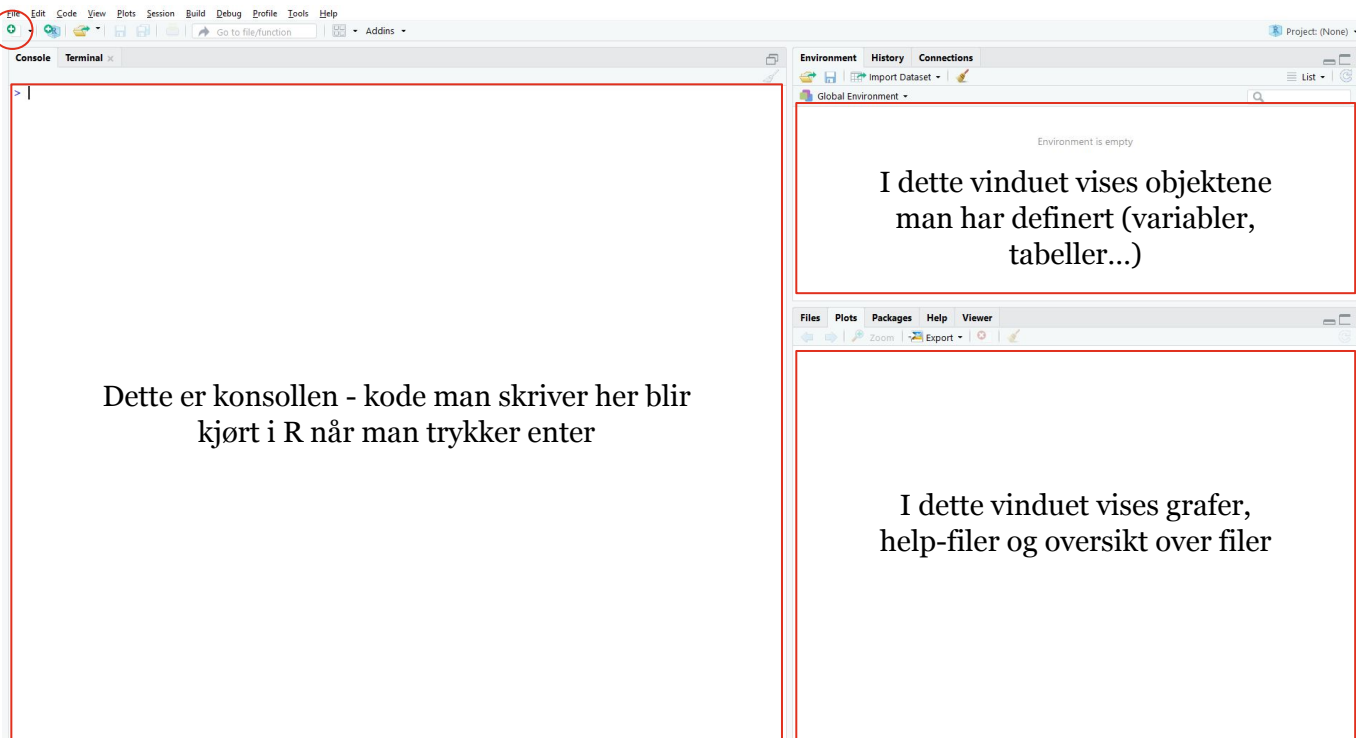
Merk: En dataframe kan inneholde alle typer data, men hver kolonne kan bare være én type (siden en kolonne er en vektor)

Første gang man åpner RStudio ser det slik ut

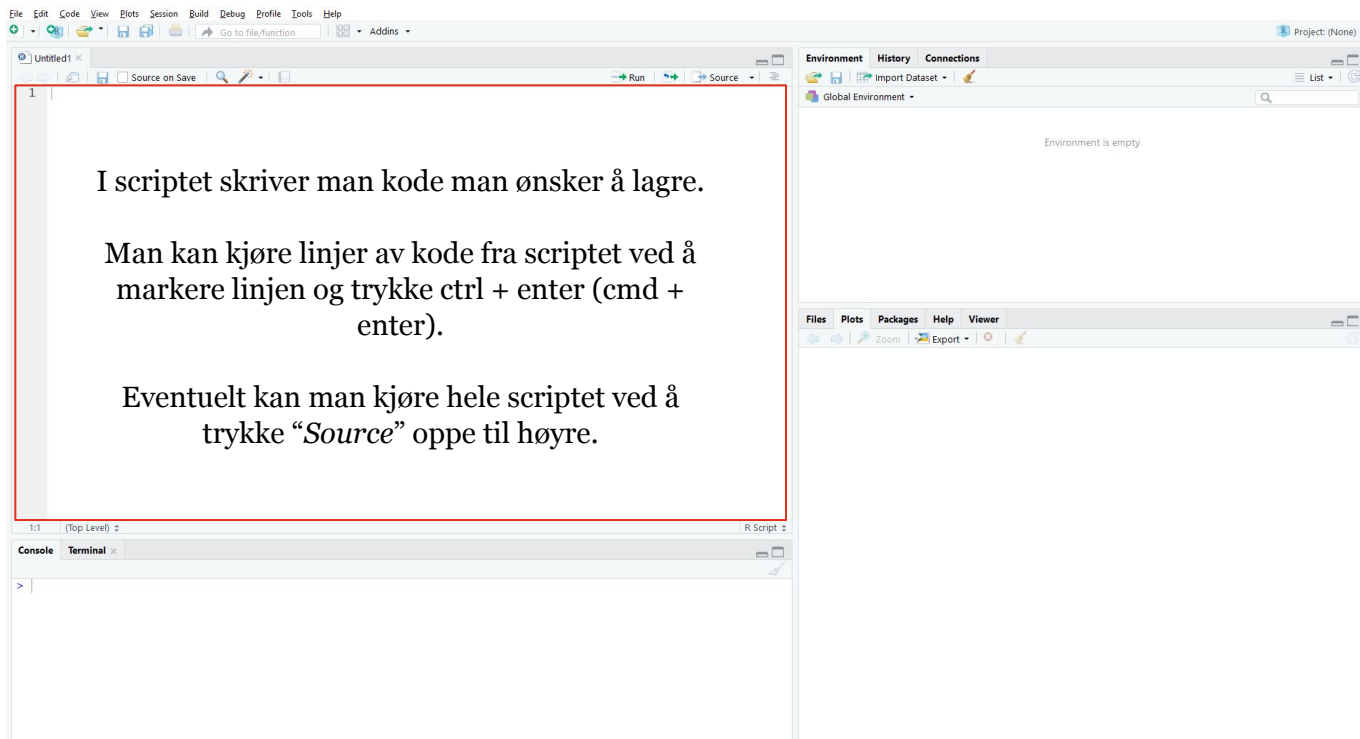


Første gang man åpner RStudio ser det slik ut

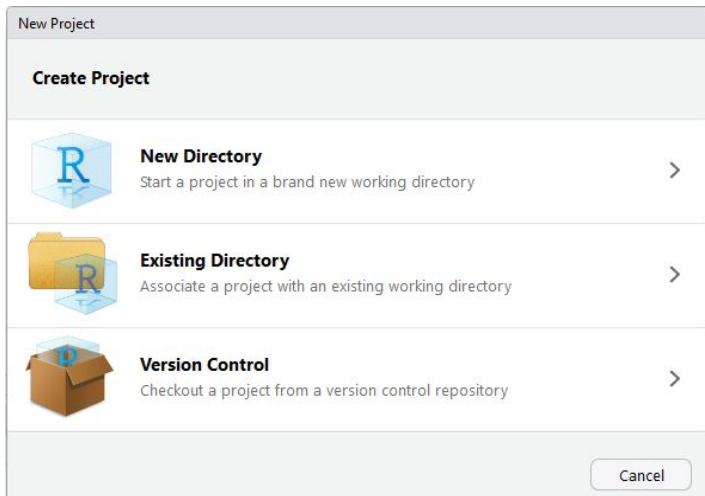
Ved å trykke her, og velge “R Script”, åpner man sitt første script



Første gang man åpner RStudio ser det slik ut



R-kode bør ligge i R-prosjekter for å unngå mange unødvendige problemer knyttet til organisering og bruk



- Ved å samle koden din i et R-prosjekt sikrer du at alle relevante filer er samlet og tilgjengelig på samme sted.
- Tre ulike måter man kan oppnå dette på
 - **New Directory** lager et nytt og tomt prosjekt
 - **Existing Directory** brukes om man skal åpne et allerede eksisterende prosjekt (e.g. fra en kollega)
 - **Version Control** brukes om man skal koble R-prosjektet til Git, for eksempel GitHub
- I dag vil vi bruke “New Directory”, så alle kan begynne med å lage et tomt prosjekt med et passende navn for kurset.

04

Bruk av pakker

R-pakker er samlinger av funksjoner og data som noen har organisert slik at det er enkelt å ta i bruk for andre

Pakker er en ekstremt viktig del av R. Det er den foretrukkede måten å dele og gjenbruke både kode og data.

Ved å bruke pakker får man tilgang til et stort bibliotek av funksjonalitet som ellers vil være enten umulig eller svært tidkrevende å lage selv.

Per september 2020 er det over 16 000 pakker på CRAN, og enda flere på Github. Det er derfor viktig å være noe kritisk til hvilke pakker en tar i bruk - ikke alle er av like god kvalitet.



Vi anbefaler å bruke *tidyverse*-samlingen av pakker



Veldig mye i R kan gjøres med samlingen av pakker kalt **“tidyverse”**.

Disse pakkene vedlikeholdes og utvikles av RStudio.

Pakkene erstatter flere “base R”-funksjoner, og har betydelig forbedret syntaks og er generelt mye raskere enn grunnpakkene.

Noen av de viktigste pakkene i tidyverse

Pakke	Formål	Eksempelfunksjoner
readr	Lese inn data i csv-format e.l. (ca 5x raskere enn <i>base R</i>)	read_csv, write_csv
dplyr	Databehandling, filtrering, aggregering	group_by, filter, left_join, %>%
ggplot2	Visualisering	ggplot, geom_point, geom_line
tidyr	Reformater data (f.eks. gjør bred data smal)	gather, spread
shiny	Lag interaktive web-applikasjoner og visualiseringer	NA
stringr	Behandling av tekst	str_replace, str_trim
forcats	Behandling av faktorvariabler	fct_reorder, fct_lump
purrr	Utfører samme funksjon på flere elementer	map
lubridate	Behandling av datoer	ymd, ymd_hms, %m-%

Installere og laste inn pakker

Før man bruker en pakke for første gang må den lastes ned.

Bruk kommandoen:

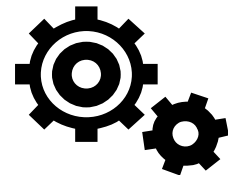
```
install.packages("<PakkeNavn>")
```



Før man bruker en pakke må den også lastes inn.

Dette gjøres ved å skrive:

```
library(<PakkeNavn>)
```



Pakker må lastes inn hver gang man starter R på nytt, men trenger ikke installeres på nytt (med mindre man vil ha ny versjon)

05

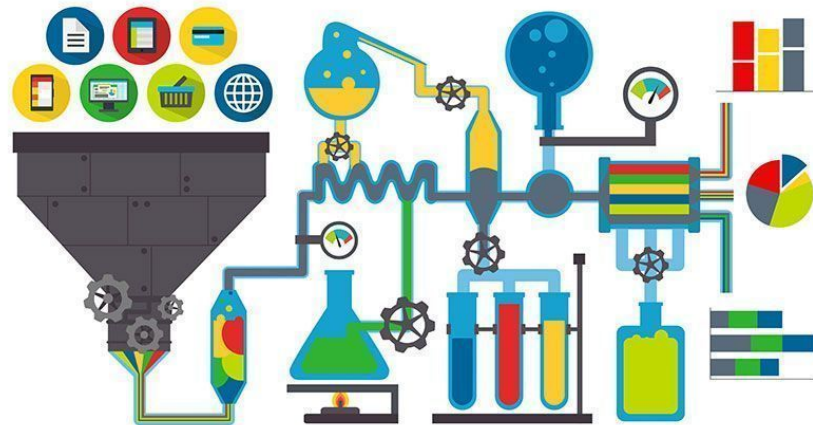
Import og rydding av data

Importering av CSV/Excel-filer

- For å lese inn en excel-fil eller csv-fil er det enkelt nok bare å legge filen i prosjekt-folderen og skrive f.eks.

```
df <- read_csv2("csvfil.csv") eller  
df <- read_xlsx("excelfil.xlsx")
```

- Unngå manuell databehandling før man importerer til R - dette vil føre til analyser som ikke er reproducerbare!



Navnet på venstresiden av pilen er valgfritt; **df** (*data-frame*) er bare et eksempel på hva man kan kalle datasettet man leser inn. Funksjonene i eksempelet er hentet fra hhv. readr og readxl-pakkene. Disse må lastes inn først

Eksempel: Import fra SQL-database

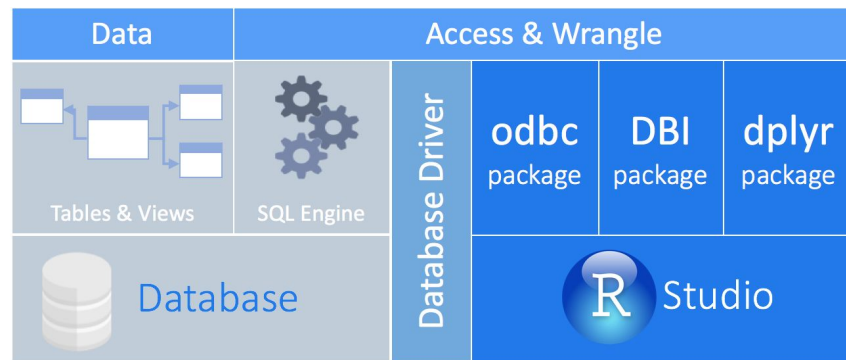
```
library(odbc)  
library(DBI)
```

```
con <- dbConnect(odbc(),  
  driver = "SQLServer",  
  server = "mysqlhost",  
  database = "mydbname",  
  uid = "myuser",  
  pwd = "password")
```

```
df <- dbReadTable(con, "tabellnavn")
```

For å skrive til databasen igjen:

```
dbWriteTable(con, "mydbname", df, overwrite = TRUE)
```



Obs! For å unngå å lagre database-passordet i plain text kan man bruke R-pakken *keyring*.

Det bør være et mål at all data er *tidy*

Når man henter data fra f.eks. Excel, er ofte dataen i et format som er uegnet for analyse i R. Både i databasen og for optimal analysestruktur ønsker vi data som er såkalt “tidy”

- hver variabel er en kolonne
- hver observasjon er en rad
- i en tabell er det kun en type “observasjonsenhet”

Vanlige årsaker til at data ikke er tidy:

- Kolonneoverskrifter er verdier, ikke variabelnavn
- Flere variabler er lagret i en kolonne
- Variabler lagret som en blanding av rader og kolonner

The diagram illustrates the concept of tidy data by showing three representations of the same data. The first is a standard table. The second is a table with vertical arrows indicating variables. The third is a table with horizontal arrows indicating observations.

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	17206362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

table1

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	17206362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

variables

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	17206362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

observations

Kilde: Hadley Wickham, Tidy data (2014)

For å rydde opp i rotete data, kan vi bruke pakken tidyr

Fra pakken *tidyr* finner vi spesielt to funksjoner som er viktig når man har messy data:

pivot_wider: Brukes dersom man har observasjoner som skulle vært variabler (eks: to variabler er laget i samme kolonne). Merk: het tidligere “spread”

pivot_longer: Brukes dersom man har variabler som skulle vært observasjoner (eks: en variabel er lagret i to kolonner). Merk: het tidligere “gather”.

```
fish station seen
<fct> <fct> <int>
1 4842 Release 1
2 4842 I80_1 1
3 4842 Lisbon 1
4 4842 Rstr 1
5 4842 Base_TD 1
6 4842 BCE 1
7 4842 BCW 1
8 4842 BCE2 1
9 4842 BCW2 1
10 4842 MAE 1
# ... with 104 more rows
```

```
fish_encounters %>%
  pivot_wider(names_from = station,
              values_from = seen)
```

```
# A tibble: 19 x 12
  fish Release I80_1 Lisbon Rstr Base_TD BCE BCW BCE2 BCW2 MAE HAW
  <fct> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
1 4842 1 1 1 1 1 1 1 1 1 1 1
2 4843 1 1 1 1 1 1 1 1 1 1 1
3 4844 1 1 1 1 1 1 1 1 1 1 1
4 4845 1 1 1 1 1 0 0 0 0 0 0
5 4847 1 1 1 0 0 0 0 0 0 0 0
6 4848 1 1 1 1 0 0 0 0 0 0 0
7 4849 1 1 0 1 1 1 0 0 0 0 0
8 4850 1 1 0 1 1 1 0 0 0 0 0
9 4851 1 1 0 0 0 0 0 0 0 0 0
10 4854 1 1 0 0 0 0 0 0 0 0 0
11 4855 1 1 1 1 1 0 0 0 0 0 0
12 4857 1 1 1 1 1 1 1 1 1 0 0
13 4858 1 1 1 1 1 1 1 1 1 1 1
14 4859 1 1 1 1 0 0 0 0 0 0 0
15 4861 1 1 1 1 1 1 1 1 1 1 1
16 4862 1 1 0 0 0 0 0 0 0 0 0
17 4863 1 1 0 0 0 0 0 0 0 0 0
18 4864 1 1 0 0 0 0 0 0 0 0 0
19 4865 1 1 1 0 0 0 0 0 0 0 0
```

```
A tibble: 18 x 11
  religion <$10k> <$10-20k> <$20-30k> <$30-40k> <$40-50k> <$50-75k> <$75-100k> <$100-150k> <>150k> <Don't know/ref-
  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Agnostic 12 27 37 52 35 70 73 59 74 76
2 Atheist 27 21 30 34 33 58 62 39 53 54
3 Buddhist 418 617 732 670 638 1116 949 792 633 1469
4 Catholic 15 14 15 11 10 35 21 17 18 116
5 Don't know/- 575 869 1064 982 881 1486 949 723 414 1529
6 Evangelical- 1 9 7 11 47 34 47 48 54 37
7 Hindu 228 244 236 238 197 223 131 81 78 339
8 Historical- 20 27 24 24 21 30 15 11 6 37
9 Jehovah's W- 19 19 25 25 30 95 69 87 151 162
10 Jewish 289 495 619 655 1107 939 753 634 1328
11 Mainline Pr- 29 40 48 51 56 112 85 49 42 69
12 Mormon 6 7 9 10 9 23 16 8 6 22
13 Orthodox 13 17 23 32 32 47 38 42 46 73
14 Other Chris- 9 7 11 13 13 14 18 14 12 18
15 Muslim 20 33 40 46 49 63 46 40 41 71
16 Other Faiths 5 2 3 4 2 7 3 4 4 8
17 Unaffiliated 217 299 374 365 341 528 407 321 258 597
```

```
relig_income %>%
  pivot_longer(-religion,
              names_to = "income",
              values_to = "count")
```

```
# A tibble: 180 x 3
  religion income count
  <chr> <chr> <dbl>
1 Agnostic <$10k> 27
2 Agnostic $10-20k 34
3 Agnostic $20-30k 60
4 Agnostic $30-40k 81
5 Agnostic $40-50k 76
6 Agnostic $50-75k 137
7 Agnostic $75-100k 122
8 Agnostic $100-150k 109
9 Agnostic >150k 84
10 Agnostic Don't know/refused 96
# ... with 170 more rows
```

Oppgaver - import og vasking av data



Start et nytt R-prosjekt og åpne “**oppgaver_s1.Rmd**”. Oppgaver ligger inne i dokumentet. Ikke vær redd for å spørre om hjelp!

Eksempel på R Markdown

```
## Scatter plots
Use the smaller dataset "df_mdp" for the scatter plots.

### a) Visualize the relationship between the duration and length of the trips with a scatter plot
Use geom_point
```{r}
...

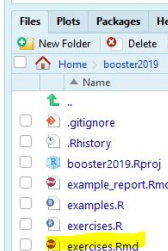
b) Same as in a), but use geom_jitter instead of geom_point
"geom_jitter" avoids overplotting (overlapping points) by nudging the points slightly in different directions.
```{r}
...

### c) Add some transparency and color
Hint: The geom-functions take in arguments. alpha = [a number between 0 and 1] and color = "[name of a color]" sets the
transparency and color of points.
```{r}
...

d) Same as in b), but only show trips that lasted 30 minutes or less
Use ylim(). The argument of ylim is a vector with the lower bound as the first element and upper bound as the second element. A
vector is created by the function c().
```{r}
...
```

Skriv svar
her

Klikk grønn
pil for å kjøre
kode-chunk



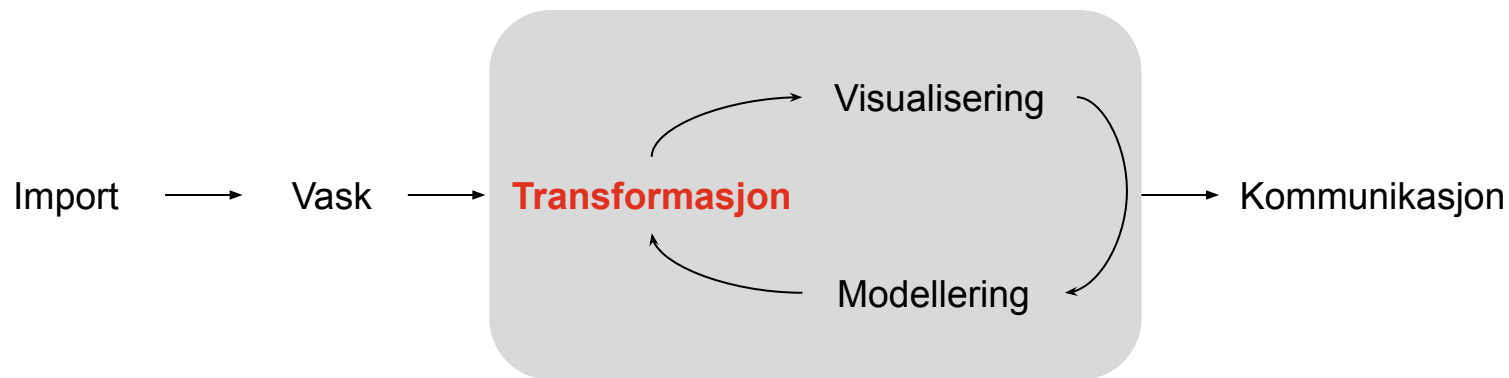
Sammenligning og mulige filtre

Logiske operatører		Eksempel		
==	er nøyaktig lik	4 == 4	→	TRUE
>=	er større enn eller lik	4 >= 5	→	FALSE
<=	er mindre enn eller lik	4 <= 4	→	TRUE
<	er mindre enn	5 < 4	→	FALSE
>	er større enn	5 > 4	→	TRUE
&	og	TRUE & FALSE	→	FALSE
	eller	TRUE FALSE	→	TRUE
!=	er ikke lik	4 != 4	→	FALSE

06

Datatransformasjon

R er svært godt egnet for datatransformasjon



dplyr er den mest populære pakken for datatransformasjon i R. Her er de viktigste *dplyr*-verbene for databehandling

Hva	Funksjon	Eksempel
Legg til variabel	mutate	<pre>df %>% mutate(x = y + 1)</pre>
Velg variabler	select	<pre>df %>% select(x)</pre>
Filtrer data	filter	<pre>df %>% filter(x == 1)</pre>
Grupper etter variabel	group_by	<pre>df %>% group_by(x)</pre>
Summer etter gruppert variabel	summarise	<pre>df %>% group_by(x) %>% summarise(snitt_x = mean(x))</pre>
Slå sammen to datasett basert på nøkkel	left_join, inner_join, right_join, full_join	<pre>df1 %>% left_join(df2, by = "id")</pre>

Bruk av *pipeline* - “%>%”



En av de viktigste bærebjelkene i *tidyverse* er bruken av den såkalte pipe-operatoren “%>%”.

Pipe-operatoren er inspirert av pipeline-karakteren i Linux “|”, og lar oss kjede sammen funksjoner.

Muligheten til å kjede sammen funksjoner er kanskje en av de største fordelene med *tidyverse*!

Bruk av *pipeline* gjør koden enklere å skrive - men også å forstå!



Istedenfor ...

```
hold_r_kurs(gå_inn_på_hjemmekontor(spis_frokost(stå_opp(meg))), for="Bulder")
```



... kan vi få mer lesbar kode med pipelines:

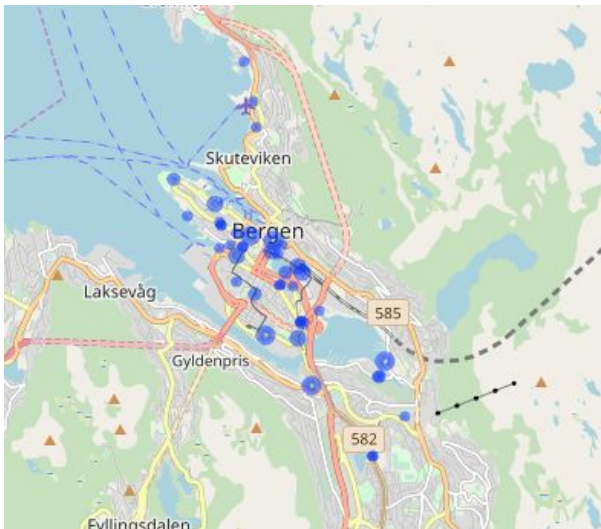
```
meg %>%  
  stå_opp() %>%  
  spis_frokost() %>%  
  gå_inn_på_hjemmekontor() %>%  
  hold_r_kurs(for = "Bulder")
```



- Symbolet “%>%” kan leses som “*then*” eller “*deretter*”
- Rent teknisk sender *pipelinen* objektet til venstre for *pipen* inn i første argument av den påfølgende funksjonen.

For resten av oppgavene og innleveringen vil vi bruke data fra Bergen Bysykkel

Vi vil jobbe med data fra Bergen bysykkel-prosjektet, som deler dataen sin via et gratis API.



Variabler i rådata

started_at	datetime	Date and time for when the trip started
ended_at	datetime	Date and time for when the time ended
duration	integer	Duration of trip in seconds
start_station_id	integer	Unique ID for start station
start_station_name	character	Name of start station
start_station_description	character	Description of location of start station
start_station_latitude	numeric	Latitude of start station
start_station_longitude	numeric	Longitude of start station
end_station_id	integer	Unique ID for end station
end_station_name	character	Name of end station
end_station_description	character	Description of location of end station
end_station_latitude	numeric	Latitude of end station
end_station_longitude	numeric	Longitude of end station

Eksempler på variabler som kan legges til

duration_minutes	numeric	Duration of trip in minutes
month	numeric	Which month the trip started
wday	factor	Day of the week
time_of_day_started	numeric	Which hour the trip started
distance_trip	numeric	Length of trip in meters
isWeekday	logical	Indicates whether the trip was made in a weekday or not

Oppgaver - datatransformasjon







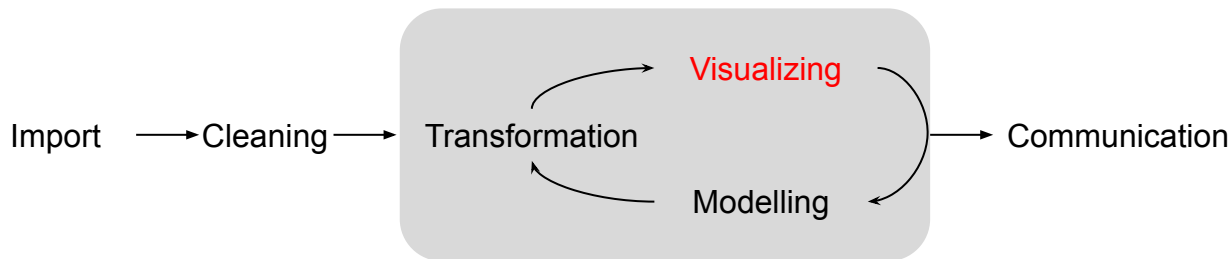
Fortsett på oppgavene i ditt R Markdown dokument
“oppgaver_s1.Rmd”

07

Visualisering

Hvorfor bør du visualisere dataen din?

-  Det lar deg oppdage nye potensielle sammenhenger i datasettet ditt og utforme hypoteser
-  Du vil ofte oppdage feil og dataproblemer mye lettere når du ser det visuelt
-  Det er en utmerket måte å kommunisere dine resultater
-  Det er gøy!



Visualisering med ggplot2



“ggplot2” er den mest brukte pakken i R for data-visualisering



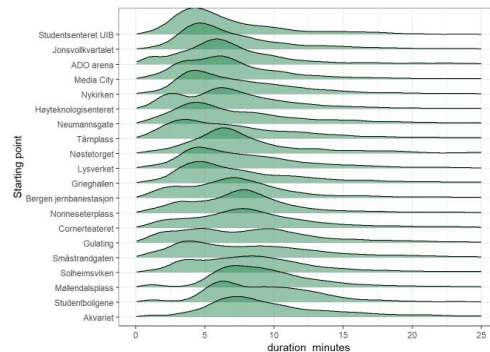
Pakken er bygget på “*The Grammar of Graphics*” og er mest kjent for sin intuitive syntaks.



Pakken brukes blant annet hyppig av BBC og Financial Times.



Mens “grunnpakken” kun inneholder statiske plot, finnes det mange tilleggspakker som gir ekstra funksjonalitet som gifs o.l.



ggplot bygges opp av *aesthetics* og *layers*

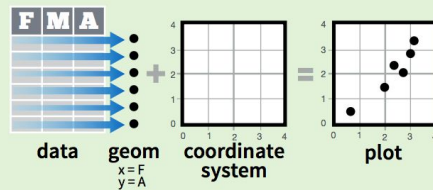
Syntaksen i alle visualiseringer i ggplot er bygget opp på samme måte:

```
ggplot(data = df, aes(x = var1, y = var2)) +  
  geom_point()
```

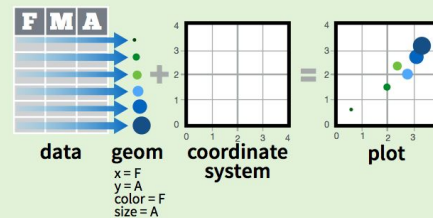
Man angir først hvilke data man bruker, så sier man hvilke variabler som skal være på x- og y-aksen (aesthetics), så hvilket *geom* dataen skal illustreres med.

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.

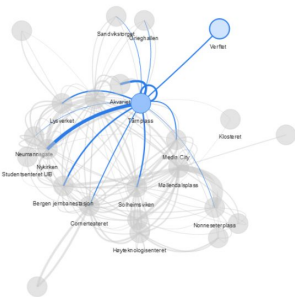


45

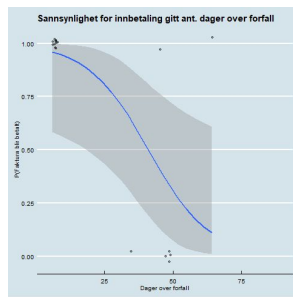
Mer avanserte plot: GIFS, interaktive kart, nettverk

Mens ggplot2 er utmerket for statiske plot, må man bruke andre pakker for å lage interaktive plot eller GIFs. Det finnes også mange utvidelser av ggplot2 som bygger på mer funksjonalitet, f.eks. ggforce.

Sjekk ut <https://www.htmlwidgets.org/> for et sett med skikkelig kule visualiseringspakker for å lage interaktive plots.



VisNetwork



gganimate

Pakke	Bruksområde
ggplot2	Normale plots
plotly	Interaktive plots. Inneholder også en enkel funksjon for å gjøre et ggplot interaktiv (ggplotly).
gganimate	Lag GIFs fra ggplot-objekter
leaflet	Interaktive kart
visNetwork	Interaktive nettverk

Oppgaver - visualisering



Fortsett med oppgaver under visualisering i
“**oppgaver_s1.Rmd**”. Ikke vær redd for å
spørre om hjelp!

08

Avslutning - hjemmeoppgaver, tips og triks

For de ekstra ivrige har vi også laget et sett med hjemmeoppgaver

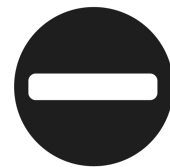


Vi sender dere “**homework_lecture1.html**”. Oppgaver ligger inne i dokumentet. Oppgavene kan løses i RMarkdown og løsning sendes til andre.rivenaes@pwc.com og jan.petter.iversen@pwc.com innen 2 uker. Send *både* html-fil og Rmd-fil. Vi gir tilbakemeldinger og tips til alle som sender inn.

Noen tips og råd

- Test koden underveis - kjør en og en linje og sjekk at *output* er som forventet
- Sjekk at variabelen er i riktig format - dette er en vanlig feil blant nybegynnere
 - F.eks. kan en *numeric* verdi være kodet som *character* - 7 vs. "7"
 - Alltid sjekk dette når data importeres, f.eks. ved å bruke `str(df)`
- Ved å google smart finner du nesten alltid noen som har løst nesten akkurat samme problemstilling som deg. Hvis ikke du finner noe fornuftig etter noen minutter med googling - spør om hjelp!
- Det er veldig ofte mulig å løse et gitt problem i R med veldig få linjer med kode. Dersom dere løser noe som bør være et vanlig problem og det krever mange linjer med kode har dere trolig gjort det unødvendig komplisert

Unngå dette



- Ikke bruk `setwd()` i starten av scriptet
 - Dette er dårlig praksis og fører til kode som ikke kan kjøres for andre
 - Bruk *relative paths* og RStudio-prosjekter for å angi filplassering
- Ikke lag for-løkker med mindre det er eneste utvei
 - For-løkker er generelt trege og det finnes nesten alltid bedre løsninger
 - R er laget og optimalisert for vektorisert kode!
 - Der for-løkker er nødvendig - erstatt med funksjoner fra *purrr*: `map`, `map_df`, `pmap` osv.
- Ikke lag egne, komplekse funksjoner dersom anerkjente pakker allerede gjør det samme!

Gode bøker om dagens tema

Bok	Forfatter	Kommentar
<u>R for Data Science</u>	Hadley Wickham & Garrett Grolemund	Data science med Tidyverse - obligatorisk lesning!
<u>Advanced R</u>	Hadley Wickham	For spesielt interesserte - dekker det aller meste du bør vite om R
<u>Data visualisation - a practical introduction</u>	Kieran Healy	Utrolig god bok om datavisualisering

Lenker i boktittelen

Neste del av kurset

- Maskinlæring og modellering av Bulder-data