

INTRODUKSJON TIL DATABEHANDLING, STATISTISK PROGRAMMERING, R, & R-STUDIO

Av Einar E. Låker

Hvem er jeg?

- Einar E. Låker | 25 år
- Ph.d. kandidat i Miljøvitenskap / Jordfag
- Uteksaminert masterstudent (2024) i miljøvitenskap med A på master-oppgaven
- 2 år som R-klubb-veileder



Hvorfor skal dere stole på meg?

- 2 år med veiledning av master, Ph.d. og bachelor studenter i bruk av R for gradsoppgaveprosjekter
- 1 års Ph.d. kandidat i miljøvitenskap
 - Påvirkningen av klimaendringer på jordfysiske egenskaper
 - Fikk stillingen basert bla.a. på statistikk ferdigheter
- 60p | Master oppgave: avansert maskinlæring & ML-modell analyse for å beregne hvor lett vann beveger seg gjennom en prøve basert på 3D-xray bilder
- Skrevet bok om bruk av Excel for Miljødirektoratet
- 10 Års arbeidserfaring med Excel
- Først introdusert til Excel før jeg var 1 år gammel

“

YOU THINK
DATA IS YOUR
ALLY? BUT YOU MERELY
ADOPTED DATA.
I WAS BORN IN IT.
MOLDED BY IT.

Einar M

Hvorfor bryr jeg meg?

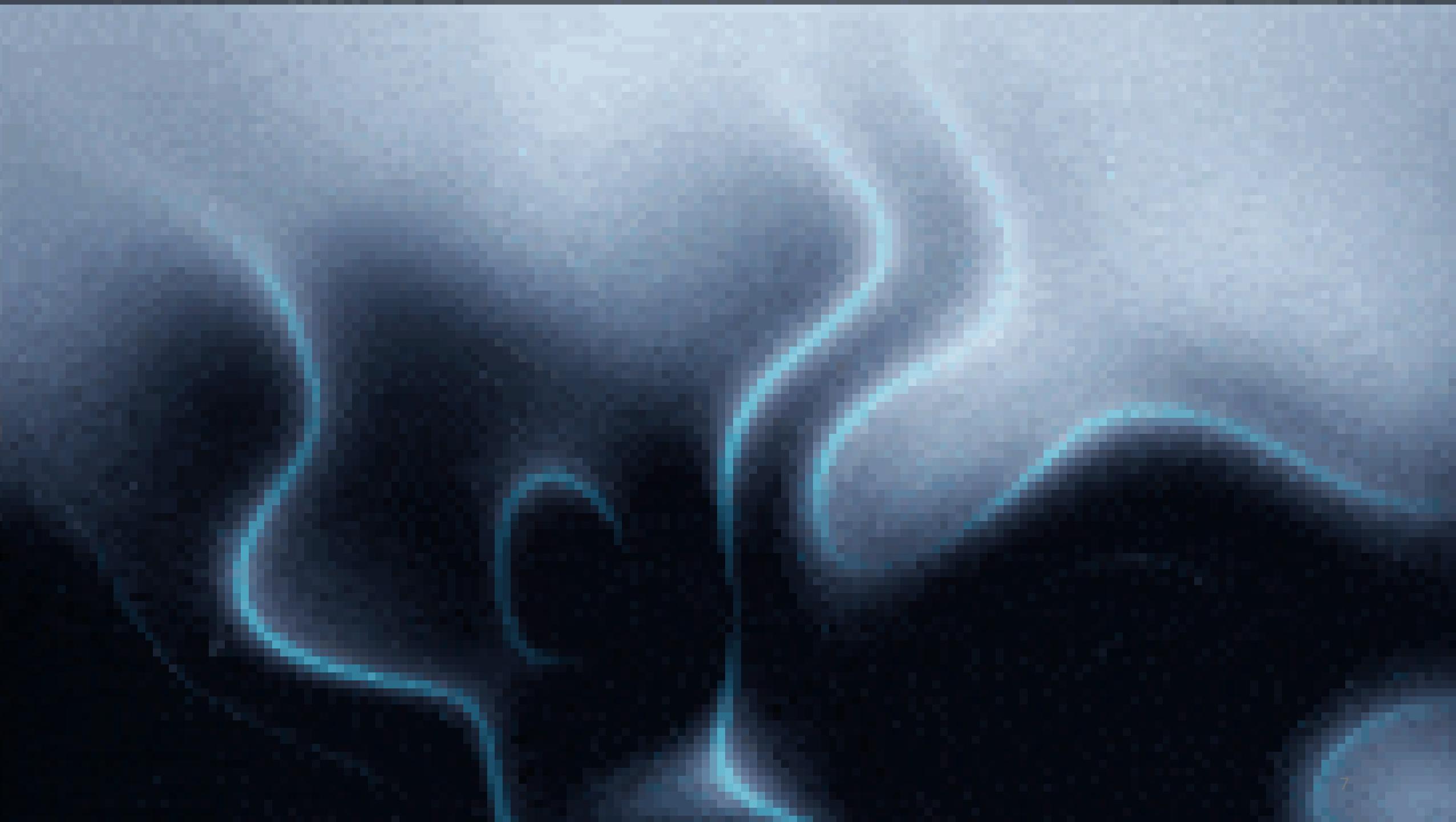
- Jeg mener at data arbeid er viktig arbeid
- Jeg tror at forståelse av statistikk, grafer, og dataarbeid burde være allmennkunnskap – *verden blir bedre om folk forstår statistikk, når de kan kjenne igjen løgn med tall, eller alternative begrunnelser*
- Jeg vil at dere skal gjøre det best mulig i det arbeidet dere gjør nå, og eventuelt senere

3 eksempler på hvorfor forståelse av data, statistikk og programering er viktig

- **Voldtektsaker i Sverige**
- Svensk voldtekts statistikk viser at Sverige er et av landene med høyest antall voldtekter pr kapita.
Hvorfor er det sånn?
- **Nyrestein operasjoner**

| Treatment Stone size | Treatment A | Treatment B |
|-------------------------|--------------------------|--------------------------|
| Small stones | Group 1 93% (81/87) | Group 2 87% (234/270) |
| Large stones | Group 3 73% (192/263) | Group 4 69% (55/80) |
| Both | 78% (273/350) | 83% (289/350) |

 - *Kan «Treatment A» fortsatt være bedre enn «Treatment B»?*
- **Elon Musks «150 år gamle svindlere»**
 - Elon Musk skryter av DOGEs evne til å finne snyltere, blant annet at det er flere 150 åringer som snylter til seg penger.
Hvorfor ler de data-folk av dette?



Hva prøver vi å gjøre i dag?

- Jeg ønsker å gi dere et sett med **nøkkel-innsikter** og **nøkkel-verktøy** slik at dere kan komme i gang med **R, databehandling, og statistisk programmering** og videre **bygge kjerneferdighetene** for disse oppgavene.
- Jeg ønsker å gi dere grobunnen for å bli selvstendige
- Vi gjør dette ved å angripe de overordnede problemene studenter ofte har

Hvordan gjør vi dette?

- Et *spesifikt synspunkt* — *mitt synspunkt*
 - Vi kommer ikke til å bruke mye tid på å diskutere mange forskjellige metoder, vi bruker heller denne tiden til å gi dere et spesifikt ståsted som dere kan jobbe ut ifra.
- Fugle- og froske-perspektiv  & 
 - Hva skal du gjøre når? Hvordan ser arbeidsflytene ut? Hva skal jeg gjøre når problem X oppstår?
- Håndfast og mindre håndfast kunnskap
 - Ofte er det man *gjør* like viktig som *hvorfor* man gjør det
- *Ferdigheter VS Informasjon*
- *Bli en «story-teller»; ikke en statistikk mitraljøse*



Kom ut av nybegynner-hullet!

Det aller vanskeligste er å komme seg ut
av «nybegynner-hullet»

Start enkelt og bygg utover

Det er helt greit å *glemme ting*

Det er helt greit å *google ting*

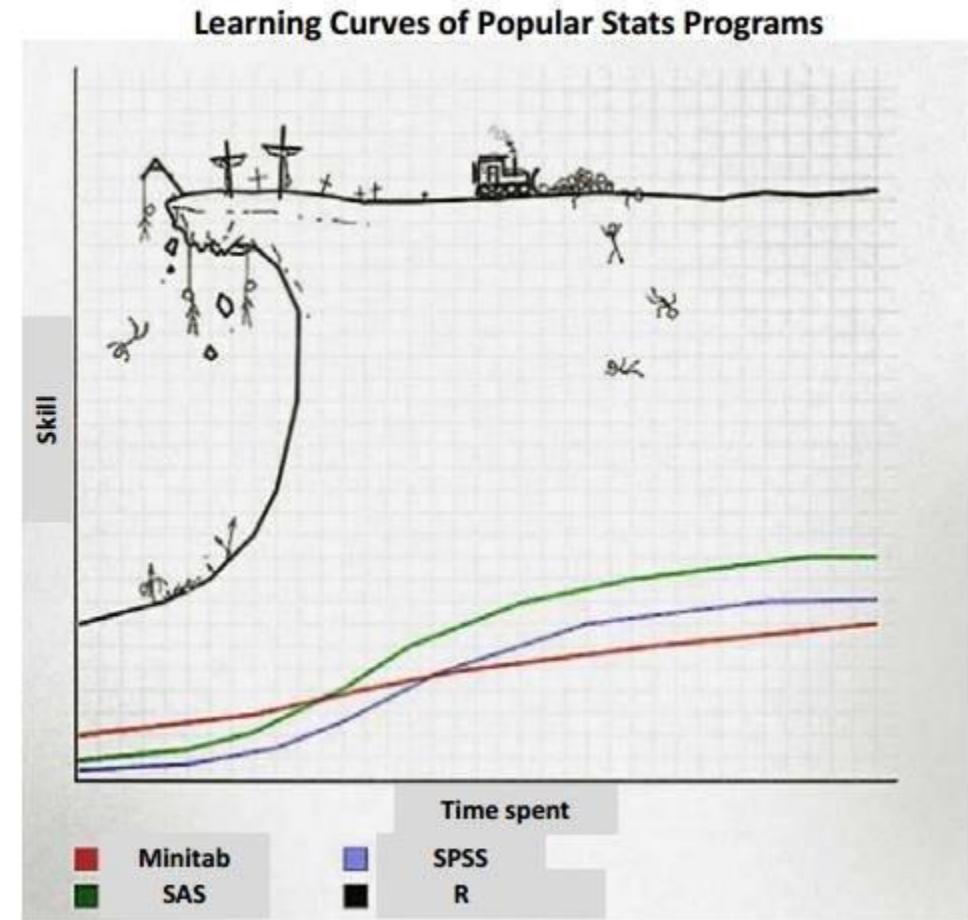
Det er helt greit å *be om hjelp*

Det er helt greit å synes det er *vanskelig og tungt*

Det er helt greit å *ta pauser*

...men du **må** opp på hesten igjen

Og det har aldri vært lettere å lære R



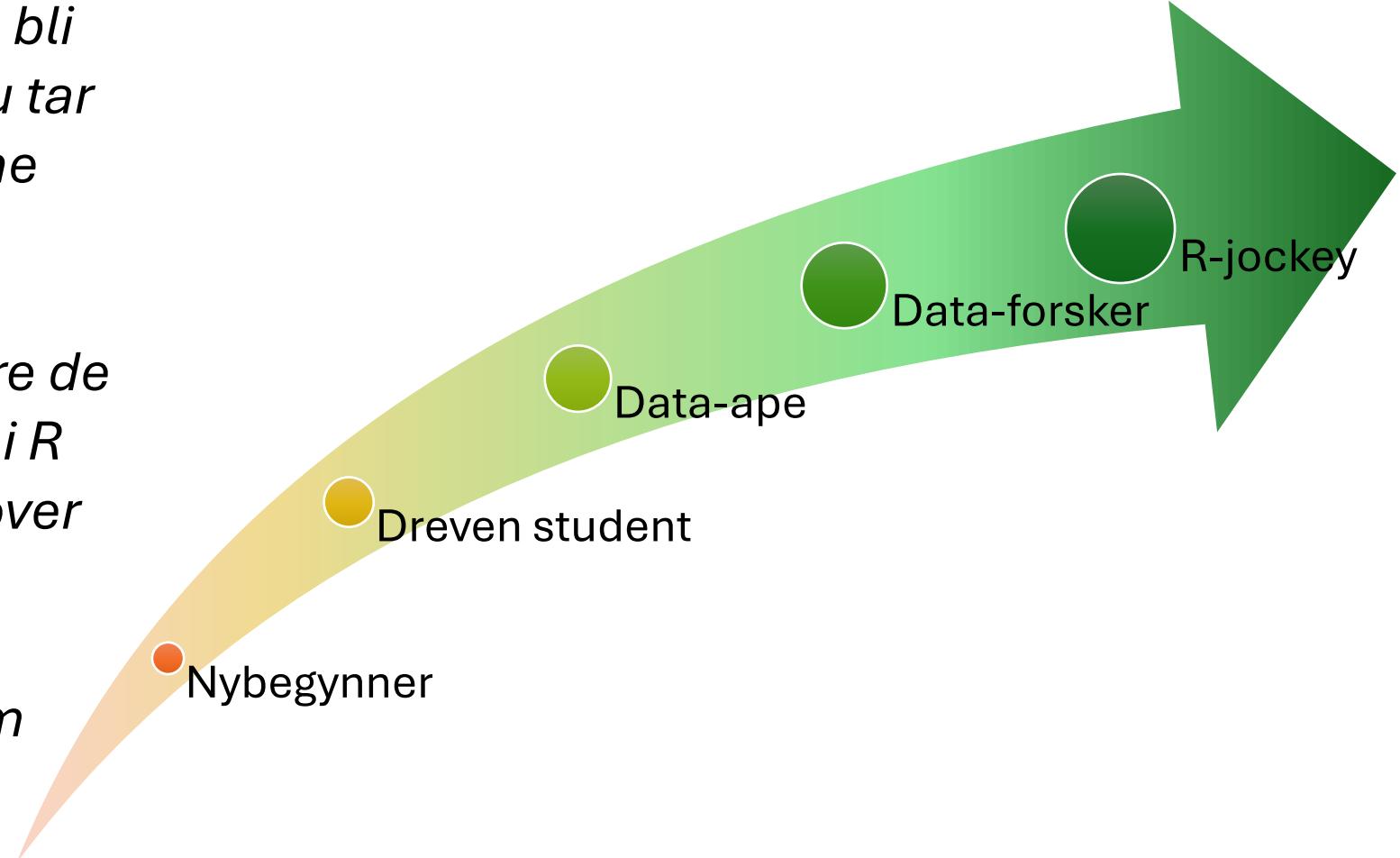
Å lære R anno 2019

Din reise fremover

Du kommer kanskje ikke til å bli verdensmester i dag, men du tar de første konstruktive stegene

Du trenger heller ikke bli verdensmester, men å mestre de grunnleggende ferdighetene i R vil gjøre ditt liv i dag og fremover lettere.

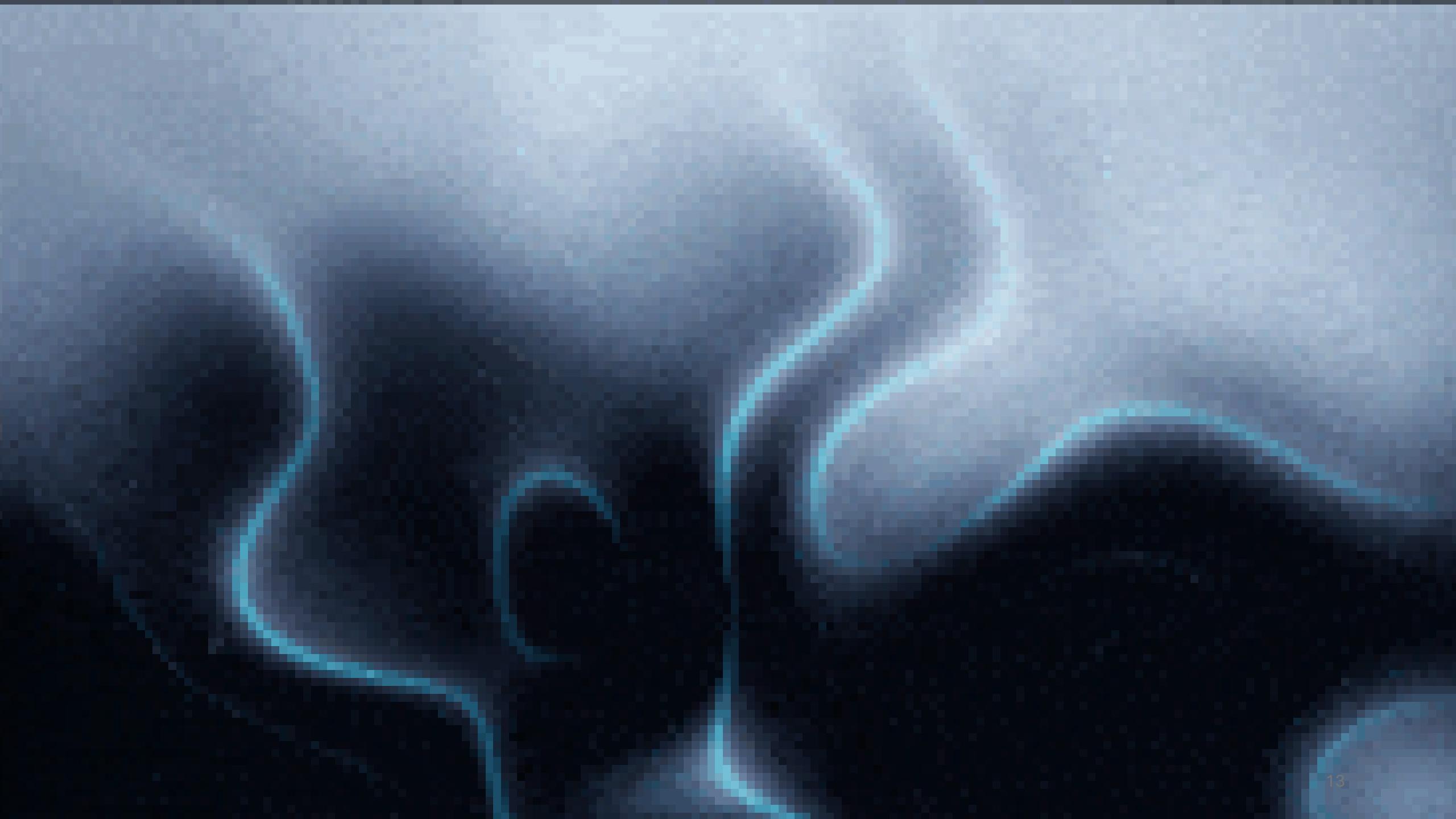
Du kan stoppe på hvilket som helst av disse punktene



Hva skal vi gå gjennom?

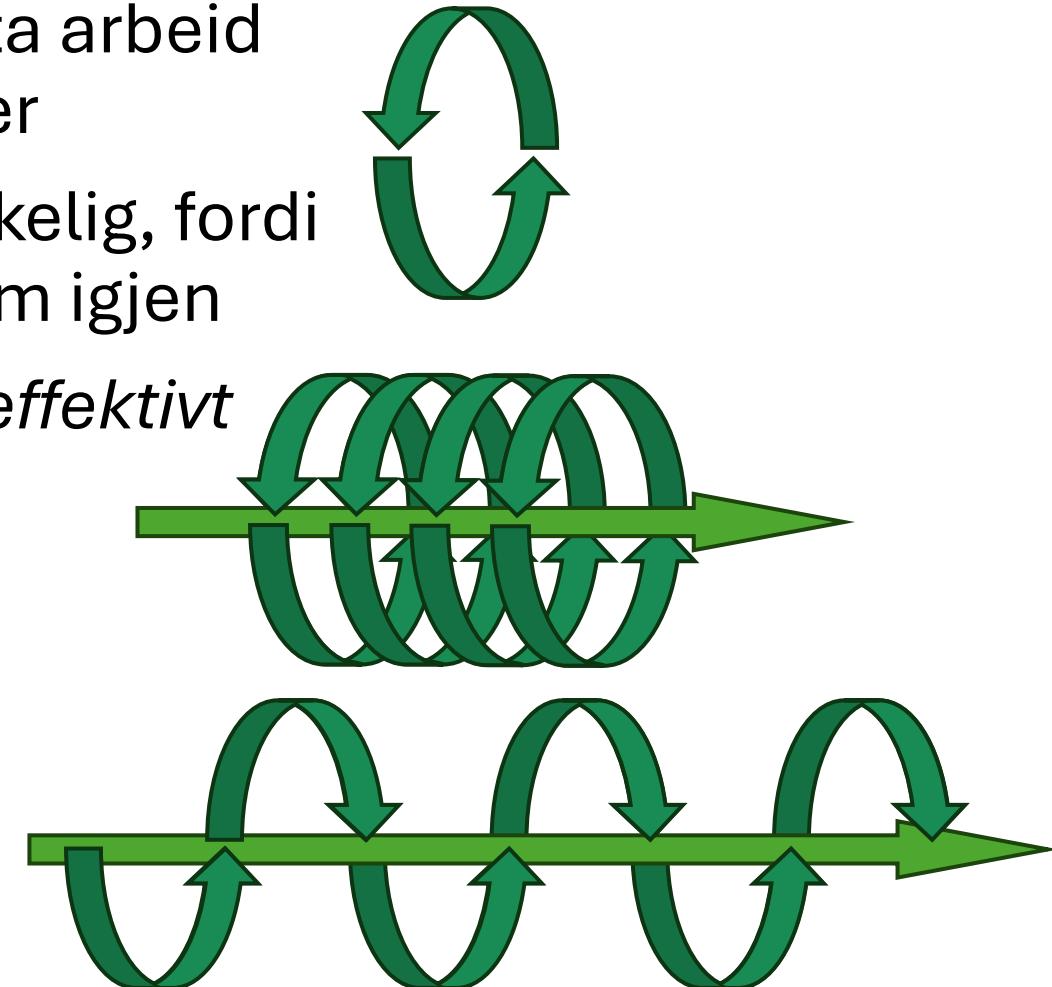
- Statistisk arbeid – *Hvorfor sliter folk med dette?*
aka: ÆÆÆ HVOR F*EN BEGYNNER JEG????
- Arbeidsflyten fra rå data til ferdig analyse
- Hvordan bli selvstendig med dataarbeid
- Den viktigste kunnskapen
- **R vs. Excel;** *hvilken, når og hvorfor?*





Iterativt arbeid

- Noe av det folk synes er tungt med data arbeid er at de ofte føler at ting ikke går forover
- «Iterativt arbeid» kan fort føles langtekkelig, fordi du sitter og gjør «det samme» om og om igjen
- *Men desto mer du gjør det desto mer effektivt blir arbeidet, og ting føles mye lettere*
- *Det er værst i starten*
- ***Omfavn den iterative prosessen, og ting vil bli lettere, nye innsikter skaper nytt arbeid***



Se prosessen

- Hva prøver du å gjøre?
- Hvilke steg er mellom deg og målet?
- Hvor må du starte?
- Hvordan gjør du det første steget?
- Hva trenger du for å gjøre dette?
- Hvordan implementerer du det?



Arbeidsflyt – fra rådata til ferdig analyse

- Datastrukturer – hva er de og hvorfor er de viktige?
- Arbeidsflytdiagram for dataarbeid



Datastrukturer

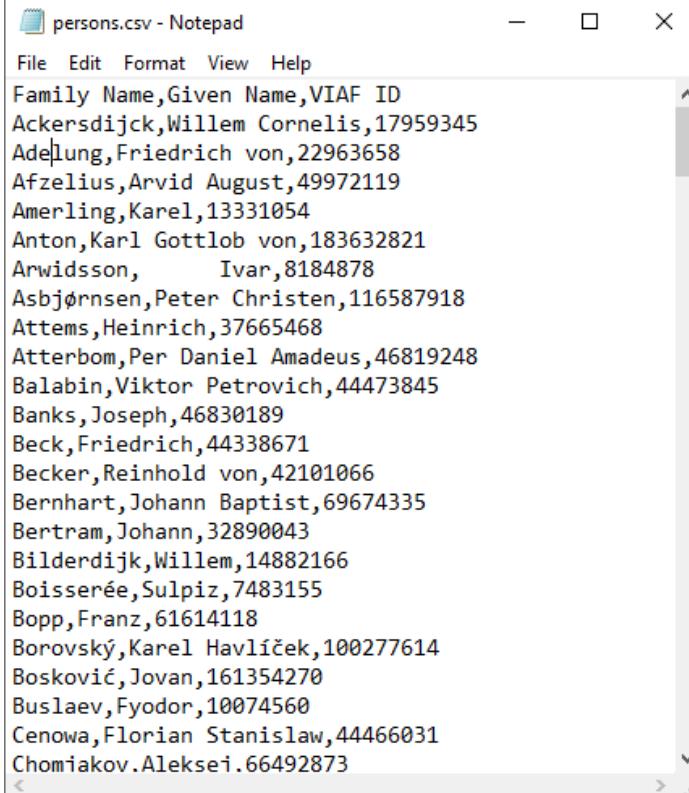
mer organisering; mindre arbeid etter på

- Datastrukturer er forskjellige formater vi kan lagre dataene våre på, for eksempel lister, databaser, matriser, etc.
- De interessante:
 - Filtyper
 - xlsx, csv, json, rds
 - Strukturtyper
 - Lister, vektorer, matriser, array, datasett, og store lister
 - Lange og brede datasett
 - Verdityper
 - Numeriske, tekst, logiske, null, NA, NaN, Inf, -Inf, og faktor



Filtyper

- **XLSX & XLS** – Excels filformat for arbeid m. excel og import til R
- **CSV (comma separated values)** – et tekstfil format for data hvor kolonner er separert med , eller ; og linjedeling (enter / ↵) separerer rader
- **JSON** – Er et mer avansert filformat som støtter mer komplekse datastrukturer som nestede lister
- **RDS** – filformat for lagrede R-datastrukturer, eks, vektorer, lister, datasett, matriser



persons.csv - Notepad

| Family Name | Given Name | VIAF ID |
|-------------|--------------------|-----------|
| Ackersdijck | Willem Cornelis | 17959345 |
| Adelung | Friedrich von | 22963658 |
| Afzelius | Arvid August | 49972119 |
| Amerling | Karel | 13331054 |
| Anton | Karl Gottlob von | 183632821 |
| Arwidsson | Ivar | 8184878 |
| Asbjørnsen | Peter Christen | 116587918 |
| Attems | Heinrich | 37665468 |
| Atterbom | Per Daniel Amadeus | 46819248 |
| Balabin | Viktor Petrovich | 44473845 |
| Banks | Joseph | 46830189 |
| Beck | Friedrich | 44338671 |
| Becker | Reinhold von | 42101066 |
| Bernhart | Johann Baptist | 69674335 |
| Bertram | Johann | 32890043 |
| Bilderdijk | Willem | 14882166 |
| Boisserée | Sulpiz | 7483155 |
| Bopp | Franz | 61614118 |
| Borovský | Karel Havlíček | 100277614 |
| Bosković | Jovan | 161354270 |
| Buslaev | Fyodor | 10074560 |
| Cenowa | Florian Stanislaw | 44466031 |
| Chomiakov | Aleksei | 66492873 |



Verdityper

- **Numeriske**
 - Integer (Int) – Heltall – ex: 5L
 - Doubles (Dbl)
- **Tekst**
 - Character (Chr)
- **Faktor (Fct) –**
Hund (1),katt (2),fugl (3)
- **Logiske (lgl) –** False // True
- **Null –** null verdi
- **NA, NaN –**
Not available / Not a Number
- **Inf, -Inf –** uendelig / -uendelig

Strukturtyper - enkle

- **Vektorer** (vec / c())

- Et sett med verdier i en rekkefølge
 - Ex: c(1,2,3) c(4,2,5,6,1)
c(katt, hund, fugl)
c(FLASE,TRUE,TRUE,FALSE)

- **Matriser**

- 2 dimensjonal datastruktur
 - 1,3,2
3,4,5
7,2,2

- **Array**

- Flerdimensjonal datastruktur
 - 1,3,2 | 2,4,3 | 1,3,4
3,4,5 | 3,3,3 | 5,5,2
7,2,2 | 1,1,3 | 7,7,2



Strukturtyper

Datasett (data.frame)

- Et sett med data med navngitte kolonner (og evt rader)
- *Dette er den aller mest brukte strukturen*
- Om du kan beskrive det med ett datasett, gjør det! Det gjør ting langt lettere
- ... lange og brede datasett



Brede og smale datasett

-

menneskevennlig VS datavennlig

← →
Hvilket datasett synes dere er
enklest å forholde seg til?

a)

| Målerunde | Skog 1 | | Skog 2 | |
|-----------|--------|------|--------|------|
| | C | N | C_ | N_ |
| 1 | 0.98 | 0.49 | 0.96 | 0.19 |
| 2 | 0.59 | 0.78 | 0.93 | 0.53 |
| 3 | 0.52 | 0.46 | 0.51 | 0.55 |
| 4 | 0.23 | 0.44 | 0.69 | 0.62 |
| 5 | 0.31 | 0.88 | 0.31 | 0.77 |
| 6 | 0.86 | 0.05 | 0.20 | 0.92 |
| 7 | 0.89 | 0.63 | 0.50 | 0.17 |
| 8 | 0.76 | 0.03 | 0.33 | 0.72 |
| 9 | 0.58 | 0.70 | 0.28 | 0.59 |
| 10 | 0.51 | 0.86 | 0.80 | 0.84 |
| 11 | 0.11 | 0.02 | 0.89 | 0.51 |
| 12 | 0.99 | 0.57 | 0.60 | 0.93 |
| 13 | 0.44 | 0.19 | 0.80 | 0.67 |
| 14 | 0.05 | 0.74 | 0.12 | 0.20 |

b)

| Skog | Målerunde | C | N |
|--------|-----------|-------------|-------------|
| SKOG 1 | 1 | 0.980270418 | 0.486237488 |
| SKOG 1 | 2 | 0.585451075 | 0.78018173 |
| SKOG 1 | 3 | 0.516075386 | 0.464914187 |
| SKOG 1 | 4 | 0.228802192 | 0.43514731 |
| SKOG 1 | 5 | 0.305263837 | 0.87644531 |
| SKOG 1 | 6 | 0.863699764 | 0.048536789 |
| SKOG 1 | 7 | 0.885743539 | 0.628891245 |
| SKOG 1 | 8 | 0.757093852 | 0.033264009 |
| SKOG 1 | 9 | 0.584263476 | 0.69620073 |
| SKOG 1 | 10 | 0.507787833 | 0.864883943 |
| SKOG 1 | 11 | 0.109070918 | 0.019942794 |
| SKOG 1 | 12 | 0.987490202 | 0.570049803 |
| SKOG 1 | 13 | 0.435433534 | 0.185244507 |
| SKOG 1 | 14 | 0.046132709 | 0.74213929 |
| SKOG 2 | 1 | 0.955629467 | 0.185017096 |
| SKOG 2 | 2 | 0.931120708 | 0.531618252 |
| SKOG 2 | 3 | 0.511481439 | 0.550389962 |
| SKOG 2 | 4 | 0.686821505 | 0.624869738 |
| SKOG 2 | 5 | 0.313919028 | 0.774259633 |
| SKOG 2 | 6 | 0.1984879 | 0.915271407 |
| SKOG 2 | 7 | 0.499677866 | 0.166899884 |
| SKOG 2 | 8 | 0.333013704 | 0.717435115 |
| SKOG 2 | 9 | 0.277863879 | 0.593039664 |
| SKOG 2 | 10 | 0.80250597 | 0.838436348 |
| SKOG 2 | 11 | 0.888249711 | 0.513618114 |
| SKOG 2 | 12 | 0.60071996 | 0.927093361 |
| SKOG 2 | 13 | 0.799504677 | 0.671489644 |
| SKOG 2 | 14 | 0.117707284 | 0.204523501 |



Brede og smale datasett, menneskevennlig VS datavennlig

- **Brede datasett** er **lettere å lese** og fylle inn, om du skal presentere dataene for noen (f.eks. i en avhandling) eller skrive inn dataene manuelt bør du bruke brede datasett
- **Smale datasett** er **nødvendig for nesten alt av databehandling.** For de aller fleste arbeidsoppgaver må du *via* lange datasett.
- *Så hva gjør vi?*



A promotional image for HBO Max. On the right side, a man with dark hair and a beard is peeking out from behind a grey sofa. He is wearing a black zip-up hoodie over a white t-shirt. The background is a dimly lit room with a staircase visible on the right. The HBO Max logo is in the top right corner.

HBOmax

PIVOT!

Pivoting

- Å gå fra brede til smale eller smale til brede
- Enklest utført i R
 - Tidyverse → Tidyr → pivot_wider() og pivot_longer()
- Vi bruker oftere pivot_longer

| country | year | cases |
|----------|------|-------|
| Angola | 1999 | 800 |
| Angola | 2000 | 750 |
| Angola | 2001 | 925 |
| Angola | 2002 | 1020 |
| India | 1999 | 20100 |
| India | 2000 | 25650 |
| India | 2001 | 26800 |
| India | 2002 | 27255 |
| Mongolia | 1999 | 450 |
| Mongolia | 2000 | 512 |
| Mongolia | 2001 | 510 |
| Mongolia | 2002 | 586 |

| country | 1999 | 2000 | 2001 | 2002 |
|----------|-------|-------|-------|-------|
| Angola | 800 | 750 | 925 | 1020 |
| India | 20100 | 25650 | 26800 | 27255 |
| Mongolia | 450 | 512 | 510 | 586 |

Pivot data wider

```
data %>%
  pivot_wider(
    names_from = "year",
    values_from = "cases"
  )
```

| country | 1999 | 2000 | 2001 | 2002 |
|----------|-------|-------|-------|-------|
| Angola | 800 | 750 | 925 | 1020 |
| India | 20100 | 25650 | 26800 | 27255 |
| Mongolia | 450 | 512 | 510 | 586 |

Pivot data longer

```
data %>%
  pivot_longer(
    cols = 1999:2002,
    names_to = "year",
    values_to = "cases"
  )
```

| country | year | cases |
|----------|------|-------|
| Angola | 1999 | 800 |
| Angola | 2000 | 750 |
| Angola | 2001 | 925 |
| Angola | 2002 | 1020 |
| India | 1999 | 20100 |
| India | 2000 | 25650 |
| India | 2001 | 26800 |
| India | 2002 | 27255 |
| Mongolia | 1999 | 450 |
| Mongolia | 2000 | 512 |
| Mongolia | 2001 | 510 |
| Mongolia | 2002 | 586 |

Se på datasett

Base R

- **DF navnet** – *standard visning av datasett*
- **View** – *åpne datasettet*
- **Summary** – *oppsummering av datasettet*
- **Names** – *kolonne navn*
- **Unique** – *unike verdier*

Tidyverse

- **Tibble** – *mer ryddig visning av datasett*
- **Glimpse** – *annen ryddig visning av datasett*
- **Slice**
 - **Slice_head** – *de første verdiene*
 - **Slice_tail** – *de siste verdiene*
 - **Slice_max** – *topp verdiene*
 - **Slice_min** – *bunn verdiene*
 - **Slice_sample** – *tilfeldige verdier*
- **Distinct** – *unike verdier for gitte rader*



eksempler

```
> iris %>% summary()
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
Min.   :4.300  Min.   :2.000  Min.   :1.000  Min.   :0.100 setosa   :50
1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300 versicolor:50
Median  :5.800  Median :3.000  Median :4.350  Median :1.300 virginica:50
Mean    :5.843  Mean   :3.057  Mean   :3.758  Mean   :1.199
3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800
Max.    :7.900  Max.   :4.400  Max.   :6.900  Max.   :2.500
```

```
> iris %>% slice_head(n=5)
#> #> #> #> #>
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |

```
> iris %>% slice_max(Sepal.Length, n=5)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          7.9        3.8       6.4        2.0  virginic
2          7.7        3.8       6.7        2.2  virginic
3          7.7        2.6       6.9        2.3  virginic
4          7.7        2.8       6.7        2.0  virginic
5          7.7        3.0       6.1        2.3  virginic
```

```
> iris %>% slice_sample(n=5)
#> #> #> #> #>
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|------------|
| 1 | 5.1 | 3.8 | 1.6 | 0.2 | setosa |
| 2 | 5.6 | 3.0 | 4.5 | 1.5 | versicolor |
| 3 | 6.4 | 2.8 | 5.6 | 2.2 | virginica |
| 4 | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| 5 | 5.6 | 2.9 | 3.6 | 1.3 | versicolor |



Strukturtyper – Lister / Store lister

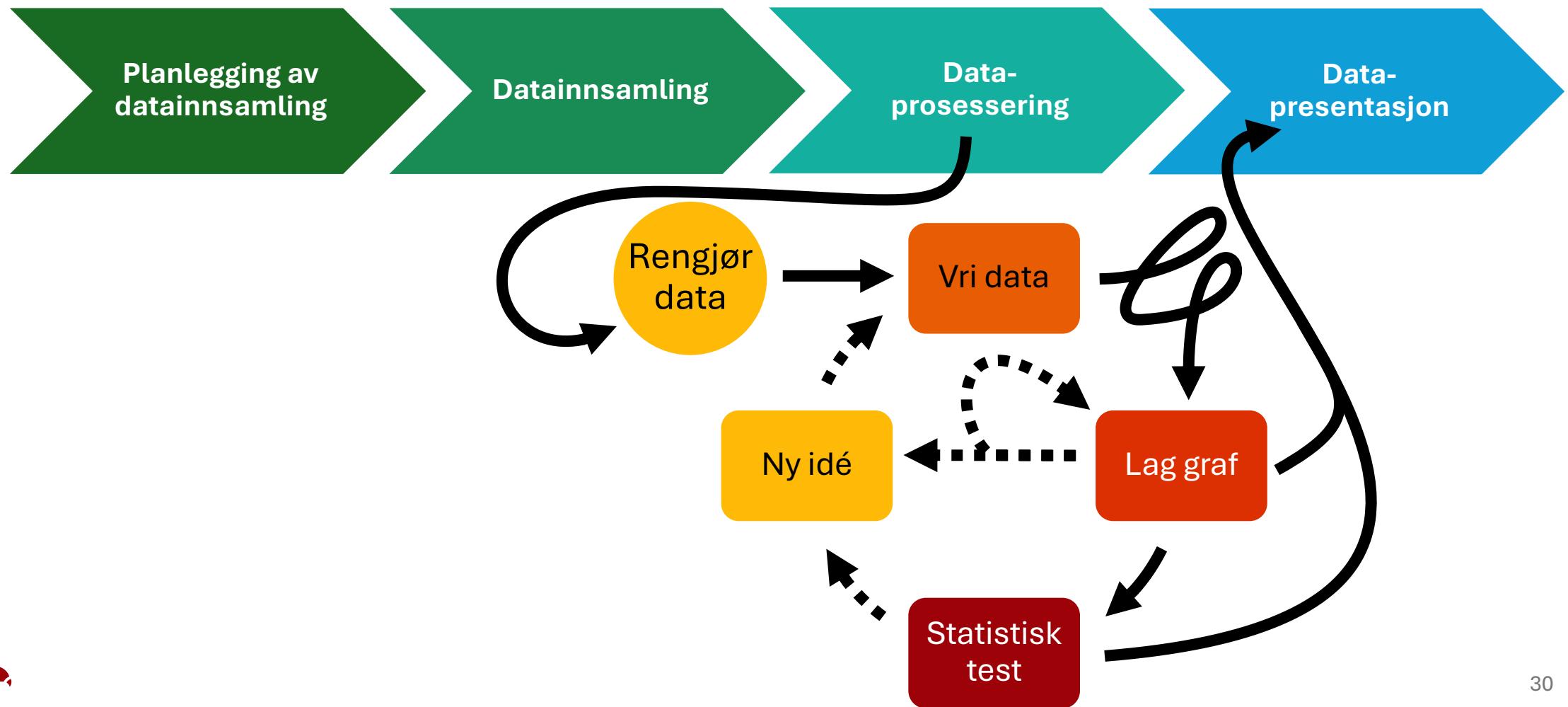
- Lister er tillater nestede data
 - Dvs at du har mer enn et objekt i samme objektet
- Hvis du jobber med offentlige datasett kan det du møter disse via *.JSON* fil-formatet
- Det kan være nyttig å kombinere flere dataset til en liste i noen tilfeller
- Personlig har jeg brukt lister for å bygge komplekse objekter for analyser og for personlige prosjekter



```
list(  
  list(  
    author = "Test Author",  
    author_metadata =  
      list(  
        Tags = c(  
          "Test"  
        ),  
        Year_of_birth = 1900,  
        Year_of_death = 2000,  
        Updated = "2024.11.19"  
      ),  
    text =  
      list(  
        list(  
          title = "I Know Why the Caged Bird Sings",  
          ISBN = 21982377217,  
          publication_year = 1969,  
          genre = "Autobiography",  
          read_on = "Kindle",  
          text_type = "Book",  
          tags =  
            list(  
              ),  
            quotes =  
              list(  
                list(  
                  Qoute_text = "Lorem Ipsum",  
                  Chapter = "Lorem Lorem",  
                  PageNumber = 1  
                )  
              )  
            )  
          )  
        )  
      )  
    )  
  )  
)
```



Arbeidsflyten for dataarbeid



Hvordan gå frem for å finne løsningen

Finn ut hva du egentlig prøver å gjøre?

1. Hvilket spørsmål er det du prøver å svare på egt?
2. Hvordan svarer du på det spørsmålet?
 - Graf eller statistiske test? (veldig ofte begge deler)
 - Hvilken test? > se flow chart [\[https://statsandr.com/blog/what-statistical-test-should-i-do/\]](https://statsandr.com/blog/what-statistical-test-should-i-do/)
 - Hvilken graf? > se data-to-viz [\[https://www.data-to-viz.com/\]](https://www.data-to-viz.com/)
3. Hvordan må datasettet se ut for å gjøre dette?
4. Hvordan former du datasettet slik?
 1. Finn funksjonen
 2. Sjekk dokumentasjonen til funksjonen [`?Func()`]
 3. Google problemet ditt
 4. Spør AI **OG FINN UT HVA DET DU HAR COPY/PASTET FAKTISK GJØR**
(eller helst `github copilot` for kode via `VS code`)
 5. **Spør gjerne r-klubben!**



Datainnsamling

Før datainnsamling

- Lag en plan for datastrukturen og lag en mal som passer i Word eller Excel
- Om du skal ut i felt lag ferdige papirmaler med notat felt og alle felt du tror du kan kunne trenge siden & print ut.

Under datainnsamling

- Fyll ut med godt leseelige tall og skriv notater om alt som kan virke interresant.
- ***Aldri, aldri aldri aldri, tenk at du skal huske noe selv***
- ***Behold alltid originalene***



Eksempler

| | Rad 1 | Rad 2 | Rad 3 |
|-----------|-------|-------|-------|
| Kolonne 1 | 1 | 4 | |
| Kolonne 2 | 2 | 5 | |
| Kolonne 3 | 3 | 6 | |

Dato:

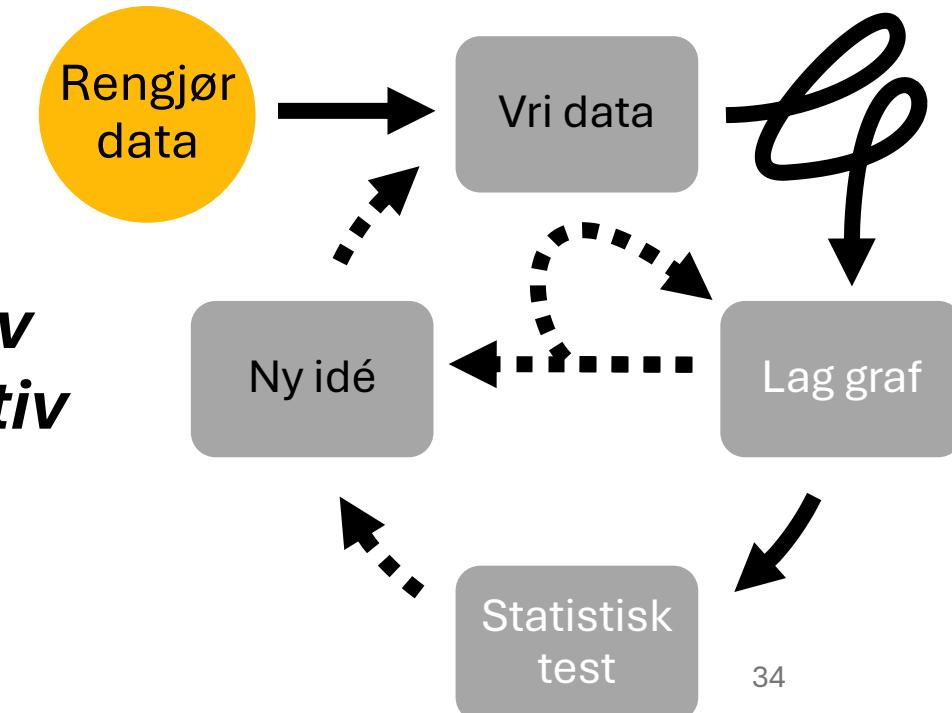
Målemetode:

Sted:



Data rengjøring

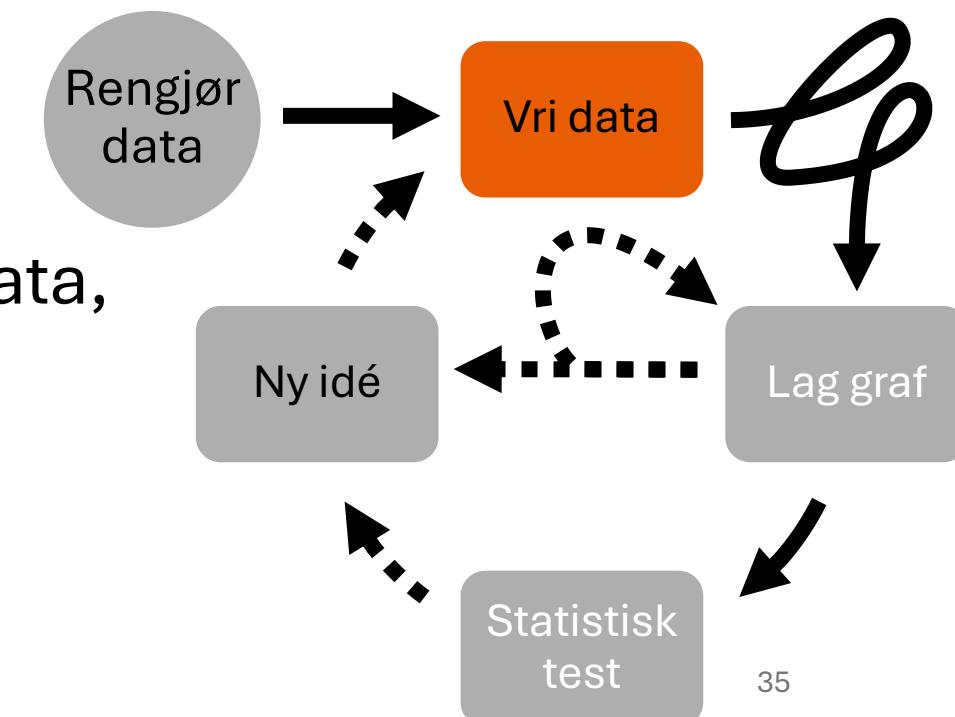
- Data rengjøring er det nødvendige steget før vi kan gjøre noe ordentlig
- Vi må sjekke at alle dataene som skal være like **er** like
- Dette er ofte noe vi gjør i Excel
- **Lag en kopi av dataene dine og gjør rengjøringen i denne versjonen av filen**
- Rengjøring i **Excel** er i hovedsak **destruktiv**
Rengjøring i **R** er i hovedsak **IKKE destruktiv**



Data wrangling

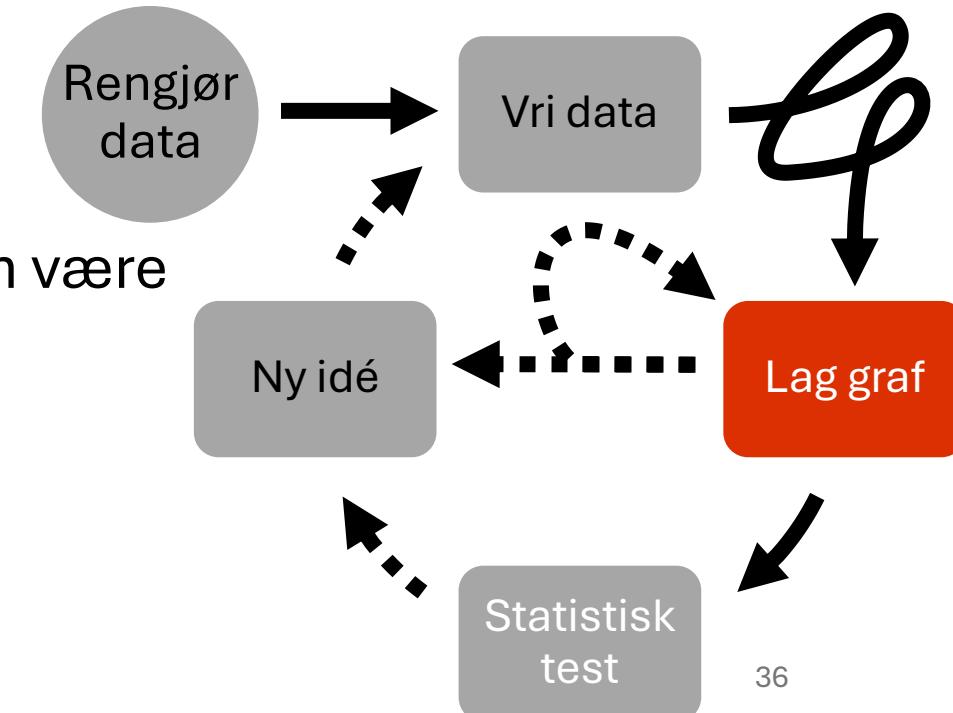


- Ofte må vi vri, vende, dele opp og rekombinere, splitte tekst og m.m.
- Dette er den “ville vesten” i data behandling, og hvor du føler deg kulest når du får noe til
- Dette er en ferdighet som du må bygge over tid.
- F.eks. dele tekst, flytte data, kombinere data, lage unike IDer, velge kolonner, lage nye kolonner, endre kolonner, + mye mer
- *Tidyverse, tidyverse, tidyverse*



Grafer – som innsynsgiver

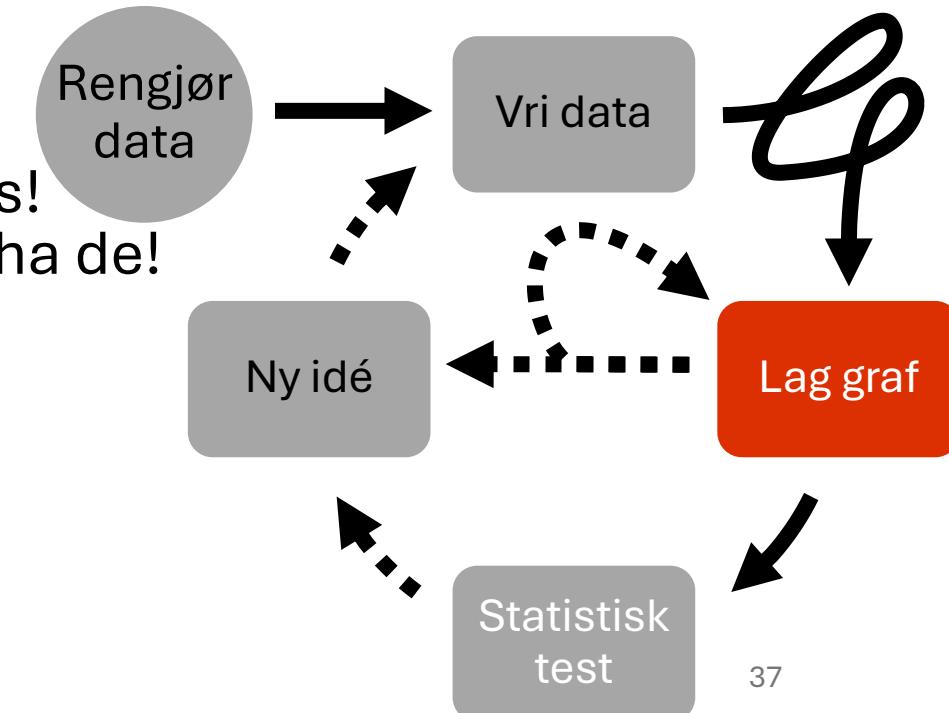
- Når du har et *bruksbart* datasett begynn å lage grafer.
- Grafene kommer til å informere deg bedre om hva du bør gjøre etter dette.
- Mange liker å starte her i Excel, men dette er en fallgruve...
 - hvorfor? Det er *mye* enklere å bytte og trikse med grafer i R enn Excel. Å endre graf type kan være så enkelt som å endre ett ord i koden din!
- ggplot >> base R



Grafer – som innsynsgiver

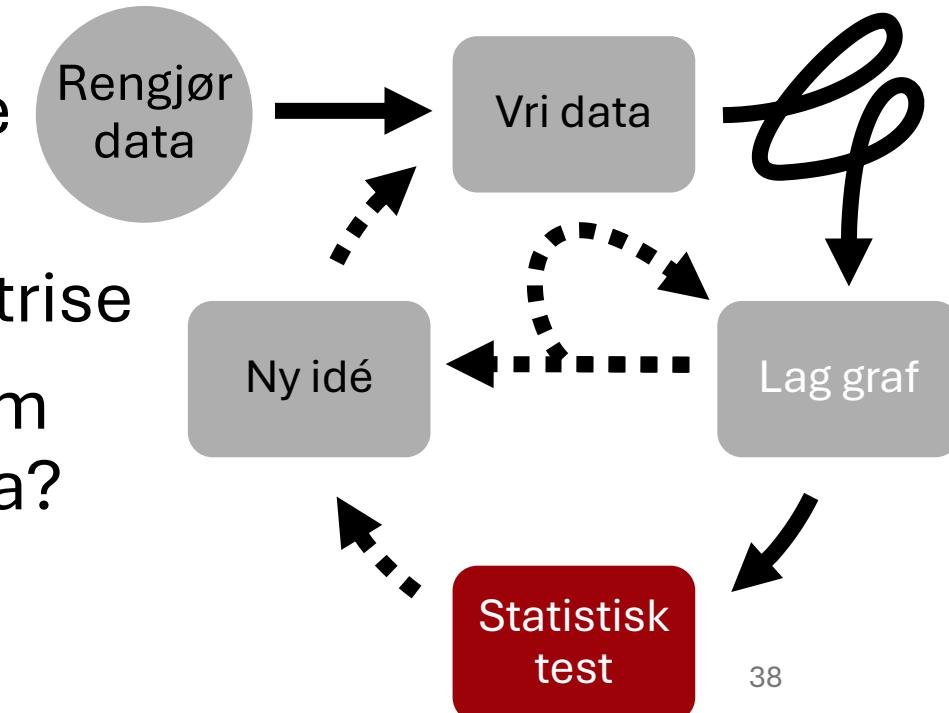
- Hvilke grafer?
 - Distribusjoner: Density // Histogram
 - Grupper: Boxplot (evt. violin + boxplot)
 - 2 numeriske: Scatter plot
 - Tidsserier: Linje
- Disse grafene er «standard» grafene, om du skal bruke noen andre bør dette begrunnes! Det finnes selvfølgelig gode grunner, men du bør ha de!

<https://www.data-to-viz.com/>
- en kjempe ressurs for å finne riktig graf



Statistiske tester

- Statistiske tester er veldig viktige.
- Som oftest lar de deg si at de trendene som er i dataene dine ikke er tilfeldige
- Noen statistiske tester er bygd slik at man kan finne trender i dataene man ellers ikke kunne se! (Ordinasjon, Maskinlæring)
- Mange variabler? Prøv en korrelasjons matrise
- Men å velge type test og vite alt man må om testen kan være vanskelig, så hva gjør vi da?

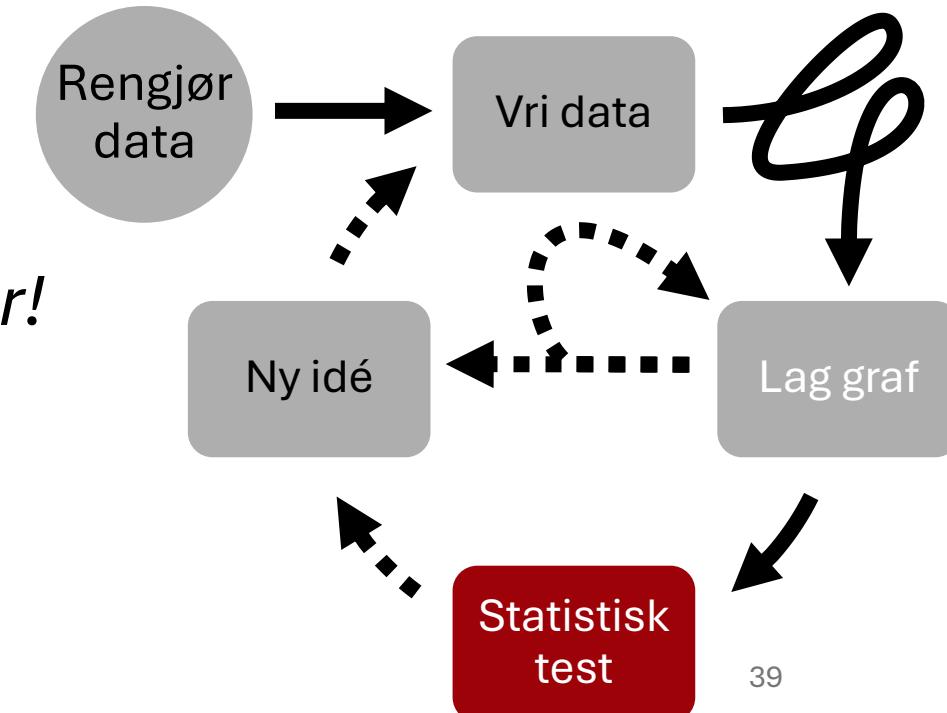


Statistiske tester

1. Spør veileder om det er noen spesiell test du bør gjøre
2. Bruk flyt-diagrammet for å | [<https://statsandr.com/blog/what-statistical-test-should-i-do/>]

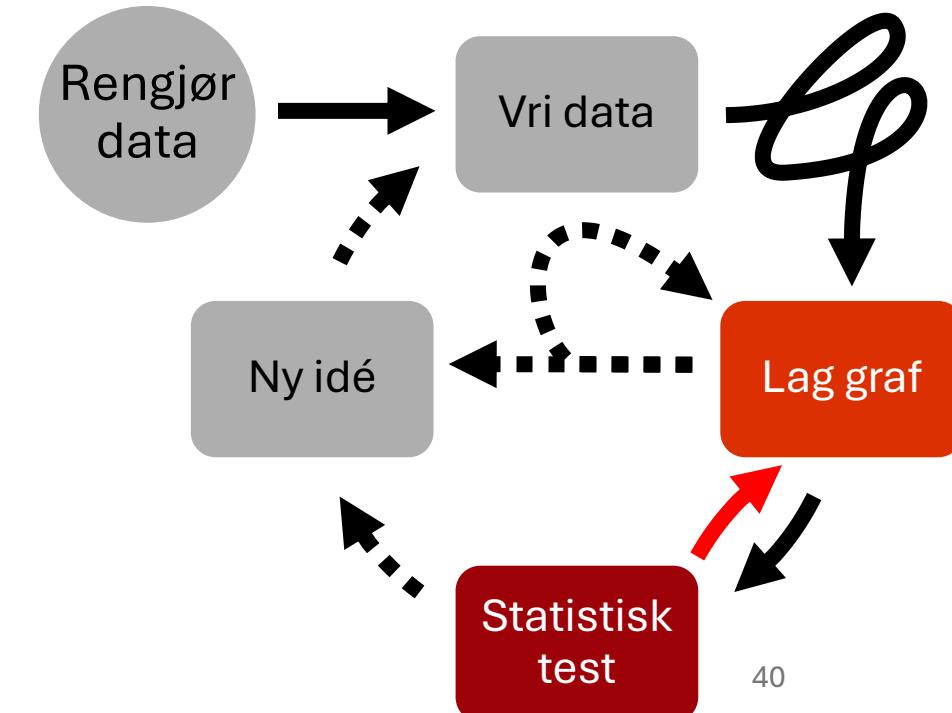
 1. Velge test
 2. Finne info om testen

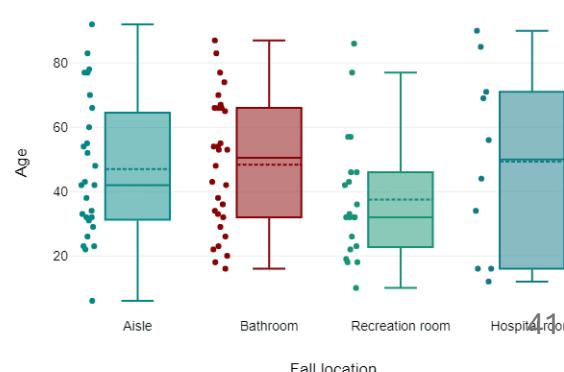
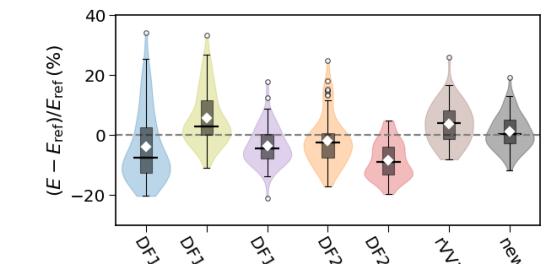
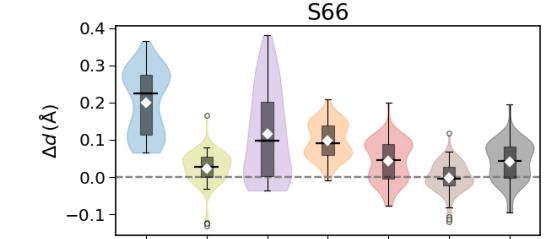
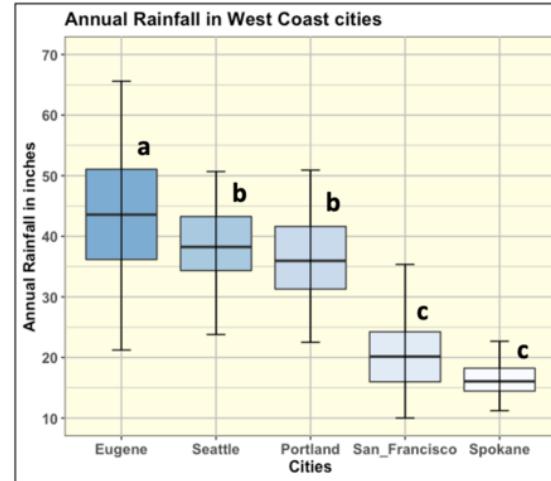
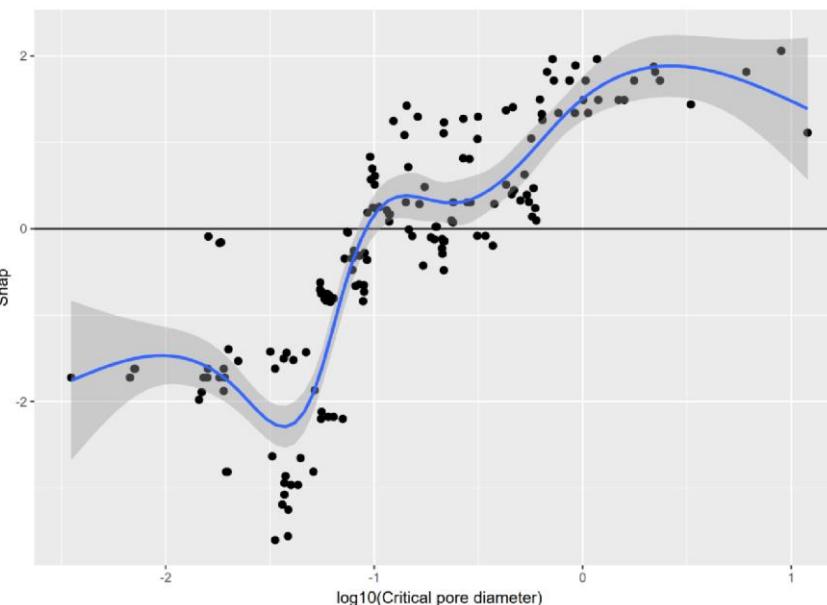
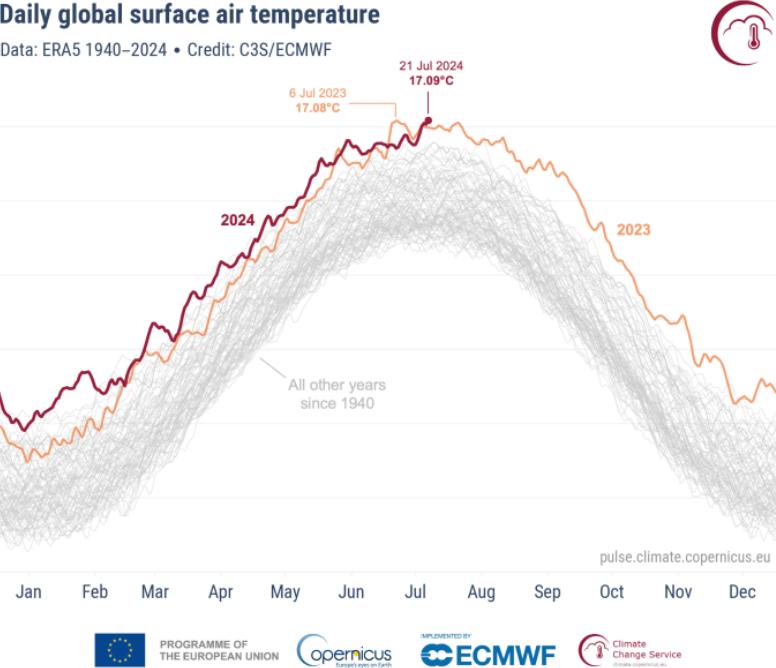
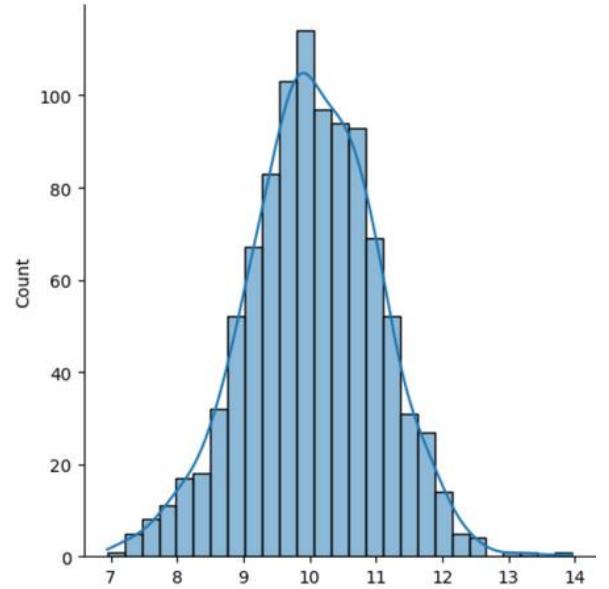
 - Du trenger ikke kunne alle testene!
Men du må lese deg opp på den du trenger!



Grafer – til datapresentasjon

- Grafer er GOD «story telling», det er langt enklere å forstå et datasett eller en analyse gjennom en graf enn via tall og tabeller!
- Men de er aller aller best sammen med statistiske tester
- *Graf & test > graf > test > tabell > tall*
- Spør gjerne en fag-kyndig graf-snekker om hjelp med å lage best mulige grafer (R-klubben)





Grafer – til datapresentasjon

Hva er en god graf?

- **Tydelig gjør historien dataene faktisk forteller; dette avhenger av testene du har gjort**
- 0-punktet er ikke nødvendigvis viktig
- **Sannferdig representasjon er viktig; statistiske tester er viktig!**
- Forskningsområdet ditt, typen data, m.fl. må tas til betrakting!

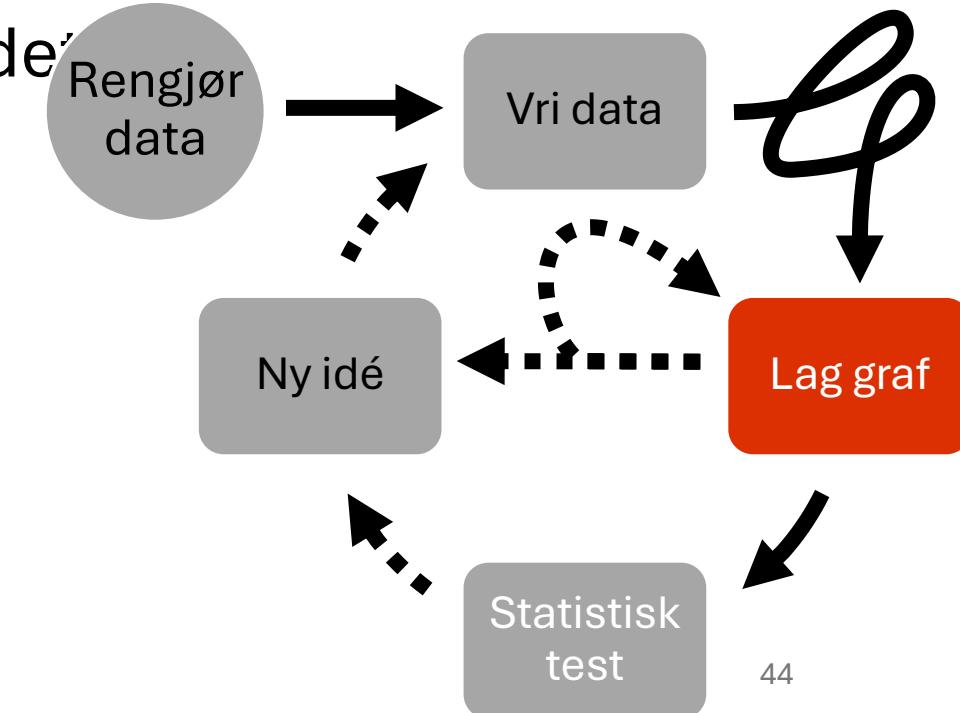




Grafer – til datapresentasjon

Hvordan få høy kvalitet på grafene i ferdig produkt?

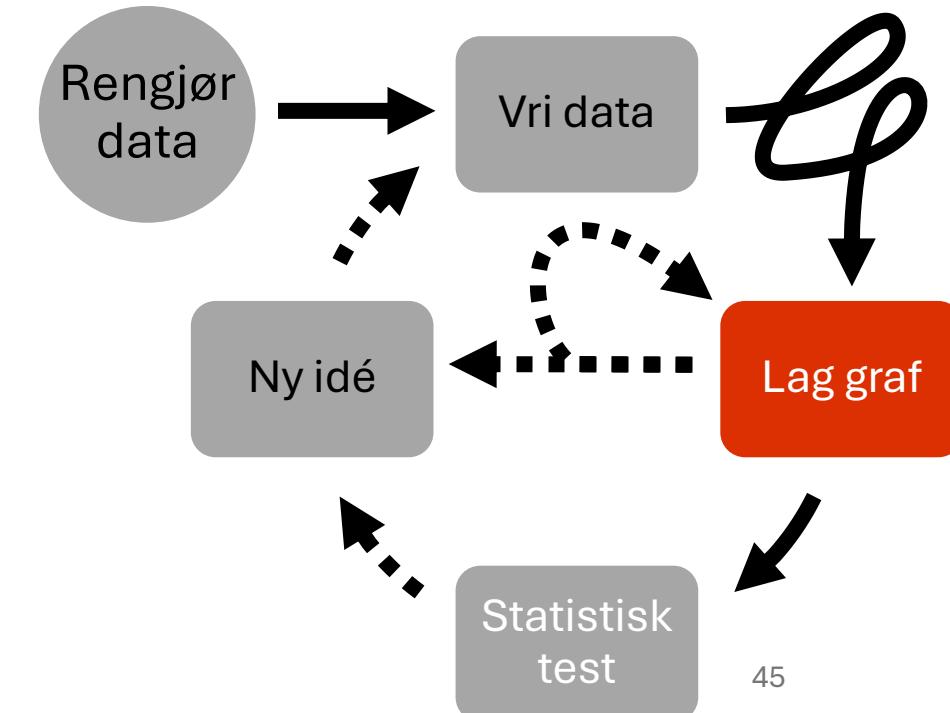
- Eksporter som PDF
- Åpne PDFen i Zotero
- Merker med «select area» og kopier området
- Dette gir deg grafer i ***print-kvalitet!***



Grafer – til datapresentasjon

Hvordan velge bra farger

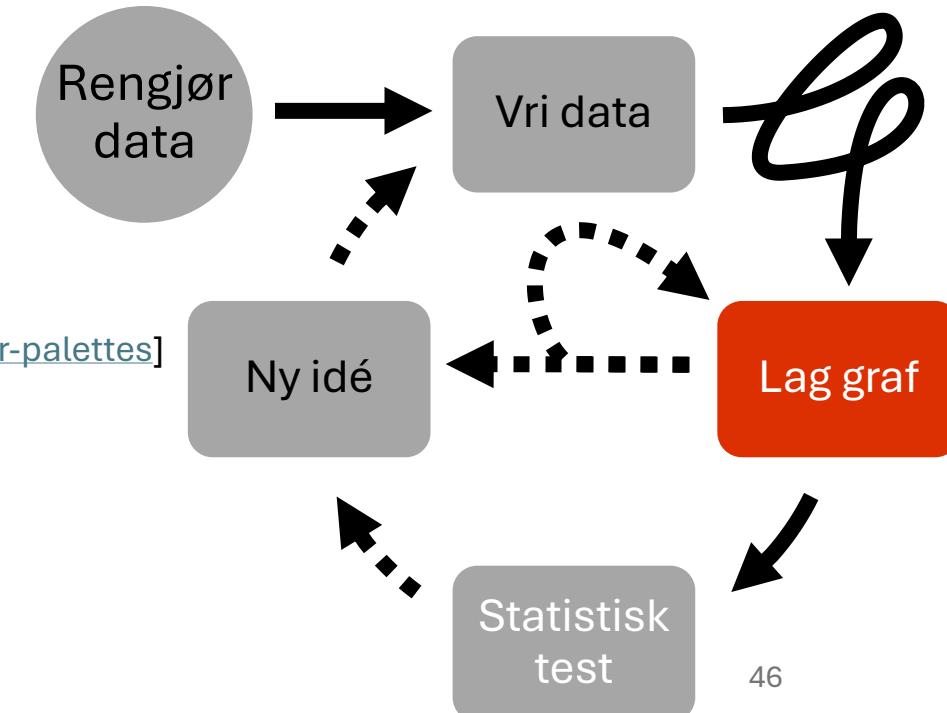
- Fargene dine burde være **fargeblindhetsvennlige** *uavhengig om noen som er fargeblinde skal se på de!*
- Hvorfor? Enkelt sagt: **Print**
 - Kontraster blir dårligere i print
 - Du vil kanskje også printe i sort/hvitt når du skal lage «lese-kopier»



Grafer – til datapresentasjon

Hvordan velge bra farger

- Velg farger som representerer det du beskriver!
 - Farge skaler må gjerne gå fra rødt til blått m nøytralt 0 punkt
 - Er det en farge som relaterer til det ditt forskningsområde? *Bruk den*
- Ferdig lagde fargeskaler:
 - Virdis, andre paletter [<https://github.com/EmilHvitfeldt/r-color-palettes>]
 - Standard farger i R [<https://rpubs.com/kylebrown/r-colors>]
 - Lage egne farge sett? [<https://coolors.co/>]



Grafer – til datapresentasjon

| | | | | | | | | | | |
|----------------|-----------------|--------------|--------|-------------|--------|-----------------|-------------------|----------------|-------------|--------------|
| white | coral4 | deepskyblue | gray28 | gray88 | grey40 | grey100 | lightpink2 | mistyrose2 | plum | slategray2 |
| aliceblue | cornflowerblue | deepskyblue1 | gray29 | gray89 | grey41 | honeydew | lightpink3 | mistyrose3 | plum1 | slategray3 |
| antiquewhite | cornsilk | deepskyblue2 | gray30 | gray90 | grey42 | honeydew1 | lightpink4 | mistyrose4 | plum2 | slategray4 |
| antiquewhite1 | cornsilk1 | deepskyblue3 | gray31 | gray91 | grey43 | honeydew2 | lightsalmon1 | moccasin | plum3 | slategray |
| antiquewhite2 | cornsilk2 | deepskyblue4 | gray32 | gray92 | grey44 | honeydew3 | lightsalmon2 | navajowhite1 | powderblue | snow |
| antiquewhite3 | cornsilk3 | dimgray | gray33 | gray93 | grey45 | honeydew4 | lightsalmon3 | navajowhite2 | purple | snow1 |
| antiquewhite4 | cornsilk4 | dimgray | gray34 | gray94 | grey46 | hotpink | lightsalmon4 | navajowhite3 | purple1 | snow2 |
| aquamarine | cyan | dodgerblue | gray35 | gray95 | grey47 | hotpink1 | lightsalmon5 | navajowhite4 | purple2 | snow3 |
| aquamarine1 | cyan1 | dodgerblue1 | gray36 | gray96 | grey48 | hotpink2 | lightsalmon6 | navajowhite5 | purple3 | snow4 |
| aquamarine2 | cyan2 | dodgerblue2 | gray37 | gray97 | grey49 | hotpink3 | lightseagreen | navajowhite6 | springgreen | |
| aquamarine3 | cyan3 | dodgerblue3 | gray38 | gray98 | grey50 | hotpink4 | lightskyblue1 | navyblue | purple4 | springgreen1 |
| aquamarine4 | cyan4 | dodgerblue4 | gray39 | gray99 | grey51 | indianred1 | lightskyblue2 | oldlace | red | springgreen2 |
| azure | darkblue | firebrick | gray40 | gray100 | grey52 | indianred2 | lightskyblue3 | olivedrab | red1 | springgreen3 |
| azure1 | darkcyan | firebrick1 | gray41 | green | grey53 | indianred3 | lightskyblue4 | olivedrab1 | red2 | springgreen4 |
| azure2 | darkgoldenrod | firebrick2 | gray42 | green1 | grey54 | indianred4 | lightslategray | olivedrab2 | red3 | steelblue |
| azure3 | darkgoldenrod1 | firebrick3 | gray43 | green2 | grey55 | indianred4 | lightslategray1 | olivedrab3 | red4 | steelblue1 |
| azure4 | darkgoldenrod2 | firebrick4 | gray44 | green3 | grey56 | ivory | lightslategray2 | olivedrab4 | rosybrown | steelblue2 |
| beige | darkgoldenrod3 | floralwhite | gray45 | green4 | grey57 | ivory1 | lightsteelblue | orange | rosybrown1 | steelblue3 |
| bisque | darkgoldenrod4 | forestgreen | gray46 | greenyellow | grey58 | ivory2 | lightsteelblue1 | orange1 | rosybrown2 | steelblue4 |
| bisque1 | darkgray | gainsboro | gray47 | grey | grey59 | ivory3 | lightsteelblue2 | orange2 | rosybrown3 | tan |
| bisque2 | darkgreen | ghostwhite | gray48 | grey0 | grey60 | ivory4 | lightsteelblue3 | orange3 | rosybrown4 | tan1 |
| bisque3 | darkgrey | gold | gray49 | grey1 | grey61 | khaki | lightsteelblue4 | orange4 | royalblue | tan2 |
| bisque4 | darkkhaki | gold1 | gray50 | grey2 | grey62 | khaki1 | lightyellow | orangered | royalblue1 | tan3 |
| black | darkmagenta | gold2 | gray51 | grey3 | grey63 | khaki2 | lightyellow1 | orangered1 | royalblue2 | tan4 |
| blanchedalmond | darkolivegreen | gold3 | gray52 | grey4 | grey64 | khaki3 | lightyellow2 | orangered2 | royalblue3 | thistle |
| blue | darkolivegreen1 | gold4 | gray53 | grey5 | grey65 | khaki4 | lightyellow3 | orangered3 | royalblue4 | thistle1 |
| blue1 | darkolivegreen2 | goldenrod | gray54 | grey6 | grey66 | lavender | lightyellow4 | orangered4 | saddlebrown | thistle2 |
| blue2 | darkolivegreen3 | goldenrod1 | gray55 | grey7 | grey67 | lavenderblush | limegreen | orchid | salmon | thistle3 |
| blue3 | darkolivegreen4 | goldenrod2 | gray56 | grey8 | grey68 | lavenderblush1 | linen | orchid1 | salmon1 | thistle4 |
| blue4 | darkorange | goldenrod3 | gray57 | grey9 | grey69 | lavenderblush2 | magenta | orchid2 | salmon2 | tomato |
| blueviolet | darkorange1 | goldenrod4 | gray58 | grey10 | grey70 | lavenderblush3 | magenta1 | orchid3 | salmon3 | tomato1 |
| brown | darkorange2 | gray | gray59 | grey11 | grey71 | lavenderblush4 | magenta2 | orchid4 | salmon4 | tomato2 |
| brown1 | darkorange3 | gray0 | gray60 | grey12 | grey72 | lawngreen | magenta3 | palegoldenrod | sandybrown | tomato3 |
| brown2 | darkorange4 | gray1 | gray61 | grey13 | grey73 | lemonchiffon | magenta4 | palegreen | seagreen | tomato4 |
| brown3 | darkorchid | gray2 | gray62 | grey14 | grey74 | lemonchiffon1 | maroon | palegreen1 | seagreen1 | turquoise |
| brown4 | darkorchid1 | gray3 | gray63 | grey15 | grey75 | lemonchiffon2 | maroon1 | palegreen2 | seagreen2 | turquoise1 |
| burlywood | darkorchid2 | gray4 | gray64 | grey16 | grey76 | lemonchiffon3 | maroon2 | palegreen3 | seagreen3 | turquoise2 |
| burlywood1 | darkorchid3 | gray5 | gray65 | grey17 | grey77 | lemonchiffon4 | maroon3 | palegreen4 | seagreen4 | turquoise3 |
| burlywood2 | darkorchid4 | gray6 | gray66 | grey18 | grey78 | lightblue | maroon4 | paleturquoise | seashell | turquoise4 |
| burlywood3 | darkred | gray7 | gray67 | grey19 | grey79 | lightblue1 | mediumaquamarine | paleturquoise1 | seashell1 | violet |
| burlywood4 | darksalmon | gray8 | gray68 | grey20 | grey80 | lightblue2 | mediumblue | paleturquoise2 | seashell2 | violetred |
| cadetblue | darkseagreen | gray9 | gray69 | grey21 | grey81 | lightblue3 | mediumblue1 | paleturquoise3 | seashell3 | violetred1 |
| cadetblue1 | darkseagreen1 | gray10 | gray70 | grey22 | grey82 | lightblue4 | mediumblue2 | paleturquoise4 | seashell4 | violetred2 |
| cadetblue2 | darkseagreen2 | gray11 | gray71 | grey23 | grey83 | lightcoral | mediumblue3 | paleturquoise1 | sienna | violetred3 |
| cadetblue3 | darkseagreen3 | gray12 | gray72 | grey24 | grey84 | lightcyan | mediumblue3 | paleturquoise2 | sienna1 | wheat |
| cadetblue4 | darkseagreen4 | gray13 | gray73 | grey25 | grey85 | lightcyan1 | mediumblue4 | paleturquoise3 | sienna2 | wheat |
| chartreuse | darkslateblue | gray14 | gray74 | grey26 | grey86 | lightcyan2 | mediumpurple | paleturquoise3 | sienna3 | wheat1 |
| chartreuse1 | darkslategray | gray15 | gray75 | grey27 | grey87 | lightcyan3 | mediumpurple1 | paleturquoise4 | sienna4 | wheat2 |
| chartreuse2 | darkslategray1 | gray16 | gray76 | grey28 | grey88 | lightcyan4 | mediumpurple2 | papayawhip | skyblue | wheat3 |
| chartreuse3 | darkslategray2 | gray17 | gray77 | grey29 | grey89 | lightgoldenrod | mediumpurple3 | peachpuff | skyblue1 | wheat4 |
| chartreuse4 | darkslategray3 | gray18 | gray78 | grey30 | grey90 | lightgoldenrod1 | mediumpurple4 | peachpuff1 | skyblue2 | whitesmoke |
| chocolate | darkslategray4 | gray19 | gray79 | grey31 | grey91 | lightgoldenrod2 | mediumseagreen | peachpuff2 | skyblue3 | yellow |
| chocolate1 | darkslategray5 | gray20 | gray80 | grey32 | grey92 | lightgoldenrod3 | mediumslateblue | peachpuff3 | skyblue4 | yellow1 |
| chocolate2 | darkturquoise | gray21 | gray81 | grey33 | grey93 | lightgoldenrod4 | mediumspringgreen | peachpuff4 | slateblue | yellow2 |
| chocolate3 | darkviolet | gray22 | gray82 | grey34 | grey94 | lightgoldenrod5 | mediumturquoise | peru | slateblue1 | yellow3 |
| chocolate4 | deeppink | gray23 | gray83 | grey35 | grey95 | lightgray | mediumvioletred | pink | slateblue2 | yellow4 |
| coral | deeppink1 | gray24 | gray84 | grey36 | grey96 | lightgreen | midnightblue | pink1 | slateblue3 | yellowgreen |
| coral1 | deeppink2 | gray25 | gray85 | grey37 | grey97 | lightgrey | mintcream | pink2 | slateblue4 | |
| coral2 | deeppink3 | gray26 | gray86 | grey38 | grey98 | lightpink | mistyrose | pink3 | slategray | |
| coral3 | deeppink4 | gray27 | gray87 | grey39 | grey99 | lightpink1 | mistyrose1 | pink4 | slategray1 | |

viridis

viridis



inferno



magma



plasma

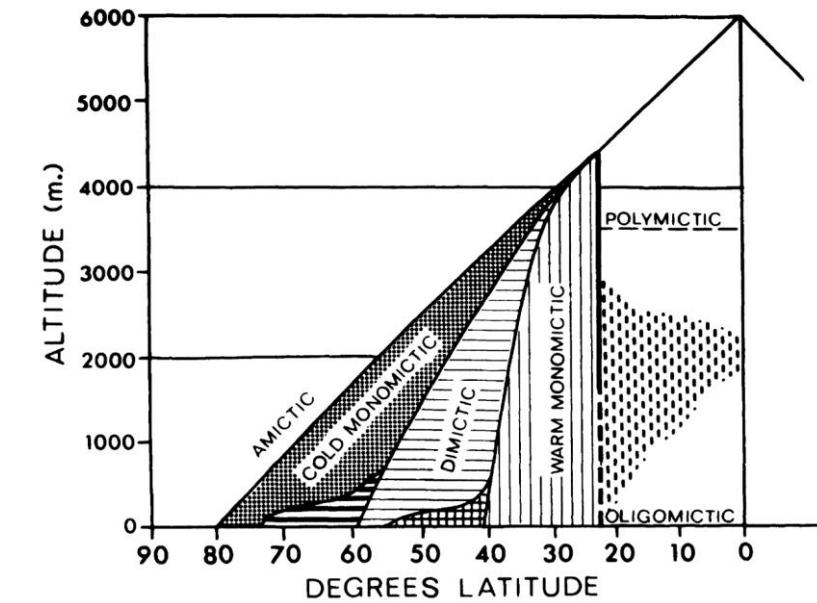
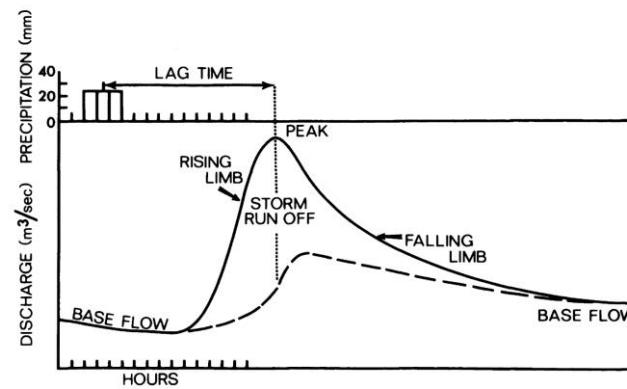
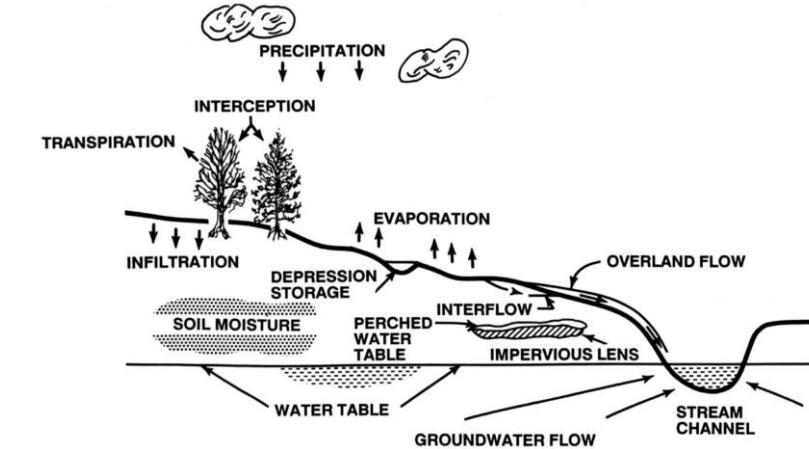
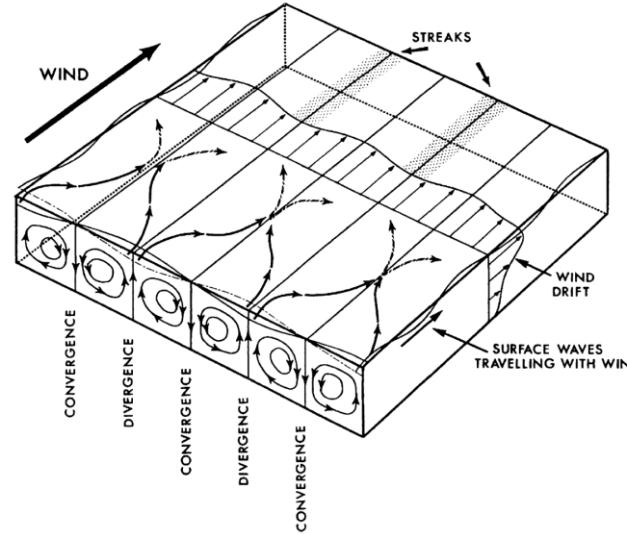


cividis



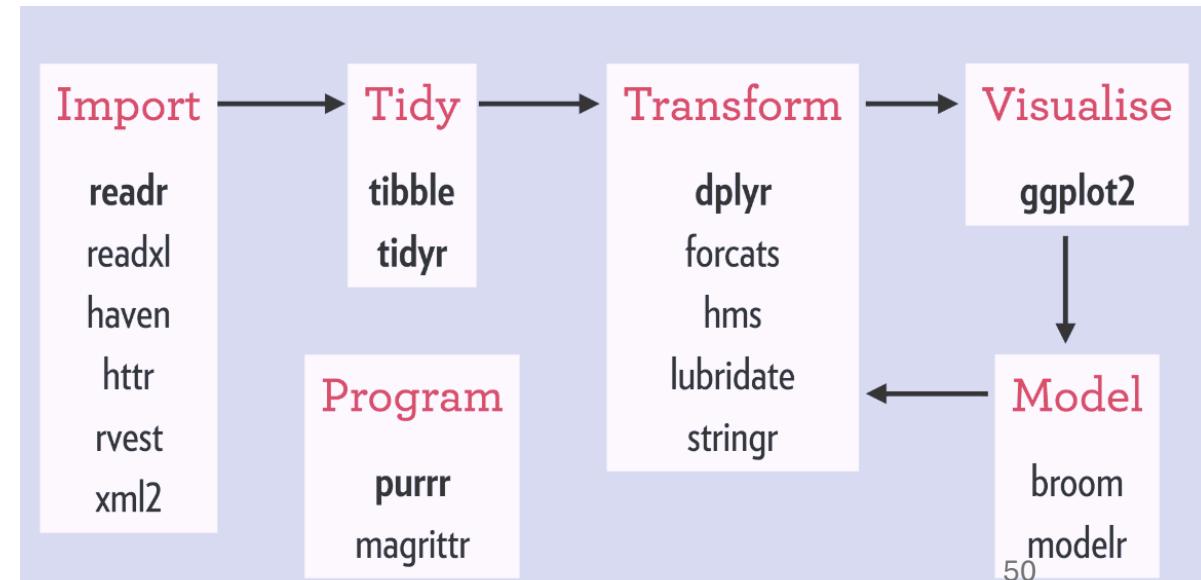
+ bonus poeng: du trenger kanskje ikke farger!

- Wetzel's «Limnology: lake and river systems» 3rd edt. (2001) er en mesterklasse i grafer og illustrasjoner i sort/hvitt.
- Uten farger klarer han å lage bedre, mer forklarende bilder enn mange proffe gjør i dag.



Tidyverse – *nesten alt du trenger finnes her*

- Tidyverse er en rekke pakker som opprettholdes av Posit, skaperne av R-studio.
- De har et stort sett med pakker som kan gjøre nesten alt du trenger for å ta i bruk dataene dine (- statistiske tester)
- Strukturen til Tidyverse gjør det mye enklere å jobbe iterativt



Rør-strømmer, funksjoner, looper, og funksjonell programering

Røret – « %>% »

- Tidyverse innførte %>% røret, og skapte med dette i prinsippet «moderne R»
- Røret fører datasettet på venstre side inn i funksjonen til høyre – dette gjør det langt lettere å skrive og lese kode
- Base R har nå også |> røret, men for øyeblikket er %>% noe bedre

```
Dates <- Dates %>%
  mutate(name = cumsum(!is.na(name)) + 1) %>%
  group_by(name) %>%
  mutate(name = row_number()) %>%
  ungroup()
```

```
Dates <-
  ungroup(
    mutate(
      group_by(
        mutate(
          Dates, name = cumsum(!is.na(name)) + 1),
          name
        ),
        name = row_number()
      )
    )
```



Rør-strømmer, funksjoner, looper, og funksjonell programering

- **Funksjoner**
- Funksjoner er et sett med instruksjoner som blir gitt informasjon fra et gitt sett med variabler
- Om du må gjøre noe komplekst mer enn en gang kan funksjoner være løsningen

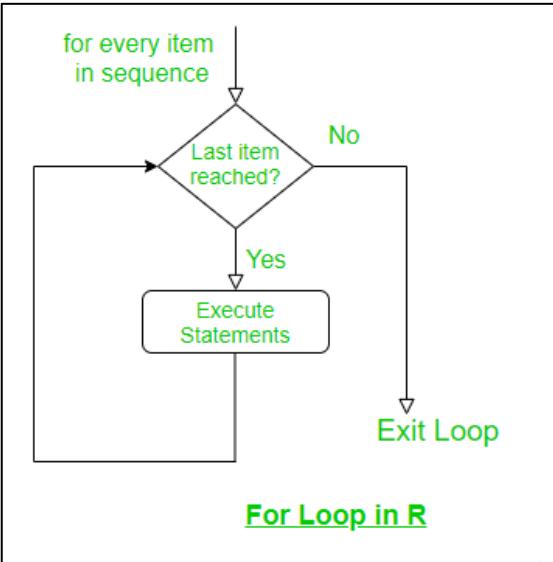
```
custom_predict ←  
  function(model, data) {  
    pred ← predict(model, na.omit(data))  
    response ← pred$.pred  
    return(response)  
  }
```



Rør-strømmer, funksjoner, looper, og funksjonell programering

- Looper

for()



For Loop in R

while()

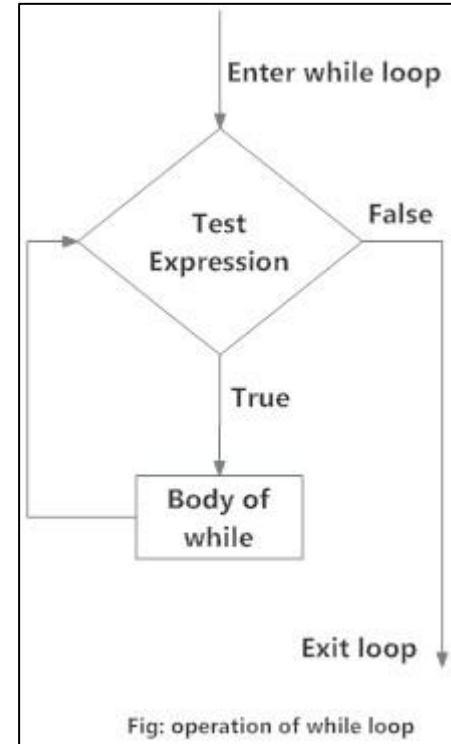
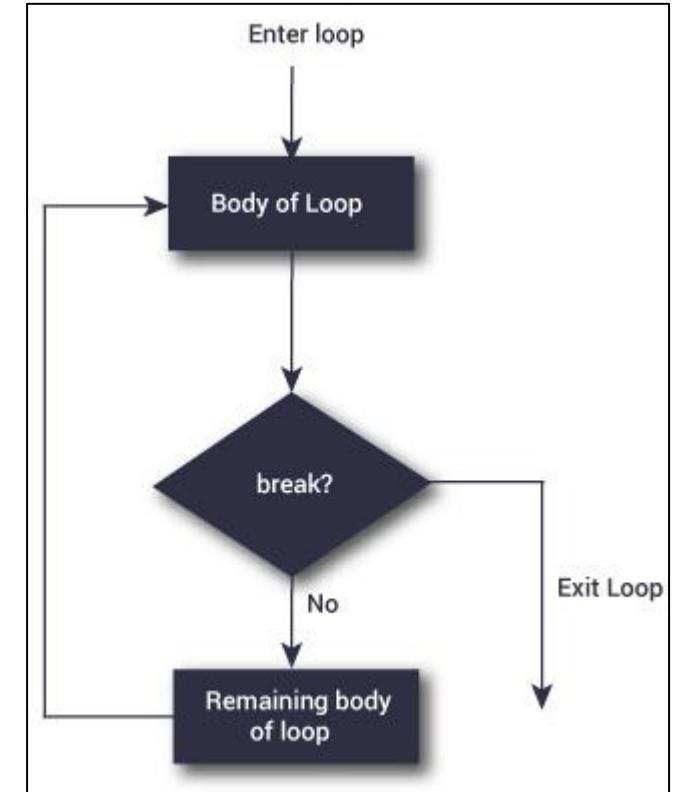


Fig: operation of while loop

repeat()



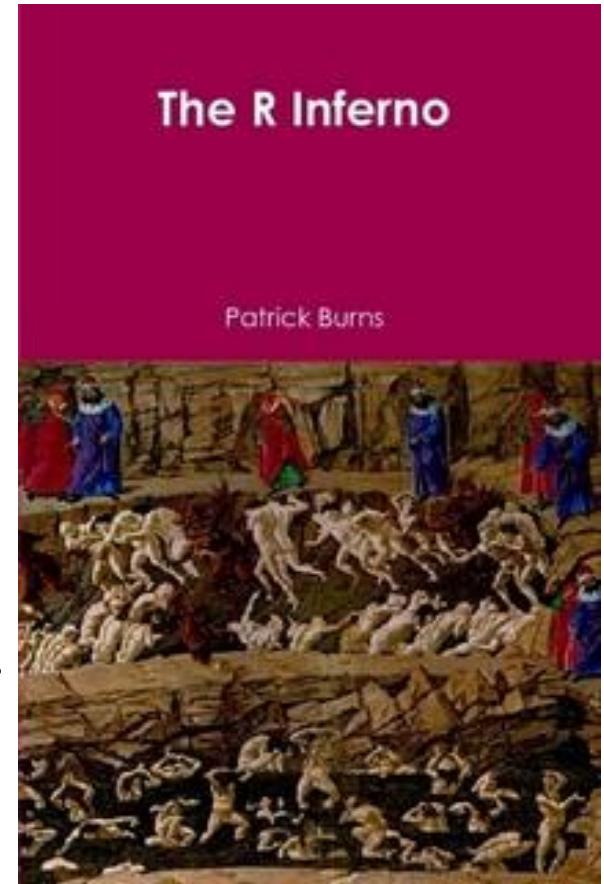
Unngå looper om mulig

*This is where Virgil said to me,
“Remember your science*

*the more perfect a thing,
the more its pain or pleasure”*

Looper er ikke egentlig anbefalt i R, de lånes fra andre språk hvor de er effektive, i R er de faktisk veldig trege.

Spesielt for store oppgaver kan det å bruke looper gjøre at oppgaven tar mye mye lengere tid



Here is some sample code:

```
lsum <- 0
for(i in 1:length(x)) {
    lsum <- lsum + log(x[i])
}
```

No. No. No.

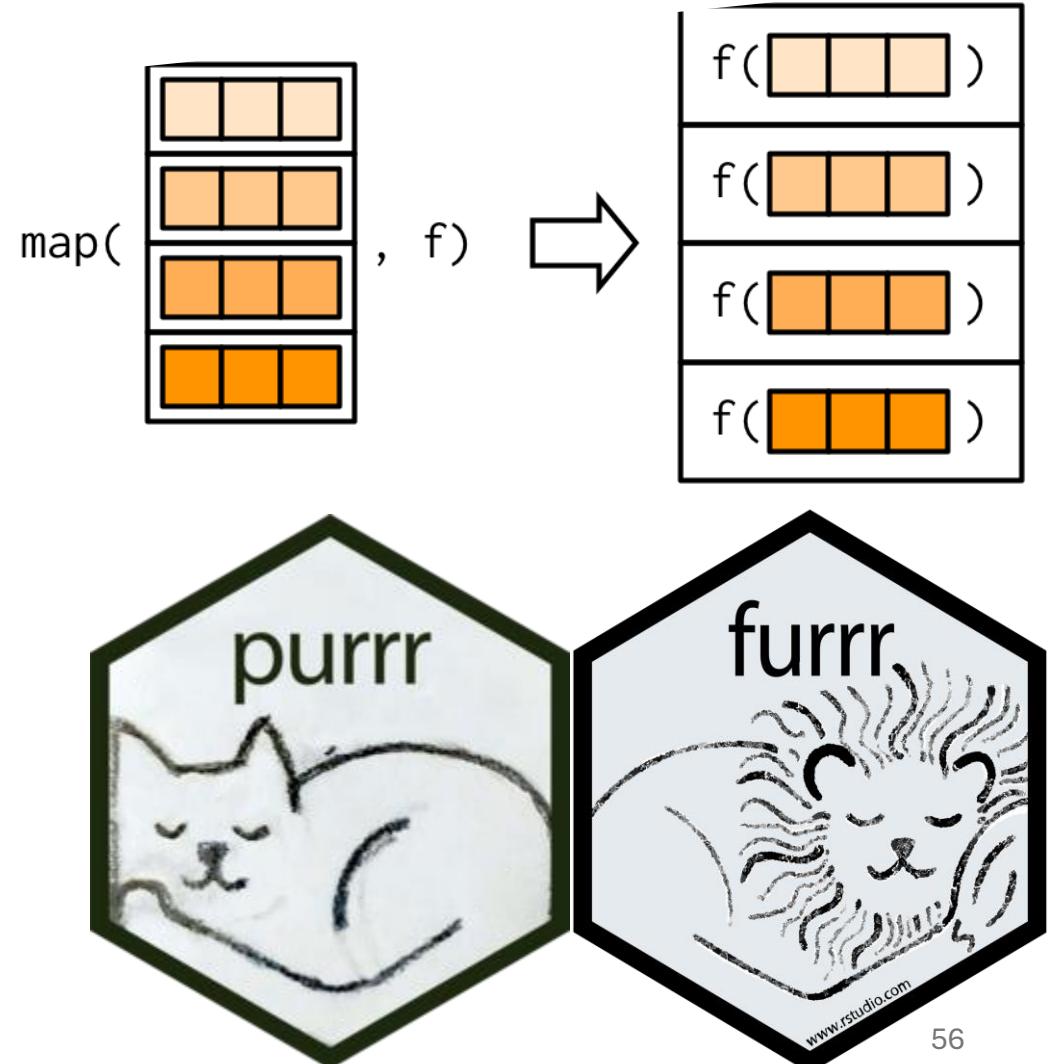
This is speaking R with a C accent—a strong accent. We can do the same thing much simpler:

```
lsum <- sum(log(x))
```



Rør-strømmer, funksjoner, looper, og funksjonell programering

- **Funksjonell programering**
- I programmerings verdenen er det en sekt med de som elsker funksjonell programering.
- `map()` tar et sett med vektorer og kjører de gjennom funksjonen “f”
- For spesielt store oppgaver kan funksjonell programering paralleliseres



Funksjon + map

```
FullFilteredTukey ← function(data, formula, filterObj, filterWord){  
  dataFiltered ← data %>% filter (!!dplyr::sym(filterObj) == filterWord)  
  
  TestList ← list(  
    "aov" = aov(formula, data = dataFiltered),  
    "Tukey" = aov(formula, data = dataFiltered) %>% TukeyHSD(),  
    "CLD" = multcompLetters4(  
      aov(formula, data = dataFiltered),  
      aov(formula, data = dataFiltered) %>% TukeyHSD()  
    )[[ "Position"]][[ "Letters"]]  
  )  
  return(TestList)  
}  
  
map(  
  c("Large Ring", "MiniDisk", "Small Ring"),  
  ~ FullFilteredTukey(AllMethods2PosCor, formula = formula(log10(mlPerMinutePerCmSq) ~ Position), "Method", .x)  
)
```



GitHub Copilot – *ChatGPT for kode*

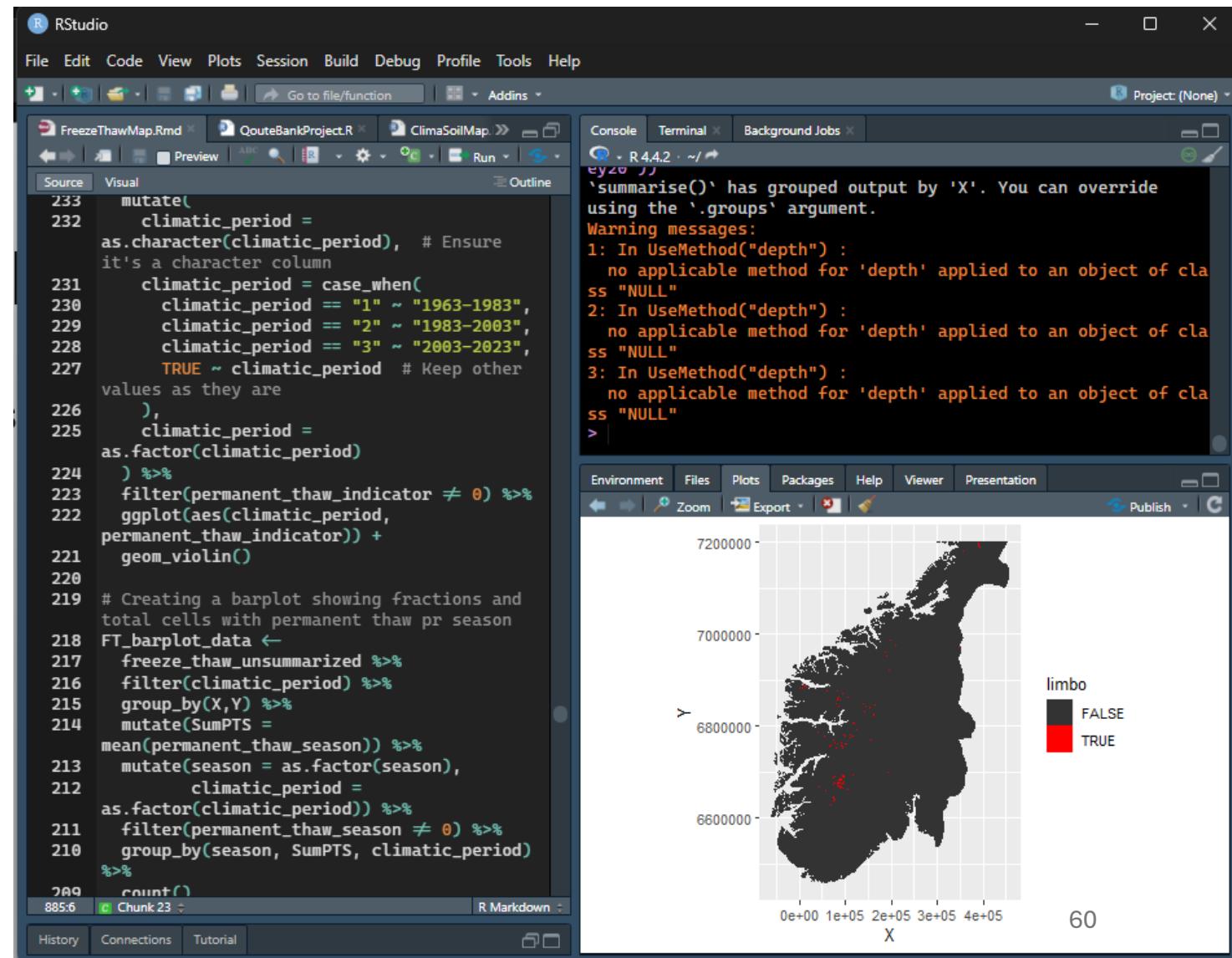
- Langt bedre for kode enn ChatGPT
- Kan gi deg forslag på kode endringer (med forklaringer) og forklaringer på eksisterende kode
- Interager via:
 - Best i VS-Code (*etter min mening*) – Vi kommer tilbake til dette
 - Finnes som chat på nettet
 - Kan integreres som «auto complete» i R-studio, men dette anbefales *ikke*, det er i all hovedsak mer irriterende enn hjelpsomt...



Oppsett av *IDE* (*integrated development environment*)

R-studio

- Pane layout ➔
 - Grupper;
 - Kode | Konsoll
ubrukte | Plots & Env.
- Sett det opp slik at det funker best for deg!
- Velg en fargeprofil og font som du liker
(men den bør også være *leselig* for andre)



The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The left sidebar has tabs for Source, Visual, and Outline. The main workspace contains three tabs: 'FreezeThawMap.Rmd', 'QouteBankProject.R', and 'ClimaSoilMap.R'. The 'Source' tab displays R code for data manipulation and visualization. The 'Console' tab shows R session output with several warning messages about 'NULL' objects. The 'Plots' tab displays a map of Norway with a grid overlay, colored according to a legend for 'limbo' status. The legend indicates 'limbo' is FALSE (dark grey) and TRUE (red). The bottom status bar shows 'Count 1' and '8856'.

okjava -src/main/webapp/xuly.jsp - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

* dichden.jsp Home.jsp xuly.jsp

```
19  * <%@
20  int a = Integer.parseInt(request.getParameter("a"));
21  int b = Integer.parseInt(request.getParameter("b"));
22  int c = Integer.parseInt(request.getParameter("c"));
23  if (a == 0) {
24      if (b == 0) {
25          out.println("Phương trình vô nghiệm!");
26      } else {
27          out.println("Phương trình có một nghiệm: "
28                  + "x = " + (-c / b));
29      }
30      return;
31  }
32  float delta = b * b - 4 * a * c;
33  float x1;
34  float x2;
35
36  if (delta > 0) {
37      x1 = ((float) (-b + Math.sqrt(delta)) / (2 * a));
38      x2 = ((float) (-b - Math.sqrt(delta)) / (2 * a));
39      out.println("Phương trình có 2 nghiệm là: "
40                  + "x1 = " + x1 + " và x2 = " + x2);
41  } else if (delta == 0) {
42      x1 = (-b / (2 * a));
```

Oppsett av *IDE* (*integrated development environment*)

Bruk av VS-code og R-studio sammen

- VS-Code har en langt bedre integrert versjon av *GitHub copilot*
- Du jobber på samme kode-fil i begge IDE-ene på en gang
- ***Man må lagre hver gang man bytter IDE***
- På tross av at dette er et mye mindre «enkelt» oppsett kan det gjøre prosessen av å skrive noe avansert kode mye enklere!



Stilguide for kode

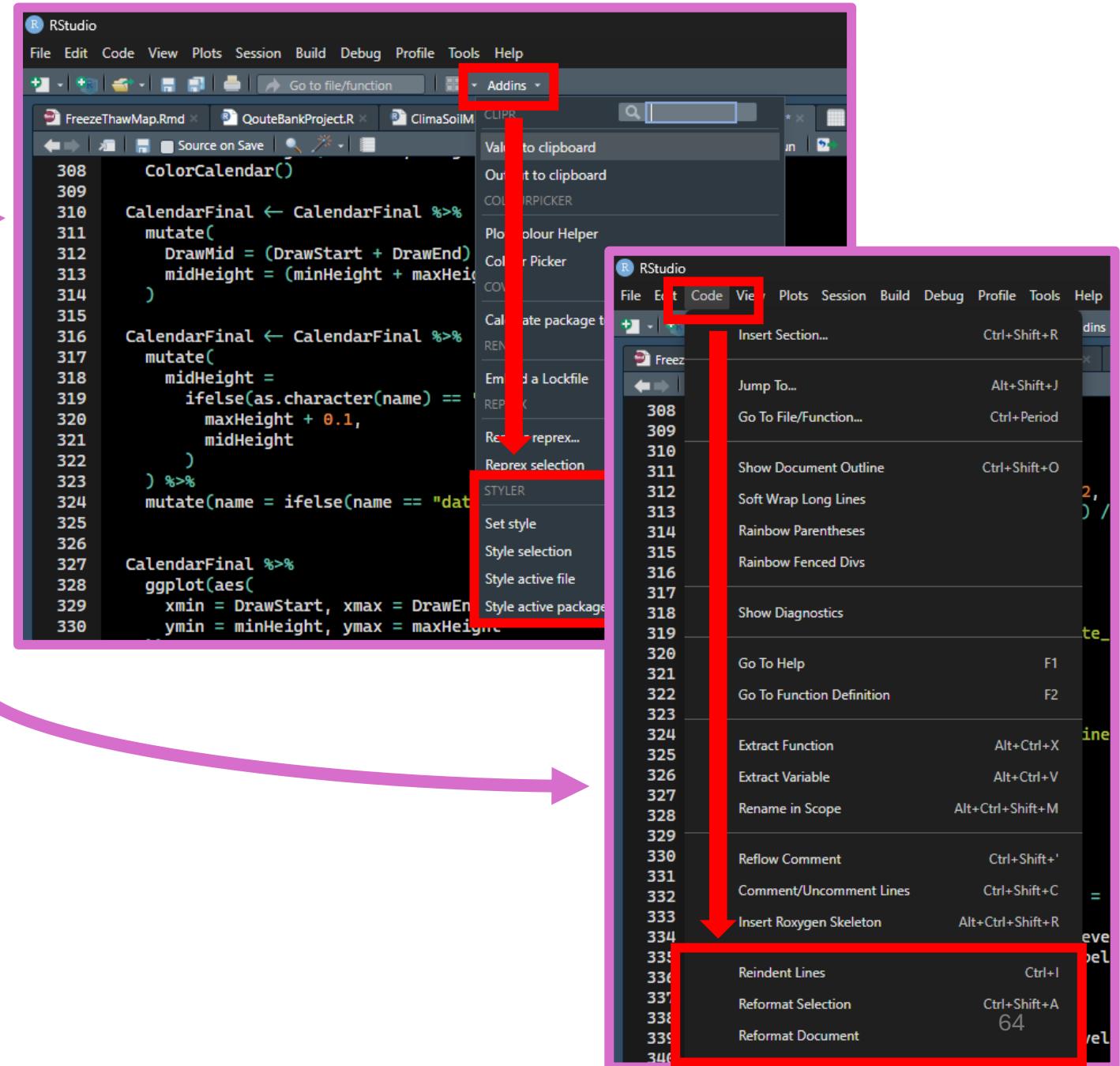
- Måten du formaterer koden din påvirker hvor enkelt det er å lese koden din. Gjør koden lett å lese
- Posit har «tidyverse» stil-guiden [<https://style.tidyverse.org>]
- Hva er viktigst?
 - Tydelige navn –
 - Engelsk | *substantiv_substantiv* | små bokstaver | korte <*forståelig*>
 - EKS: day_one, day_1, forest
 - Iterative navn for variasjoner av datasett
 - forest_soil_moisture_long, day_one_after_eight
 - *Skriv samme kode på samme måte*



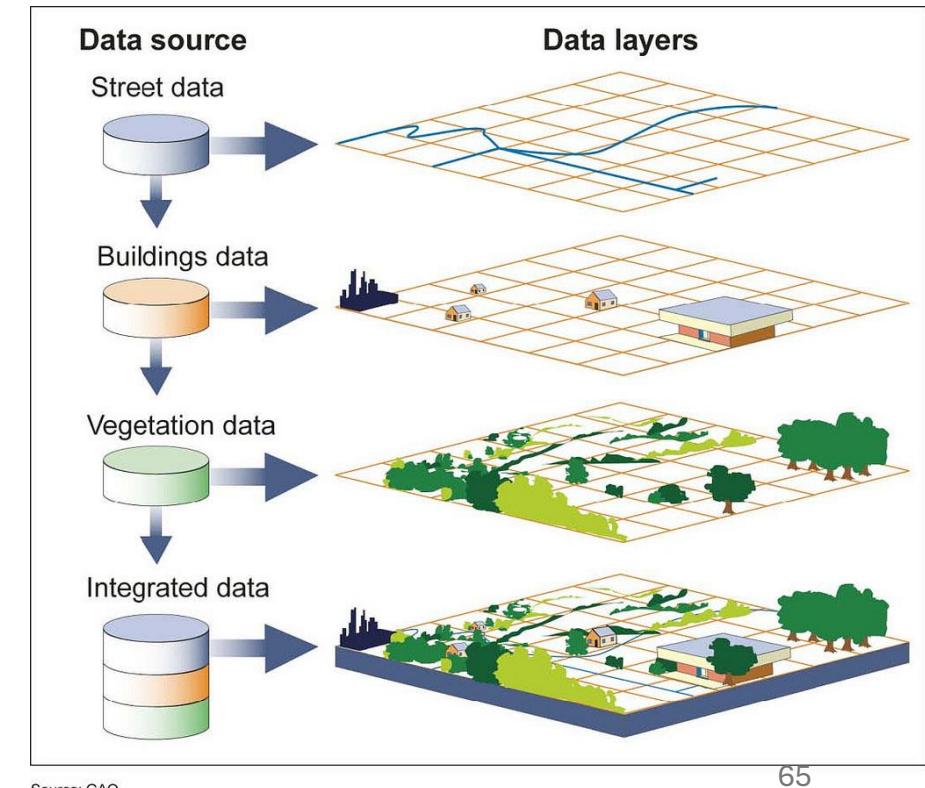
Stilguide for kode

Auto stil for kode:

- Tidyverse – **styler** pakken
 - Krever installasjon
- R-studios innebygde stil format



- Mange GIS oppgaver kan utføres i R.
- Dette er mer for viderekomne, både innenfor GIS og R
- Her er det ekstra viktig at du vet hva du prøver å gjøre for å klare å gjennomføre det
- Men fordelen VS QGIS/Arc GIS er at kode gjør det langt enklere å finne feil og repetere prosessen.
- R-GIS er derfor «tryggere»



R i fremtiden? Hvor er kursen rettet?

- R er et eldende språk, men blir fortsatt oppdatert og er fortsatt «best» for statistisk programmering, spesielt takket POSIT
- R-Studio *kan bli* byttet ut med Posits nye IDE *Positron* (*VS Code basert*)
- Andre språk har bedre evner innenfor spesifikke sektorer
 - **Python** – dyplæring/tyngre AI modeller
 - **Julia** – matematikk, behandling av store data, m.m.
- *Det ser ut som flere bruker fler språk* (generell trend i koding)



Steintavlen

1. *Aldri, aldri aldri aldri, tenk at du skal huske noe selv*
2. *Behold alltid originalene*
3. *Omfavn den iterative prosessen*
4. *Vit hva koden din faktisk gjør!!!*
5. *Excel er til produksjon og endring av rådata;
R er til alt annet*
6. *?func() -> R-klubb nettsiden
-> google -> ekstern hjelp/AI*
7. *Graf & test > graf > test > tabell > tall*



Vanlige funksjoner

- **Data manipulasjon**

| Funksjon | Forklaring | Eks |
|--------------------------|---|---|
| <code>filter()</code> | filtrer rader | <code>filter(name == "Einar")</code> <code>filter(tre != 2 & skog %in% c("North", "South"))</code> |
| <code>select()</code> | velg kolonner | <code>select(temp, lengde, id), select(!id)</code> |
| <code>group_by()</code> | grupper etter kolonne | <code>group_by(species)</code> |
| <code>ungroup()</code> | fjern gruppering | <code>ungroup()</code> |
| <code>mutate()</code> | endre eller skap kolonner | <code>mutate(min = seconds_to_period(min * 60))</code> |
| <code>summarize()</code> | oppsummer data m. 1 argument pr verdi | <code>group_by(cyl) %>% summarize(mean = mean(disp), n = n())</code> |
| <code>reframe()</code> | rekonstruerer datasettet med gitte funksjoner, ligner mutate og summarize men har færre regler | <code>group_by(cyl) %>%</code> <code>reframe(qs = quantile(disp, c(0.25, 0.75)), prob = c(0.25, 0.75))</code> |
| <code>arrange()</code> | sorter rader | <code>arrange(age)</code> |
| <code>rename()</code> | gi kolonner nye navn | <code>rename(new_name = old_name)</code> |
| <code>distinct()</code> | behold kun unike rader | <code>distinct()</code> |
| <code>count()</code> | antall | <code>count()</code> |
| <code>pull()</code> | Henter vektoren fra en gitt rad | <code>pull(station_ID)</code> |

dplyr (For data-manipulasjon)

| Navn på formelen | Forklaring | Eksempel |
|--------------------------|--|---|
| <code>filter()</code> | Filtrerer rader basert på betingelser. | <code>filter(df, age > 30)</code> |
| <code>select()</code> | Velger spesifikke kolonner fra en data frame. | <code>select(df, name, age)</code> |
| <code>mutate()</code> | Oppretter eller endrer kolonner. | <code>mutate(df, age_next_year = age + 1)</code> |
| <code>group_by()</code> | Grupperer data for aggregering. | <code>group_by(df, category)</code> |
| <code>summarize()</code> | Beregner sammendragsstatistikk for grupperte data. | <code>summarize(df, avg_income = mean(income, na.rm = TRUE))</code> |

tidyr (For datatransformasjon og rensing)

| Navn på formelen | Forklaring | Eksempel |
|-----------------------------|--|---|
| <code>pivot_longer()</code> | Gjør brede data lange ved å samle kolonner i to variabler (navn og verdi). | <code>pivot_longer(df, cols = c("var1", "var2"), names_to = "variable", values_to = "value")</code> |
| <code>pivot_wider()</code> | Gjør lange data brede ved å spre rader basert på en variabel. | <code>pivot_wider(df, names_from = category, values_from = value)</code> |
| <code>separate()</code> | Deler en kolonne inn i flere kolonner basert på en separator. | <code>separate(df, col = "full_name", into = c("first", "last"), sep = " ")</code> |
| <code>unite()</code> | Kombinerer flere kolonner til én enkelt kolonne. | <code>unite(df, "full_name", c(first, last), sep = " ")</code> |
| <code>fill()</code> | Fyller manglende verdier med nærmeste ikke-manglende verdi. | <code>fill(df, column_name, .direction = "down")</code> |

forcats (For kategoriske variabler - faktorer)

| Navn på formelen | Forklaring | Eksempel |
|----------------------------|--|--|
| <code>fct_reorder()</code> | Endrer rekkefølgen på faktorene basert på en numerisk variabel. | <code>fct_reorder(df\$category, df\$value, .fun = median)</code> |
| <code>fct_lump()</code> | Grupperer sjeldne faktornivåer under en felles kategori ("other"). | <code>fct_lump(df\$category, n = 5)</code> |
| <code>fct_relevel()</code> | Flytter spesifikke faktornivåer til en ny rekkefølge. | <code>fct_relevel(df\$category, "A", "B", "C")</code> |
| <code>fct_infreq()</code> | Sorterer faktorene etter hyppighet. | <code>fct_infreq(df\$category)</code> |
| <code>fct_rev()</code> | Reverserer rekkefølgen på faktorene. | <code>fct_rev(df\$category)</code> |

lubridate (For håndtering av datoer og tid)

| Navn på formelen | Forklaring | Eksempel |
|---------------------------|--|---|
| <code>ymd()</code> | Konverterer en streng til en dato i "år-måned-dag" format. | <code>ymd("2024-03-12")</code> |
| <code>mdy()</code> | Konverterer en streng til en dato i "måned-dag-år" format. | <code>mdy("March 12, 2024")</code> |
| <code>dmy()</code> | Konverterer en streng til en dato i "dag-måned-år" format. | <code>dmy("12-03-2024")</code> |
| <code>now()</code> | Returnerer gjeldende tidspunkt. | <code>now()</code> |
| <code>floor_date()</code> | Runder datoer ned til nærmeste enhet (f.eks. måned). | <code>floor_date(ymd("2024-03-12"), "month")</code> |

purrr (For funksjonell programmering)

| Navn på formelen | Forklaring | Eksempel |
|------------------------|--|--|
| <code>map()</code> | Bruker en funksjon på hver verdi i en liste. | <code>map(c(1, 2, 3), sqrt)</code> |
| <code>map_dbl()</code> | Som <code>map()</code> , men returnerer en numerisk vektor. | <code>map_dbl(c(1, 2, 3), sqrt)</code> |
| <code>map_chr()</code> | Som <code>map()</code> , men returnerer en tegnstreng-vektor. | <code>map_chr(c(TRUE, FALSE), as.character)</code> |
| <code>map_df()</code> | Som <code>map()</code> , men returnerer en data frame. | <code>map_df(list(df1, df2), bind_rows)</code> |
| <code>reduce()</code> | Kombinerer en liste til en enkelt verdi ved å bruke en funksjon. | <code>reduce(c(1, 2, 3), +)</code> |