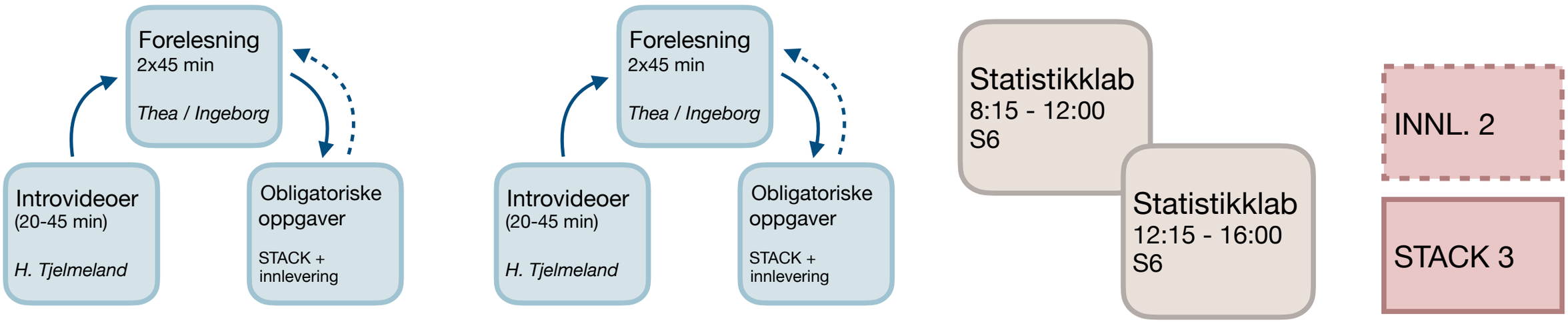
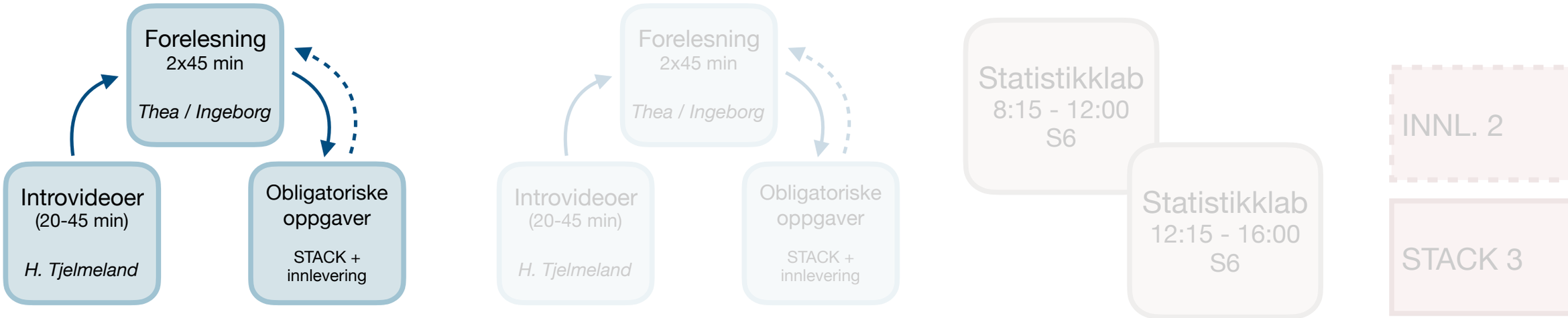
TMA4245 Statistikk

Uke 4 - mandag



Thea Bjørnland,
Institutt for matematiske fag, NTNU

Uke 4 (man 22. jan - fredag 26. januar)



Introvideoer

Stokastisk simulering og pseudo-tilfeldige tall

TMA4240/TMA4245 Statistikk

Håkon Tjelmeland
Institutt for matematiske fag
Norges teknisk-naturvitenskapelige universitet

Simulere fra diskrete og kontinuerlige fordelinger

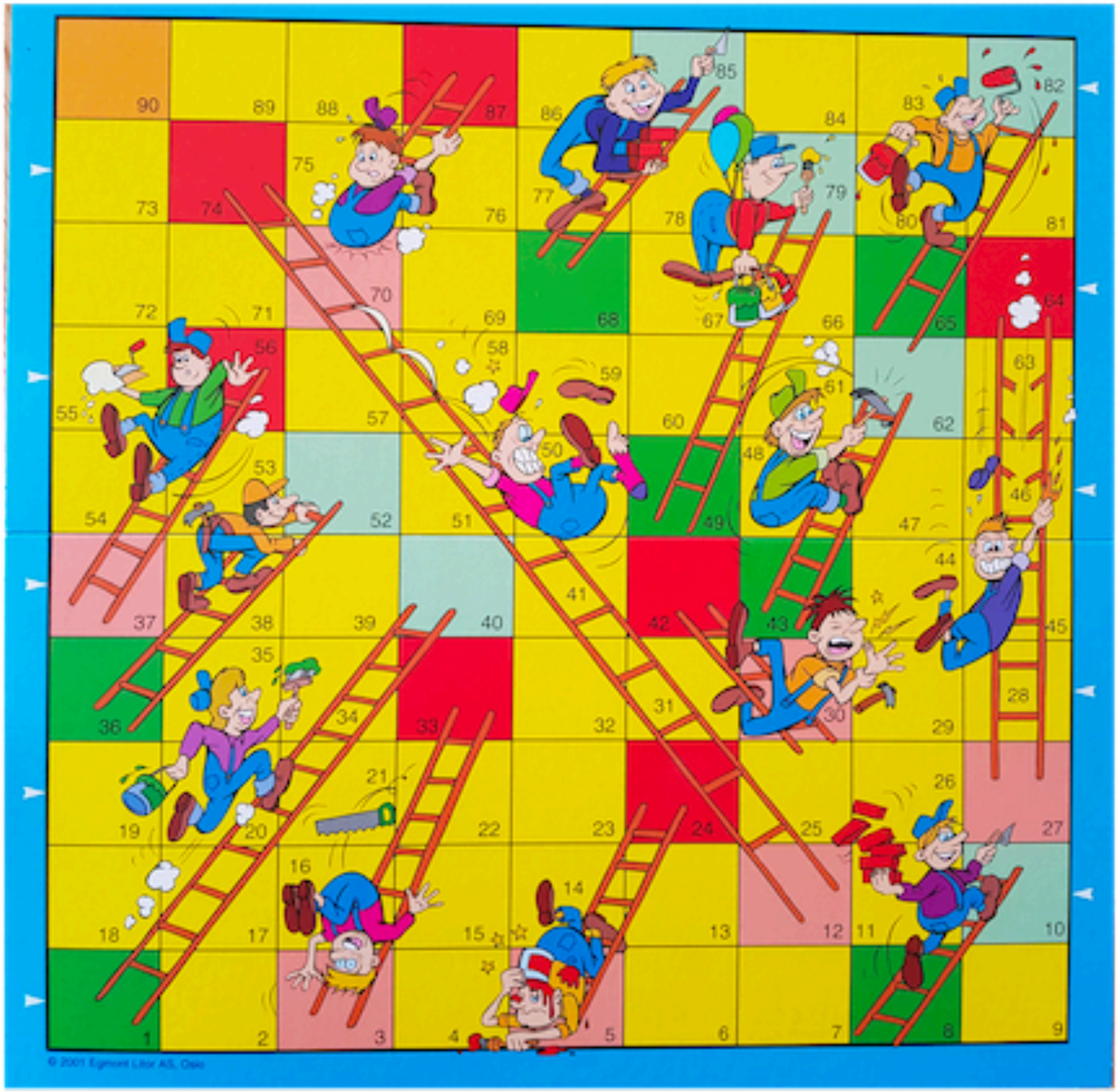
TMA4240/TMA4245 Statistikk

Håkon Tjelmeland
Institutt for matematiske fag
Norges teknisk-naturvitenskapelige universitet

Oppgaver

STACK 3: Oppg. 1-2
INNL. 2: Oppg. 1 og 2

Eksempel 1: Stigespill



Hva er sannsynligheten for å vinne dersom man starter først?
(2 spillere)

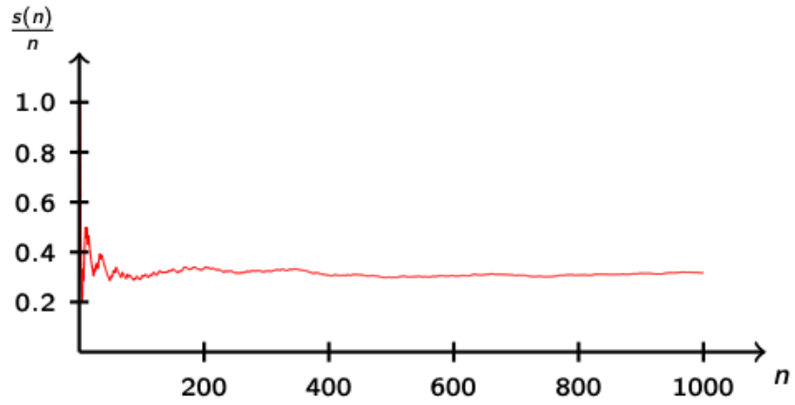
Repetisjon fra forrige uke

Tolkning av sannsynlighet

- Eks: Kast terning, registrer antall øyne, $S = \{1, 2, 3, 4, 5, 6\}$
  - la $A = \{5, 6\}$, hva betyr det at $P(A) = 1/3$?
- Stokastisk forsøk:
  - gjenta forsøket n ganger
  - $s(n)$: antall ganger hendelsen A skjer
  - relativ hyppighet av A: $\frac{s(n)}{n}$

$$\frac{s(n)}{n} \xrightarrow[n \to \infty]{} P(A)$$

  - simulering:



Dersom vi simulerer stigespill veldig mange ganger kan vi approksimere sannsynligheten for at spiller 1 vinner

Python: kode terningkast

```python
# kaster terning med unif[0,1]-fordeling
def kaste_terning_unif():
    u = np.random.rand()
    x = int(min(6,np.floor(6*u)+1))
    return x
```

← Fra video

```python
terningkast = kaste_terning_unif()
print(terningkast)
```

```
4
```

```python
# kaster terning med innebygd randint-funksjon
def kaste_terning():
    return np.random.randint(1, 7)
```

```python
terningkast = kaste_terning()
print(terningkast)
```

```
6
```

Python: kode stigespill

```
def stigespill_2player():
    spiller1 = 0
    spiller2 = 0
    vinner = 0

    while spiller1 < 90 and spiller2 < 90:
        kast_resultat1 = kaste_terning()
        kast_resultat2 = kaste_terning()

        spiller1 = flytteregler(spiller1, kast_resultat1)
        spiller2 = flytteregler(spiller2, kast_resultat2)

    if spiller1 == 90:
        vinner = 1
    elif spiller2 == 90:
        vinner = 2

    return vinner
```

Merk: Koden er skrevet vha ChatGPT

Python: kode stigespill

```python
# Spesielle felter (stiger og avslutning)
def flytteregler(posisjon, kast_resultat):
    spesielle_felter = [1, 8, 24, 33, 36, 42, 43, 49, 56, 64, 65, 68, 74, 87]
    nye_posisjoner = [40, 10, 5, 3, 52, 30, 62, 79, 37, 27, 82, 85, 12, 70]

    ny_posisjon = posisjon + kast_resultat

    if ny_posisjon in spesielle_felter:
        ny_posisjon_indeks = spesielle_felter.index(ny_posisjon)
        ny_posisjon = nye_posisjoner[ny_posisjon_indeks]
        return ny_posisjon
    elif ny_posisjon > 90:
        gå_tilbake = ny_posisjon-90
        return (90-gå_tilbake)
    else:
        return ny_posisjon
```

Merk: Koden er skrevet vha ChatGPT

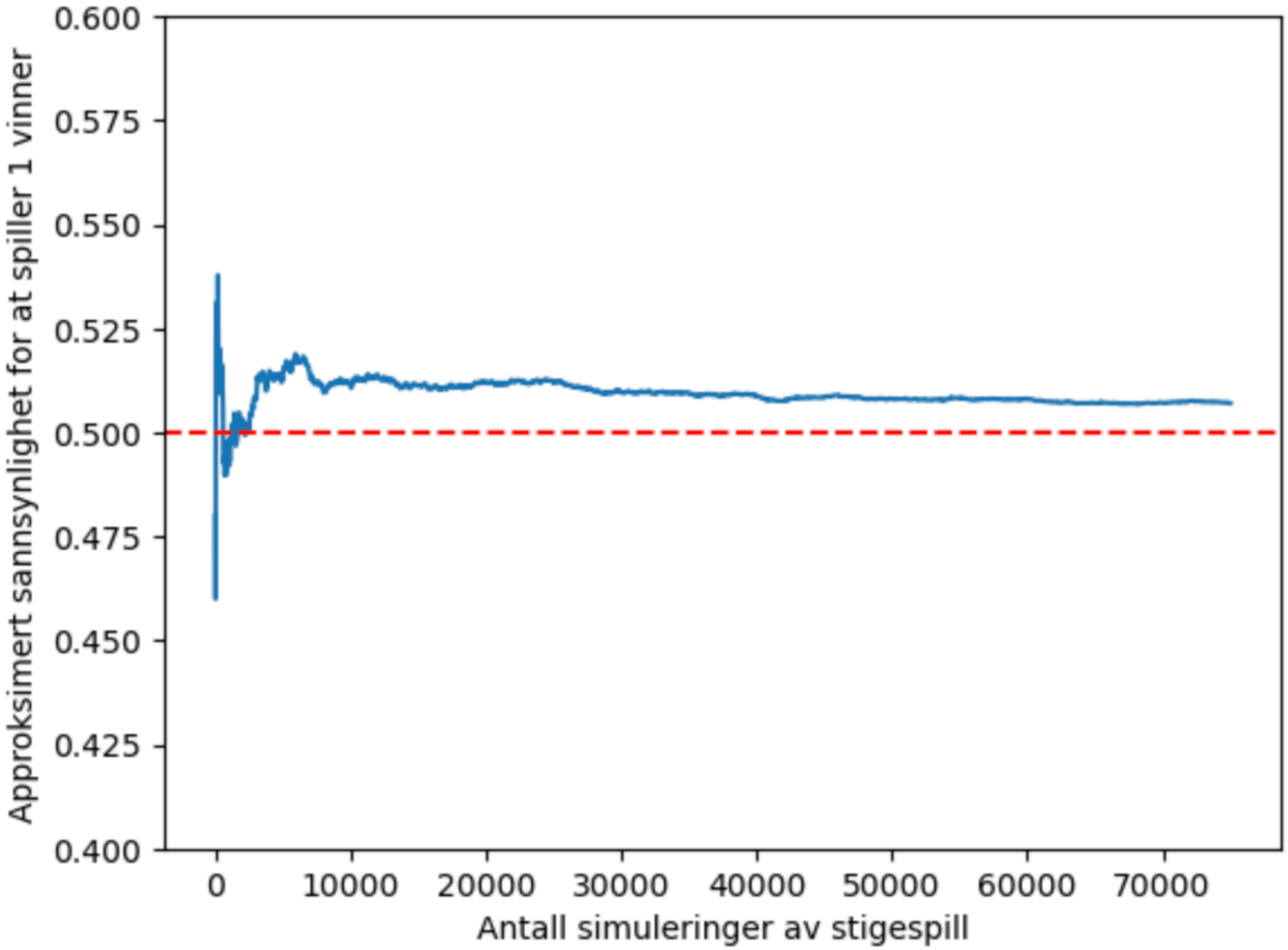
Python: simulere mange stigespill

```
# approksimert sannsynlighet for at spiller1 vinner
n = 1000
teller_spiller1_vinn = 0
for _ in range(n):
    vinner = stigespill_2player()
    if vinner == 1:
        teller_spiller1_vinn += 1
print(teller_spiller1_vinn/n)
```

```
0.47
```

Merk: Koden er skrevet vha ChatGPT

Python: simulere mange stigespill

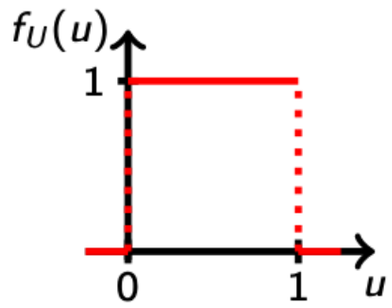


Stokastisk simulering, kort oppsummert

1. Vi kan bruke datamaskinen til å generere pseudo-tilfeldige tall

som hovedregel fra
en Unif[0,1]-fordeling

$$f_U(u) = \begin{cases} 1 & \text{for } x \in [0,1], \\ 0 & \text{ellers.} \end{cases}$$



2. Vi kan **transformere** uniformfordelte stokastiske variabler slik at vi også kan generere tall fra mer interessante sannsynlighetsfordelinger, både diskrete og kontinuerlige

Oppgave 1 og 2 i dag

https://docs.python.org/3/library/random.html#module-random

https://numpy.org/doc/stable/reference/random/generator.html

Oppgave 1: Simulere fra en diskret fordeling

En stokastisk variabel X har verdimengde {0, 1, 2, 3} med punktsannsynligheter $P(X=0) = 0.44$, $P(X=1) = 0.40$, $P(X=2) = 0.13$ og $P(X=3) = 0.03$.

a) Hvordan kan vi, fra en Unif(0,1)-fordeling, generere realisasjoner av X?

b) Dersom vi genererer u = 0.52, hva blir realisasjonen x av X?

Oppgave 1: Simulere fra en diskret fordeling

[1]:
```
import numpy as np
```

[2]:
```
#verdimengde
x = np.arange(4)
#punktsannsynligheter
f_x = np.array([0.44, 0.40, 0.13, 0.03])
# kumulative sannsynligheter
F_x = np.cumsum(f_x)
print(F_x)
```

```
[0.44 0.84 0.97 1.  ]
```

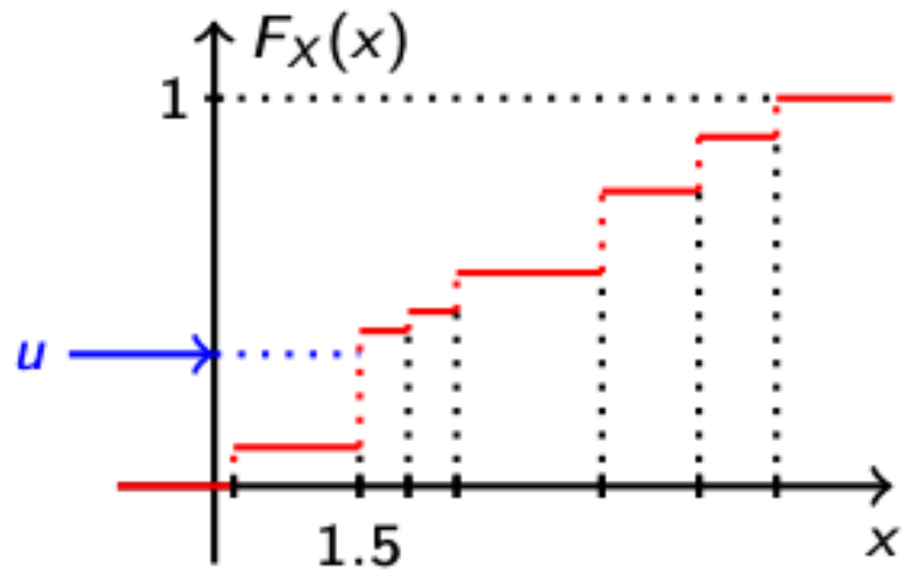
[3]:
```
u = np.random.uniform()
if(u < F_x[0]): # dersom u er mindre enn 0.44 vil vi at realisasjonen av X skal være 0
    x = 0
elif(u <= F_x[1]): # hvis u er mindre enn 0.84 vil vi at realisasjonen av X skal være 1
    x = 1
elif(u <= F_x[2]):
    x = 2
elif(u > F_x[2]):
    x = 3

print("u ble", u, "og x ble dermed", x)
```

```
u ble 0.4642365102448054 og x ble dermed 1
```

Fra video: Simulere fra en diskret fordeling

$U \sim \text{Unif}[0, 1]$

$X = \min\{x : F_x(x) \geq U\}$



Oppgave 2a: Simulere fra kontinuerlige fordelinger

Oppgave 2

I figur 1 er det gitt en python-funksjon med to input-parametre, et positivt heltall n og en parameter phi (φ) som må være større enn null. Funksjonen genererer n uavhengige realisasjoner av en kontinuerlig stokastisk variabel Y. Ut fra denne koden, bestem sannsynlighetstettheten til Y, dvs $f_Y(y)$ for $-\infty < y < \infty$.

```
import numpy as np

def simY(n, phi):
    u = np.random.uniform(size=n)
    y = phi * np.sqrt(u)

    return y
```

Figur 1: Phyton-funksjon som genererer n uavhengige realisasjoner av en kontinuerlig stokastisk variabel Y, der phi (φ) er en parameter i sannsynlighetsfordelingen til Y.

i. Hva er $F_U(u) = P(U \leq u)$?

ii. I hvilket intervall vil Y ta tallverdier?

iii. Hva blir $F_Y(y) = P(Y \leq y)$, uttrykt ved $F_U(u) = P(U \leq u)$?

iv. Hva blir $f_Y(y)$?

Eksamen TMA4240 Des 2023

Oppgave 2a: Simulere fra kontinuerlige fordelinger

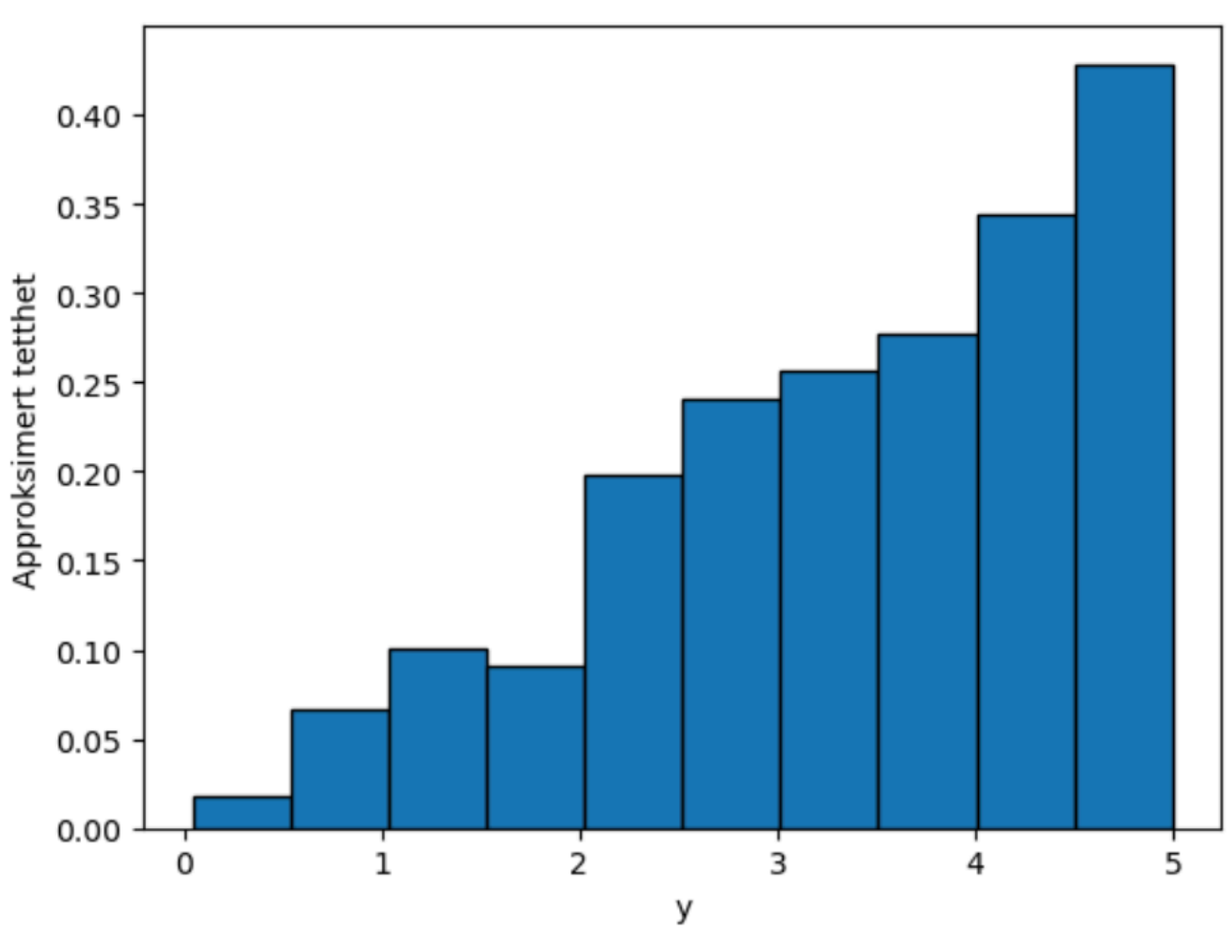
```
:
import numpy as np
```

```
:
def simY(n,phi):
    u = np.random.uniform(size=n)
    y = phi*np.sqrt(u)
    return(y)
```

```
:
sim1 = simY(n=1000,phi=5)
print(sim1[0:5])
```

```
[3.58275121 4.49293669 4.46034152 4.96549151 3.10240576]
```

```
:
import matplotlib.pyplot as plt

plt.hist(sim1, density=True,bins=10,edgecolor="black") #density=True gjør at vi får et sannsynlighetshistogram
plt.ylabel('Approksimert tetthet'); plt.xlabel('y')
plt.show()
```



Oppgave 2b: Simulere fra kontinuerlige fordelinger

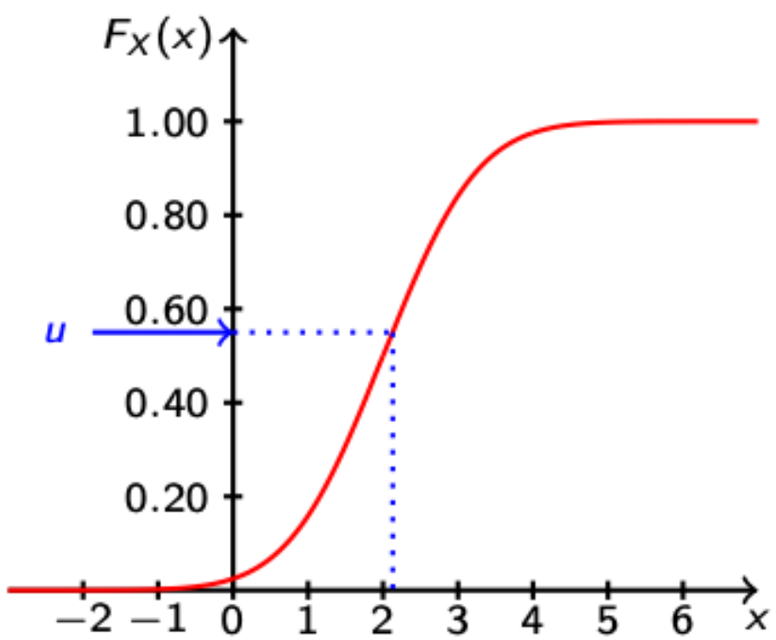
La en annen kontinuerlig stokastisk variabel X ha kumulativ fordelingsfunksjon gitt ved

$$F_X(x) = \begin{cases} 1 - \exp\left\{-\frac{x^2}{\theta}\right\} & \text{dersom } x \geq 0, \\ 0 & \text{ellers,} \end{cases}$$

der $\theta > 0$ er en parameter. Utled hvordan man ved å ta utgangspunkt i en uniformfordelt variabel på intervallet $[0, 1]$ kan generere en realisasjon av X. Skriv så en python-funksjon, simX, som genererer n uavhenige realisasjoner av X.

$U \sim \text{Unif}[0, 1]$

$X = F_X^{-1}(U)$




```python
import numpy as np

def simY(n, phi):
    u = np.random.uniform(size=n)
    y = phi * np.sqrt(u)

    return y
```

Eksamen TMA4240 Des 2023

Hvorfor er det nyttig at en datamaskin kan generere (pseudo-)tilfeldige tall?

1. Vi kan bruke datamaskiner til å approksimere sannsynligheter i komplekse situasjoner (vanskelig/umulig å regne for hånd)

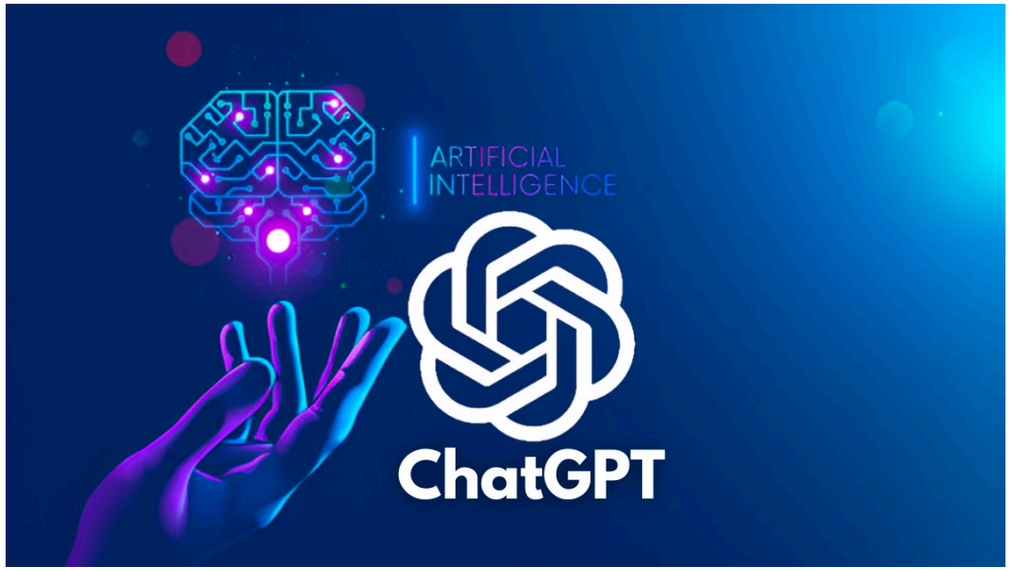
2. Vi kan bruke datamaskiner til å generere (fra en passende sannsynlighetsfordeling) tilfeldig "output" til brukere

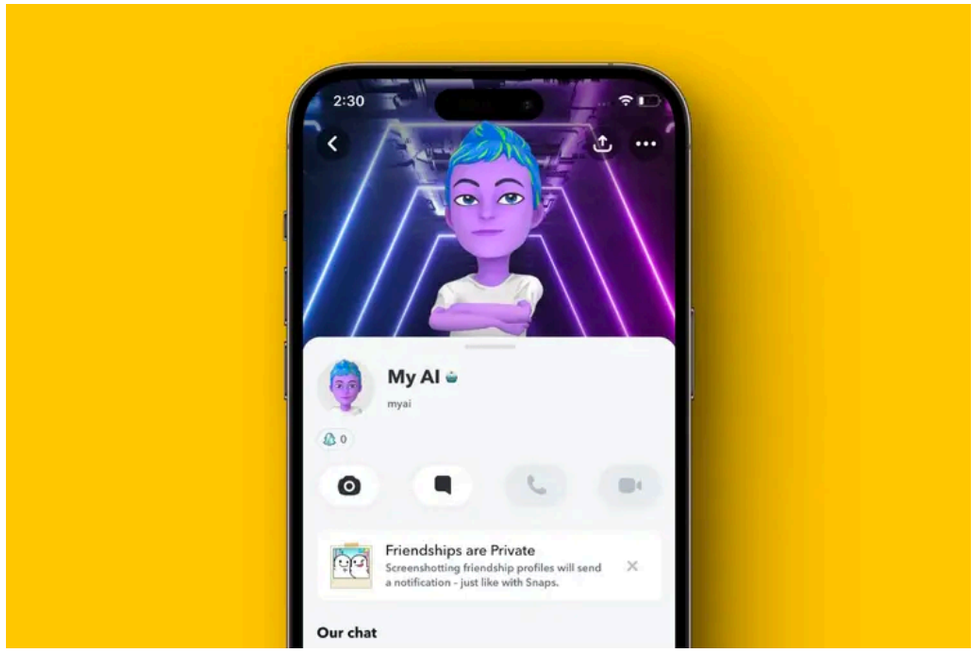
3. Vi kan bruke datamaskinen til å visualisere stokastiske forsøk for å bedre forstå hvordan innsamlede data *kan* komme til å se ut

Hvorfor er det nyttig at en datamaskin kan generere (pseudo-)tilfeldige tall?

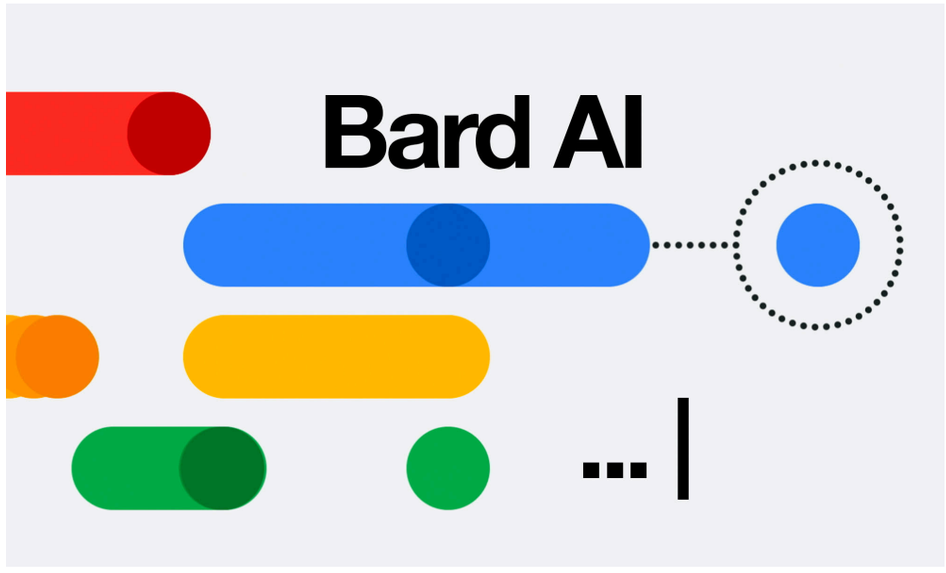
1. Vi kan bruke datamaskiner til å approksimere sannsynligheter i komplekse situasjoner (vanskelig/umulig å regne for hånd)
2. Vi kan bruke datamaskiner til å generere (fra en passende sannsynlighetsfordeling) tilfeldig "output" til brukere
3. Vi kan bruke datamaskinen til å visualisere stokastiske forsøk for å bedre forstå hvordan innsamlede data *kan* komme til å se ut

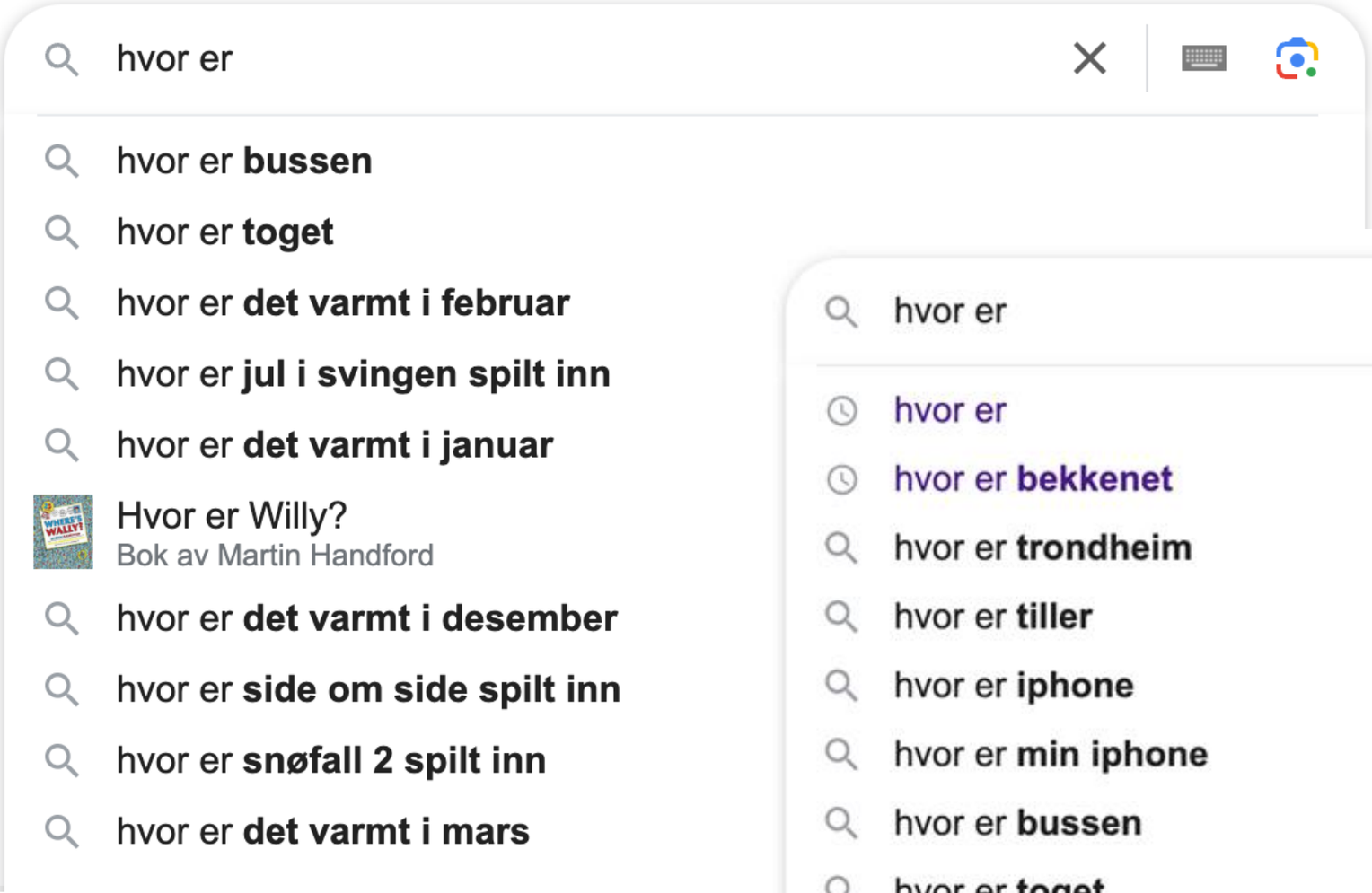
Eksempel 2: Språkmodeller



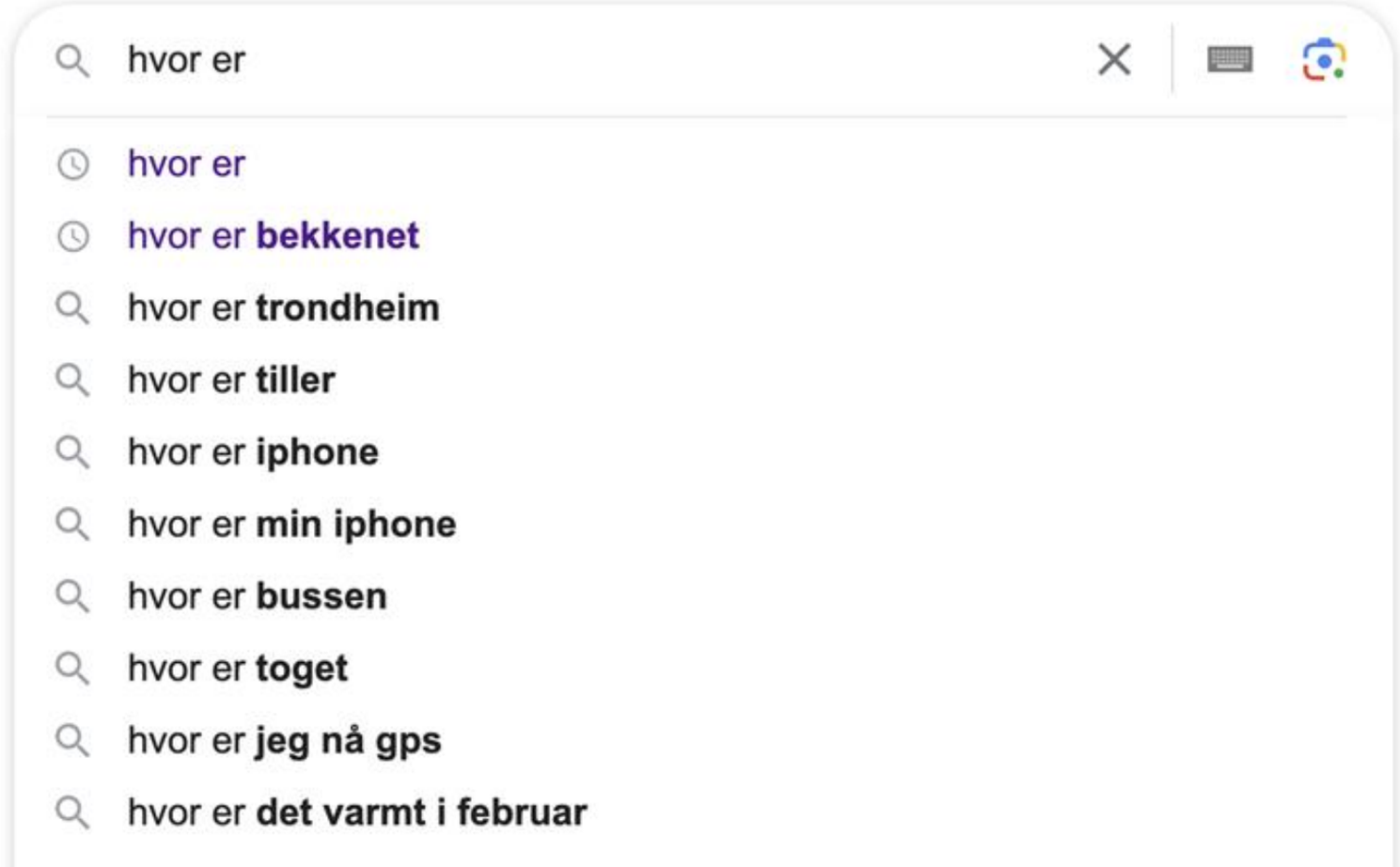


Bing





18. januar 2024



Ingeborg

Hvorfor er det nyttig at en datamaskin kan generere (pseudo-)tilfeldige tall?

1. Vi kan bruke datamaskiner til å approksimere sannsynligheter i komplekse situasjoner (vanskelig/umulig å regne for hånd)

2. Vi kan bruke datamaskiner til å generere (fra en passende sannsynlighetsfordeling) tilfeldig "output" til brukere

3. Vi kan bruke datamaskinen til å visualisere stokastiske forsøk for å bedre forstå hvordan innsamlede data *kan* komme til å se ut

Eksempel 3: Hvor mange søsken har dere?

https://www.ssb.no/befolkning/artikler-og-publikasjoner/slik-er-norske-barnefamilier

Statistikken er basert på barn under 18 år i 2018