Syfte

Syftet med dagens laborationen är att du skall:

- träna på olika sätt att illustrera och beskriva ett datamaterial i Python
- få förståelse för begreppen fördelningsfunktion, täthets- och sannolikhetsfunktion samt sambandet mellan stickprov och population
- träna på att beräkna sannolikheter i Python
- bli bekant med inversmetoden för att generera slumptal

Förberedelseuppgifter

- Förvissa dig om att du förstår vad täthetsfunktion och fördelningsfunktion är och hur de förhåller sig till varandra.
- Vad är en kvantil?
- Ta reda på hur man använder inversmetoden för att transformera slumptal till en önskad fördelning. (Kap.~8.4).

Python paket

Vi kommer använda följande paket i Python

- **numpy** Grundläggande linjär algebra och nummerik.
- **scipy.stats** För beräkningar av täthets- och fördelningsfunktioner.
- **pandas** För data hantering.
- **matplotlib.pyplot** Grundläggande plot-funktioner.
- **seaborn** Bättre plot-funktioner för pandas objekt.

Importera moduler och ladda upp filer till Colab

Börja med att importera ett antal Python paket (om du inte använder *colabs* kan du behöva installera paketen först)

```
import numpy as np
import scipy.stats as stats
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Utöver modulerna ovan använder laborationen data materialet i **kroppsTemp.csv**. För att komma åt data måste du:

1. Ladda ner **kroppsTemp.csv** från kurshemsidan
2. Klicka på mappen *Filer* till vänster i *google colab* menyn
3. Ladda upp **kroppsTemp.csv** genom att klicka på *Ladda upp till sessionens lagringsutrymme* (eller drag-n-drop filen)
4. Om du får ett **FileNotFoundError** när du försöker importera **kroppsTemp.csv** högerklicka på **kroppsTemp.csv** i google colab och *Kopiera sökväg*. Klistra sedan katalog namnet på platsen för **/content/kroppsTemp.csv**

```
#google colabs
T = pd.read_csv(r"/content/kroppsTemp.csv")
```

Om du använder Python lokal på datorn, använd istället följande för att läsa in **kroppsTemp.csv**.

```
#egen dator med kroppsTemp.csv i samma folder.
T = pd.read_csv("kroppsTemp.csv")
```

Relativa frekvenser och fördelningar

I denna del av laborationen ska vi använda ett datamaterial av kroppstemperatur hos 130 st friska 18--40 åringar (Mackowiak, Wasserman, Levine. (1992) *A critical appraisal of 98.6 degrees F, the upper limit of the normal body temperature*). Data finns i filen **kroppsTemp.csv** på kurshemsidan.

Kroppstemperatur

Läsa in och titta på data (kom ihåg att lägga csv-filen i samma katalog som ditt Python script, eller ange en lämplig relativ sökväg)

```
print(T)
print(T.describe())
```

Data består av två kolumner där den första är temperaturer (i C) och den andra indikerar om personen var man (=0) eller kvinna (=1).

En god regel, när man står inför ett nytt datamaterial, är att rita upp det på några olika sätt. Vi börjar med ett histogram, titta först på hjälp texten för att förstå hur funktionen fungerar

```
help( sns.histplot )
```

Vi använder nu funktionen för att göra ett histogram:

```
sns.histplot(T, x="Temperatur")
```

Vi kan också plotta data som en funktion av index

```
sns.scatterplot(T, x=T.index, y="Temperatur")
```

och relatera det till histogrammet. Här anger **T** vilket datamaterial att använda och **x=** och **y=** vilka kolumner i **T** som ska plottas på vilka axlar. Bättre är att lägga båda i samma figur

```
fig, axs = plt.subplots(1, 2, constrained_layout=True)
sns.scatterplot(T, x=?, y=?, ax=axs[0])
sns.histplot(T, x=?, ax=axs[1])
```

Här genererar **plt.subplots** två del figurer och **ax=** anger vilken delfigur respektive plot ska hamna i.

Mozquizto: Jämför histogrammet med ploten. Hur syns egenskaperna hos data i histogrammet, och tvärtom?

sns.scatterplot ser vi att data är grupperat i män och kvinnor. För att jämföra data mellan flera grupper kan vi dela data baserat på kön.

```
fig, axs = plt.subplots(1, 2, constrained_layout=True)
sns.scatterplot(T, x=?, y=?, hue=?, ax=axs[0])
sns.histplot(T, x=?, hue=?, ax=axs[1])
```

Genom att använda **displot** och **col** istället för **hue** för vi två histogram bredvid varandra, pröva.

```
sns.displot(T, x=?, col=?)
```

Ett annat alternativ för att jämföra data är att använda **sns.boxplot**

```
sns.boxplot(T, x=?, y=?)
```

Ett bättre alternativ till **sns.boxplot** är ibland **sns.violinplot**.

```
fig, axs = plt.subplots(1, 2, constrained_layout=True)
sns.boxplot(T, x=?, y=?, ax=axs[0])
sns.violinplot(T, x=?, y=?, ax=axs[1])
```

Mozquizto: Jämför histogram och box-/violinplot. Är det någon skillnad mellan män och kvinnor? Hur stor verkar skillnaden vara?

Vi kan också använda **groupby** funktionen i pandas för att titta på skillnader mellan män och kvinnor

```
print(T.groupby(?).describe().transpose())
```

Eftersom skillnaden inte är särskilt stor fortsätter vi att analysera alla data på en gång. Ett annat sätt är att rita en empirisk tähets- eller fördelningsfunktionen.

```
fig, axs = plt.subplots(1, 3, constrained_layout=True)
sns.scatterplot(T, x=?, y=?, ax=axs[0])
sns.histplot(T, x=?, stat=?, kde=True, ax=axs[1])
sns.ecdfplot(T, x=?, ax=axs[2])
```

Här anger **stat** vilken skalning av histogram vi vill ha:

- **count** Antal observationer i varje indelning.
- **frequency** Antalet observationer delat med intervall bredd.
- **probability/proportion** Andel observationer i varje indelning
- **density** Standardisera så att arean av histogrammet är 1 (dvs täthetsfunktion).

och **kde=True** anger att en ***k**ernal **d**ensity **e**stimation* (dvs skattning av tätheten) ska ritas in i histogrammet. Även här kan vi jämföra kvinnor och män

```
fig, axs = plt.subplots(1, 3, constrained_layout=True)
sns.scatterplot(T, x=?, y=?, hue=?, ax=axs[0])
sns.histplot(T, x=?, hue=?, stat=?, kde=True, ax=axs[1])
sns.ecdfplot(T, x=?, hue=?, ax=axs[2])
```

Uppgift: Jämför scatterplot, histogram/täthet och fördelningsfunktionerna. Hur hänger de ihop med varandra?

Uppgift: Välj ut några datapunkter i de olika figurerna och försök hitta dem i de andra figurerna.

I fördelningsfunktionen kan vi avläsa hur många av observationerna som är mindre än eller lika med ett visst tal.

Uppgift: Välj $x = 37$ och försök avgöra i figurerna (histogram/täthet och fördelningsfunktionerna) hur många av värdena som är mindre än eller lika med 37.

När antalet observationer i stickprovet ökar kan vi tolka kvoten som sannolikheten att få ett värde mindre än eller lika med x. Antalet och kvoten kan beräknas som:

```
print('Antal mindre än:', np.sum(T.Temperatur<=?), 'av', T.shape[0] )
print('Andel mindre än:', np.mean(T.Temperatur<=?) )
```

Uppgift: Stämmer det med din uppskattning från figuren?

För att förstå hur **T.Temperatur<=?** fungerar så jämför vi det med ursprungsdata:

```
print( T.Temperatur )
print( T.Temperatur<=? )
```

Vad är det som händer?

Mozquizto: Pröva med några andra värden på x. Hur borde andelen ändra sig när x ökar respektive minskar? Jämför med figuren.

Den omvända proceduren, hitta det värde x som motsvarar en given sannolikhet, dvs en given kvantil, är ofta viktigare. Vi återkommer till det lite senare.

Kommer data från en standardfördelning?

Histogrammet och den empiriska fördelningsfunktionen kan jämföras med täthet- och fördelningsfunktioner för standardfördelningar för att undersöka om någon sådan passar som modell för data.

Uppgift: Påminner histogrammet och fördelningsfunktionen om någon vanliga modell för stokastiska variabler?

Mozquizto: Innan vi kan jämföra data med standard fördelningar behöver vi beräkna medelvärde och stickprovsstandardavvikelse, använd att en **pandas dataframe** har funktioner för **mean** och **std**.

Spara värdena i variablerna **mu** och **sigma** de behövs senare.

```
mu = T.Temperatur.?()
sigma = T.Temperatur.?()
print("medelvärde=", mu  )
print("standardavvikelse=", sigma )
```

Vi kan nu jämföra histogrammet med täthetsfunktionen för en normalfördelning. För att beräkna normalfördelningen använder vi

```
help( stats.norm )
```

```
fig, axs = plt.subplots(1, 2, constrained_layout=True)
sns.histplot(T, x=?, stat=?, kde=True, ax=axs[0])
sns.ecdfplot(T, x=?, ax=axs[1])

#konstruera en vektor med 100 värden från minsta till största temperatur
x = np.linspace(min(T.Temperatur), max(T.Temperatur), 100)
#beräkna normaltäthet med de skattade parametrarna
p = stats.norm.pdf(x, ?, ?)
#och fördelningsfunktion med de skattade parametrarna
F = stats.norm.cdf(x, ?, ?)

#addera dessa till figuren
axs[0].plot(x,p,'r')
axs[1].plot(x,F,'r')
```

Ett alternativ är att jämföra empiriska och teoretiska kvantiler i data, om vi har rätt fördelnings antagande bör dessa ligga på en rak linje.

```
fig = stats.probplot(T.Temperatur, dist="norm", fit=True, plot=plt)
```

Här anger **dist="norm"** att vi är intresserade av en normalfördelning, **fit=True** att parametrarna ska skattas från data och **plot=plt** att **matlibplot** ska användas för att illustrerar data. Vi kan också jämföra data med rektangel- (uniform) och exponentialfördelningar:

```
fig = stats.probplot(T.Temperatur, dist=stats.uniform, fit=True, plot=plt)
```

```
fig = stats.probplot(T.Temperatur, dist=stats.expon, fit=True, plot=plt)
```

Mozquizto: Vilken fördelning verkar data komma från?

Större stickprov --- Fördelningsfunktionen för en slumpvariabel

En intressant fråga är hur storleken på datamaterialet påverkar hur bra anpassningarna kan förväntas bli. För att undersöka effekten av antalet observationer vill vi nu studera både ett större och ett mindre datamaterial, t.ex. 20 eller 2000 observationer från samma fördelning som tidigare. Eftersom vi bara har 130 observationer simulerar vi ny data

```
N = 20 #pröva 20, 200 eller 2000
data = stats.norm.rvs(?, ?, size=?)
#placera den resulterande vektorn som en kolumn i en DataFrame
data = pd.DataFrame({"x":data})

#samma plottar som tidigare (tänk på namnet av kolumnen i data)
fig, axs = plt.subplots(1, 2, constrained_layout=True)
sns.histplot(data, x=?, stat=?, kde=True, ax=axs[0])
sns.ecdfplot(data, x=?, ax=axs[1])

#addera exakta täthets- och fördelningsfunktioner till figuren
axs[0].plot(x,p,'r')
axs[1].plot(x,F,'r')
```

Vi kan också titta på kvantil ploten.

```
fig = stats.probplot(data.x, dist="norm", fit=True, plot=plt)
```

Uppgift: Hur förändras histogram och empirisk fördelningsfunktion när antalet observationer ökar? Hur bra stämmer dessa med sina teoretiska motsvarigheterna?

Uppgift: Vad blir nu andelen värden som är mindre än eller lika med 37? Hur mycket ändrar det sig mellan olika simuleringar?

```
print("Från simulerad data P(X<=37) = ", np.mean( stats.norm.rvs(?, ?, size=?) <= ? ) )
print("Från observationer data P(X<=37) = ", np.mean(T.Temperatur <= ?))
```

Eftersom resultatet närmare sig en normalfördelning kan vi beräkna andelen värden som är ≤ 37 som $\mathrm{P}(X \leq 37) = F_X(37)$ där $X \in \mathrm{N}(\mu, \sigma)$. Använd funktionen **stats.norm.cdf** för att beräkna $\mathrm{P}(X \leq 37)$.

```
print("P(X<=37) = ", stats.norm.cdf(?, ?, ?))
```

Mozquizto: Hur stämmer sannolikhets beräkningen med tidigare beräkningar av andelen värden som är mindre än eller lika med 37 (eller andra värden på x)?

Kvantiler

Begreppet **kvantil** är viktigt. Kvantilen kan definieras på olika sätt men vi (och många andra) använder följande definition: kvantilen är det tal x_α som uppfyller

$$\mathrm{P}(X \leq x_\alpha) = 1 - \alpha$$

där α är ett tal mellan 0 och 1 (vanliga val är: 0.05, 0.01, 0.001).

Mozquizto: Läs av kvantilen $x_{0.05}$ där $\alpha = 0.05$ ur dina figurer, med hjälp av definitionen ovan. Både som skattningar i de två empiriska fördelningsfunktionerna och exakt i den teoretiska.

Mozquizto: Jämför med det exakta värdet, som kan fås med funktionen **stats.norm.ppf**

```
help(stats.norm)
```

```
print("Kvantil från simulerad data:", np.quantile( stats.norm.rvs(?, ?, size=?), ?))
print("Teoretisk kvantil:", stats.norm.ppf(?, ?, ?) )
```

Uppgift: Hur mycket skiljer $x_{0.05}$ baserat på $50, 100, 500$ eller 2000 simulerade observationer?

Hur datasetets storlek påverkar osäkerheten i uppskattningarna kommer vi tillbaka till under hela resten av kursen.

Andra fördelningar

Vi ska nu rita upp några andra normalfördelningar, $\mathrm{N}(\mu, \sigma)$, och se hur de ändrar sig när vi ändrar på parametrarna μ och σ.

```
x = np.linspace(0, 10, 1000)

fig, axs = plt.subplots(1, 2, constrained_layout=True)
#N(2, 0.5) täthet
axs[0].plot(x, stats.norm.pdf(x, ?, ?))
#Plotta lite andra normaltätheter här, t.ex. N(7, 0.5), N(5, 2) & N(5, 0.2)
axs[0].plot(x, stats.norm.pdf(x, ?, ?))
axs[0].plot(x, stats.norm.pdf(x, ?, ?))
axs[0].plot(x, stats.norm.pdf(x, ?, ?))

#N(2, 0.5) fördelningsfunktionen
axs[1].plot(x, stats.norm.cdf(x, ?, ?))
#Plotta lite andra normalfördelningar här, t.ex. N(7, 0.5), N(5, 2) & N(5, 0.2)
axs[1].plot(x, stats.norm.cdf(x, ?, ?))
axs[1].plot(x, stats.norm.cdf(x, ?, ?))
axs[1].plot(x, stats.norm.cdf(x, ?, ?))
```

Uppgift: Vad händer med fördelningen när μ och σ ändras? Vad representerar μ och σ i fördelningen?

Uppgift: Fördelningsfunktionen är ju integralen av täthetsfunktionen.

Relatera dem till varandra i figuren. Hur ändrar sig, t.ex. fördelningsfunktionen när x ligger nära μ jämfört med när x ligger långt från μ? Hur ser täthetsfuktionen ut då (stor eller liten?)

Mozquizto: Experimentera med andra värden på μ och σ och se vad som händer. Du kan behöva ändra **x** för att för att få plats i figuren (tips: det allra mesta av en normalfördelning ryms inom $\mu \pm 4\sigma$).

Inversmetoden

Ett sätt att få fram slumptal från olika fördelningar är att använda inversmetoden. Då genereras först slumptal från en $\mathrm{U}(0, 1)$-fördelning. Dessa stoppas sedan in i inversen till den önskade fördelningens fördelningsfunktion, dvs i $F^{-1}(u)$.

I Python kan man få en vektor med n slumptal från en $\mathrm{U}(0, 1)$-fördelning med funktionen **stats.uniform**

```
help(stats.uniform)
```

Uppgift: Gör en vektor u med $n = 1000$ slumptal från denna fördelning och gör ett histogram och övertyga dig om att det verkar vara rätt fördelning.

```
n = 1000
u = stats.uniform.rvs(size=?)
print(u)

sns.histplot(x=?, stat="density")
```

Nu vill vi göra om dessa slumptal till att komma från $\mathrm{Exp}(\lambda)$-fördelning. Fördelningsfunktionen för en sådan fördelning är som bekant

$$F_X(x) = 1 - e^{-\lambda x}, \quad x \geq 0.$$

```
Lambda = ?
u = stats.uniform.rvs(size=?)
x = ?

sns.histplot(x=x, stat="density")
```

Mozquizto: Beräkna fördelningsfunktionens invers och använd den för att transformera slumptalen i vektorn **u** till att komma från en exponentialfördelning med $\lambda = 3$. (dvs lös ut x som funktion av u i $u = 1 - e^{-\lambda x}$.

Uppgift: Gör ett histogram över de exponentialfördelade slumptalen. Ser det rimligt ut? Gör även en empirisk fördelningsfunktion och se så att det verkar vara rätt fördelning.

Du kan också jämföra med direkt simulering från exponential fördelningen

```
Lambda = ?
#invers metoden
u = stats.uniform.rvs(size=?)
x = ?
#exakt simulering
x2 = stats.?.?(scale=?, size=?)

#jämförelse av simuleringarna
fig, axs = plt.subplots(2, 2, constrained_layout=True)
#våra rektangelfördelade slumptal
sns.histplot(x=u, stat="density", ax=axs[0,0])
#våra transformerade, exponential slumptal
sns.histplot(x=x, stat="density", ax=axs[0,1])
#Kvantilplot för exponential fördelning för att se om simuleringarna stämmer
stats.probplot(x, dist=stats.?, fit=True, plot=axs[1,0])
#simulerade exponential i ett histogram att jämföra med
sns.histplot(x=x2, stat="density", ax=axs[1,1])
```