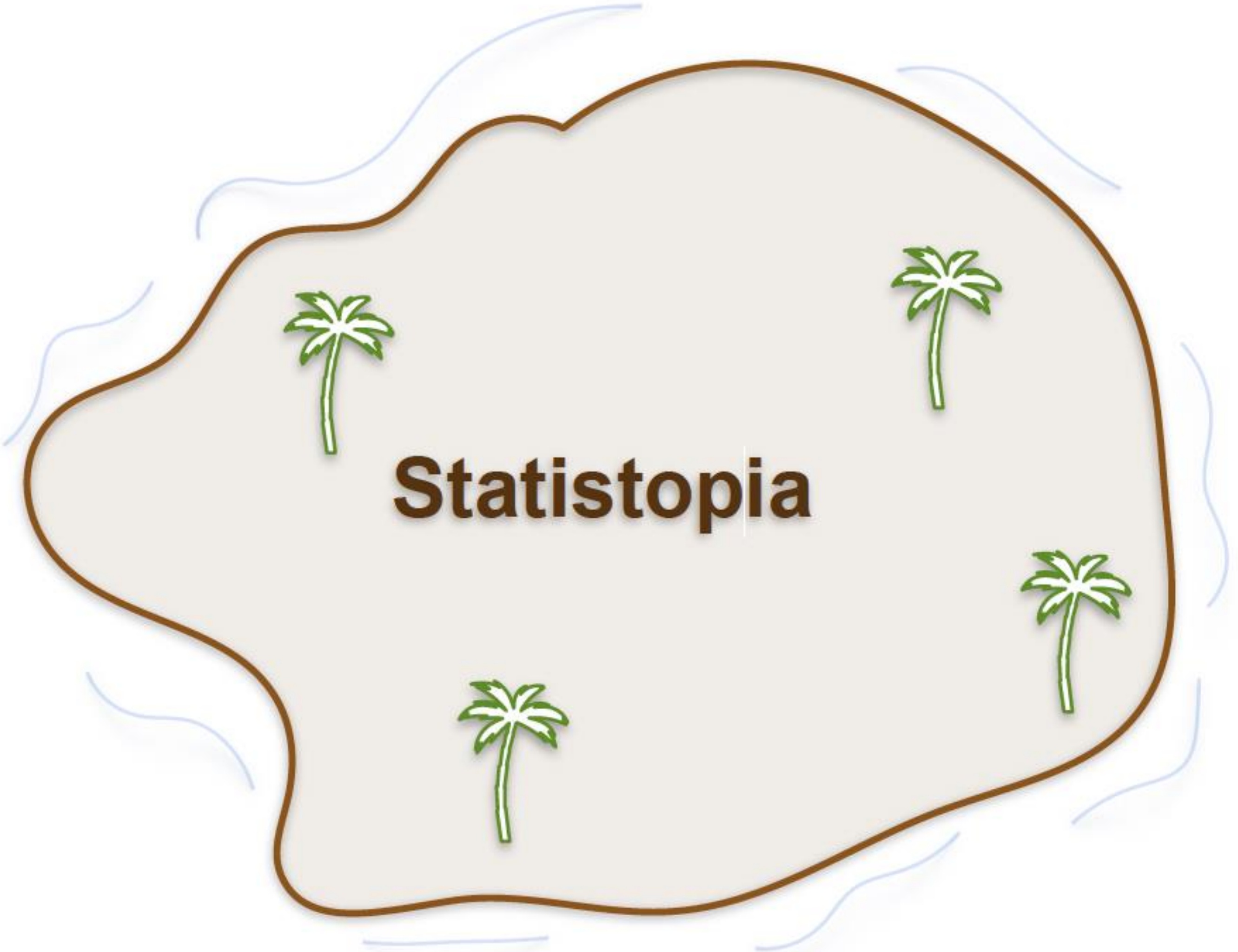
Sampling and Empirical Distribution

SAMPLING

Population and sample

Find the average height of

the people living on Statistopia.



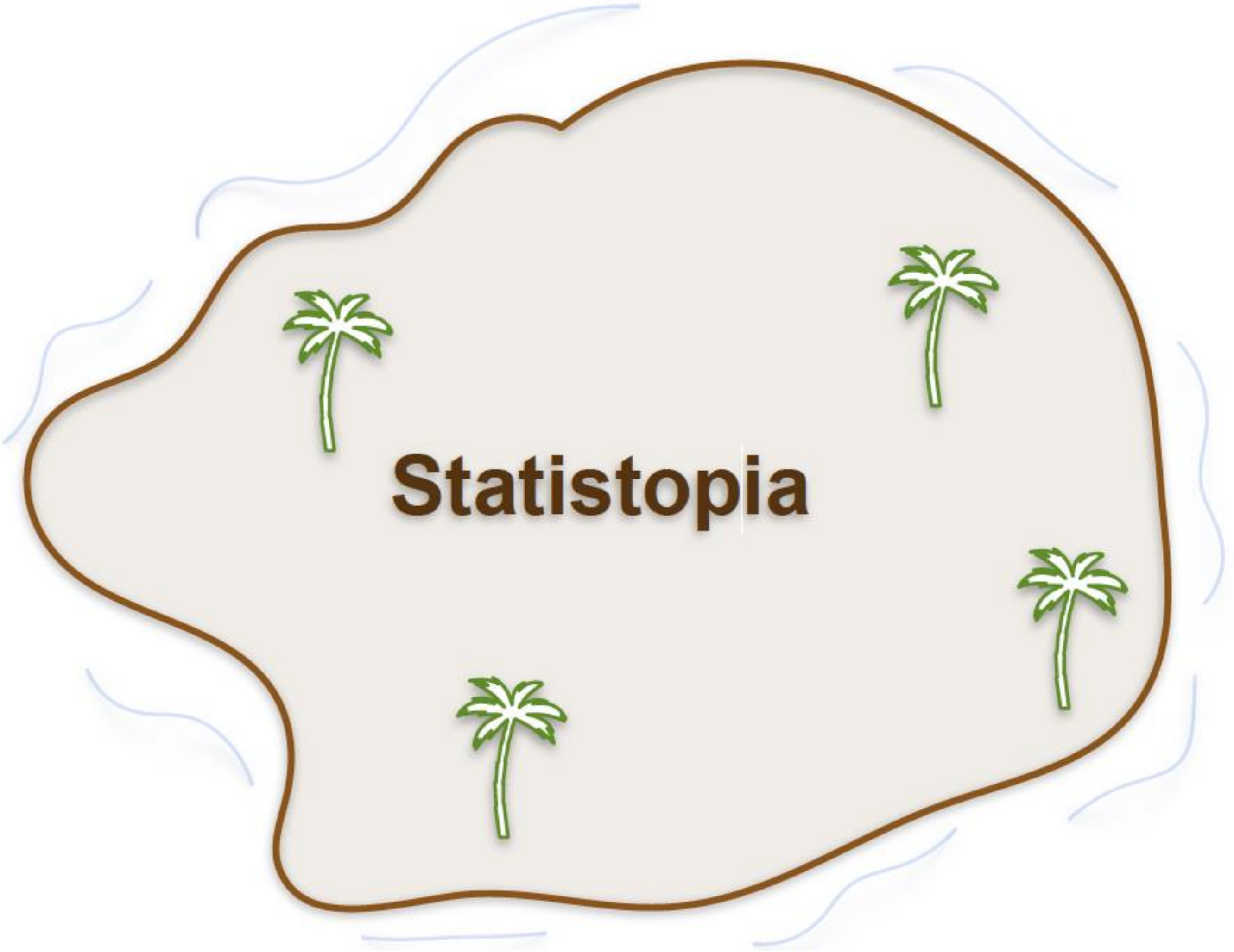
Population and sample

Find the average height of

the people living on Statistopia.

→ Ask everyone on the island for th
eir height.

→ Divide by the total number



Population and sample

Find the average height of

the people living on Statistopia.

→ Ask everyone on the island for th
eir height.

→ Divide by the total number

Population and sample

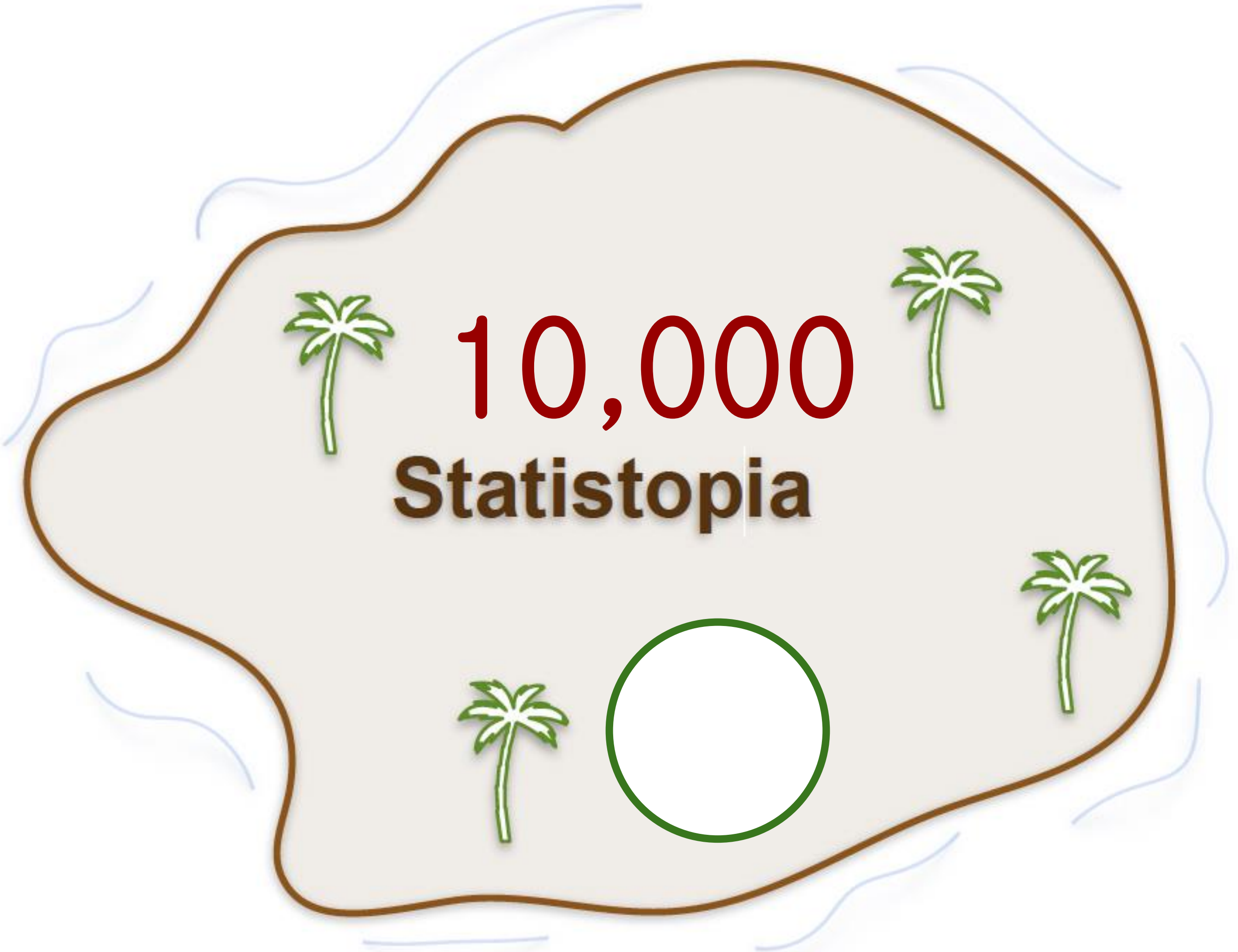
Find the average height of

the people living on Statistopia.

→ Ask everyone on the island for their height.

→ Divide by the total number

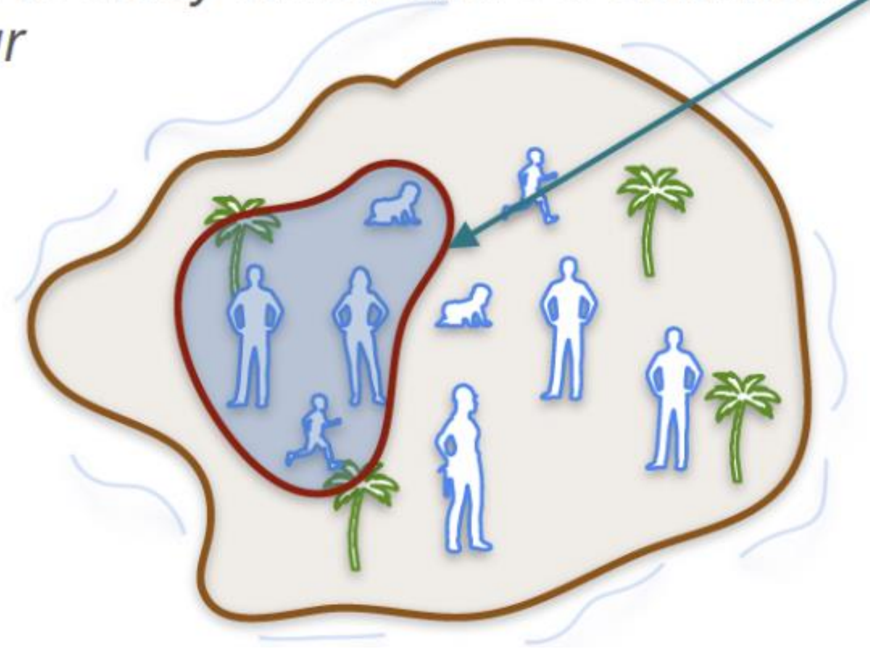
→ Only ask a subset of the group to estimate the average height



Population and sample

Population

the entire group of individuals or elements you want to study which share a common behaviour



Sample

subset of the population you use to draw conclusions about the population as a whole

Population Size: N

Sample Size: n

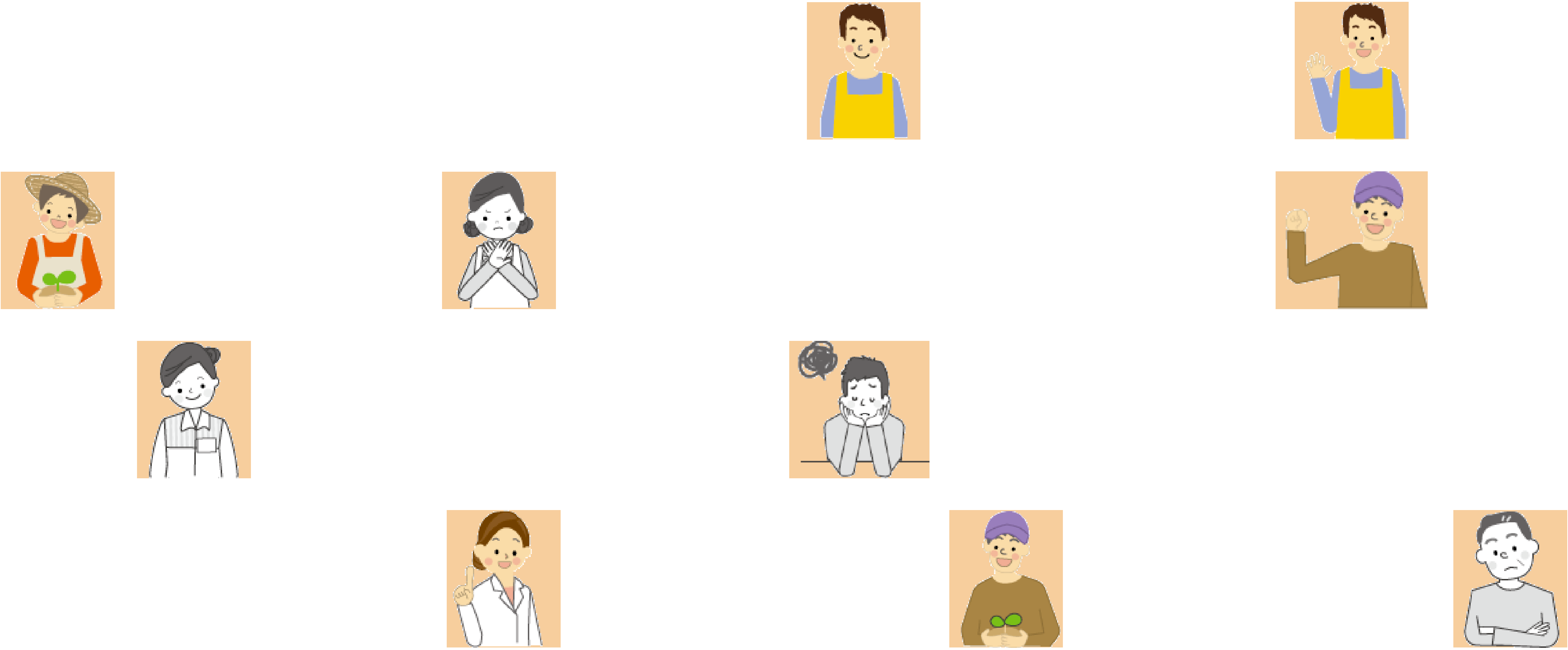
Population (모집단)

This is a population.

Sample (표본) (1/2)

A sample is selected from a population.

Sample (표본) (2/2)



A sample is selected from a population.

Dataset: Top Movies

- Import libs

```
import pandas as pd
import numpy as np
```

- Read csv to DataFrames

```
# upload actor.csv file
from google.colab import files
file_uploaded = files.upload()
```

```
top = pd.read_csv('top_movies_2017.csv')
top
```

	Title	Studio	Gross	Gross (Adjusted)	Year
0	Gone with the Wind	MGM	198676459	1796176700	1939
1	Star Wars	Fox	460998007	1583483200	1977
2	The Sound of Music	Fox	158671368	1266072700	1965
3	E.T.: The Extra-Terrestrial	Universal	435110554	1261085000	1982
4	Titanic	Paramount	658672302	1204368000	1997
...	...	...	...	...	...

Deterministic sample

- Deterministic sample:
  - simply specify which elements of a set you want to choose
  - Sampling scheme doesn't involve chance

```
# Deterministic Sample
top.iloc[[3, 18, 100], :]
```

	Title	Studio	Gross	Gross (Adjusted)	Year
3	E.T.: The Extra-Terrestrial	Universal	435110554	1261085000	1982
18	The Lion King	Buena Vista	422783777	792511700	1994
100	The Hunger Games	Lionsgate	408010692	452174400	2012

https://pandas.pydata.org/docs/reference/api/pandas.Series.str.contains.html

```
# Deterministic Sample
top.loc[top.Title.str.contains('Harry Potter', regex=False)]
```

	Title	Studio	Gross	Gross (Adjusted)	Year
74	Harry Potter and the Sorcerer's Stone	Warner Brothers	317575550	497066400	2001
114	Harry Potter and the Deathly Hallows Part 2	Warner Brothers	381011219	426630300	2011
131	Harry Potter and the Goblet of Fire	Warner Brothers	290013036	401608200	2005
133	Harry Potter and the Chamber of Secrets	Warner Brothers	261988482	399302200	2002
154	Harry Potter and the Order of the Phoenix	Warner Brothers	292004738	377314200	2007
175	Harry Potter and the Half-Blood Prince	Warner Brothers	301959197	359788300	2009
177	Harry Potter and the Prisoner of Azkaban	Warner Brothers	249541069	357233500	2004

Random sample

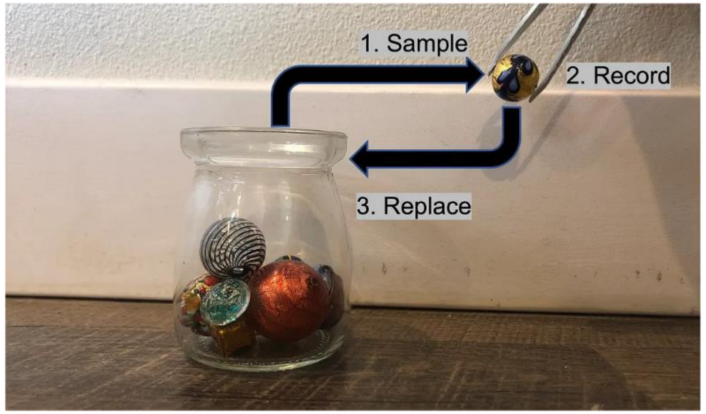
- Random sample:
  - Before the sample is drawn, you have to know the selection probability of every group of people in the population
  - Not all individuals / groups have to have equal chance of being selected

```
"""Choose a random start among rows 0 through 9;
then take every 10th row."""

start = np.random.choice(np.arange(10))
top.iloc[np.arange(start, len(top), 10), :]
```

	Title	Studio
9	Snow White and the Seven Dwarves	Disney
19	The Sting	Universal
29	Thunderball	United Artists
39	Home Alone	Fox
49	Bambi	RKO
59	The Greatest Showman on Earth	Paramount
69	Tootsie	Columbia

Random Sample: Two methods

- 1. Random sampling with replacement
  - the default behavior of np.random.choice when it samples from an array



- 2. Simple random sample
  - a sample drawn at random *without* replacement
  - To use np.random.choice for simple random sampling
    - Set replace=False

Image sources: https://towardsdatascience.com/understanding-sampling-with-and-without-replacement-python-7aff8f47ebe4

Examples of Sampling without Replacement

- train test split procedure in Machine Learning

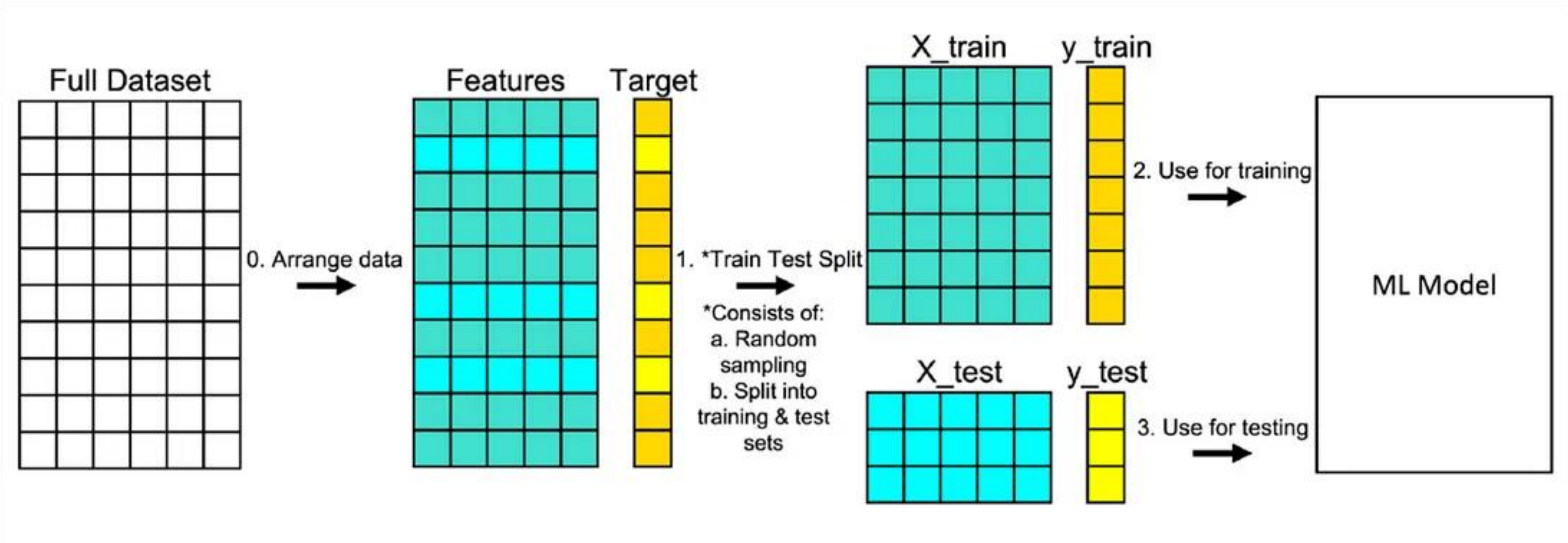


Image sources: https://towardsdatascience.com/understanding-sampling-with-and-without-replacement-python-7aff8f47ebe4

Sample of Convenience

- Example: sample consists of whoever visits your store
- Just because you think you're sampling "randomly", doesn't mean you have a random sample.
- If you can't figure out ahead of time
  - what's the population
  - what's the chance of selection, for each group in the population

→then you don't have a random sample

Sampling Methods

- Probability Sampling
  - Systemetic Sampling
  - Random Sampling
    - with or without replacement
- Non-Probability Sampling
  - Deterministic Sampling
  - Convenience Sampling

DISTRIBUTIONS

Probability Distribution

- Defined over a random quantity

- **Probability distribution**:
  - For all the possible values of the quantity
  - The probability of each of those values

- If you can do the math, you can work out the probability distribution without simulation (analytically)
- But... simulation is often easier and **requires less assumptions**!

Probability Distribution

Distribution	Parameters	Possible Description	Range Ω_X	$\mathbb{E}[X]$	$\mathsf{Var}(X)$	PDF ($f_X(x)$ for $x \in \Omega_X$)	CDF ($F_X(x) = \mathbb{P}(X \leq x)$)
Uniform	$X \sim \text{Unif}(a,b)$ for $a < b$	Equally likely to be any real number in $[a,b]$	$[a,b]$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$	$\frac{1}{b-a}$	$\begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } a \leq x < b \\ 1 & \text{if } x \geq b \end{cases}$
Exponential	$X \sim \text{Exp}(\lambda)$ for $\lambda > 0$	Time until first event in Poisson process	$[0,\infty)$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$	$\lambda e^{-\lambda x}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 - e^{-\lambda x} & \text{if } x \geq 0 \end{cases}$
Normal	$X \sim \mathcal{N}(\mu, \sigma^2)$ for $\mu \in \mathbb{R}$, and $\sigma^2 > 0$	Standard bell curve	$(-\infty,\infty)$	μ	σ^2	$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	$\Phi\left(\frac{x-\mu}{\sigma}\right)$
Gamma	$X \sim \text{Gam}(r, \lambda)$ for $r, \lambda > 0$	Sum of r iid $\text{Exp}(\lambda)$ rvs. Time to r^{th} event in Poisson process. Conjugate prior for Exp, Poi parameter λ	$[0,\infty)$	$\frac{r}{\lambda}$	$\frac{r}{\lambda^2}$	$\frac{\lambda^r}{\Gamma(r)} x^{r-1} e^{-\lambda x}$	Note: $\Gamma(r) = (r-1)!$ for integers r.
Beta	$X \sim \text{Beta}(\alpha, \beta)$ for $\alpha, \beta > 0$	Conjugate prior for Ber, Bin, Geo, NegBin parameter p	$(0,1)$	$\frac{\alpha}{\alpha+\beta}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$	
Dirichlet	$\mathbf{X} \sim \text{Dir}(\alpha_1, \alpha_2, \ldots, \alpha_r)$ for $\alpha_i, r > 0$ and $r \in \mathbb{N}, \alpha_i \in \mathbb{R}$	Generalization of Beta distribution. Conjugate prior for Multinomial parameter $\mathbf{p}$	$x_i \in (0,1)$; $\sum_{i=1}^r x_i = 1$	$\mathbb{E}[X_i] = \frac{\alpha_i}{\sum_{j=1}^r \alpha_j}$		$\frac{1}{B(\alpha)} \prod_{i=1}^r x_i^{\alpha_i - 1}$, $x_i \in (0,1), \sum_{i=1}^r x_i = 1$	
Multivariate Normal	$\mathbf{X} \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$	Generalization of Normal distribution	$\mathbb{R}^n$	$\boldsymbol{\mu}$	$\boldsymbol{\Sigma}$	$\frac{1}{(2\pi)^{n/2}\lvert\Sigma\rvert^{1/2}} \cdot \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))$	

Empirical Distribution

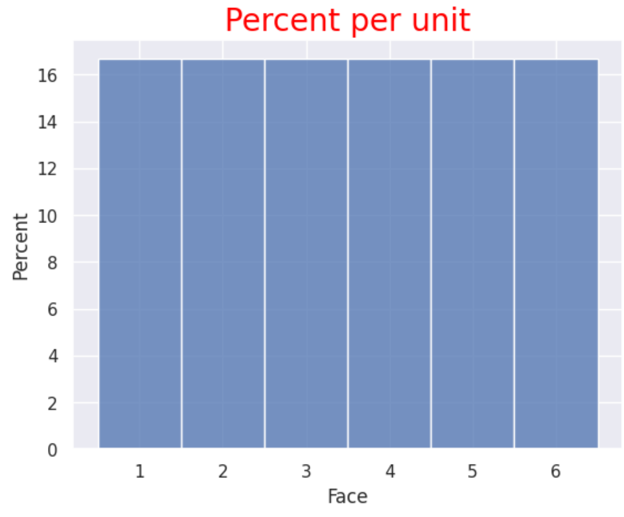
- **Empirical**: based on observations

- Observations can be from repetitions of an experiment

- **Empirical Distribution**:
  - All observed unique values
  - The proportion of times each value appears

One dice roll

```
# Define all possible results
die = pd.DataFrame({'Face': np.arange(1, 7, 1)})
die
```




```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_theme(style="darkgrid")
fig, ax = plt.subplots()
ax=sns.histplot(data=die, x="Face", bins=die_bins, stat="percent"
, discrete=True )

ax.set_title("Percent per unit", size=20, color="red")
plt.show()
```

Simulation: Roll dice (1/3)

```
# Numpy np.random.choice
# rng = random number generator, seed 2003
rng = np.random.default_rng(2023)
rng.choice(die.Face, 10, replace=True)
```

```
# Pandas pd.DataFrame.sample
die.sample(n=10, replace=True, random_state=2023)
```


https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html

Simulation: Roll dice (2/3)

- Define empirical_hist_die function

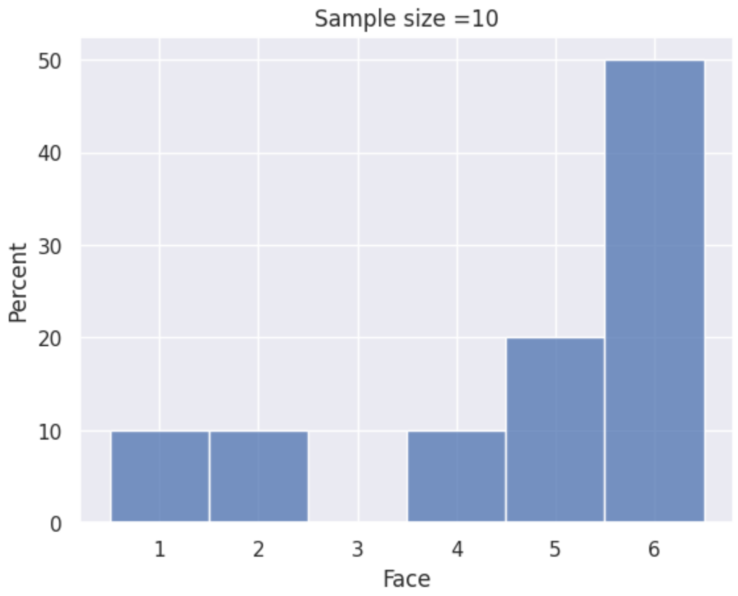
```
# Define Empirical Histogram Function
def empirical_hist_die(n):
    # Run one experiment
    die_sample = die.sample(n=n, replace=True, random_state=2023)

    # set die bins
    die_bins = np.arange(0.5, 6.6, 1)
    # Define Empirical Histogram
    sns.set_theme(style="darkgrid")
    fig, ax = plt.subplots()
    ax=sns.histplot(data=die_sample, x="Face", bins=die_bins, stat="
percent", discrete=True )

    ax.set_title("Sample size =" + str(n))
    plt.show()
```

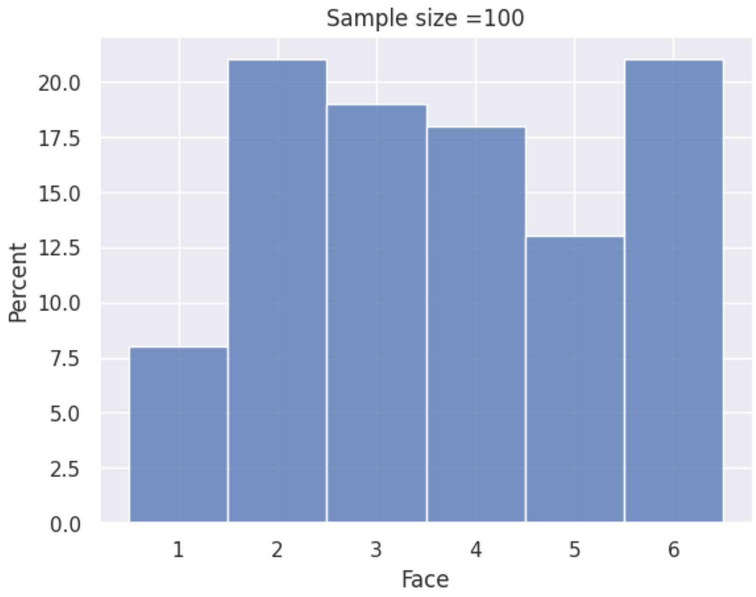
Simulation: Roll dice (3/3)

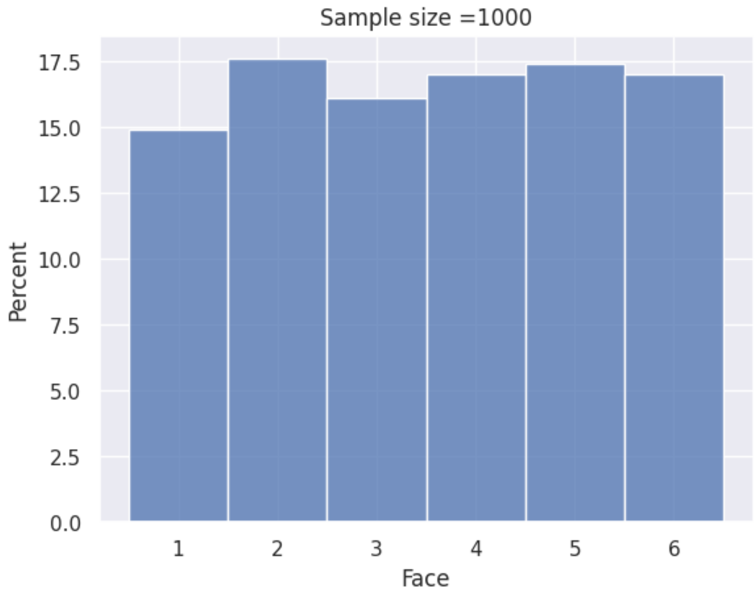
```
empirical_hist_die(10)
```

```
empirical_hist_die(100)
```

```
empirical_hist_die(1000)
```







LARGE RANDOM SAMPLES

Law of Averages / Law of Large Numbers

Law of Averages / Law of Large Numbers

If a chance experiment is **repeated many times**, **independently** and under the **same conditions**,
then
the **Empirical Distribution** gets closer to the
Probability Distribution.

- Example
  - As you increase the number of rolls of a die, the proportion of times you see the face with five spots gets closer to 1/6

Empirical Distribution of a Sample

- If the sample size is large, then
  - the empirical distribution of a uniform random sample resembles the distribution of the population, with high probability

Dataset: United airline delay

- Read csv to DataFrames

```
# upload actor.csv file
from google.colab import files
file_uploaded = files.upload()
```

```
united = pd.read_csv('united_summer2015.csv')
united
```

- Information

```
united.Delay.min()
```

```
united.Delay.max()
```

Manual histogram (1/2)

- Create a bins

```
delay_bins = np.append(np.arange(-20, 301, 10), 600)
delay_bins
```

```
array([-20, -10,   0,  10,  20,  30,  40,  50,  60,  70,  80,  90, 100,
       110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230,
       240, 250, 260, 270, 280, 290, 300, 600])
```

- Dataframe Bins

```
# Construct bins and Delay Count
bins = united.groupby(pd.cut(united['Delay'], delay_bins, right=False))
bins = pd.DataFrame(bins['Delay'].count().reset_index(name='Delay Count'))
bins.rename(columns={'Delay': 'bins'}, inplace=True)
bins
```

	bins	Delay Count
0	[-20, -10)	55
1	[-10, 0)	4994
2	[0, 10)	4059
3	[10, 20)	1445
4	[20, 30)	773
5	[30, 40)	590
6	[40, 50)	357

https://pandas.pydata.org/docs/reference/api/pandas.cut.html

Manual histogram (2/2)

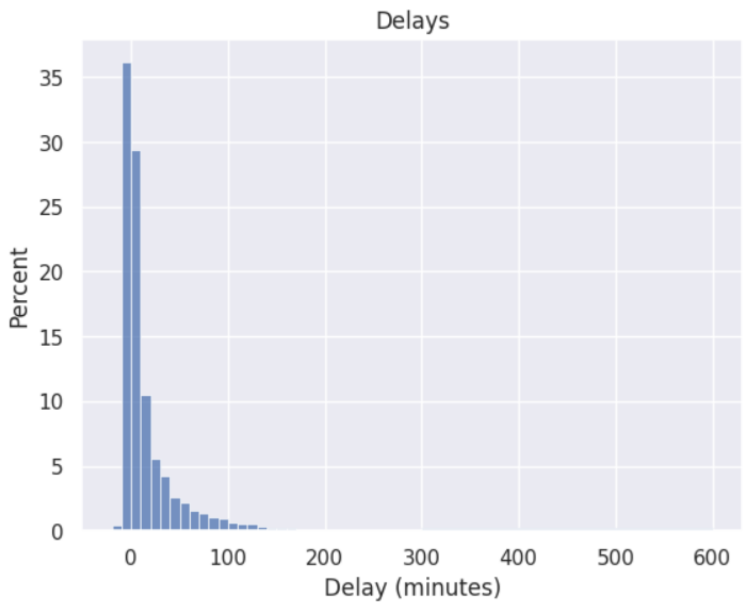
- Compute a percent to each bin

```
# Define population. Since we only extract -
20 < Delay < 201, we set this part of data as population.
data_amount = len(united[(-20 < united['Delay']) & \
        (united['Delay'] < 600)])

# Add percent data to bins
bins['Percent %'] = round(bins['Delay Count']/data_amount*100, 3)
bins
```

	bins	Delay Count	Percent %
0	[-20, -10)	55	0.398
1	[-10, 0)	4994	36.123
2	[0, 10)	4059	29.360
3	[10, 20)	1445	10.452
4	[20, 30)	773	5.591
5	[30, 40)	590	4.268
6	[40, 50)	357	2.582
7	[50, 60)	301	2.177

Draw a histogram using Seaborn

```
delay_bins = np.append(np.arange(-20, 301, 10), 600)
# draw histogram
fig, ax = plt.subplots()
ax=sns.histplot(data=united, x="Delay", \
      bins=delay_bins, stat="percent", discrete=False)
ax.set_ylabel("Percent")
ax.set_xlabel("Delay (minutes)")
ax.set_title("Delays")

plt.show()
```



Distribution of the Sample (1/2)

- Define empirical_hist_delay

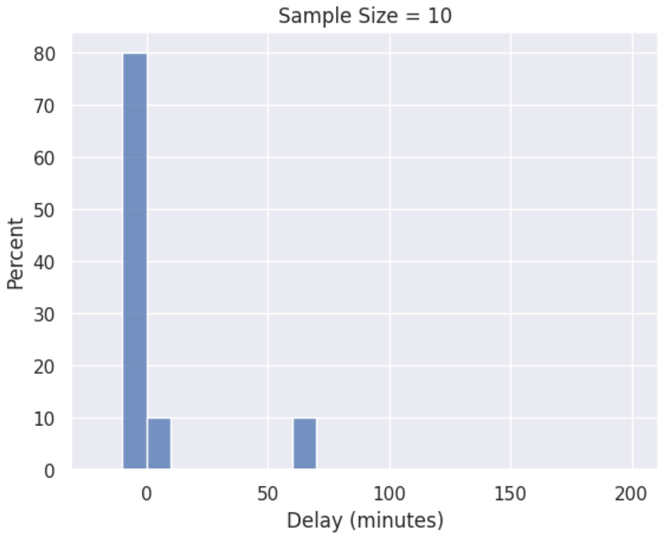
```
# Define Empirical Histogram Function
def empirical_hist_delay(n):
    # Extract sample
    united_sample = united.sample(n=n, replace=True, random_state=2023)

    # set delays_bins, only consider minutes in  -20 < delay <= 200
    delay_bins = np.arange(-20, 201, 10)
    # draw histogram
    fig, ax = plt.subplots()
    ax=sns.histplot(data=united_sample, x="Delay", \
       bins=delay_bins, stat="percent", discrete=False)
    ax.set_ylabel("Percent")
    ax.set_xlabel("Delay (minutes)")
    ax.set_title('Sample Size = ' + str(n))

    plt.show()
```

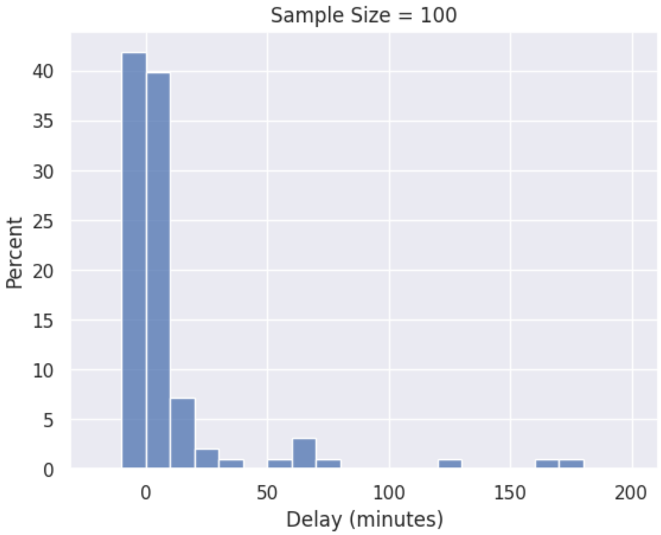
Distribution of the Sample (2/2)

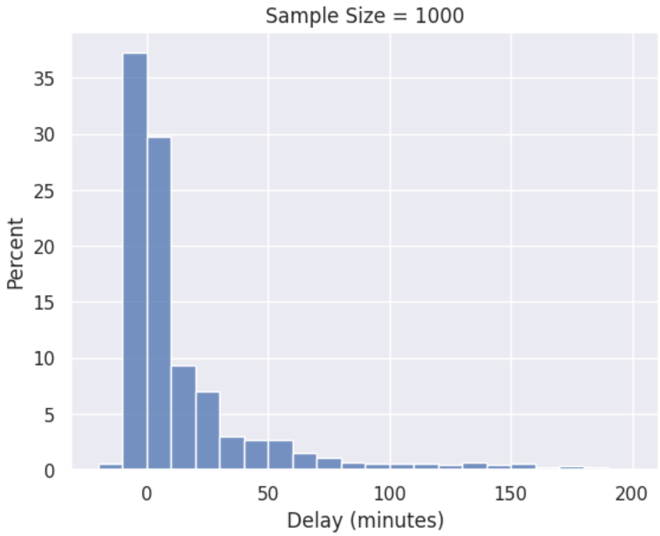
```
empirical_hist_delay(10)
```

```
empirical_hist_delay(100)
```

```
empirical_hist_delay(1000)
```



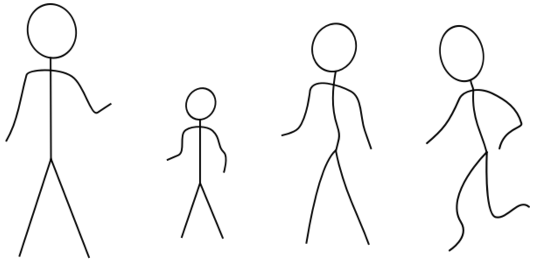




INFERENCE AND STATISTICS

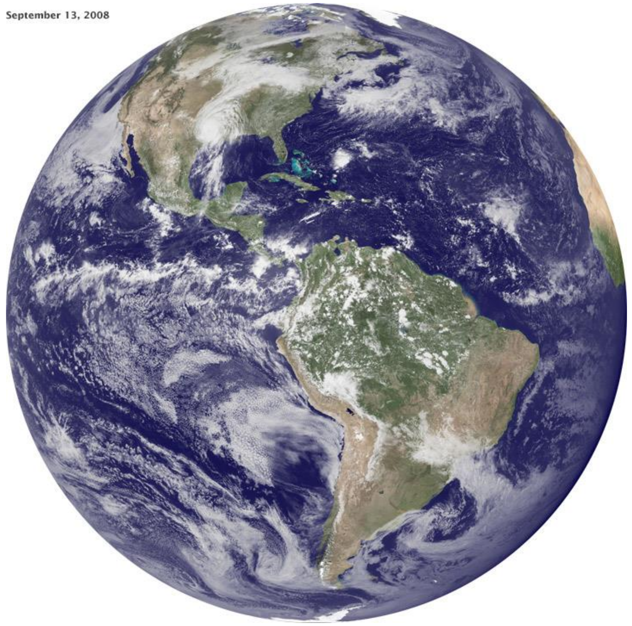
Inference

Sample Data



Inference

Population



Inference

- Statistical Inference:
  - Making conclusions based on data in random samples

- Example:

fixed

Use the data to guess the value of an **unknown number**

Create an **estimate** of the unknown quantity

depends on the **random** sample

Terminology

- Parameter
  - A number associated with the population
- Statistic
  - A number calculated from the sample

A *statistic* can be used as an **estimate** of a *parameter*

Demo: Parameter

- Load dataset as a population

```
united = pd.read_csv('united_summer2015.csv')
united
```

- Parameters

```
# Median of Delay
np.median(united.Delay)
```

```
# Percentage of Delay that is lower or equal than 2.0
len(united.loc[united.Delay <= 2.0])/len(united)
# 0.5018444846292948
```

```
# Amount of Delay that equal than 2.0
len(united.loc[united.Delay == 2.0])
```

Demo: Statistic

- Define sample

```
# Define sample_1000
sample_1000 = united.Delay.sample(n=1000, \
      replace=True)
```

- Statistic

```
# Median of smaple_1000
np.median(sample_1000)
```

```
# Median of another 1000 sample from united
np.median(united.sample(n=1000, replace=True).Delay)
```

Probability Distribution of a Statistic

- Values of a statistic vary because of random samples
- Probability (Sampling) Distribution of a statistic:
  - All possible values of the statistic,
  - and all the corresponding probabilities

- Often challenging to calculate analytically
  - Either have to do the math (may not be possible…)
  - Or generate all possible samples and calculate the statistic based on each sample (lots of compute!)

Empirical Distribution of a Statistic

- **Empirical distribution** of the **statistic**:
  - Based on **simulated values** of the statistic
  - Consists of all the **observed values** of the statistic,
  - and the **proportion of times** each value appeared

- Good approximation to the probability distribution of the statistic
  - if the number of repetitions in the simulation is large

Simulating a Statistic (1/3)

- **Step 1: Decide which statistic to simulate**
  - the median of a random sample of size 1000 drawn from the population of flight delays

- **Step 2: Write the code to generate one value of the statistic.**

```
# Define function to generate one value of the statistic
def random_sample_median(size):
    return np.median(united.sample(n=size, replace=True).Delay)
```

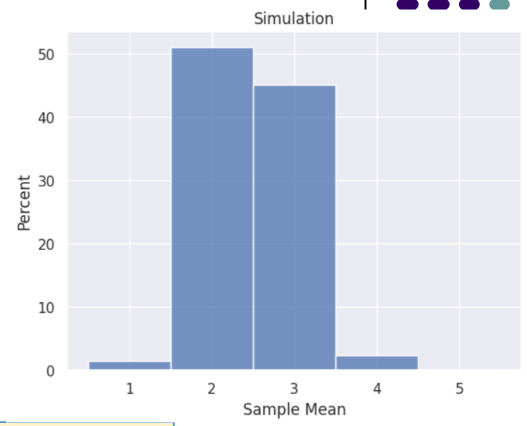
Simulating a Statistic (2/3)

- **Step 3: Decide how many simulated values to generate**
  - 5,000 repetitions

- **Step 4: Write the code to generate an array of simulated values.**

```
# Create Array medians to store repitions
medians = np.array([])

# Repeat the function 5,000 times.
for i in np.arange(5000):
    medians = np.append(medians, random_sample_median(1000))
```

Simulating a Statistic (3/3)



- Visualization
  - Create a Dataframe

```
simulated_medians = pd.DataFrame({'Sample Median': medians})
simulated_medians
```

  - Draw a histogram

```
median_bins = np.arange(0.5, 5, 1)
fig, ax = plt.subplots()
ax=sns.histplot(data=simulated_medians, x="Sample Median", \
      bins=median_bins, stat="percent", discrete=True)
ax.set_ylabel("Percent")
ax.set_xlabel("Sample Mean")
ax.set_title("Simulation")
plt.show()
```

Q&A