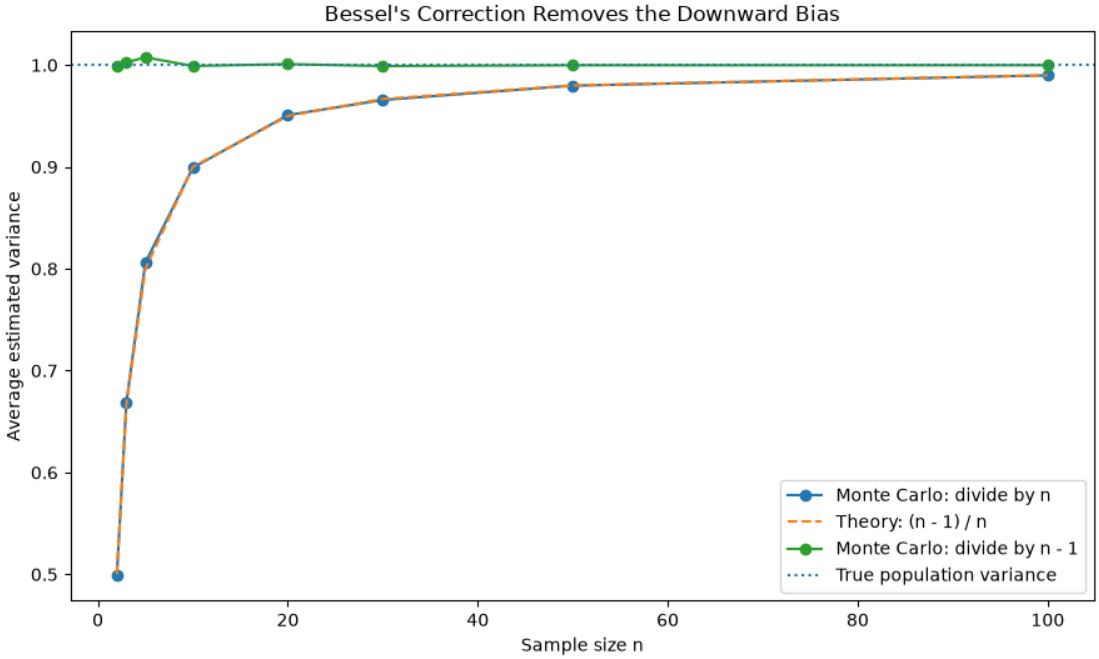
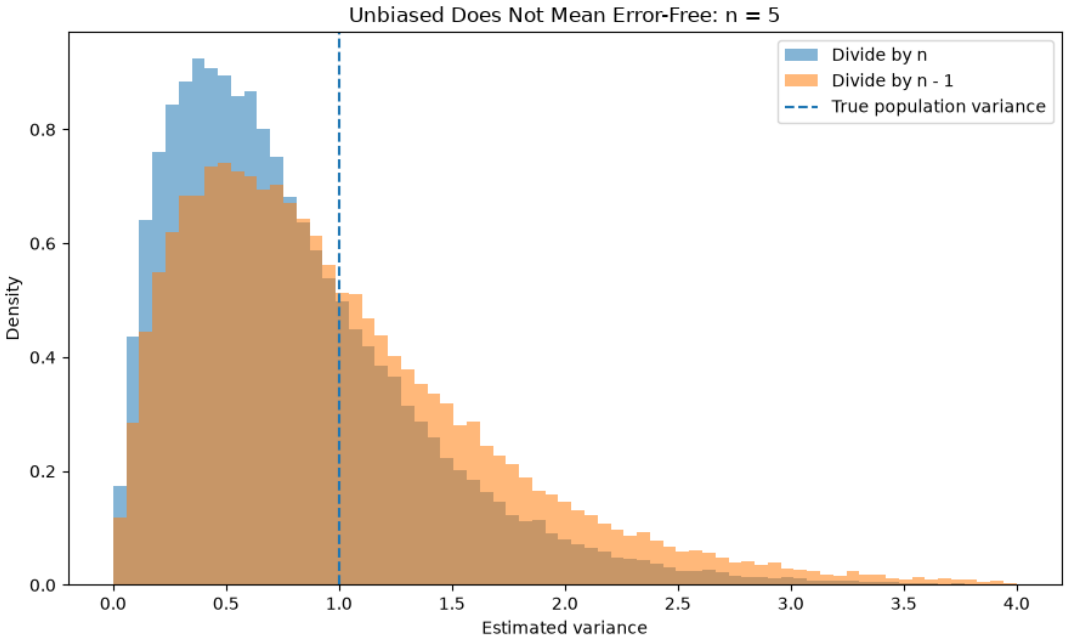
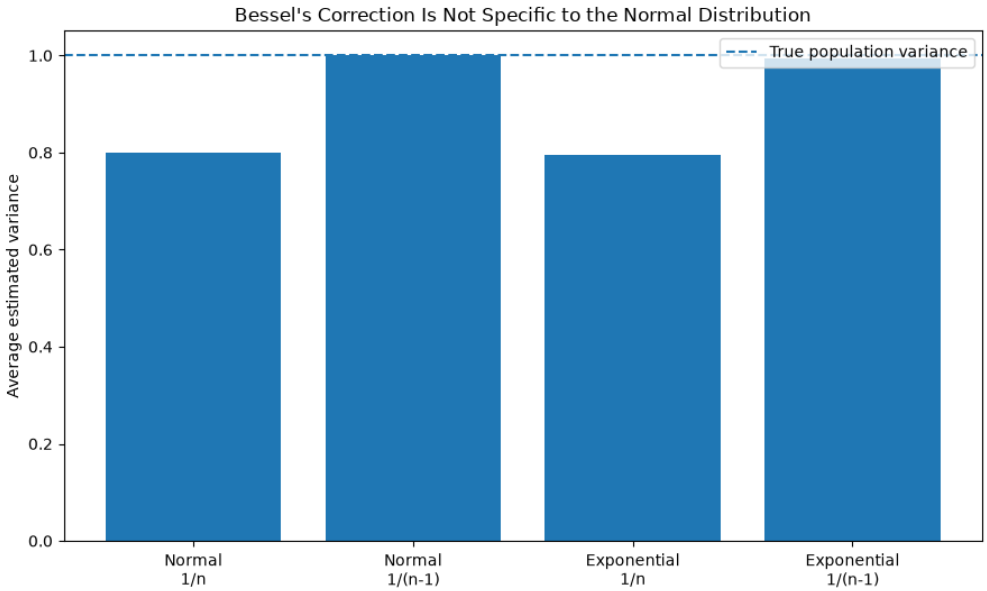
Output as explained in the video







The standard deviation of the sample and Bessel's correction

The **standard deviation**, named by Pearson, is the most important measure of dispersion since it considers all values of a distribution and not only two extreme values as the range does. The standard deviation can be interpreted as the average scatter around the mean for a ratio- or interval-scaled variable. However, there are different standard deviations for the population and the sample. The **standard deviation of the population**—denoted by the Greek letter σ (sigma)—is calculated as the square root of the squared differences around the mean divided by the size of the population *N*:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (X_i - \mu)^2}{N}}$$

The **standard deviation of the sample** is denoted by *s* and computed as the square root of the squared differences around the mean divided by the sample size *minus 1*:

$$s = \sqrt{\frac{\sum_{i=1}^{N} (X_i - \bar{X})^2}{n-1}}$$

It should be noted that in contrast to the standard deviation of the population, the sum of squares is divided by n – 1 (the so-called Bessel's correction, named after Friedrich Wilhelm Bessel), which refers to the concept of the degrees of freedom. The reason is that, once the sample mean has been computed, only n – 1 of the deviations can vary freely—the last one is determined by the others; hence the term degrees of freedom.

> "Using *n* – 1 in the denominator … solves a problem in inferential statistics associated with generalizations from samples to populations. The adequacy of these generalizations usually depends on *accurately* estimating unknown variability in the population with known variability in the sample. But if we were to use *n* rather than *n* – 1 in the denominator of our estimates, they would tend to underestimate variability in the population because n is too large. This tendency would compromise any subsequent generalizations, such as whether observed mean differences are real or merely transitory. On the other hand, when the denominator is made smaller by using *n* – 1, variability in the population is estimated more accurately, and subsequent generalizations are more likely to be valid." (Witte & Witte, 2017, pp. 73 et seq.)

A related but distinct quantity is the standard error of the sample mean, which measures how strongly sample means scatter around the population mean.

References:

Witte, R. S., & Witte, J. S. (2017). Statistics (11th ed.). Wiley.

Python

```
"""
Marcus Oehlrich
Code developed with assistance from ChatGPT (OpenAI)
Bessel's Correction
CS109 Probability Challenge

"""

import numpy as np
import matplotlib.pyplot as plt


# ------------------------------------------------------------
# Settings
# ------------------------------------------------------------

RNG = np.random.default_rng(109)

TRUE_VARIANCE = 1.0
N_SIMULATIONS = 50_000
SAMPLE_SIZES = [2, 3, 5, 10, 20, 30, 50, 100]


# ------------------------------------------------------------
# Core functions
# ------------------------------------------------------------

def variance_estimators(samples):
  """
  Compute two variance estimators for each row of 'samples'.

  samples shape:
    (number_of_simulations, sample_size)

  Returns
  -------
  variance_n:
    (1/n) * sum((Xi - Xbar)^2)

  variance_n_minus_1:
    (1/(n-1)) * sum((Xi - Xbar)^2)
  """

  n = samples.shape[1]
  sample_means = np.mean(samples, axis=1, keepdims=True)
  squared_deviations = (samples - sample_means) ** 2
  sum_squared_deviations = np.sum(squared_deviations, axis=1)

  variance_n = sum_squared_deviations / n
  variance_n_minus_1 = sum_squared_deviations / (n - 1)

  return variance_n, variance_n_minus_1


def simulate_for_sample_size(n, distribution="normal"):
```

```
    """
    Draw many samples of size n from a population with variance 1.
    """

    if distribution == "normal":
        # N(0, 1), so population variance = 1
        samples = RNG.normal(
            loc=0.0,
            scale=1.0,
            size=(N_SIMULATIONS, n)
        )

    elif distribution == "exponential":
        # Exponential(scale=1), so population variance = 1
        samples = RNG.exponential(
            scale=1.0,
            size=(N_SIMULATIONS, n)
        )

    else:
        raise ValueError("Unknown distribution")

    return variance_estimators(samples)


# ------------------------------------------------------------
# Part 1: Bias as a function of sample size
# ------------------------------------------------------------

mean_variance_n = []
mean_variance_n_minus_1 = []
theoretical_variance_n = []

print("\nBESSEL'S CORRECTION MONTE CARLO EXPERIMENT")
print("-" * 67)
print(f"{'n':>5}{'mean using 1/n':>18}{'theory':>12} "
      f"{'mean using 1/(n-1)':>22}")
print("-" * 67)

for n in SAMPLE_SIZES:
    v_n, v_n1 = simulate_for_sample_size(n, distribution="normal")

    simulated_mean_n = np.mean(v_n)
    simulated_mean_n1 = np.mean(v_n1)

    # E[(1/n) sum (Xi-Xbar)^2] = ((n-1)/n) sigma^2
    theoretical_mean_n = ((n - 1) / n) * TRUE_VARIANCE

    mean_variance_n.append(simulated_mean_n)
    mean_variance_n_minus_1.append(simulated_mean_n1)
    theoretical_variance_n.append(theoretical_mean_n)

    print(
        f"{n:5d} "
        f"{simulated_mean_n:18.4f} "
        f"{theoretical_mean_n:12.4f} "
        f"{simulated_mean_n1:22.4f}"
```

```
    )


plt.figure(figsize=(9, 5.5))

plt.plot(
    SAMPLE_SIZES,
    mean_variance_n,
    marker="o",
    label="Monte Carlo: divide by n"
)

plt.plot(
    SAMPLE_SIZES,
    theoretical_variance_n,
    linestyle="--",
    label="Theory: (n - 1) / n"
)

plt.plot(
    SAMPLE_SIZES,
    mean_variance_n_minus_1,
    marker="o",
    label="Monte Carlo: divide by n - 1"
)

plt.axhline(
    TRUE_VARIANCE,
    linestyle=":",
    label="True population variance"
)

plt.xlabel("Sample size n")
plt.ylabel("Average estimated variance")
plt.title("Bessel's Correction Removes the Downward Bias")
plt.legend()
plt.tight_layout()
plt.savefig("01_bessel_bias_vs_sample_size.png", dpi=200)
plt.show()


# ------------------------------------------------------------
# Part 2: What happens for an individual small sample?
# ------------------------------------------------------------

n_small = 5
v_n, v_n1 = simulate_for_sample_size(n_small, distribution="normal")

print("\nSMALL-SAMPLE EXAMPLE: n = 5")
print("-" * 42)
print(f"Mean estimate using 1/n:    {np.mean(v_n):.4f}")
print(f"Mean estimate using 1/(n-1): {np.mean(v_n1):.4f}")
print(f"True population variance:    {TRUE_VARIANCE:.4f}")
print()
print("Mean squared error:")
print(f"Estimator using 1/n:        {np.mean((v_n - TRUE_VARIANCE)**2):.4f}")
print(f"Estimator using 1/(n-1):    {np.mean((v_n1 - TRUE_VARIANCE)**2):.4f}")
```

```
plt.figure(figsize=(9, 5.5))

bins = np.linspace(0, 4, 70)

plt.hist(
  v_n,
  bins=bins,
  alpha=0.55,
  density=True,
  label="Divide by n"
)

plt.hist(
  v_n1,
  bins=bins,
  alpha=0.55,
  density=True,
  label="Divide by n - 1"
)

plt.axvline(
  TRUE_VARIANCE,
  linestyle="--",
  label="True population variance"
)

plt.xlabel("Estimated variance")
plt.ylabel("Density")
plt.title("Unbiased Does Not Mean Error-Free: n = 5")
plt.legend()
plt.tight_layout()
plt.savefig("02_small_sample_distribution.png", dpi=200)
plt.show()


# ------------------------------------------------------------
# Part 3: Is Bessel's correction only a Normal-distribution result?
# ------------------------------------------------------------

comparison_n = 5

normal_n, normal_n1 = simulate_for_sample_size(
  comparison_n,
  distribution="normal"
)

exponential_n, exponential_n1 = simulate_for_sample_size(
  comparison_n,
  distribution="exponential"
)

labels = [
  "Normal\n1/n",
  "Normal\n1/(n-1)",
  "Exponential\n1/n",
```

```
  "Exponential\n1/(n-1)"
]

means = [
  np.mean(normal_n),
  np.mean(normal_n1),
  np.mean(exponential_n),
  np.mean(exponential_n1)
]

print("\nDISTRIBUTION COMPARISON: n = 5")
print("-" * 47)
print(f"Normal, divide by n:          {means[0]:.4f}")
print(f"Normal, divide by n-1:        {means[1]:.4f}")
print(f"Exponential, divide by n:     {means[2]:.4f}")
print(f"Exponential, divide by n-1:   {means[3]:.4f}")


plt.figure(figsize=(9, 5.5))
positions = np.arange(len(labels))

plt.bar(positions, means)
plt.axhline(
  TRUE_VARIANCE,
  linestyle="--",
  label="True population variance"
)

plt.xticks(positions, labels)
plt.ylabel("Average estimated variance")
plt.title("Bessel's Correction Is Not Specific to the Normal Distribution")
plt.legend()
plt.tight_layout()
plt.savefig("03_normal_vs_exponential.png", dpi=200)
plt.show()


# ------------------------------------------------------------
# Part 4: One very small numerical illustration of degrees of freedom
# ------------------------------------------------------------

example = np.array([2.0, 4.0, 9.0])
example_mean = np.mean(example)
deviations = example - example_mean

print("\nDEGREES OF FREEDOM ILLUSTRATION")
print("-" * 42)
print(f"Sample:           {example}")
print(f"Sample mean:      {example_mean:.4f}")
print(f"Deviations from mean: {deviations}")
print(f"Sum of deviations:   {np.sum(deviations):.10f}")
print()
print(
  "Once n - 1 deviations are known, the final deviation is forced "
  "because all deviations from the sample mean must sum to zero."
)
```