Estimation and Confidence Intervals

a) 99% Confidence Interval using Sample Standard Deviation (t-distribution)

- **Sample size (n): 15**
- **Sample mean (x¯\bar{x}x¯): 1.239 million characters**
- **Sample standard deviation (s): 0.193 million characters**
- **Why t-distribution?**
  Since the population standard deviation is not known and the sample size is small (n<30n < 30n<30), the appropriate distribution for constructing the confidence interval is the *t-distribution* with n−1=14n-1 = 14n−1=14 degrees of freedom.
- **Critical value: t0.005,14≈2.9768t_{0.005,14} \approx 2.9768t0.005,14 ≈2.9768**
- **Margin of error:**

  E=t×sn=2.9768×0.19315≈0.149E = t \times \frac{s}{\sqrt{n}} = 2.9768 \times \frac{0.193}{\sqrt{15}} \approx 0.149E=t×ns=2.9768×150.193≈0.149

- **99% CI:**

  (1.090, 1.387)(1.090, \; 1.387)(1.090,1.387)

Interpretation: With 99% confidence, the true mean durability of print-heads lies between 1.09 and 1.39 million characters.

b) 99% Confidence Interval using Known Population Standard Deviation (z-distribution)

- **Population standard deviation (σ\sigmaσ): 0.2 million characters**
- **Why z-distribution?**
  When σ\sigmaσ is known, the sampling distribution of the mean follows the *standard normal distribution* (z).
- **Critical value: z0.005≈2.576z_{0.005} \approx 2.576z0.005≈2.576**
- **Margin of error:**

  E=z×σn=2.576×0.215≈0.133E = z \times \frac{\sigma}{\sqrt{n}} = 2.576 \times \frac{0.2}{\sqrt{15}} \approx 0.133E=z×nσ=2.576×150.2≈0.133

- **99% CI:**

  (1.106, 1.372)(1.106, \; 1.372)(1.106,1.372)

Interpretation: With 99% confidence, the true mean durability lies between 1.11 and 1.37 million characters.

Key Insight

Both approaches give very similar confidence intervals. The t-based interval is slightly wider because it accounts for the uncertainty of estimating the standard deviation from a small sample. When the population standard deviation is known, the z-based interval is slightly narrower.

Code used:

```
import numpy as np

data =
np.array([1.13,1.55,1.43,0.92,1.25,1.36,1.32,0.85,1.07,1.48,1.20,1.33,1.18,1.22,1.2
9])
n = len(data)
mean = data.mean()
s = data.std(ddof=1)

alpha = 0.01
# critical values
t_crit_14 = 2.9768   # t_{0.005, df=14}
z_crit = 2.5758      # z_{0.005}

me_t = t_crit_14 * (s / np.sqrt(n))
ci_t = (mean - me_t, mean + me_t)

pop_sigma = 0.2
me_z = z_crit * (pop_sigma / np.sqrt(n))
ci_z = (mean - me_z, mean + me_z)

print(f"n = {n}")
print(f"sample mean = {mean:.6f}")
print(f"sample sd (s) = {s:.6f}")
print()
print("99% CI using t (s unknown):")
print(f"({ci_t[0]:.6f}, {ci_t[1]:.6f})")
print()
print("99% CI using z (sigma known = 0.2):")
print(f"({ci_z[0]:.6f}, {ci_z[1]:.6f})")
```

basicstats.2.1.py

```python
import numpy as np

data = np.array([1.13,1.55,1.43,0.92,1.25,1.36,1.32,0.85,1.07,1.48,1.20,1.33,1.18,1.22,1.29])
n = len(data)
mean = data.mean()
s = data.std(ddof=1)

alpha = 0.01
# critical values
t_crit_14 = 2.9768   # t_{0.005, df=14}
z_crit = 2.5758      # z_{0.005}

me_t = t_crit_14 * (s / np.sqrt(n))
ci_t = (mean - me_t, mean + me_t)

pop_sigma = 0.2
me_z = z_crit * (pop_sigma / np.sqrt(n))
ci_z = (mean - me_z, mean + me_z)

print(f"n = {n}")
print(f"sample mean = {mean:.6f}")
print(f"sample sd (s) = {s:.6f}")
print()
print("99% CI using t (s unknown):")
print(f"({ci_t[0]:.6f}, {ci_t[1]:.6f})")
print()
print("99% CI using z (sigma known = 0.2):")
print(f"({ci_z[0]:.6f}, {ci_z[1]:.6f})")
```

TERMINAL

```
PS D:\python apps> & "D:/python apps/.venv/Scripts/python.exe" "d:/python apps/basicstats.2.1.py"
n = 15
sample mean = 1.238667
sample sd (s) = 0.193164

99% CI using t (s unknown):
(1.090199, 1.387134)

99% CI using z (sigma known = 0.2):
(1.105653, 1.371680)
PS D:\python apps>
```