

Chapter 4

Simulation, bootstrap and error propagation

Exercises

Contents

4 Simulation, bootstrap and error propagation

Exercises	1
4.1 Reliability: System lifetime	3
4.2 Basic bootstrap CI	12
4.3 Various bootstrap CIs	15
4.4 Two-sample TV data	20
4.5 Non-linear error propagation	24
4.6 Simulation of statistical power	27

4.1 Reliability: System lifetime

|||| Exercise 4.1 Reliability: System lifetime

In this exercise we will practice to do our own simulation.

You should use Python to solve this exercise and getting help from internet and generative AI is allowed. In an exam situation you should be able to read and understand code that carries out a simulation. Therefore it is also a good idea to study the solution for this exercise (after you have made your own solution) - you do not need to make your own code in the exact same way, but it is good to practice reading and understanding code from solutions (and from the book in general).

A device comprising three components; "A", "B" and "C", is only functioning as long as all three components are functioning individually. The lifetime (in months) of the individual components are assumed to follow exponential distributions: The average lifetime of components of type "A" is 2 months, the average lifetime of components of type "B" is 3 months and the average lifetime of components of type "C" is 5 months. Hence we can describe the lifetime of each component type with the random variables, X_A , X_B and X_C that follow exponential distributions:

$$\begin{aligned}X_A &\sim \text{Exp}(\lambda_A = 1/2), \\X_B &\sim \text{Exp}(\lambda_B = 1/3), \\X_C &\sim \text{Exp}(\lambda_C = 1/5).\end{aligned}$$

Describing the overall lifetime of devices is however not as straight forward. We will therefore simulate the situation.

- a) Simulate a large number of devices (at least 1000 – go for 10000 or 100000 if your computer is up for it). For each device you will need to simulate the lifetimes of component A, B and C. Store this information in a DataFrame. In your simulation, how long does the first three devices stay functioning?

|||| Solution

First, note that the solution below has been generated with the following seed:

```
np.random.seed(82719)
```

If you want to simulate the exact same pseudo-random numbers as shown in this solution, you will need to use the same seed (and generate the random variables in the same order).

The following Python-code generates 10.000 simulated devices:

```
np.random.seed(82719)

# Number of simulations
k = 10000
# Generating k component A lifetimes
xA = stats.expon.rvs(scale=2,size=k)
# Generating k component B lifetimes
xB = stats.expon.rvs(scale=3,size=k)
# Generating k component C lifetimes
xC = stats.expon.rvs(scale=5,size=k)

# Store the simulated lifetimes in a DataFrame:
df = pd.DataFrame({
    'Lifetime_A':xA,
    'Lifetime_B':xB,
    'Lifetime_C':xC
})

# Print the top part of your DataFrame containing simulated
# data:
print(df.head())
```

	Lifetime_A	Lifetime_B	Lifetime_C
0	0.320960	4.895210	11.179703
1	0.491043	3.006795	4.386929
2	1.608826	1.485084	0.929437
3	2.285528	4.850018	9.774913
4	2.409813	2.420582	1.164840

From the simulated data we see that "device0" stays functioning for 0.32 months (after which component A fails). We also see that "device1" stops functioning after 0.49 months (after which component A fails) and "device 2" stops functioning after 0.93

■ months (after which component C fails).

- b) To check our simulation, compute the average lifetime of each component type. Do these values match your expectations?

|||| Solution

```
print(df.mean())
Lifetime_A  2.003508
Lifetime_B  3.026452
Lifetime_C  5.047804
dtype: float64
```

These values match our expectations as they are very close to the theoretical population means used for the simulation (which were 2, 3 and 5 months).

We now have a simulated dataset that we can play with. We can compute the overall lifetime for each simulated device, and investigate how this *simulated random variable* behaves:

- c) For each device calculate the device lifetime and store this value in a new column in your DataFrame. Make a histogram of the device lifetimes (hence investigating the distribution of this *simulated random variable*).

(hint: consider how the random variable Y = "device lifetime" is a function of the three component lifetimes: is it the sum, the mean, the median, the minimum, the maximum, the range or something else?)

|||| Solution

For a series system, all components must function, so the system lifetime is:

$$Y = \min(X_A, X_B, X_C)$$

In Python we compute the minimum of each row using the `.min()` method (and we add `axis=1` to indicate the minimum should be computed for every row not for the columns).

```

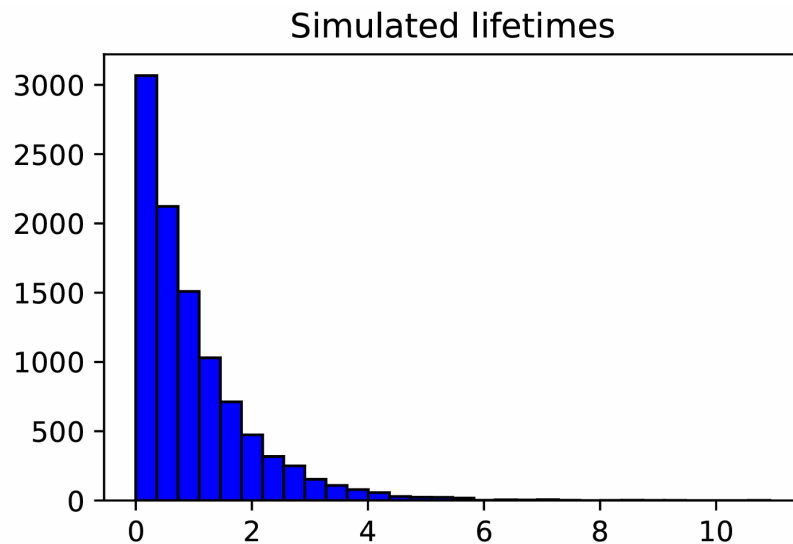
# Add a new column that is the minimum of the three original
  columns:
df['Lifetime'] = df[['Lifetime_A', 'Lifetime_B', 'Lifetime_C
  ']].min(axis=1)

print(df.head())

    Lifetime_A Lifetime_B Lifetime_C Lifetime
0    0.320960    4.895210    11.179703  0.320960
1    0.491043    3.006795     4.386929  0.491043
2    1.608826    1.485084     0.929437  0.929437
3    2.285528    4.850018     9.774913  2.285528
4    2.409813    2.420582     1.164840  1.164840

# Histogram of the simulated lifetimes
plt.hist(df['Lifetime'], bins=30, color='blue', edgecolor='
  black')
plt.title('Simulated lifetimes')
plt.show()

```



The device lifetime looks like it follows an exponential distribution.

d) Estimate the mean device lifetime from the simulated data.

||| Solution

```
# The estimated mean lifetime
print(df['Lifetime'].mean())
0.9880015386365172
```

e) Estimate the standard deviation of device lifetimes.

||| Solution

```
# The estimated standard deviation
print(df['Lifetime'].std(ddof=1))
0.9959225338611547
```

f) Estimate the probability that the device fails within 1 month.

||| Solution

We need to count how often the device lifetimes are smaller than or equal to 1 month - this can in Python be achieved by use of a logical operator (e.g., `<=`):

```
# Add a new column with an "indicator variable":
df['below_1_month'] = (df['Lifetime'] <= 1)

print(df.head())
```

	Lifetime_A	Lifetime_B	Lifetime_C	Lifetime	below_1_month
0	0.320960	4.895210	11.179703	0.320960	True
1	0.491043	3.006795	4.386929	0.491043	True
2	1.608826	1.485084	0.929437	0.929437	True
3	2.285528	4.850018	9.774913	2.285528	False
4	2.409813	2.420582	1.164840	1.164840	False

```
# The fraction of times the simulated lifetime was below or
equal to 1:
```

```
print(df['below_1_month'].mean())  
0.6391
```

Notice the output of the logical operator is either True or False. It is however still possible to apply the `.mean()` method - in this case Python treats True as one and False as zero.

g) Estimate the median system lifetime.

|||| Solution

```
# The estimated median  
print(np.median(df['Lifetime']))  
0.6878542614451192
```

h) Estimate the 10th percentile of system lifetimes.

|||| Solution

```
print(np.quantile(df['Lifetime'], 0.1, method='averaged_inverted_cdf'))  
0.10348342692625076
```

i) What seems to be the distribution of system lifetimes?

||| Solution

We already made the histogram above. It appears that the minimum of the three exponential variables also follows a distribution that looks like an exponential. In fact, there is a theoretical result (beyond the syllabus of this course) that states that the distribution of the minimum of these three exponential distributions is again an exponential distribution with rate:

$$\lambda_{\min} = \lambda_A + \lambda_B + \lambda_C = 1/2 + 1/3 + 1/5 = 31/30$$

Corresponding to the average lifetime:

$$1/\lambda_{\min} = 30/31 \approx 0.968$$

Consistent with the simulation result.

Exam-style questions

- j) For a particular low-expression gene, the number of sequencing reads observed in a sample is expected to follow a Poisson distribution with mean $\lambda = 4$. This means that the gene is expected to generate, on average, 4 reads per experiment, while the exact number varies randomly from sample to sample. Let X denote the number of reads observed for the gene in one experiment:

$$X \sim Po(\lambda = 4)$$

Researchers are interested in investigating the probability that the gene produces 10 or more reads in an experiment. They decide to estimate this probability using simulation. Which of the following simulations estimates this probability correctly?

1)

```
# Simulate RNA sequencing read counts
reads = stats.poisson.pmf(mu=4, k=10)

# Estimate P(X >= 10)
estimated_prob = np.mean(reads)

print(estimated_prob)
0.00529
```

The probability is 0.00529

2)

```
# Simulate RNA sequencing read counts
reads = stats.poisson.pmf(mu=4, k=4)

# Estimate  $P(X \geq 10)$ 
estimated_prob = np.min(reads)

print(estimated_prob)
0.19537
```

The probability is 0.19537

3)

```
# Simulate RNA sequencing read counts
reads = stats.binom.rvs(p=4/10, n=10, size=1000000)

# Estimate  $P(X \geq 10)$ 
estimated_prob = np.mean(reads >= 10)

print(estimated_prob)
0.0001
```

The probability is 0.0001

4)

```
# Simulate RNA sequencing read counts
reads = stats.poisson.rvs(mu=4, size=1000000)

# Estimate  $P(X \geq 10)$ 
estimated_prob = np.mean(reads >= 10)

print(estimated_prob)
0.008
```

The probability is 0.008

5)

```
# Simulate RNA sequencing read counts
reads = stats.norm.rvs(scale=10, size=1000000)

# Estimate  $P(X \geq 10)$ 
estimated_prob = np.mean(reads >= 10)

print(estimated_prob)
0.158
```

The probability is 0.16

|||| **Solution**

Correct answer: 4).

4.2 Basic bootstrap CI

|||| Exercise 4.2 Basic bootstrap CI

(Can be handled without using Python/R)

The following measurements are the cylindrical compressive strength (in MPa) for 11 prestressed concrete beams:

38.43, 38.43, 38.39, 38.83, 38.45, 38.35, 38.43, 38.31, 38.32, 38.48, 38.50.

1000 bootstrap samples (each of size 11) were generated from these data, and the 1000 bootstrap means were arranged in order. Denote the i -th smallest as $\bar{x}_{(i)}^*$. Assume that:

$$\begin{aligned}\bar{x}_{(25)}^* &= 38.3818, & \bar{x}_{(26)}^* &= 38.3818, \\ \bar{x}_{(50)}^* &= 38.3909, & \bar{x}_{(51)}^* &= 38.3918, \\ \bar{x}_{(950)}^* &= 38.5218, & \bar{x}_{(951)}^* &= 38.5236, \\ \bar{x}_{(975)}^* &= 38.5382, & \bar{x}_{(976)}^* &= 38.5391.\end{aligned}$$

- a) Compute a 95% bootstrap confidence interval for the mean compressive strength.

|||| Solution

The 95% CI uses the 2.5th and 97.5th percentiles of the 1000 bootstrap means. With $k = 1000$, the 2.5th percentile is the average of the 25th and 26th ordered values:

$$\begin{aligned}q_{0.025} &= \frac{\bar{x}_{(25)}^* + \bar{x}_{(26)}^*}{2} = 38.3818, \\ q_{0.975} &= \frac{\bar{x}_{(975)}^* + \bar{x}_{(976)}^*}{2} = \frac{38.5382 + 38.5391}{2} = 38.5387.\end{aligned}$$

The 95% bootstrap confidence interval is $[38.3818; 38.5387]$.

- b) Compute a 90% bootstrap confidence interval for the mean compressive strength.

||| Solution

The 90% CI uses the 5th and 95th percentiles:

$$q_{0.05} = \frac{\bar{x}_{(50)}^* + \bar{x}_{(51)}^*}{2} = \frac{38.3909 + 38.3918}{2} = 38.3914,$$

$$q_{0.95} = \frac{\bar{x}_{(950)}^* + \bar{x}_{(951)}^*}{2} = \frac{38.5218 + 38.5236}{2} = 38.5227.$$

The 90% bootstrap confidence interval is $[38.3914; 38.5227]$.

Exam-style questions

- c) Suppose $k = 2000$ bootstrap samples are generated, and their means are sorted in increasing order $\bar{x}_{(1)}^* \leq \dots \leq \bar{x}_{(2000)}^*$. Which pair of ordered values should be averaged to obtain the *lower* bound of the 99% bootstrap confidence interval (the 0.5th percentile)?
- 1) $\bar{x}_{(10)}^*$ and $\bar{x}_{(11)}^*$
 - 2) $\bar{x}_{(5)}^*$ and $\bar{x}_{(6)}^*$
 - 3) $\bar{x}_{(50)}^*$ and $\bar{x}_{(51)}^*$
 - 4) $\bar{x}_{(1990)}^*$ and $\bar{x}_{(1991)}^*$
 - 5) Only $\bar{x}_{(10)}^*$, without averaging with a neighbour

||| Solution

Correct answer: **1)**. For the lower bound of a 99% CI, the tail probability is $0.5\% = 0.005$, so the index is $0.005 \times 2000 = 10$ — exactly the same computation used above for the 95% and 90% cases, just with a different tail probability and k . As there, the two neighbouring ordered values are averaged: $\bar{x}_{(10)}^*$ and $\bar{x}_{(11)}^*$. Option 2) mistakenly reuses $k = 1000$ from the earlier exercise instead of the new $k = 2000$. Option 3) uses the 95% CI's 2.5% tail instead of the 99% CI's 0.5% tail. Option 4) computes the index for the *upper* bound ($0.995 \times 2000 = 1990$) instead of the lower bound.

- d) Using the data from the exercise above, the 95% bootstrap CI was found to be $[38.3818, 38.5387]$. The 90% bootstrap CI, found in the same way, is $[38.3914, 38.5227]$. This second interval is narrower. Why?
- 1) A higher confidence level requires capturing a larger proportion of the bootstrap distribution's mass, which means including more extreme (further from the centre) ordered values, widening the interval — exactly the same trade-off as for the classical t -based confidence intervals from Chapter 3.
 - 2) The 90% interval is narrower purely due to random simulation noise; with a different random seed, the 95% interval could easily end up narrower than the 90% interval.
 - 3) The intervals should actually be the same width; the difference shown must be a computational error.
 - 4) A lower confidence level means less data was used to construct the interval.
 - 5) Narrower confidence intervals always indicate more precise (lower-variance) estimates, so the 90% interval is unconditionally more useful than the 95% interval here.

|||| Solution

Correct answer: 1). This is the bootstrap analogue of a familiar fact: wanting to be more confident of capturing the true mean requires a wider interval, regardless of whether the interval is built from a t -distribution formula or empirically from resampled percentiles. Option 5) is a common trap: an interval's width cannot be judged as "better" without accounting for the confidence level attached to it — a very narrow interval with low confidence is not obviously more useful than a wider one with high confidence.

4.3 Various bootstrap CIs

|||| Exercise 4.3 Various bootstrap CIs

Consider the data from Exercise 2:

$x = 38.43, 38.43, 38.39, 38.83, 38.45, 38.35, 38.43, 38.31, 38.32, 38.48, 38.50$.

Generate $k = 1000$ (or more) bootstrap samples and compute the 1000 means.

- a) What are the 2.5% and 97.5% quantiles (i.e. the 95% confidence interval for μ without assuming any distribution)?

|||| Solution

Non-parametric bootstrap: resample with replacement from the data k times and compute the mean of each resample. The 2.5th and 97.5th percentiles of these k means give the 95% CI. Results vary with the random seed but are close to the result from Exercise 2: approximately [38.38; 38.54].

```
x = np.array([38.43, 38.43, 38.39, 38.83, 38.45,
              38.35, 38.43, 38.31, 38.32, 38.48, 38.50])
np.random.seed(6287)
boot = [np.random.choice(x, len(x), replace=True).mean() for _
        in range(10000)]
print(np.quantile(boot, [0.025, 0.975]))
```

- b) Find the 95% confidence interval for μ by the parametric bootstrap assuming the normal distribution. Compare with the classical t -based approach.

||| Solution

Parametric bootstrap: sample k times from $N(\bar{x}, s^2)$ and record the mean of each sample. The 2.5th and 97.5th percentiles give the CI.

The classical t -based CI (from Chapter 3) will give essentially the same result, since the normal assumption is the basis for both methods.

```
n = len(x)
simsamples = [np.random.normal(x.mean(), x.std(ddof=1), n).
               mean() for _ in range(10000)]
print(np.quantile(simsamples, [0.025, 0.975]))
print(stats.t.interval(0.95, df=n-1, loc=x.mean(), scale=x.
                       std(ddof=1)/np.sqrt(n)))
```

- c) Find the 95% confidence interval for μ by the parametric bootstrap assuming the log-normal distribution.

||| Solution

Fit a log-normal to the data by computing $\hat{\mu}_{\log} = \overline{\log x}$ and $\hat{\sigma}_{\log} = s_{\log x}$, then simulate from $LogNormal(\hat{\mu}_{\log}, \hat{\sigma}_{\log}^2)$:

```
logx = np.log(x)
simsamples = [np.random.lognormal(logx.mean(), logx.std(ddof=1), n).mean()
              for _ in range(10000)]
print(np.quantile(simsamples, [0.025, 0.975]))
```

- d) Find the 95% confidence interval for the lower quartile Q_1 by the parametric bootstrap assuming the normal distribution.

|||| Solution

```
simQ1s = [np.percentile(np.random.normal(x.mean(), x.std(
    ddof=1), n), 25)
          for _ in range(10000)]
print(np.quantile(simQ1s, [0.025, 0.975]))
```

- e) Find the 95% confidence interval for the lower quartile Q_1 by the non-parametric bootstrap.

|||| Solution

```
simQ1s = [np.percentile(np.random.choice(x, n, replace=True)
    , 25)
          for _ in range(10000)]
print(np.quantile(simQ1s, [0.025, 0.975]))
```

Exam-style questions

- f) A researcher has strong reason to believe her data are heavily skewed and does not know which parametric family (if any) fits well. Which bootstrap approach is most appropriate for constructing a CI for the mean, and why?
- 1) The non-parametric bootstrap, since it resamples directly from the observed data without assuming any particular distributional family, making it more robust when the correct parametric form is unknown or uncertain.
 - 2) The parametric bootstrap assuming a normal distribution, since the normal distribution is always a safe default regardless of the data's actual shape.
 - 3) The parametric bootstrap assuming a log-normal distribution, since log-normal is always correct for skewed data.
 - 4) Neither method is valid unless the sample size exceeds 1000.
 - 5) It does not matter; parametric and non-parametric bootstrap always give identical results.

|||| Solution

Correct answer: **1**). This is exactly why the exercise above walks through non-parametric, normal-parametric, and log-normal-parametric bootstrap on the *same* data: each makes a different assumption, and the non-parametric version is the safest choice when there is genuine uncertainty about the underlying distributional family. Option 3) is a common overcorrection — skewed data does not automatically mean log-normal is the right model, just a plausible candidate worth checking.

g) Suppose the true underlying data are strongly right-skewed (like income data), but an analyst mistakenly performs a *parametric* bootstrap assuming a normal distribution. What is the most likely consequence for the resulting confidence interval, compared to a non-parametric bootstrap on the same data?

- 1) The normal-based parametric bootstrap CI may be poorly calibrated (e.g. too symmetric), since it forces the resampled means to come from a symmetric distribution that does not reflect the actual skew in the data.
- 2) The two methods will always give identical intervals, since bootstrap resampling automatically corrects for any distributional misspecification.
- 3) The parametric bootstrap will simply fail to run (raise an error) if the true distribution is not normal.
- 4) The parametric bootstrap CI will always be exactly half the width of the non-parametric one.
- 5) Skewness in the original data has no effect on the sampling distribution of the mean, by the Central Limit Theorem, so both methods will always agree closely regardless of sample size.

|||| Solution

Correct answer: **1**). A parametric bootstrap simulates entirely from the *assumed* distribution, so if that assumption (normality) is wrong, the resulting CI reflects the wrong model rather than the true data generating process. Option 5) invokes the Central

Limit Theorem, but that guarantee only kicks in for sufficiently *large* sample sizes — for a small sample (like $n = 11$ here), skewness in the underlying data can still noticeably affect the sampling distribution of the mean, so “regardless of sample size” is false. Option 2) is also false: correcting for misspecification is precisely something resampling from a wrong assumed distribution cannot do.

4.4 Two-sample TV data

|||| Exercise 4.4 Two-sample TV data

A TV producer had 20 consumers evaluate the quality of two different TV flat screens — 10 consumers for each screen. A scale from 1 (worst) to 5 (best) was used, with the following results:

TV screen 1	TV screen 2
1	3
2	4
1	2
3	4
2	2
1	3
2	2
3	4
1	3
1	2

- a) Compare the two means without assuming any distribution for the two samples (non-parametric bootstrap CI and hypothesis test interpretation).

|||| Solution

Non-parametric bootstrap of the mean difference $\bar{x}_1 - \bar{x}_2$:

```
x1 = np.array([1,2,1,3,2,1,2,3,1,1])
x2 = np.array([3,4,2,4,2,3,2,4,3,2])
np.random.seed(98273); k = 10000
diffs = [np.random.choice(x1,10,replace=True).mean()
          - np.random.choice(x2,10,replace=True).mean() for _
          in range(k)]
print(np.quantile(diffs, [0.025, 0.975]))
```

Since 0 is not contained in the CI, we reject $H_0 : \mu_1 = \mu_2$.

- b) Compare the two means assuming normal distributions for the two samples — without simulation (even though the scores are discrete, the approximation is used here for illustration).

|||| Solution

The Welch two-sample t -test:

```
print(stats.ttest_ind(x1, x2))
```

The p -value is small (typically < 0.05); we reject $H_0 : \mu_1 = \mu_2$.

- c) Compare the two means assuming normal distributions — via parametric bootstrap.

|||| Solution

Parametric bootstrap: simulate k pairs of samples from fitted normals and compute the mean difference each time:

```
n = 10
diffs = [np.random.normal(x1.mean(), x1.std(ddof=1), n).mean()
          - np.random.normal(x2.mean(), x2.std(ddof=1), n).mean()
          for _ in range(k)]
print(np.quantile(diffs, [0.025, 0.975]))
```

Again zero is not contained in the CI; we reject H_0 .

Exam-style questions

- d) A UX researcher runs a non-parametric bootstrap comparing time-on-task for two app designs and obtains a 95% bootstrap CI for the mean difference (Design A – Design B) of $[-1.2, 0.3]$ seconds. What should be concluded about $H_0 : \mu_A = \mu_B$ at $\alpha = 0.05$?

- 1) H_0 is *not* rejected, since 0 lies within the interval $[-1.2, 0.3]$.
- 2) H_0 is rejected, since the interval does not contain the value 1.
- 3) H_0 is rejected, since the interval is entirely negative on one side.
- 4) H_0 is rejected, since $-1.2 \neq 0.3$.
- 5) No conclusion can be drawn from a bootstrap CI; only a formal hypothesis test can determine significance.

|||| Solution

Correct answer: **1**). This is the same decision rule used throughout this exercise: if 0 falls *outside* the bootstrap CI for the mean difference, H_0 is rejected; if 0 falls *inside* it (as here), H_0 is *not* rejected. In the TV-screen example above, 0 happened to fall outside all three computed intervals, leading to rejection every time — it is important not to over-generalise that pattern to “bootstrap CIs always reject H_0 ”. Option 5) is false: a percentile bootstrap CI *is* a valid (and commonly used) way to perform a hypothesis test.

- e) The TV-rating data above are actually discrete ordinal scores (1 to 5), yet the second question of this exercise applies a Welch t -test and the third applies a normal-based parametric bootstrap, both of which formally assume continuous, (approximately) normally distributed data. Given this, which statement is most defensible?
 - 1) The t -test and normal-parametric bootstrap results should be interpreted with some caution, as approximations; the non-parametric bootstrap (which makes no distributional assumption at all) is the most directly justified of the three methods for this discrete, non-normal data.
 - 2) Since all three methods happen to agree numerically in this case, the discreteness of the data is entirely irrelevant and causes no concern in general.
 - 3) A t -test can never be applied to discrete data under any circumstances, so the second question of this exercise is simply invalid.
 - 4) Discrete ordinal data should always be converted to ranks before any statistical test can be performed.
 - 5) The normal-parametric bootstrap is the most appropriate of the three, since simulating from a fitted normal distribution fully resolves the discreteness mismatch.

|||| Solution

Correct answer: **1)**. The exercise itself flags this: “even though the scores are discrete, the approximation is used here for illustration.” The non-parametric bootstrap re-samples the actual discrete values observed, making no assumption about continuity or normality, and is therefore the most directly defensible of the three methods here — the other two are useful approximations, not exact procedures for this data type. Option 2) overgeneralises from a single example where the methods happened to agree; that agreement is not guaranteed in general. Option 5) overstates what fitting a normal distribution actually fixes — it approximates around the discreteness issue rather than resolving it.

4.5 Non-linear error propagation

|||| Exercise 4.5 Non-linear error propagation

Solve question (a)-(c) without using Python, but use Python to simulate in question (d).

The pressure P , and the volume V of one mole of an ideal gas are related by the equation $PV = 8.31T$, when P is measured in kilopascals (kPa), T is measured in kelvin (K), and V is measured in liters (L).

- a) Assume that P is measured to be 240.48 kPa and V to be 9.987 L with known measurement errors (given as standard deviations): $\sigma_P = 0.03$ kPa and $\sigma_V = 0.002$ L. Estimate T and find its uncertainty (also given as standard deviation).

|||| Solution

This is an almost direct copy of the rectangle example from the book ($A = XY$) (see Example 4.1), since $T = PV/8.31$.

$$\hat{T} = \frac{240.48 \times 9.987}{8.31} = 289.01 \text{ K.}$$

To use the approximate error propagation rule, we must differentiate the function $T = f(P, V) = PV/8.31$ with respect to both P and V . The partial derivatives are $\partial f / \partial P = V/8.31$ and $\partial f / \partial V = P/8.31$, so by the error propagation formula:

$$\sigma_{\hat{T}} = \frac{1}{8.31} \sqrt{V^2 \sigma_P^2 + P^2 \sigma_V^2} = \frac{\sqrt{9.987^2 \times 0.03^2 + 240.48^2 \times 0.002^2}}{8.31} = 0.0682 \text{ K.}$$

- b) Assume that P is measured to be 240.48 kPa and T to be 289.12 K with known measurement errors (given as standard deviations): $\sigma_P = 0.03$ kPa and $\sigma_T = 0.02$ K. Estimate V and find its uncertainty (also given as standard deviation).

||| Solution

Since $V = f(P, T) = 8.31T/P$:

$$\hat{V} = \frac{8.31 \times 289.12}{240.48} = 9.9908 \text{ L.}$$

The partial derivatives are $\partial f / \partial T = 8.31/P$ and $\partial f / \partial P = -8.31T/P^2$:

$$\sigma_{\hat{V}} = 8.31 \sqrt{\frac{\sigma_T^2}{P^2} + \frac{T^2 \sigma_P^2}{P^4}} = 8.31 \sqrt{\frac{0.02^2}{240.48^2} + \frac{289.12^2 \times 0.03^2}{240.48^4}} = 0.00143 \text{ L.}$$

- c) Assume that V is measured to be 9.987 L and T to be 289.12 K with known measurement errors (given as standard deviations): $\sigma_V = 0.002$ L and $\sigma_T = 0.02$ K. Estimate P and find its uncertainty (also given as standard deviation).

||| Solution

Since $P = f(V, T) = 8.31T/V$ (symmetrically to the previous question):

$$\hat{P} = \frac{8.31 \times 289.12}{9.987} = 240.57 \text{ kPa.}$$

$$\sigma_{\hat{P}} = 8.31 \sqrt{\frac{\sigma_T^2}{V^2} + \frac{T^2 \sigma_V^2}{V^4}} = 8.31 \sqrt{\frac{0.02^2}{9.987^2} + \frac{289.12^2 \times 0.002^2}{9.987^4}} = 0.0510 \text{ kPa.}$$

- d) Verify one or more of the above results by simulation. Assume normally distributed measurement errors.

||| Solution

Simulation for question (c), estimating P :

```
np.random.seed(28973)
k = 10000
Vs = np.random.normal(9.987, 0.002, k)
Ts = np.random.normal(289.12, 0.02, k)
Ps = 8.31*Ts/Vs
print(Ps.std(ddof=1))
```

The simulation standard deviation should be close to the analytical 0.051 kPa.

Exam-style questions

- e) The resistance of a wire is given by $R = \rho L / A$, where ρ (resistivity) is known exactly, L is length, and A is cross-sectional area (both measured with uncertainty). Which of the following correctly gives an expression for σ_R ?

- 1) $\sigma_R = \sqrt{\left(\frac{\rho}{A}\right)^2 \sigma_L^2 - \left(\frac{\rho L}{A^2}\right)^2 \sigma_A^2}$
- 2) $\sigma_R = \sqrt{\left(\frac{\rho}{A}\right)^2 \sigma_L^2 + \left(\frac{\rho L}{A^2}\right)^2 \sigma_A^2}$
- 3) $\sigma_R = \frac{\rho}{A} \sigma_L + \frac{\rho L}{A^2} \sigma_A$
- 4) $\sigma_R = \sqrt{\left(\frac{\rho}{A}\right)^2 \sigma_L^2 + \left(\frac{\rho}{A}\right)^2 \sigma_A^2}$
- 5) $\sigma_R = \sqrt{\left(\frac{L}{A}\right)^2 \sigma_L^2 + \left(\frac{L}{A^2}\right)^2 \sigma_A^2}$

||| Solution

Correct answer: **2)**. With $R = f(L, A) = \rho L / A$: $\partial R / \partial L = \rho / A$ and $\partial R / \partial A = -\rho L / A^2$.

4.6 Simulation of statistical power

|||| Exercise 4.6 Simulation of statistical power

This exercise should be solved without using Python.

A company wants to measure the effect of a newly developed coating that may improve the shelf-life of their product. Without the coating the product has an average shelf-life of 27 days. They decide that the new coating will be worth the extra expense if the average shelf-life is increased by 1.5 days or more. They also know that the shelf-life of their products is subject to rather large variability and expect the standard deviation to be of order ~ 5 days.

The company plans to make an experiment where they apply the new coating to 100 products and measure the shelf-life of these. But before they initiate the actual experiment they decide to simulate the risk of not detecting any significant improvement, if the true improvement is only 1.5 days (or less). They assume that the measured shelf-life, X , follows a normal distribution:

$$X \sim N(\mu = 28.5, \sigma^2 = 5^2).$$

The company carries out the following simulation in Python:

```
np.random.seed(412)

k = 1000          # Number of simulations
n = 100           # Sample size

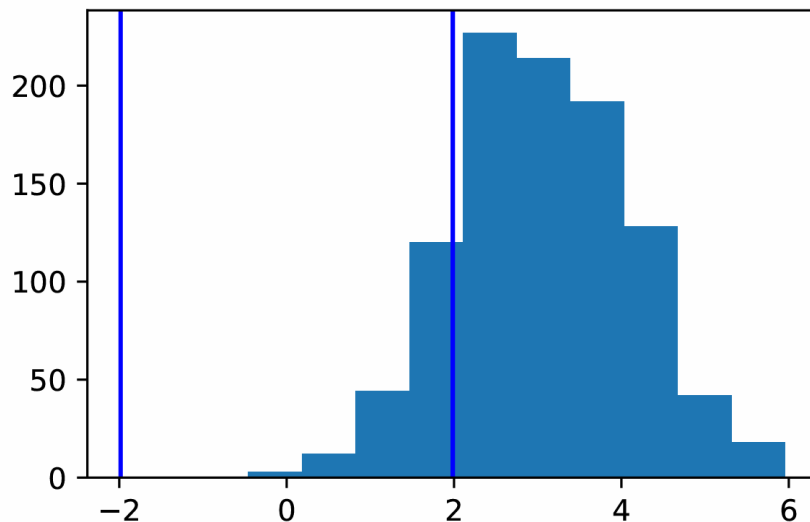
# Simulation
shelf_life_samples = stats.norm.rvs(loc=28.5, scale=5, size=(
    k, n))

# Simulation statistics
sim_means = shelf_life_samples.mean(axis=1)
sim_SEM = shelf_life_samples.std(ddof=1, axis=1)/np.sqrt(n)

# Simulation results
sim_tobs = (sim_means - 27)/sim_SEM
tcritical = stats.t.ppf(0.975, df=n-1)

# Plot t_obs for every simulation
plt.hist(sim_tobs)
plt.axvline(-tcritical, linestyle='-', color="blue", ymin=0, ymax=1)
```

```
plt.axvline(tcritical, linestyle='-',color="blue",ymin=0,ymax=1)
plt.show()
```



```
reject_null_hyp = np.abs(sim_tobs) >= tcritical
n_rejected_null_hyp = np.sum(reject_null_hyp)
print(n_rejected_null_hyp)
858
```

Hint: Notice the variable `reject_null_hyp` is a boolean variable (True or False). Here we print the first 10 values in the variable `reject_null_hyp` and the first 10 values of the variable `sim_tobs`:

```
print(reject_null_hyp[:10])
[True True True True True True True False False True]

print(sim_tobs[:10])
[2.0167269 2.44449889 2.86036866 2.37276581 3.84951488
 2.97859893
 2.83082468 1.67866428 1.37293584 3.29996005]
```

a) How many experiments (samples) were simulated?

|||| **Solution**

$k = 1000$ experiments were simulated, each of sample size $n = 100$.

b) What is `shelf_life_samples`? And what are its dimensions?

|||| **Solution**

`shelf_life_samples` is a 2D array of simulated shelf-lives with 1000 rows and 100 columns: each row is one simulated experiment (sample) of 100 measurements.

c) What is `sim_SEM`?

|||| **Solution**

`sim_SEM` is an array containing the standard error of the mean, $s/\sqrt{n}$, computed separately for each of the 1000 simulated samples.

d) Explain the hypothesis test carried out in the simulation (and the histogram). What is the null hypothesis? Which test statistic is calculated? Are the conclusions from the hypothesis tests the same for every simulation?

|||| Solution

A one-sample t -test is carried out for *each* simulated sample, testing $H_0 : \mu = \mu_0 = 27$ (no effect of the coating) against $H_1 : \mu \neq 27$, using

$$t_{\text{obs}} = \frac{\bar{x} - \mu_0}{SE_{\bar{x}}}.$$

H_0 is rejected whenever $|t_{\text{obs}}|$ exceeds the critical value $t_{0.975, 99}$, i.e. at significance level $\alpha = 0.05$. The histogram shows the 1000 simulated values of t_{obs} : most fall beyond the critical-value lines (rejecting H_0), while a minority fall inside them (failing to reject H_0), even though the true improvement is genuinely present.

- e) What is the conclusion from the simulation regarding the risk of not detecting any significant improvement?

|||| Solution

858 of the 1000 simulated experiments reject H_0 , so the estimated statistical power is ≈ 0.86 . This means there is a risk of about 0.14 (probability of making a type II error) of failing to detect the improvement in an experiment with $n = 100$, $\alpha = 0.05$, when the true effect size is 1.5 days and $\sigma = 5$.

- f) Perform the theoretical calculation of the sample size needed for a power of 0.90 (using $\alpha = 0.05$).

You can use the following quantiles from a standard normal distribution:
 $z_{0.90} = 1.282$, $z_{0.95} = 1.645$, $z_{0.975} = 1.960$.

|||| Solution

Using the sample-size formula (Method 3.65):

$$n = \left(\sigma \cdot \frac{z_{1-\alpha/2} + z_{1-\beta}}{\mu_0 - \mu_1} \right)^2 = \left(5 \cdot \frac{1.960 + 1.282}{1.5} \right)^2 = 116.8.$$

Rounding up: $n = 117$.