Exercise 1: Linear Regression is Gaussian Inference

Throughout, let $y \in \mathbb{R}^n$ be the response vector, $X \in \mathbb{R}^{n \times p}$ the design matrix (with full column rank), $\beta \in \mathbb{R}^p$ the coefficient vector, and $\varepsilon \sim \mathcal{N}\big(0, \sigma^2 I_n\big)$ the noise.

1. Equivalence of the two Gaussian statements

Claim.

$$y = f(x) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \Sigma) \quad \Longleftrightarrow \quad y \,\big|\, x \;\sim\; \mathcal{N}\big(f(x), \Sigma\big).$$

Answer:
"$\Longrightarrow$" Adding a deterministic vector to a Gaussian shifts its mean, leaving the covariance untouched: $y \mid x \;=\; f(x) + \varepsilon \sim \mathcal{N}\big(f(x), \Sigma\big)$.
"$\Longleftarrow$" Conversely, if $y \mid x$ is Gaussian with mean $f(x)$ and covariance Σ, define $\varepsilon \coloneqq y - f(x)$. Then $\varepsilon \sim \mathcal{N}(0, \Sigma)$ and trivially $y = f(x) + \varepsilon$. □

2. Log-likelihood versus residual-sum-of-squares

Assume the *classical linear model* $y = X\beta + \varepsilon$ with $\varepsilon \sim \mathcal{N}\big(0, \sigma^2 I_n\big)$. Its likelihood is

$$p(y \mid X, \beta, \sigma^2) = (2\pi\sigma^2)^{-n/2} \exp\Big[-\tfrac{1}{2\sigma^2}(y - X\beta)^\top (y - X\beta)\Big].$$

Taking logs gives

$$\ell(\beta) = \log p(y \mid X, \beta, \sigma^2) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\underbrace{\|y - X\beta\|_2^2}_{\mathrm{RSS}(\beta)}.$$

The first term is β-independent, so

$$\begin{aligned} \arg\max_\beta \ell(\beta) &= \arg\min_\beta \mathrm{RSS}(\beta) \\ &= \arg\min_\beta \tfrac{1}{n}\mathrm{RSS}(\beta) \quad \text{(MSE)}. \end{aligned}$$

Thus, *maximising the Gaussian log-likelihood is identical to ordinary least squares.*

3. Normal equations, MLE, and its covariance

3.1 Point estimator. Define $\mathrm{RSS}(\beta) = \|y - X\beta\|_2^2$. Gradient-setting yields the *normal equations*

$$\nabla_\beta \mathrm{RSS} = -2X^\top (y - X\beta) = 0 \implies X^\top X\, \beta = X^\top y.$$

Because $X^\top X$ is invertible (full rank assumption),

$$\boxed{\hat{\beta}_{\mathrm{MLE}} = (X^\top X)^{-1} X^\top y}.$$

3.2 Sampling distribution of $\hat{\beta}$. Insert the true model $y = X\beta + \varepsilon$:

$$\hat{\beta} = (X^\top X)^{-1} X^\top (X\beta + \varepsilon) = \beta + (X^\top X)^{-1} X^\top \varepsilon.$$

Since $E[\varepsilon] = 0$ and $\mathrm{Cov}(\varepsilon) = \sigma^2 I_n$,

$$\mathrm{Cov}(\hat{\beta}) = (X^\top X)^{-1} X^\top (\sigma^2 I_n) X \, (X^\top X)^{-1} = \sigma^2 (X^\top X)^{-1}.$$

Hence, each coefficient estimator is unbiased with

$$\boxed{\mathrm{Var}(\hat{\beta}_j) = \sigma^2 \big[(X^\top X)^{-1}\big]_{jj}}.$$

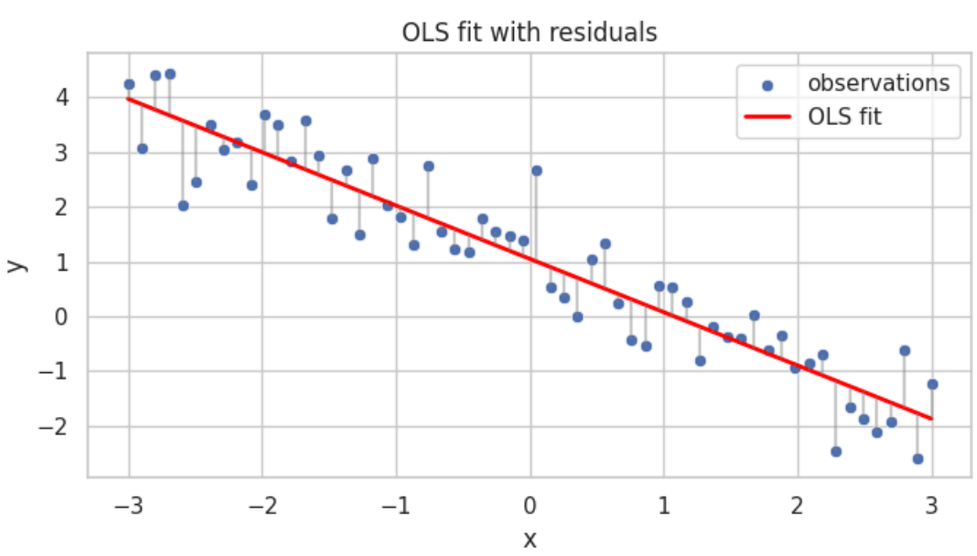
Remark. If σ^2 is unknown, replace it by the usual unbiased estimate $\hat{\sigma}^2 = \mathrm{RSS}(\hat{\beta})/(n-p)$ before computing standard errors—but that lies beyond the scope of this question.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns # For better aesthetics
from sklearn.linear_model import LinearRegression
from scipy.special import binom # Binomial coefficients for polynomial features

sns.set_theme(style="whitegrid")
```

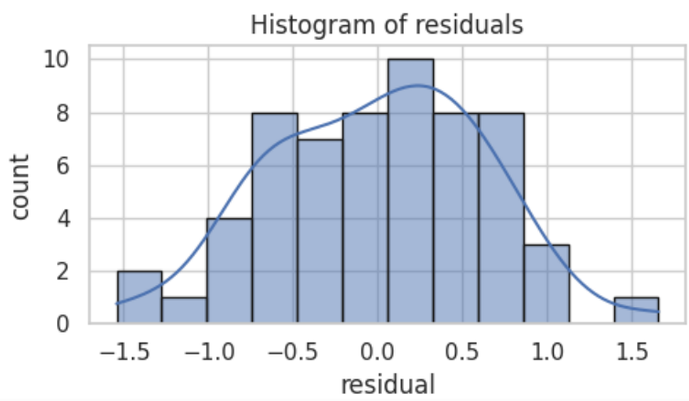
Exercise 1

Task 4

```
# Generate Synthetic data
n = 60
beta0, beta1, sigma = 1.0, -1.0, 0.8
x = np.linspace(-3, 3, n)

rng = np.random.default_rng(42)
eps = rng.normal(0, sigma, n)
y =  beta0 + beta1 * x + eps


# Fit OLS model using sklearn
X = x.reshape(-1, 1)
linreg = LinearRegression().fit(X, y)

y_hat = linreg.predict(X)
beta0_hat = linreg.intercept_
beta1_hat = linreg.coef_[0]


# Plot data, fitted line and vertical error (residual) bars
order = np.argsort(x)

plt.figure(figsize=(7, 4))
sns.scatterplot(x=x, y=y, label="observations")
plt.plot(x[order], y_hat[order], color="red", lw=2, label="OLS fit")

for xi, yi, ypred in zip(x, y, y_hat):
    plt.vlines(xi, min(yi, ypred), max(yi, ypred), color="grey", alpha=0.4)

plt.xlabel("x")
plt.ylabel("y")
plt.title("OLS fit with residuals")
plt.legend()
plt.tight_layout()
plt.show()
```




```
# Do a histogram of the residuals (e.g. sns.histplot)
residuals = y - y_hat

plt.figure(figsize=(5, 3))
sns.histplot(residuals, bins=12, kde=True, edgecolor='k')
plt.xlabel('residual'); plt.ylabel('count')
plt.title('Histogram of residuals')
plt.tight_layout(); plt.show()
```




```
# Plot the log likelihood and the RSS as a function of beta1. Show visually that MLE and OLS are equivalent for linear

beta1_grid = np.linspace(beta1_hat - 3, beta1_hat + 3, 200)
rss_list, ll_list = [], []

for b1 in beta1_grid:
    y_pred = beta0_hat + b1 * x
    res    = y - y_pred
    rss    = np.sum(res**2)
    rss_list.append(rss)
    ll     = -(n/2)*np.log(2*np.pi*sigma**2) - rss/(2*sigma**2)
    ll_list.append(ll)

fig, ax1 = plt.subplots(figsize=(7, 4))

ax1.plot(beta1_grid, rss_list, color='tab:blue', label='RSS')
ax1.set_xlabel(r'$\beta_1$')
ax1.set_ylabel('RSS', color='tab:blue')
ax1.tick_params(axis='y', labelcolor='tab:blue')

ax2 = ax1.twinx()
ax2.plot(beta1_grid, ll_list, color='tab:red', label='log-likelihood')
ax2.set_ylabel('log-likelihood', color='tab:red')
ax2.tick_params(axis='y', labelcolor='tab:red')

ax1.axvline(beta1_hat, ls='--', color='k')
fig.suptitle('Minimum RSS coincides with Maximum Likelihood')
fig.tight_layout(); plt.show()
```