Question 1:

a) MLE estimator for μ is: $\hat{\mu} = \bar{X}$

$$L_n(\bar{X}_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x_i - \bar{x}_n)^2}{2\sigma^2}\right)$$

$$L_n(\mu_0) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x_i - \mu_0)^2}{2\sigma^2}\right)$$

$$\Rightarrow \frac{L_n(\bar{X}_n)}{L_n(\mu_0)} = \prod_{i=1}^{n} \exp\left(\frac{-(x_i - \bar{x}_n)^2}{2\sigma^2}\right) \cdot \exp\left(\frac{(x_i - \mu_0)^2}{2\sigma^2}\right) = \prod_{i=1}^{n} \exp$$

$$\Rightarrow 2\log\left(\frac{L_n(\bar{X}_n)}{L_n(\mu_0)}\right) = \frac{1}{\sigma^2}\left(\sum_{i=1}^{n}(x_i - \mu_0)^2 - \sum_{i=1}^{n}(x_i - \bar{x}_n)^2\right)$$

$$= \frac{1}{\sigma^2} \cdot \sum_{i=1}^{n}\left(2x_i(\bar{x}_n - \mu_0) + \mu_0^2 - \bar{x}_n^2\right) = 2\frac{\bar{x}_n - \mu_0}{\sigma^2} \cdot \sum_{i=1}^{n} x_i + \frac{n}{\sigma^2}(\mu_0^2 - \bar{x}_n^2)$$

$$= \frac{2}{\sigma^2} \cdot n \cdot \bar{x}_n^2 - \frac{2\mu_0}{\sigma^2} \sum_{i=1}^{n} x_i + \frac{n}{\sigma^2}\mu_0^2 - \frac{n}{\sigma^2}\bar{x}_n^2 = \frac{n}{\sigma^2} \cdot \bar{X}_n^2 + \frac{1}{\sigma^2}\left(n - 2\sum_{i=1}^{n} X_i\right)\mu_0$$

$$\chi_1^2$$

$$= \frac{n}{\sigma^2}(\bar{X}_n^2 + \mu_0^2 - 2\bar{x}_n\mu_0) = \left(\frac{\bar{x}_n - \mu_0}{\sqrt{\sigma^2/n}}\right)^2 \sim \chi_1^2$$

Therefore we reject the null hypothesis if $\lambda > \chi^2_{1,\alpha}$.

b) MLE estimator for σ^2: $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$

$$L_n(\hat{\sigma}^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} \exp\left(\frac{-(x_i - \mu)^2}{2\hat{\sigma}^2}\right)$$

$$\frac{\hat{\sigma}^2}{\sigma_0^2} = \frac{1}{n} \cdot \frac{S^2}{\sigma^2} = \frac{1}{n^2} \cdot \chi_1^2$$

$$L_n(\sigma_0^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{-(x_i - \mu)^2}{2\sigma_0^2}\right)$$

$$2\log\left(\frac{L_n(\hat{\sigma}^2)}{L_n(\sigma_0^2)}\right) = 2\log\left(\left(\frac{\sigma_0^2}{\hat{\sigma}^2}\right)^{n/2}\right) \cdot \left(\sum_{i=1}^{n}\frac{(x_i - \mu)^2}{2\sigma_0^2} - \sum_{i=1}^{n}\frac{(x_i - \mu)^2}{2\hat{\sigma}^2}\right)$$

$$= n \cdot \log\left(\frac{\sigma_0^2}{\hat{\sigma}^2}\right) \cdot \left(\frac{1}{2\sigma_0^2}\sum_{i=1}^{n}(x_i - \mu)^2 - \frac{n}{2}\right) = n \cdot \log\left(\frac{\sigma_0^2}{\hat{\sigma}^2}\right) \cdot \left(\frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_i - \mu)^2 - \frac{n}{2}\right) = \lambda$$

Therefore we reject the null hypothesis if $\lambda > \chi^2_{1,\alpha}$.

Question 2:

a) $\mathbb{E}[AX] = \begin{pmatrix} \mathbb{E}(A^1 X) \\ \vdots \\ \mathbb{E}(A^m X) \end{pmatrix} = \begin{pmatrix} A^1 \mu \\ \vdots \\ A^m \mu \end{pmatrix} = A\mu$

matrix row independence

$\mathbb{E}[v^T X] = v\mu \quad \forall v \in \mathbb{R}^n$ by tutorial

b) $\Sigma = Var(X) = \mathbb{E}[(X-\mu_X)(X-\mu_X)^T]$

It is known that $\forall v \in \mathbb{R}^n$, $vv^T \succeq 0$ therefore $\mathbb{E}(vv^T) \succeq 0$
by non-negatively weighted sum of positive semidefinite matrices.

$\Rightarrow \mathbb{E}[(X-\mu_X)(X-\mu_X)^T] \succeq 0 \Rightarrow \Sigma \succeq 0$.

c) $Cov(AX, BY) = \mathbb{E}[A(X-\mu_X) \cdot B(Y-\mu_Y)] \overset{\text{by (a)}}{=} A \cdot Cov(X, BY)$

$\overset{\text{by (b)}}{=} A \cdot Cov(BY, X)^T = A \cdot \mathbb{E}[B(Y-\mu_Y)(X-\mu_X)]^T = A \cdot (B \cdot \mathbb{E}[(Y-\mu_Y)(X-\mu_X)])^T$

$= A \cdot Cov(Y, X)^T \cdot B^T = A \cdot Cov(X, Y) \cdot B^T$

Question 3:

$$\hat{\beta}_{OLS} = (X^TX)^{-1}X^Ty = (X^TX)^{-1}X^T(X\beta^* + \varepsilon) = \beta^* + (X^TX)^{-1}X^T\varepsilon$$

$$\hat{\beta}_{MLE} = \arg\max_{\beta} \prod_{i=1}^{n} P(Y_i = y_i) = \arg\max_{\beta} \prod_{i=1}^{n} P(\beta_0 + \sum_{j=1}^{k}\beta_j X_{ij} + \varepsilon_i = y_i)$$

$$= \arg\max_{\beta} \prod_{i=1}^{n} P(\varepsilon_i = y_i - \beta_0 - \sum_{j=1}^{k}\beta_j X_{ij}) = \arg\max_{\beta} \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\cdot\frac{(y_i - \beta_0 - \sum_{j=1}^{k}\beta_j X_{ij})^2}{\sigma^2}\right)$$

$$= \arg\max_{\beta} \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\frac{(y_i - \beta_0 - \sum_{j=1}^{k}\beta_j X_{ij})^2}{\sigma^2}\right) = \arg\max_{\beta} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k}\beta_j X_{ij})^2\right)$$

$$\arg\min_{\beta}\left(\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k}\beta_j X_{ij})^2\right) = \arg\min_{\beta} S(\beta)$$

log monotone increasing + e^x monotone increasing

$$\Longrightarrow \hat{\beta}_{MLE} = \arg\min_{\beta} S(\beta)$$

This is exactly the definition of $\hat{\beta}_{OLS}$ and therefore $\underline{\hat{\beta}_{MLE} = \hat{\beta}_{OLS}}$.

hw3

December 5, 2022

[94]:
```
import pandas as pd
import numpy as np
from scipy.stats import f
```

1 Question 4

1.1 a.

the exact model we assume is $Y = \beta_0 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \epsilon$

[95]:
```
df = pd.read_csv("ex3.csv")
X = df[["x2","x3","x4", "x5"]].to_numpy()
X = np.c_[np.ones(X.shape[0]), X]
y = df[["y"]].to_numpy()
beta_star =np.linalg.inv(X.T@X)@X.T@y
print(f"the beta star coeficiants are {beta_star.squeeze()}")
```

```
the beta star coeficiants are [ 7.45780659 -0.0297028   0.52051008 -0.10180238
-2.1605807 ]
```

1.2 b.

[96]:
```
y_hat = X@beta_star.squeeze()
p = X.shape[1]
n = y.shape[0]
e = y.squeeze()- y_hat
var_hat = e.T@e/(n-p)
var_hat
```

```
0.7792240642000448
```

1.3 c.

[97]:
```
SoS_res = (y.squeeze() - y_hat).T@(y.squeeze() - y_hat)
SoS_T = (y.squeeze()-np.ones(n)*np.mean(y)).T@(y.squeeze()-np.ones(n)*np.
 ↪mean(y))
SoS_R = SoS_T - SoS_res
print(f"{SoS_res=}\t {SoS_T=}\t {SoS_R=}")
```

```
SoS_res=10.909136898800627       SoS_T=33.22105263157895
SoS_R=22.311915732778324
```

[98]:
```
df_res = p-1
df_T = n-p
df_R = n-1
print(f"{df_res=}\t {df_T=}\t {df_R=}")
```

```
df_res=4         df_T=14         df_R=18
```

[99]:
```
MS_R = SoS_R/(p-1)
MS_res = SoS_res/(n-p)
MS_T = SoS_T/(n-1)
print(f"{MS_res=}\t {MS_T=}\t {MS_R=}")
```

```
MS_res=0.7792240642000448        MS_T=1.8456140350877195
MS_R=5.577978933194581
```

[100]:
```
F = MS_R/MS_res
pv = 1 - f(p-1, n-p).cdf(F)
print(f"{F=}, {pv=}")
```

```
F=7.158376119866064, pv=0.0023475553240575042
```

1.4 d.

[101]:
```
R2 = SoS_R / SoS_T
R_adj = (1- ((n-1)/(n-p)) * (1-R2))
R_adj = 1- MS_res/MS_T

print(f"{R2=}, {R_adj=}")
```

```
R2=0.6716197701565084, R_adj=0.5777968473440822
```

1.5 e.

[102]:
```
x = np.array([1,20,30,90,2])
y = beta_star.squeeze()@x.T

print(f"{y=}")
```

```
y=8.99567772483296
```

1.6 f.

[103]:
```
se_hat = np.sqrt(var_hat * x.T @ np.linalg.inv(X.T @ X) @ x)
print(f"confidence interval at 95% is [{y - 2 * se_hat}, {y + 2 * se_hat}]")
```

```
confidence interval at 95% is [8.050787013415139, 9.940568436250782]
```

1.7 g.

[104]:
```
print(f"the confidence interval is [{y - 2 * np.sqrt(se_hat ** 2 + np.
↪sqrt(var_hat))}, {y + 2 * np.sqrt(se_hat ** 2 + np.sqrt(var_hat))}]")
```

```
the confidence interval is [6.892402845682218, 11.098952603983703]
```

2 Question 5

2.1 a.

[105]:
```
best_model = (0, "")
df = pd.read_csv("ex3.csv")
for model_params in [["x2","x3","x4"], ["x2","x3","x5"], ["x2","x5","x4"],␣
↪sorted(["x5","x3","x4"])]:
    X = df[model_params].to_numpy()
    X = np.c_[np.ones(X.shape[0]), X]
    y = df[["y"]].to_numpy()
    beta_star2 =np.linalg.inv(X.T@X)@X.T@y

    y_hat = X@beta_star2.squeeze()
    p_new = X.shape[1]
    n_new = y.shape[0]
    e_new = y.squeeze()- y_hat
    var_hat = e_new.T@e_new/(n_new-p_new)

    SoS_res_new = (y.squeeze() - y_hat).T@(y.squeeze() - y_hat)
    SoS_T_new = (y.squeeze()-np.ones(n)*np.mean(y)).T@(y.squeeze()-np.
↪ones(n)*np.mean(y))
    SoS_R_new = SoS_T_new - SoS_res_new

    beta_star2 = np.linalg.inv(X.T@X)@X.T@y

    MS_R_new = SoS_R_new/(p_new-1)
    MS_res_new = SoS_res_new/(n_new-p_new)
    MS_T_new = SoS_T_new/(n_new-1)

    R2 = SoS_R_new / SoS_T_new
    R_adj_new = 1-MS_res_new/MS_T_new

    F_new = MS_R_new/MS_res_new
    pv_new = 1-f(p_new-1, n_new-p_new).cdf(F_new)

    df_res_new = p_new-1
    df_T_new = n_new-p_new
    df_R_new = n_new-1

    print(f"{R2=}, f{model_params=}")
```

```
    if R2>best_model[0]:
        best_model = (R2, model_params, beta_star2, (SoS_res_new, SoS_T_new,␣
↪SoS_R_new),
                      (MS_res_new, MS_R_new, MS_T_new), (F_new, pv_new),␣
↪(df_res_new, df_T_new, df_R_new), R_adj_new)
print(f"the best model gave an r squared value of {best_model[0]} and the␣
↪parameters used were {best_model[1]}")
```

```
R2=0.6525266745447751, fmodel_params=['x2', 'x3', 'x4']
R2=0.5863475902220495, fmodel_params=['x2', 'x3', 'x5']
R2=0.3275448510986872, fmodel_params=['x2', 'x5', 'x4']
R2=0.6713212019763024, fmodel_params=['x3', 'x4', 'x5']
the best model gave an r squared value of 0.6713212019763024 and the parameters
used were ['x3', 'x4', 'x5']
```

the exact model we assume is $Y = \beta_0 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \epsilon$

2.2 b.

[106]:
```
print(f"the parameters for beta star is this model is {best_model[2].
↪squeeze()}")
print(f"compared to the parameters of the full model {beta_star.squeeze()}")
```

```
the parameters for beta star is this model is [ 7.31012315  0.51888685
-0.10381156 -2.25553769]
compared to the parameters of the full model [ 7.45780659 -0.0297028
0.52051008 -0.10180238 -2.1605807 ]
```

it can be easily seen that both models have similar betas. if we ignore the second entry in the full model's we can see that the parameters are almost the same.

2.3 c.

[107]:
```
SoS_res_new, SoS_T_new, SoS_R_new = best_model[3]
print(f"{SoS_res=}\t {SoS_T=}\t {SoS_R=}")
print(f"{SoS_res_new=}\t {SoS_T_new=}\t {SoS_R_new=}\n")

MS_res_new, MS_R_new, MS_T_new = best_model[4]
print(f"{MS_res=}\t {MS_T=}\t {MS_R=}")
print(f"{MS_res_new=}\t {MS_T_new=}\t {MS_R_new=}\n")

df_res_new, df_T_new, df_R_new = best_model[6]
print(f"{df_res=}\t {df_T=}\t {df_R=}")
print(f"{df_res_new=}\t {df_T_new=}\t {df_R_new=}\n")

F_new, pv_new = best_model[5]
print(f"{F=}\t {pv=}")
print(f"{F_new=}\t {pv_new=}")
```

```
SoS_res=10.909136898800627      SoS_T=33.22105263157895
SoS_R=22.311915732778324
SoS_res_new=10.919055648029365  SoS_T_new=33.22105263157895
SoS_R_new=22.301996983549586

MS_res=0.7792240642000448       MS_T=1.8456140350877195
MS_R=5.577978933194581
MS_res_new=0.7279370432019577   MS_T_new=1.8456140350877195
MS_R_new=7.433998994516529

df_res=4         df_T=14         df_R=18
df_res_new=3     df_T_new=15     df_R_new=18

F=7.158376119866064       pv=0.0023475553240575042
F_new=10.212420241476915        pv_new=0.0006487416675347024
```

The values are very similar. as excpected because we've seen that the second parameter is almost redundent and doesn't affect eh model, so removing it shouldn't decrease performance by much

2.4 d.

[108]:
```
R_adj_new = best_model[7]
print(f"{R_adj_new=} compared to {R_adj}")
```

```
R_adj_new=0.6055854423715629 compared to 0.5777968473440822
```

we can the see that the new $R_(adj)$ is bigger for the new model

3 e.

The model assumptions are that the data is linear with normal noise. and we can see tell that the assumptions aren't that far from reality because we still manage to get good R2 scores and our parameters seem to make sense.

The assumptions hold the same as in the previous model because the redundant variable didn't affect much. thus the only difference is that in one model we assume that y is connected linearly to x2,x3,x4,x5 and in the other we assumed that it's connected only to x3,x4,x5