Problem Set 2

1. The exponential distribution

Problem a

The support of X is $x \geq 0$

Problem b

$L(\theta) = \prod_{i=1}^{n} \theta e^{-\theta x_i}$ $\log L(\theta|x) = n \log\theta - \theta \sum_{i=1}^{n} x_i$

Problem c

Let $\frac{d}{d\theta} \log L(\theta) = \frac{n}{\theta} - \sum_{i=1}^{n} x_i = 0$ We have $\hat{\theta} = \frac{n}{\sum_{i=1}^{n} x_i}$

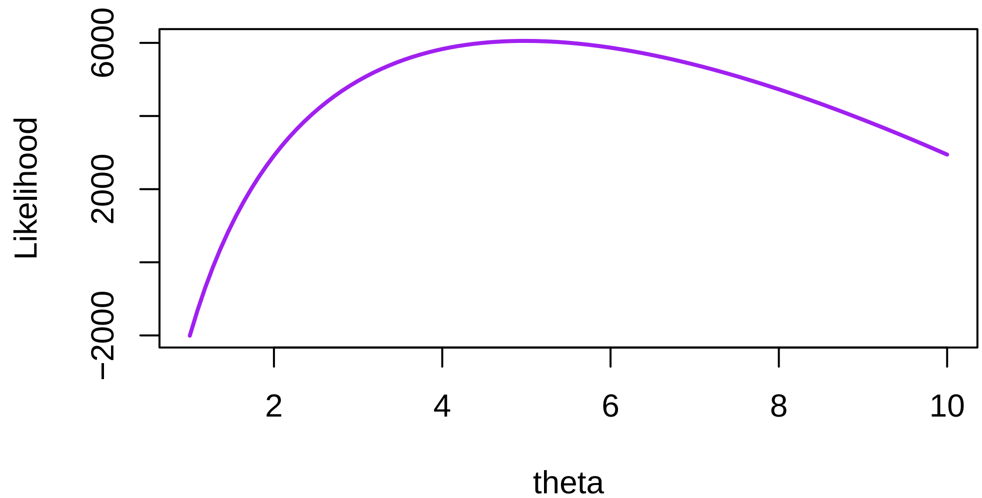
Problem d

```
library(formatR)
```

```
Warning: package 'formatR' was built under R version 4.4.2
```

```
set.seed(5)
x <- rexp(10000, 5)

exp.ll = function(theta, x){
  n = length(x)
  return(n*log(theta) - theta*sum(x))
}
```

```r
theta_seq = seq(1, 10, length.out = 100)
log_log_vals = sapply(theta_seq, exp.ll, x = x)

plot(theta_seq, log_log_vals, type="l",
     xlab="theta", ylab="Likelihood", col="purple", lwd=2)
```



It looks like around 5

```r
theta_tilde = 6
theta_hat = length(x) / sum(x)

ll_hat = exp.ll(theta_hat, x)
ll_tilde = exp.ll(theta_tilde, x)

likelihood_ratio = exp(ll_hat - ll_tilde)
print(likelihood_ratio)
```

```
[1] 2.05498e+80
```

```r
log_likelihood = function(theta, x) {
  n = length(x)
```

```r
  return(- (n*log(theta)-theta*sum(x)))
}
```

```r
start_time_BFGS = Sys.time()
result_BFGS = optim(par=1, fn=log_likelihood, x=x, method="BFGS")
```

```
Warning in log(theta): NaNs produced
Warning in log(theta): NaNs produced
Warning in log(theta): NaNs produced
Warning in log(theta): NaNs produced
Warning in log(theta): NaNs produced
Warning in log(theta): NaNs produced
Warning in log(theta): NaNs produced
```

```r
end_time_BFGS = Sys.time()
time_BFGS = end_time_BFGS - start_time_BFGS
start_time_SANN = Sys.time()
result_SANN = optim(par=1, fn=log_likelihood, x=x, method="SANN")
end_time_SANN = Sys.time()
time_SANN = end_time_SANN - start_time_SANN

print(result_BFGS$par)
```

```
[1] 4.980155
```

```r
print(result_SANN$par)
```

```
[1] 4.979926
```

```r
print(time_BFGS)
```

```
Time difference of 0.003666878 secs
```

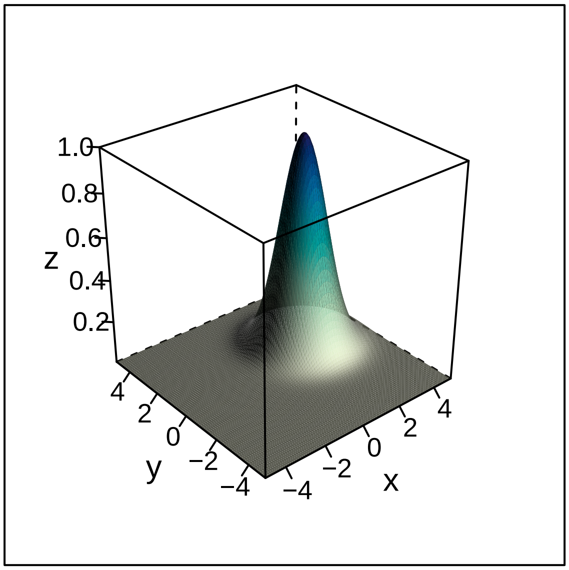
```r
print(time_SANN)
```

```
Time difference of 0.07271194 secs
```

2. Maximizing a multivariate function

Problem a

```
mvn <- function(xy) {
    x <- xy[1]
    y <- xy[2]
    z <- exp(-0.5 * ((x - 2)^2 + (y - 1)^2))
    return(z)
}

# install.packages('lattice')
library(lattice)

y <- x <- seq(-5, 5, by = 0.1)
grid <- expand.grid(x, y)
names(grid) <- c("x", "y")
grid$z <- apply(grid, 1, mvn)

wireframe(z ~ x + y, data = grid, shade = TRUE, light.source = c(10, 0, 10), scales = list(a
```



It looks like when $x = 0, y = 0$ the function achieves a maximum.

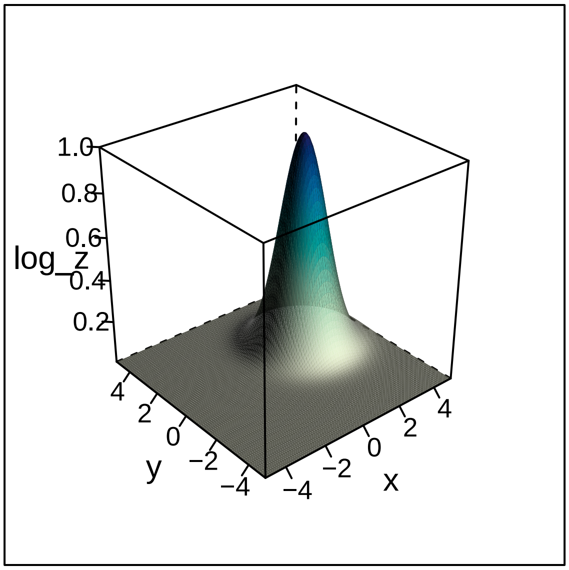
Problem b

```
neg_mvn <- function(xy) {
    return(-mvn(xy))
}

opt1 <- optim(c(1, 0), neg_mvn, method = "BFGS")

opt2 <- optim(c(5, 5), neg_mvn, method = "BFGS")

data.frame(
    Start_Point = c("(1,0)", "(5,5)"),
    Optimum_X = c(opt1$par[1], opt2$par[1]),
    Optimum_Y = c(opt1$par[2], opt2$par[2]),
    Function_Value = c(-opt1$value, -opt2$value)
)
```

```
  Start_Point Optimum_X Optimum_Y Function_Value
1       (1,0)  2.000000  1.000000   1.000000e+00
2       (5,5)  4.998888  4.998518   3.761331e-06
```

The optimization starting from (5,5) did not converge to (2,1). The function value is very small, meaning that it has not reached the peak.

```
y <- x <- seq(-5, 5, by = 0.1)
grid <- expand.grid(x, y)
names(grid) <- c("x", "y")
grid$log_z <- apply(grid, 1, mvn)
wireframe(log_z ~ x + y, data = grid, shade = TRUE, light.source = c(10, 0, 10), scales = li
```



Problem c

```
neg_log_mvn <- function(xy) {
    return(-mvn(xy))
}

opt1_log <- optim(c(1, 0), neg_log_mvn, method = "BFGS")

opt2_log <- optim(c(5, 5), neg_log_mvn, method = "BFGS")

data.frame(
    Start_Point = c("(1,0)", "(5,5)"),
    Optimum_X = c(opt1_log$par[1], opt2_log$par[1]),
    Optimum_Y = c(opt1_log$par[2], opt2_log$par[2]),
    Log_Function_Value = c(-opt1_log$value, -opt2_log$value)
)
```

```
  Start_Point Optimum_X Optimum_Y Log_Function_Value
1       (1,0)  2.000000  1.000000       1.000000e+00
2       (5,5)  4.998888  4.998518       3.761331e-06
```

Using log-likelihood improves numerical stability by preventing underflow and helps the optimizer converge faster by smoothing sharp variations. While function values change due to the logarithm, the optimal (x, y) remains the same.

3. The normal variance

Problem a

log normal distribution is:

$\ell(\mu, \sigma^2) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2.$

derivative:

$\frac{\partial\ell}{\partial\sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n}(x_i - \mu)^2.$

equal to zero:

$-\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n}(x_i - \mu)^2 = 0.$

then

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

Problem b

Step 1: MLE for Variance

From Problem a, the maximum likelihood estimator (MLE) for variance is $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$

Step 2: Expected Value of $\hat{\sigma}^2$

To check if $\hat{\sigma}^2$ is biased, take its expectation: $E[\hat{\sigma}^2] = E\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2\right]$

By linearity of expectation: $E\left[\sum_{i=1}^{n}(x_i - \mu)^2\right] = \sum_{i=1}^{n}E\left[(x_i - \mu)^2\right]$

Since $x_i \sim \mathcal{N}(\mu, \sigma^2)$, we know $E\left[(x_i - \mu)^2\right] = \sigma^2$

Thus, $E[\hat{\sigma}^2] = \frac{1}{n} \cdot n\sigma^2 = \sigma^2$

This suggests that $\hat{\sigma}^2$ is an unbiased estimator **only if** μ is known.

Step 3: Why is the MLE Biased?

In practice, we do not know μ and instead estimate it with the **sample mean**: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

Using $\bar{x}$ instead of μ, the variance estimator becomes $S^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$

Taking expectation: $E[S^2] = \frac{1}{n}\sum_{i=1}^{n} E[(x_i - \bar{x})^2]$

It can be shown that: $E[(x_i - \bar{x})^2] = \frac{n-1}{n}\sigma^2$

which leads to: $E[S^2] = \frac{n-1}{n}\sigma^2$

Since $E[S^2] \neq \sigma^2$, the MLE underestimates σ^2 by a factor of$\frac{n-1}{n}$.

Step 4: Unbiased Estimator

To correct the bias, we rescale S^2: $s^2 = \frac{n}{n-1}\hat{\sigma}^2$

This gives the **unbiased variance estimator**:$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

which satisfies: $E[s^2] = \sigma^2$

Final Answer:

- **MLE for variance**:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

- **MLE is biased**, underestimating σ^2 by a factor of $\frac{n-1}{n}$.
- **Unbiased estimator**:$s^2 = \frac{n}{n-1}\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

This adjustment ensures an unbiased estimate ofσ^2.

Appendix

I certify that we did not use any LLM or generative AI tool in this assignment.