Problem 1:

a) **Label leakage** is when information about the goal or target variable leaks into the features that are being used to train the model itself. It essentially is like giving the answers to the model. It essentially refers to a model learning from data that it realistically should not have access to. **Training example leakage** occurs in machine learning when a model is trained using information that would not be available during prediction. This means that in the real-world, the model would not be as useful as it seems when it DOES have access to additional information. The model inadvertently learns from data it shouldn't have access to, causing an overestimation of its predictive capabilities.
b) **P hacking** refers to statistical decisions during research that intend to produce results that are statistically significant in an artificial manner. Essentially, rather than actual testing and analysis, the researchers try to find a way to test that makes their results valid. One example of p hacking is premature stopping of data collection. In this, a researcher may halt an experiment as soon as significant results are observed. This issue is very important and needs to be addressed since I intend to be a data scientist/computer science where falsifying results is never a safe option. In computer science, falsifying unit tests for example can have detrimental implications for an application's reliability. **Self-Selection Bias** is a type of sampling bias that occurs when individuals select themselves into a group, leading to a non-random sample that may not represent the broader population. In the field of computer science that increasingly relies on decision-making, algorithms, and user-centered design, overlooking self-selection bias can have several negative consequences.
c) A critical insight from the lecture could be recognizing the subtle ways data leakage can manifest, such as inadvertently using future data or test data in the training process. This misstep can lead to overestimated performance and a misleading evaluation of the model's predictive power. Another takeaway might be learning specific strategies to mitigate data leakage, such as rigorous separation of training and test sets, implementing time-based splits for time-series data, or using cross-validation techniques carefully to ensure no information leaks from the test set into the training set during model evaluation

Problem 2:

a) model <- lm(total ~ expend + ratio + salary + takers, data = sat)
summary(model)

```
Call:
lm(formula = total ~ expend + ratio + salary + takers, data = sat)

Residuals:
    Min      1Q  Median      3Q     Max
-90.531 -20.855  -1.746  15.979  66.571

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1045.9715    52.8698  19.784  < 2e-16 ***
expend         4.4626    10.5465   0.423    0.674
ratio         -3.6242     3.2154  -1.127    0.266
salary         1.6379     2.3872   0.686    0.496
takers        -2.9045     0.2313 -12.559 2.61e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 32.7 on 45 degrees of freedom
Multiple R-squared:  0.8246,    Adjusted R-squared:  0.809
F-statistic: 52.88 on 4 and 45 DF,  p-value: < 2.2e-16
```

b) sat.new <- data.frame(expend = c(6),
ratio = c(16),
salary = c(35),
takers = c(30))

prediction_intervals <- predict(
model,
newdata = sat.new,
interval = "prediction",
level = 0.99
)

prediction_intervals

```
> prediction_intervals
       fit      lwr      upr
1 984.9521 895.7647 1074.139
```

c) The interval, in terms of width, will be narrower if we build a 99% confidence interval instead of a prediction interval. The center of both the confidence interval and the prediction interval is the same.

d) confidence_intervals <- predict(
model,
newdata = sat.new,
interval = "confidence",
level = 0.99
)

Confidence_intervals are as follows:

```
       fit      lwr      upr
1 984.9521 970.1753 999.7288
```

e) The interval narrows because the 95% interval captures less uncertainty compared to the 99% interval. The center stays the same. When changing the ratio to 25, the mean response would decrease because of the negative coefficient for ratio. This would shift the interval's center downward. The width may also change due to invariability.

```
> sat.new <- data.frame(expend = c(6),
+                       ratio = c(16),
+                       salary = c(35),
+                       takers = c(30))
> confidence_intervals <- predict(
+   model,
+   newdata = sat.new,
+   interval = "confidence",
+   level = 0.95
+ )
> confidence_intervals
       fit      lwr      upr
1 984.9521 973.8864 996.0177
```

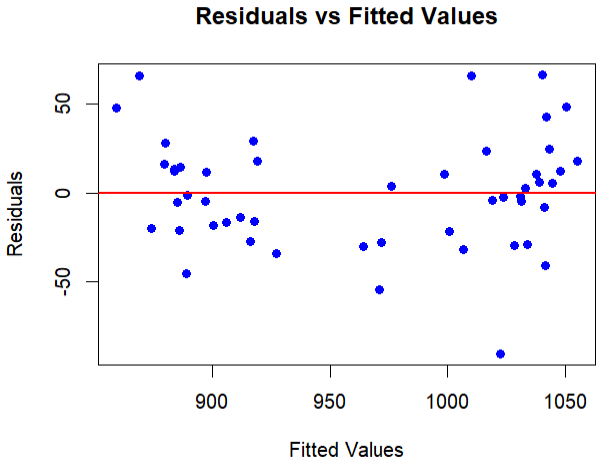
```
> sat.new <- data.frame(expend = c(6),
+                       ratio = c(25),
+                       salary = c(35),
+                       takers = c(30))
> confidence_intervals <- predict(
+   model,
+   newdata = sat.new,
+   interval = "prediction",
+   level = 0.99
+ )
> confidence_intervals
      fit      lwr      upr
1 952.334 838.6646 1066.003
```

f) The interval for the ratio of 25 is slightly wider, possibly due to greater uncertainty or variability introduced. We can see that even with the 95% confidence interval, the center is the same as the 99% confidence interval.

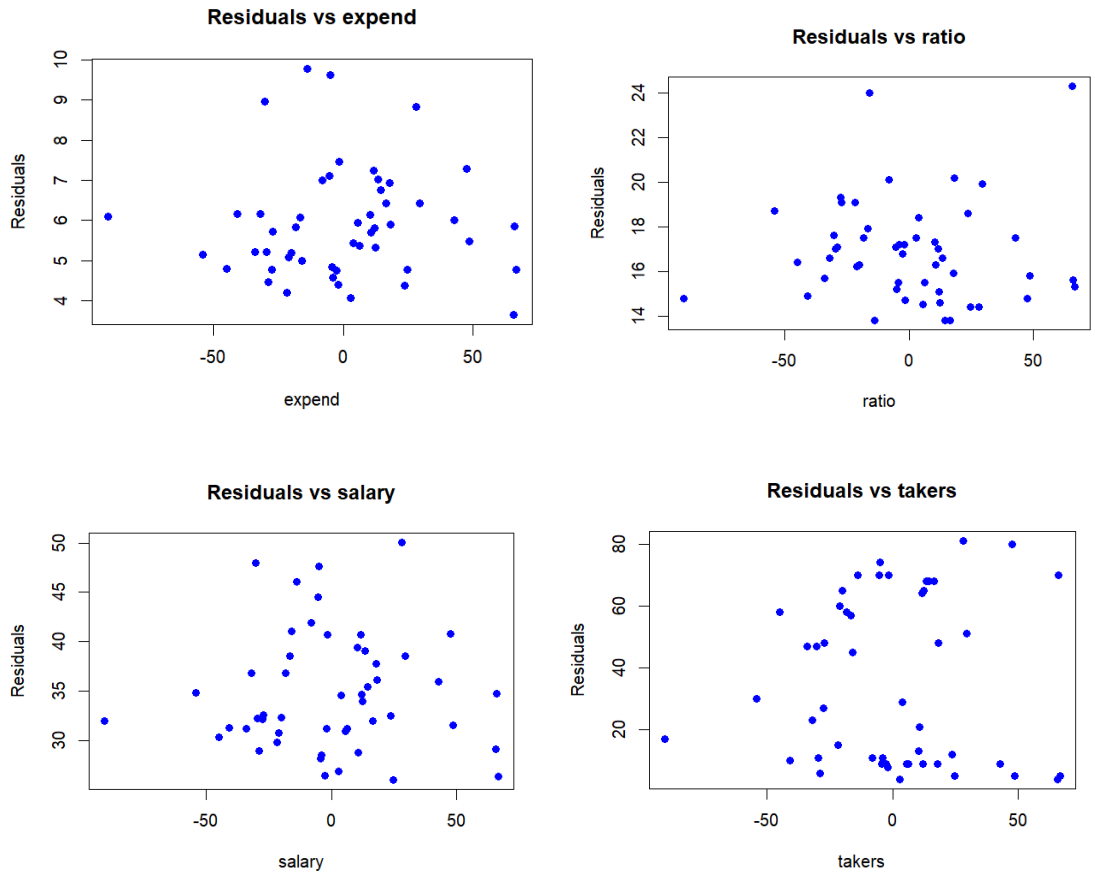
Problem 3

```
#Problem 3
model <- lm(total ~ expend + ratio + salary + takers, data = sat)
summary(model)
plot(model$fitted.values, residuals(model),
     main = "Residuals vs Fitted Values",
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "blue")

abline(h = 0, col = "red", lwd = 2)
```

a)

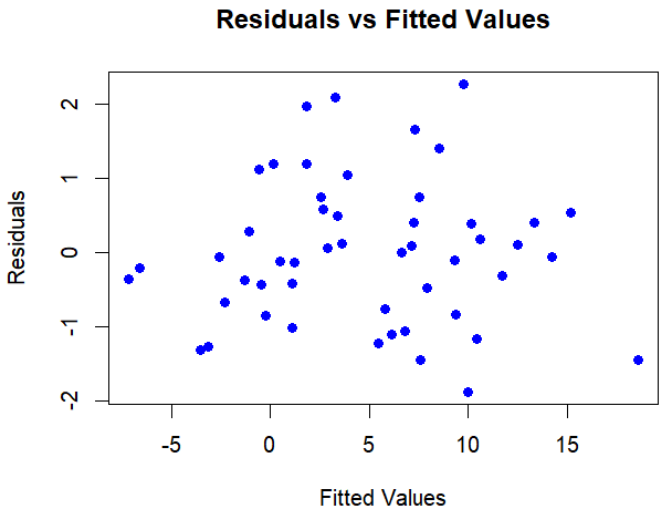


b)



The equal variance assumption appears to be reasonably satisfied for this model.

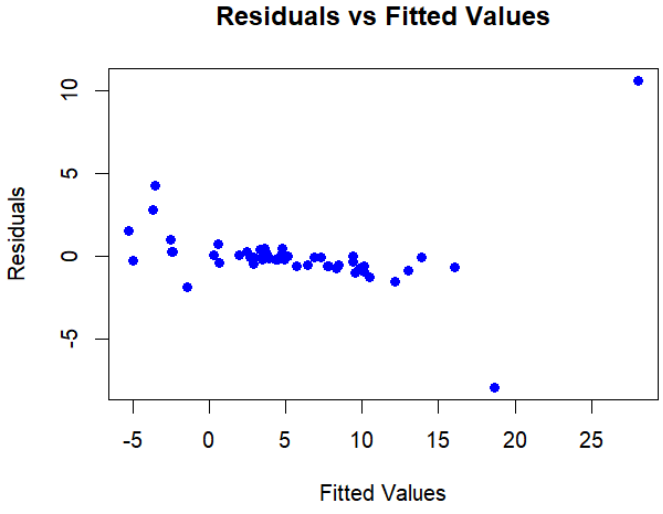
c)
```
set.seed(123)

x1 <- rnorm(50)
x2 <- rnorm(50)
y <- 5 + 6*x1 - x2 + rnorm(50)

model2 <- lm(y ~ x1 + x2)
plot(model2$fitted.values, residuals(model2),
    main = "Residuals vs Fitted Values",
    xlab = "Fitted Values",
    ylab = "Residuals",
    pch = 19,
    col = "blue")
```



The residuals are scattered randomly around the horizontal line y=0, which suggests that the model's assumptions about the residuals. It aligns with the linear nature of the data The addition of random noise ϵ~N(0,1) explains the variability seen in the residuals.

d)

```
#3d
x1 <- rnorm(50)
x2 <- rnorm(50)
y <- 5 + 6*x1 - x2 + rnorm(50, 0, x1^2)
model3 <- lm(y ~ x1 + x2)
plot(model3$fitted.values, residuals(model3),
     main = "Residuals vs Fitted Values",
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19,
     col = "blue")
```



The residuals appear to follow a noticeable pattern. At lower fitted values, residuals are clustered near zero, while at higher fitted values, there's a clear increase in variance,

with some extreme values. The residuals show no curvature, confirming that the linear relationship in the model (with coefficients for x1 and x2) is appropriate.