Assignment 12b

Satvik Saha

2024-11-26

Answer 1

(a) In the model

$$y = \beta_0 + \beta_1 e^{-x} + \beta_2 e^{-2x} + \text{error},$$

the deterministic part $\mathbb{E}[y \mid x] = \beta^\top X$ part of the model is linear in the (transformed) covariates X. In fact, the MLE $\hat{\beta}$ for the coefficients is a weighted linear combination $(X^\top X)^- X^\top y$ of the reponses, the weights being functions of the covariates.

On the other hand, in the model

$$y = \beta_0 + \beta_1 e^{-\beta_2 x} + \text{error},$$

there is no design X such that the deterministic part $\mathbb{E}[y \mid x] = \beta^\top X$. This can be easily verified by differentiating $\mathbb{E}[y \mid x]$ with respect to β.

(b) The error in y gets absorbed into the error term in the model, so our regression remains unbiased. However, our estimate of the error term will be inflated.

(c) The regression coefficients will become biased (see a short derivation in Answer 3); specifically, the estimated slope will always be shallower than the 'true' slope b.

Answer 2

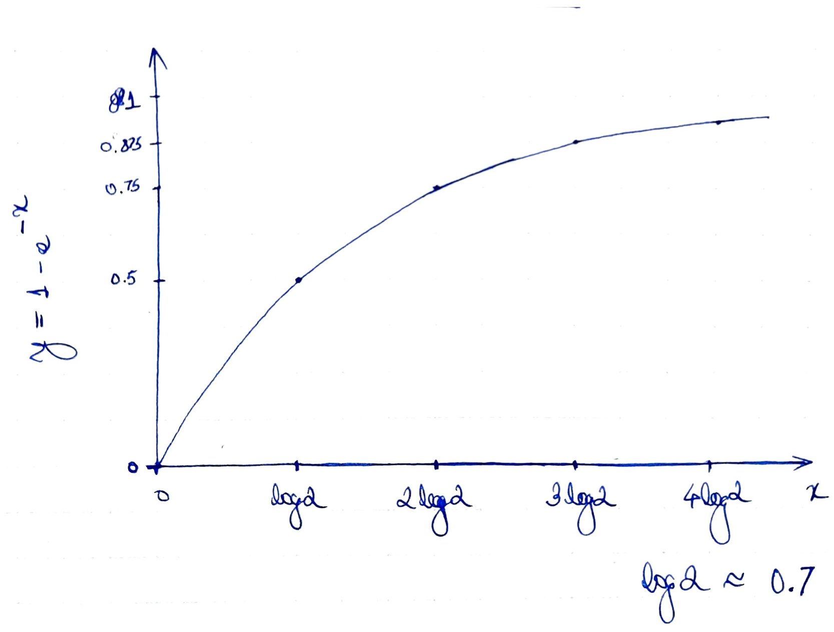


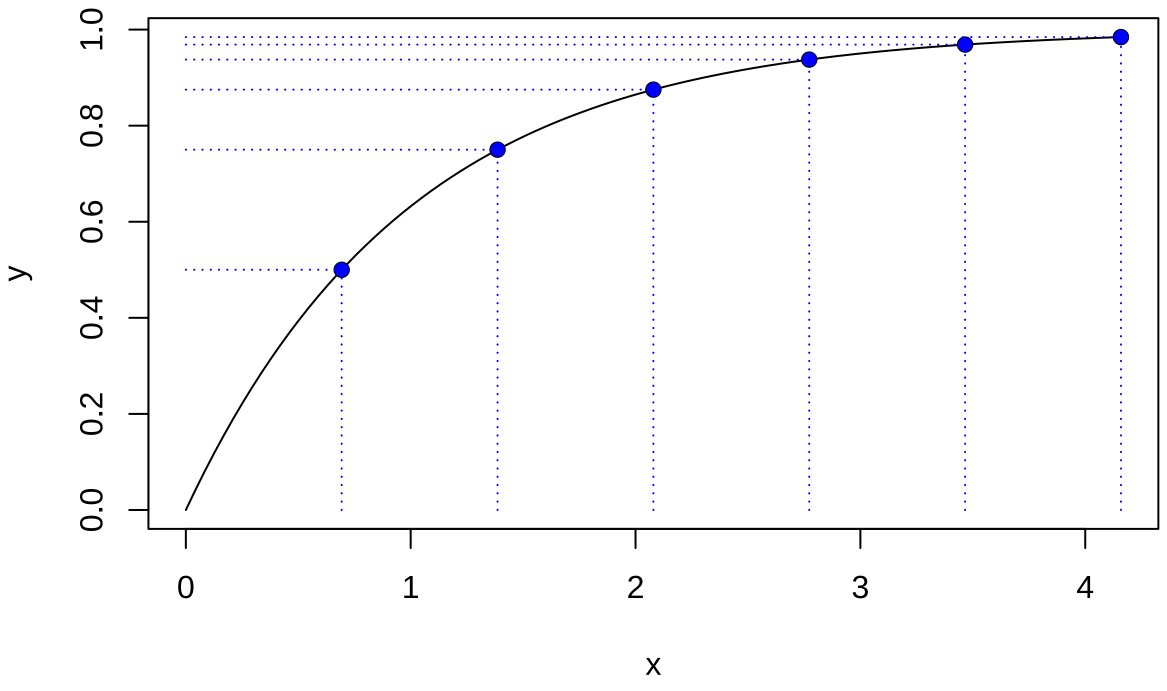
Figure 1: A plot of $y = 1 - e^{-x}$

(a)

(b)

```
f <- function(x) (1 - exp(-x))
x <- seq(0, log(2^6), length.out = 100)
y <- f(x)
plot(x, y, type = "l")

# Show special points
x_0 <- log(2^(1:6))
y_0 <- f(x_0)
points(x_0, y_0)
segments(x_0, 0, x_0, y_0, col = "blue", lty = 3)
segments(0, y_0, x_0, y_0, col = "blue", lty = 3)
points(x_0, y_0, col = "blue", pch = 16)
```



At points $x_n = n \log 2$, the curve attains values $y_n = 1 - 2^{-n}$, i.e. each step of $\log 2$ along the x-axis brings us half as close to 1 on the y-axis as before.

Answer 3

We have set up a model

$$y_i \mid x_i \sim N(a + bx, \sigma^2), \qquad u_i \mid x_i \sim N(x_i, \sigma_u^2),$$

and will perform regressions of y on $a + bu$. Note that is this case,

$$\hat{b} = \frac{\mathrm{cov}(u_i, y_i)}{\mathrm{var}(u_i)} = \frac{\mathrm{cov}(u_i - x_i, y_i) + \mathrm{cov}(x_i, y_i)}{\mathrm{var}(u_i - x_i) + \mathrm{var}(x_i)} \to^p \left(\frac{\sigma_x^2}{\sigma_u^2 + \sigma_x^2} \right) b < b.$$

Thus, this regression will be biased.

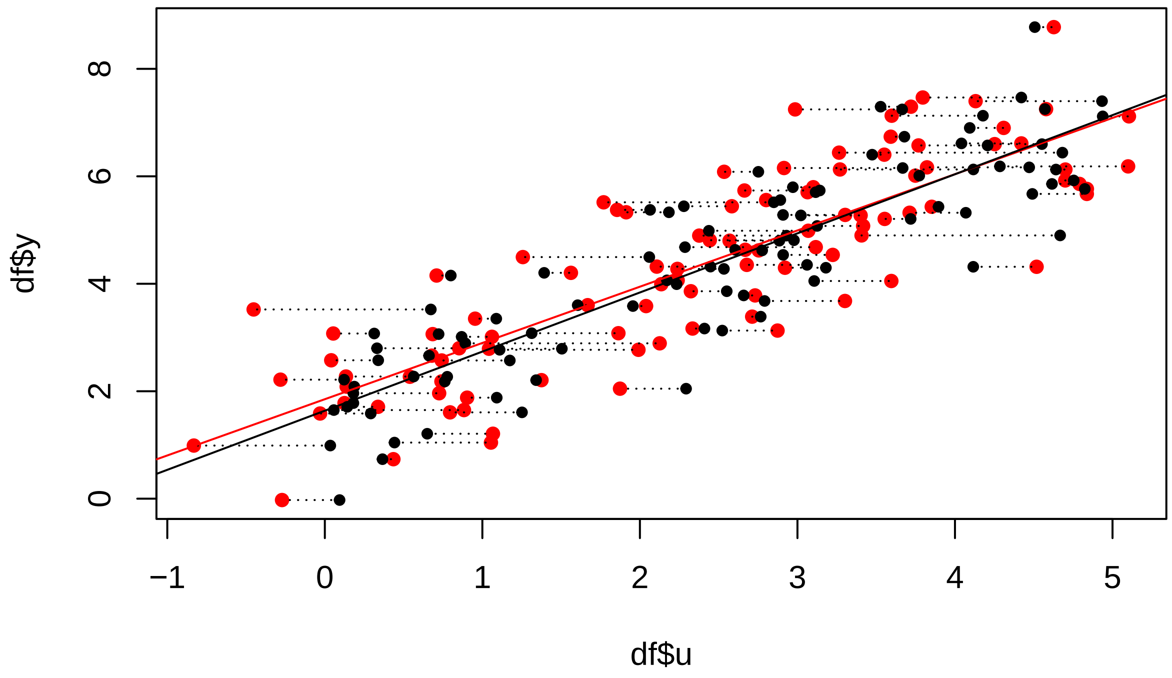
```
gendata <- function(n, x, a = 2.0, b = 1.0, sigma = 1.0, sigma_u = 0.5) {
    u <- rnorm(n, mean = x, sd = sigma_u)
    y <- rnorm(n, mean = a + b * x, sd = sigma)
    data.frame(
        x = x,
        u = u,
        y = y
    )
}
```

```
n <- 100
x <- runif(n, 0, 5)
df <- gendata(n, x)
plot(df$u, df$y, col = "red", pch = 16)
segments(df$x, df$y, df$u, df$y, lty = 3)
points(df$x, df$y, pch = 20, col = "black")

fit.u <- lm(y ~ u, data = df)
abline(fit.u$coef[1], fit.u$coef[2], col = "red")
fit.u
```

```
##
## Call:
## lm(formula = y ~ u, data = df)
##
## Coefficients:
## (Intercept)            u
##       1.851        1.047
```

```
fit.x <- lm(y ~ x, data = df)
abline(fit.x$coef[1], fit.x$coef[2], col = "black")
```



```
fit.x
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
```

```
##
## Coefficients:
## (Intercept)            x
##       1.637        1.100
```

Here, the true data in black has been perturbed on the x-axis to the data points in red. The regression line thus obtained (in red) is not as steep as the one on the true covariates (in black), consistently across multiple simulations.

Research homework assignment

Consider an experiment where we measure test scores x_i of n students, then scores y_i from a (similar) test a few months later. In the treatment group, an intervention in the form of an advanced course was administered to the students. In this scenario, the course has the effect of benefiting students who already had a better score more; it also introduces some additional variability, thus making the correlation with the pre-treatment scores weaker.

```r
gendata <- function(n, x, a = 0.0, b = 1.0, c = 0.5, d = -10, sigma = 2.0, sigma_t = 4.0) {
    z <- rbinom(n, 1, 0.5)
    y <- rnorm(
        n,
        mean = a + (b + c * z) * x + d * z,
        sd = ifelse(z == 0, sigma, sqrt(sigma^2 + sigma_t^2))
    )
    data.frame(
        x = x,
        y = y,
        z = z
    )
}
```

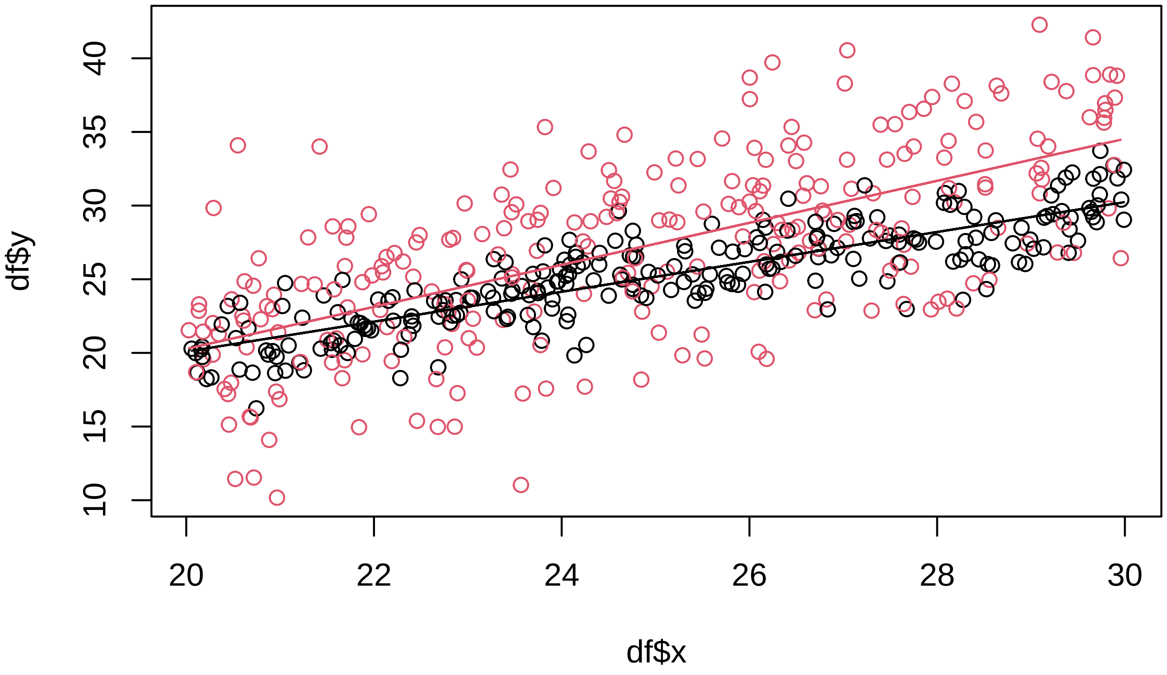
Here's an example with lots of data points to give some sense of the data generation mechanism. The black points are from the control, red from the treatment.

```r
n <- 500
x <- runif(n, 20, 30)
df <- gendata(n, x)

plot(df$x, df$y, col = df$z + 1)
fit <- lm(y ~ x * z, data = df)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x * z, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3229  -1.7540   0.1176   2.0768  13.0316
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.20407    2.08189  -0.098 0.921955
## x            1.01474    0.08262  12.282  < 2e-16 ***
## z           -8.03560    2.87689  -2.793 0.005421 **
## x:z          0.41101    0.11459   3.587 0.000368 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.661 on 496 degrees of freedom
## Multiple R-squared:  0.5067, Adjusted R-squared:  0.5037
## F-statistic: 169.8 on 3 and 496 DF,  p-value: < 2.2e-16
```

```
y.pred <- predict(fit, df)
lines(df$x[df$z == 0], y.pred[df$z == 0], col = 1)
lines(df$x[df$z == 1], y.pred[df$z == 1], col = 2)
```



```
cor(df$x[df$z == 0], df$y[df$z == 0])
```

```
## [1] 0.8500929
```

```
cor(df$x[df$z == 1], df$y[df$z == 1])
```

```
## [1] 0.651079
```

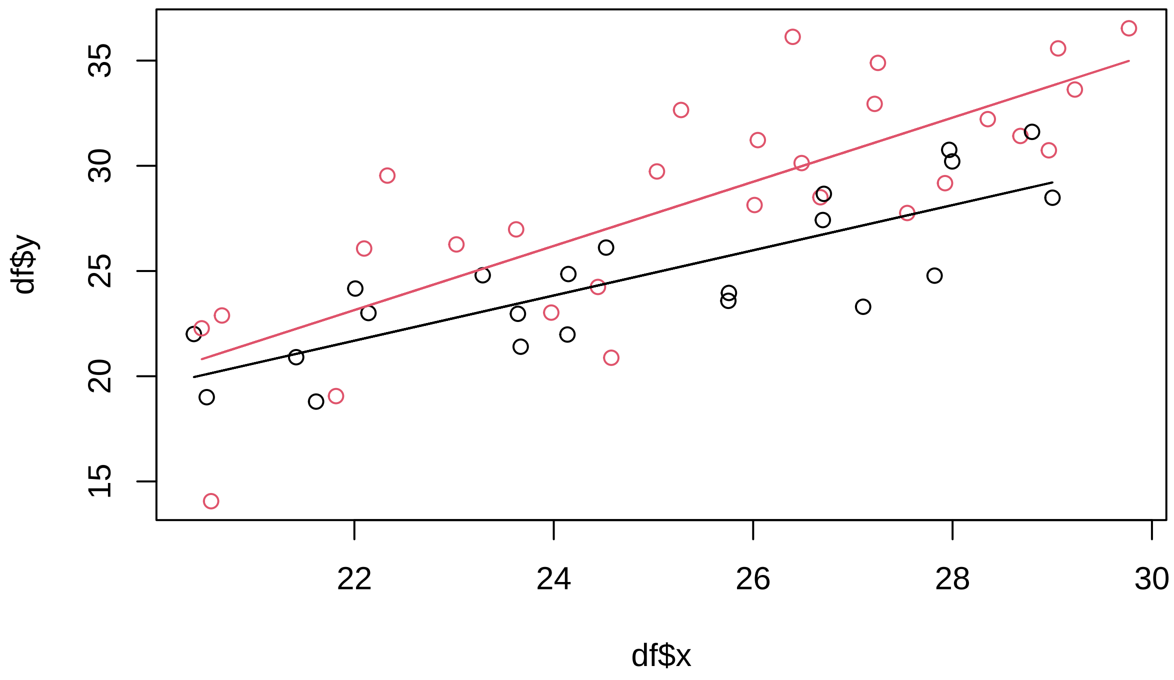
Repeating this for small n, see that the estimates are too noisy.

```
n <- 50
x <- runif(n, 20, 30)
df <- gendata(n, x)

plot(df$x, df$y, col = df$z + 1)
fit <- lm(y ~ x * z, data = df)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x * z, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.8909 -2.0591  0.4178  1.8937  6.2841
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.9421     5.7859  -0.336    0.739
## x             1.0741     0.2322   4.626 3.04e-05 ***
## z            -8.4522     7.6906  -1.099    0.277
## x:z           0.4503     0.3049   1.477    0.147
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.913 on 46 degrees of freedom
## Multiple R-squared:  0.6884, Adjusted R-squared:  0.6681
## F-statistic: 33.88 on 3 and 46 DF,  p-value: 1.034e-11
```

```r
y.pred <- predict(fit, df)
lines(df$x[df$z == 0], y.pred[df$z == 0], col = 1)
lines(df$x[df$z == 1], y.pred[df$z == 1], col = 2)
```

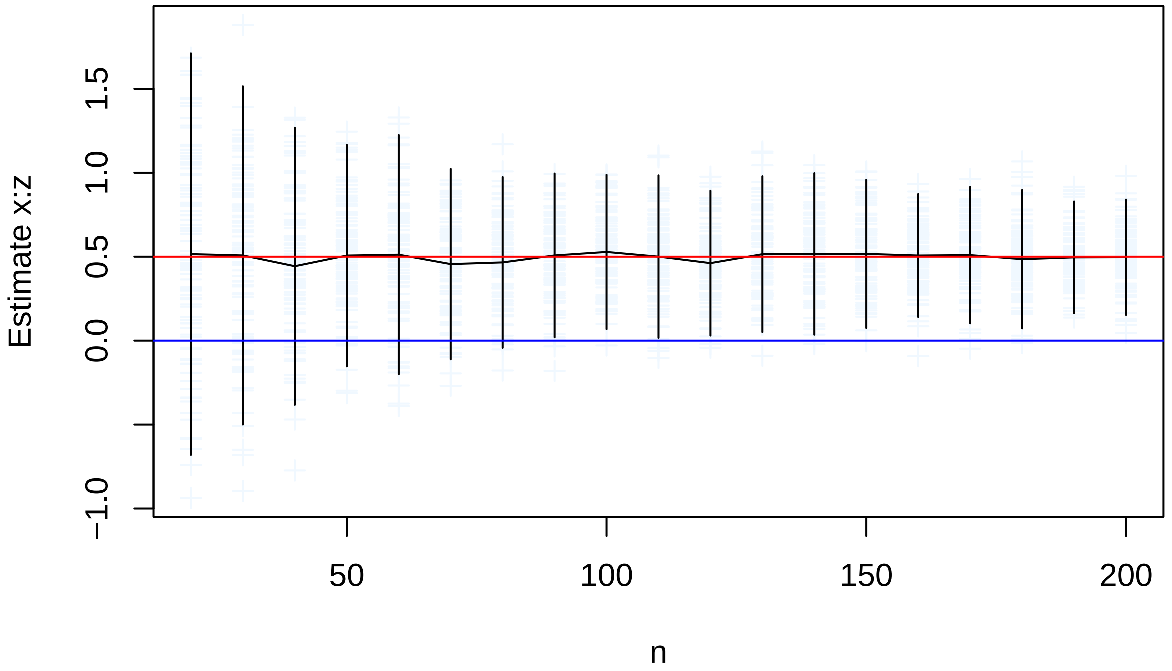


Let's focus on the estimates of the interaction term. We can run our fit for each n multiple times, then

```r
c.hat <- function(n) {
    x <- runif(n, 20, 30)
    df <- gendata(n, x)
    fit <- lm(y ~ x * z, data = df)
    fit$coefficients["x:z"]
}

n.values <- seq(20, 200, by = 10)
x.z <- sapply(n.values, function(n) replicate(100, c.hat(n)))
m <- apply(x.z, 2, mean)
s <- apply(x.z, 2, sd)
matplot(n.values, t(x.z), col = "aliceblue", pch = 3, xlab = "n", ylab = "Estimate x:z")
lines(n.values, m)
```

```r
segments(n.values, m - 2 * s, n.values, m + 2 * s)
abline(0.5, 0, col = "red")
abline(0.0, 0, col = "blue")
```



Around n between 100 and 150, the interaction term is estimated as significantly positive, which we judge as a good fit here.