Assignment 10b

Satvik Saha

2024-11-12

Answer 1

(a) The estimated treatment effect is $10.4 - 0.4 \times$ (pre-test score).

(b) If the estimated population average treatment effect is 7.1, then the population average pre-test score must be 8.25, via solving for $10.4 - 0.4x = 7.1$.

Answer 2

```
df <- data.frame(
    x = c(3, 5, 2, 8, 5, 10, 2, 11),
    z = c(0, 0, 1, 0, 0, 1, 1, 1),
    y0 = c(5, 8, 5, 12, 4, 8, 4, 9),
    y1 = c(5, 10, 3, 13, 2, 9, 1, 13)
)
df
```

```
##    x z y0 y1
## 1  3 0  5  5
## 2  5 0  8 10
## 3  2 1  5  3
## 4  8 0 12 13
## 5  5 0  4  2
## 6 10 1  8  9
## 7  2 1  4  1
## 8 11 1  9 13
```

```
df$tau <- df$y1 - df$y0
mean(df$tau)
```

```
## [1] 0.125
```

Thus, the average treatment effect in the population is 0.125.

```
treated <- df$z == 1
mean(df$tau[treated])
```

```
## [1] 0
```

Thus, the average treatment effect among the treated is 0.

```
y.treated   <- mean(df$y1[treated])
y.untreated <- mean(df$y0[!treated])
y.treated - y.untreated
```

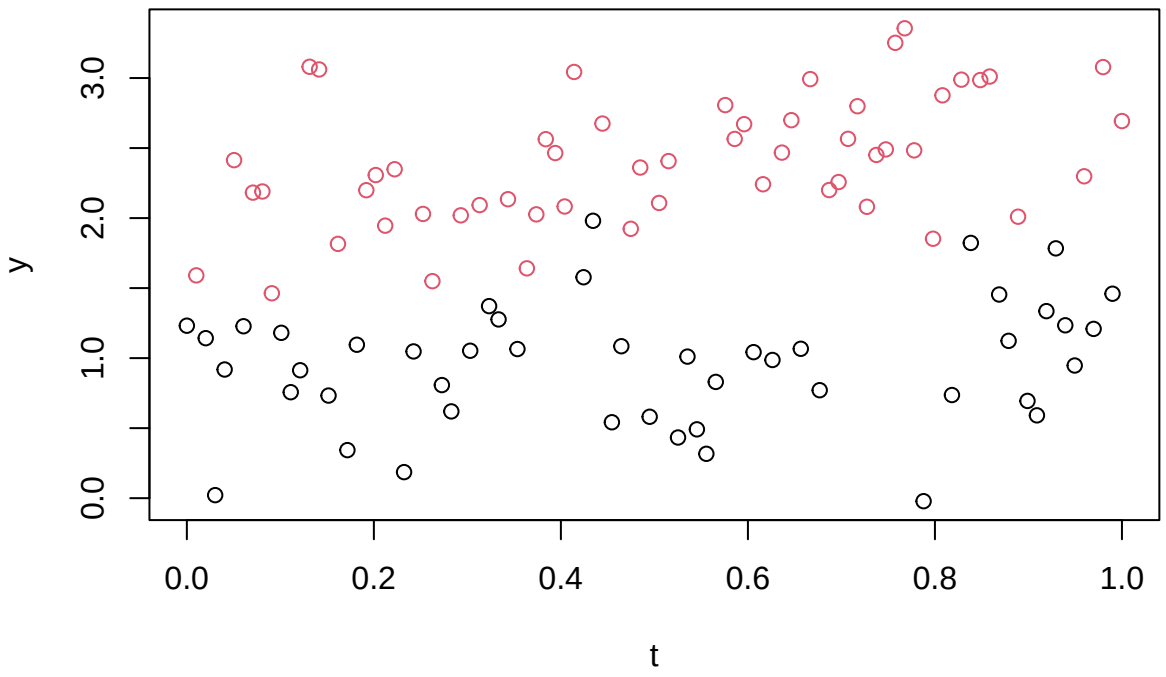
```
## [1] -0.75
```

Thus, a simple comparison between outcomes in the treated and untreated groups gives a treatment effect of -0.75.

Research homework assignment

Consider the following generation process.

```
gendata <- function(z, t, a = 1, b = 1.0, c = 0.7, sigma = 0.5) {
    y <- rnorm(1, mean = a + (b + c * t) * z, sd = sigma)
}

n <- 100
t <- seq(0, 1, length.out = n)

z <- rbinom(n, 1, 0.5)
y <- sapply(1:n, function(i) gendata(z[i], t[i]))

df <- data.frame(t = t, z = z, y = y)

plot(t, y, col = z + 1)
```



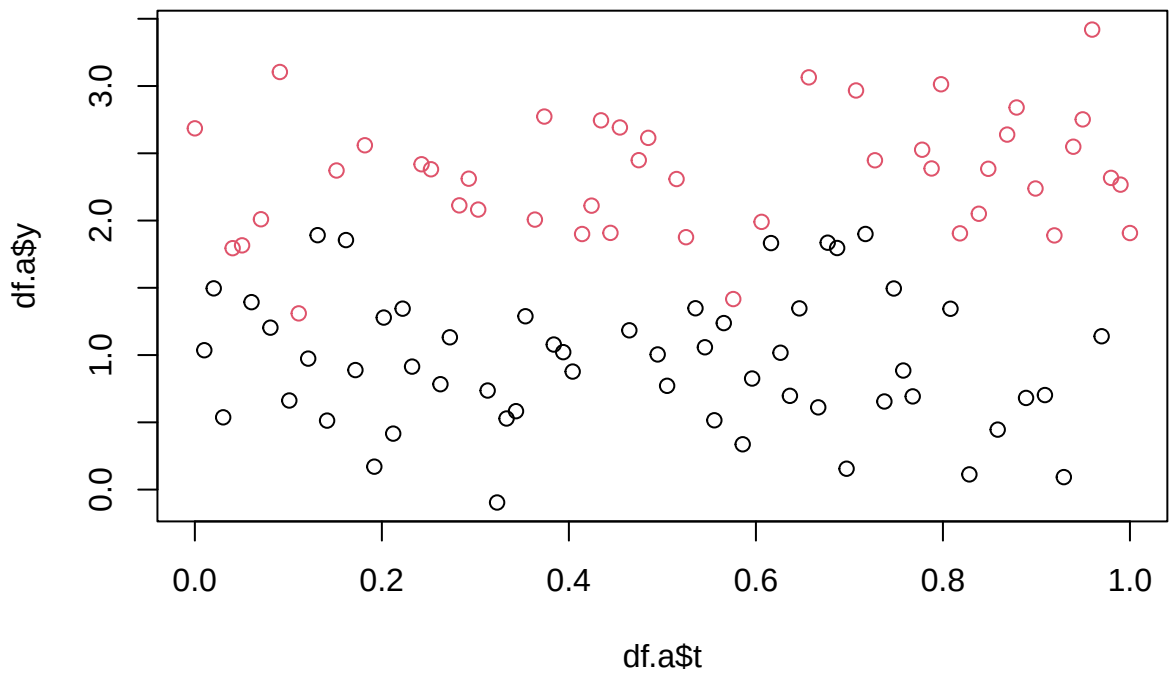

```
lm(y ~ z * t, data = df)
```

```
##
## Call:
## lm(formula = y ~ z * t, data = df)
##
## Coefficients:
## (Intercept)            z            t          z:t
##      0.8226       1.1973       0.2760       0.4837
```

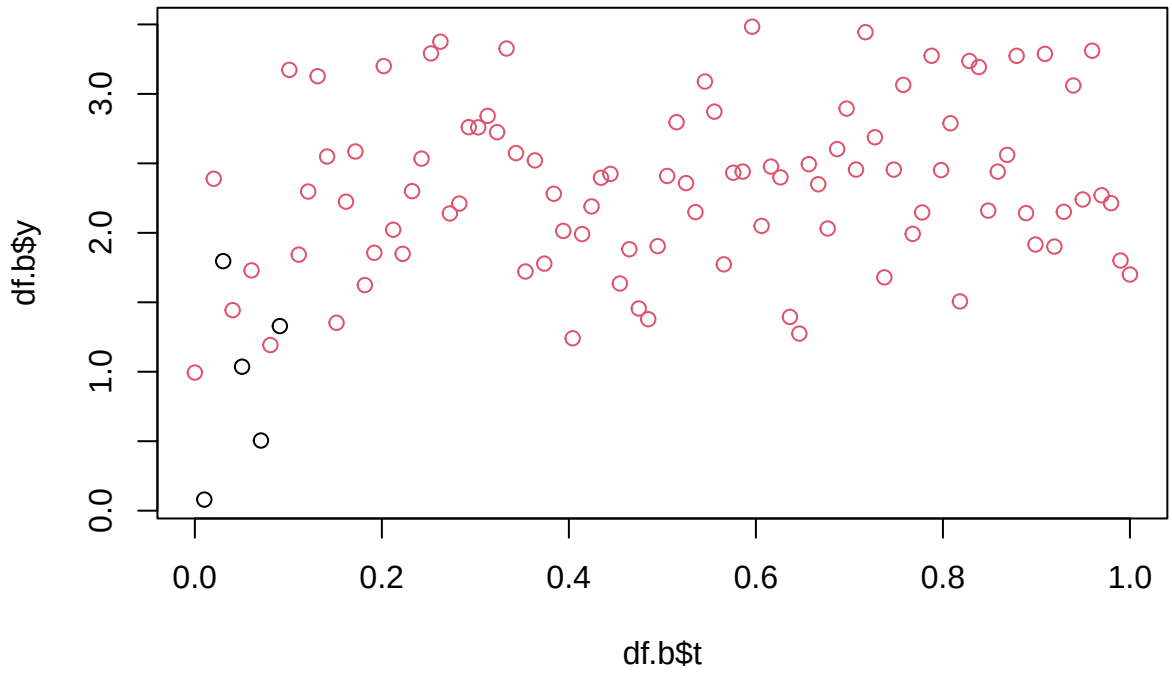
We have three methods of treatment assignment.

```
method.a <- function() {
    z <- rbinom(n, 1, 0.5)
```

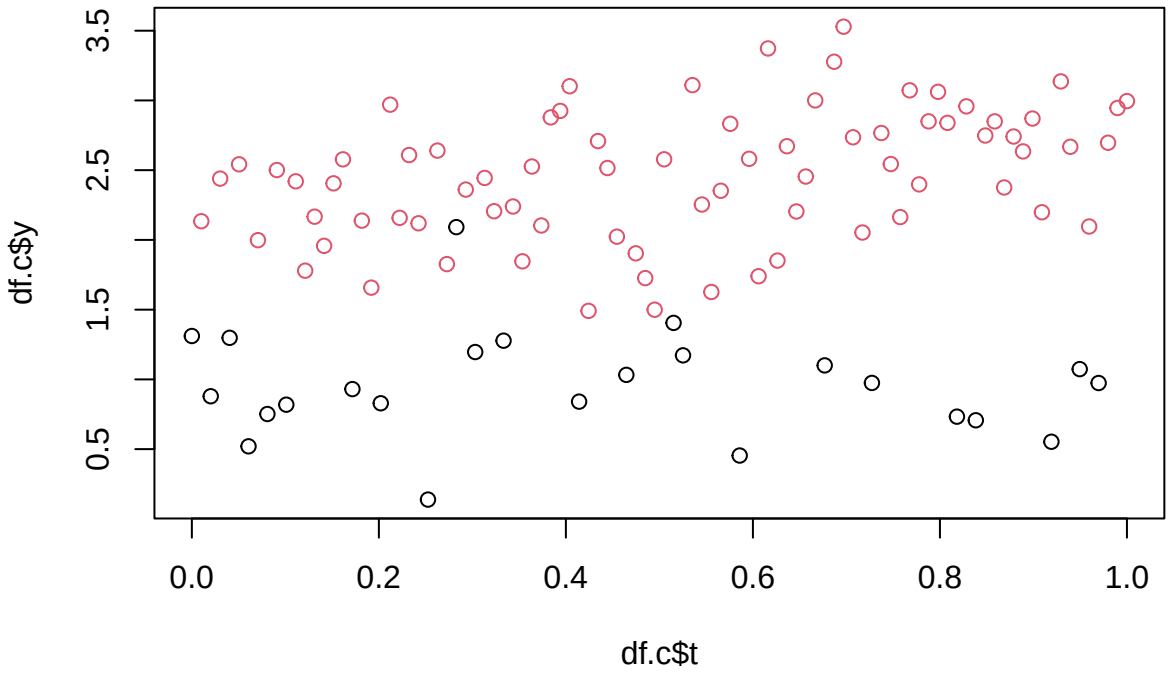
```
    y <- sapply(1:n, function(i) gendata(z[i], t[i]))
    data.frame(t = t, z = z, y = y)
}

df.a <- method.a()
plot(df.a$t, df.a$y, col = df.a$z + 1)
```




```
method.b <- function() {
    z <- rep(NA, n)
    y <- rep(NA, n)
    z[1:10] <- (rbinom(1, 1, 0.5) + rep(c(0, 1), 5)) %% 2
    y[1:10] <- sapply(1:10, function(i) gendata(z[i], t[i]))
    for (i in 11:100) {
        y0.mean <- mean(y[z == 0], na.rm = TRUE)
        y1.mean <- mean(y[z == 1], na.rm = TRUE)
        z[i] <- as.numeric(y1.mean > y0.mean)
        y[i] <- gendata(z[i], t[i])
    }
    data.frame(t = t, z = z, y = y)
}

df.b <- method.b()
plot(df.b$t, df.b$y, col = df.b$z + 1)
```



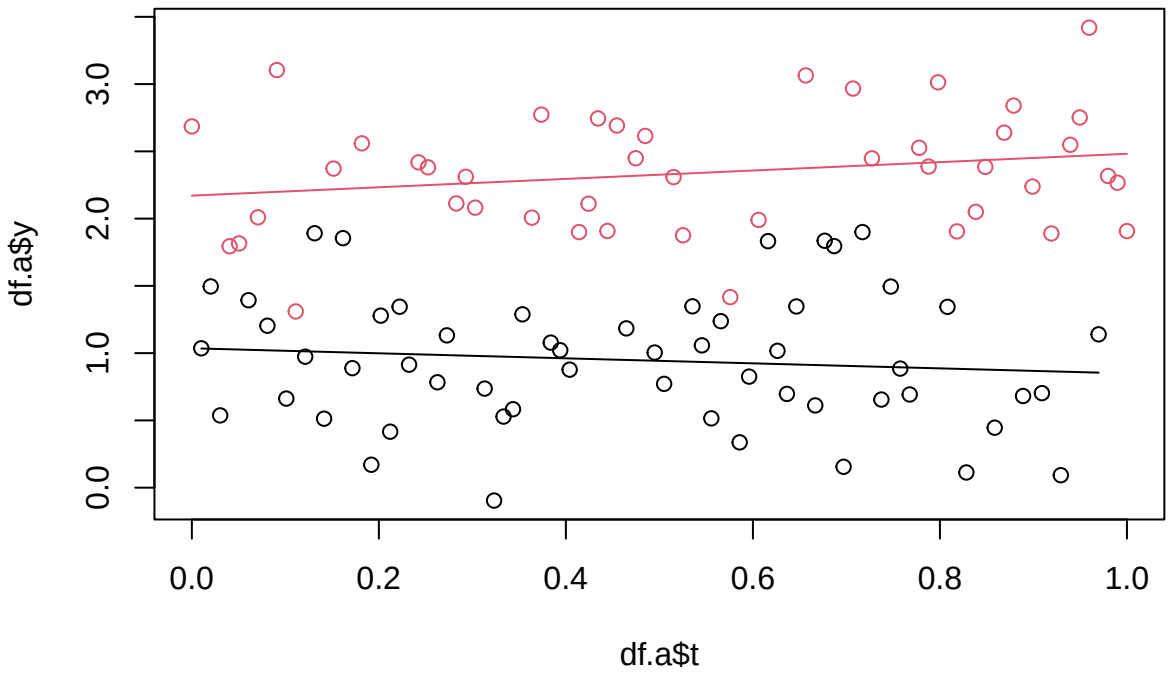

```
method.c <- function() {
    z <- rep(NA, n)
    y <- rep(NA, n)
    z[1:10] <- (rbinom(1, 1, 0.5) + rep(c(0, 1), 5)) %% 2
    y[1:10] <- sapply(1:10, function(i) gendata(z[i], t[i]))
    for (i in 11:100) {
        y0.mean <- mean(y[z == 0], na.rm = TRUE)
        y1.mean <- mean(y[z == 1], na.rm = TRUE)
        z[i] <- (rbinom(1, 1, 0.2) + as.numeric(y1.mean > y0.mean)) %% 2
        y[i] <- gendata(z[i], t[i])
    }
    data.frame(t = t, z = z, y = y)
}

df.c <- method.c()
plot(df.c$t, df.c$y, col = df.c$z + 1)
```



It seems that method (b) will nearly always severely under-represent the control group.

We can see a fit of the data generated by method (a) below, using an interaction term with time.

```
fit.a <- lm(y ~ z * t, data = df.a)
plot(df.a$t, df.a$y, col = df.a$z + 1)
y.a.pred <- predict(fit.a, df.a)
lines(df.a$t[df.a$z == 0], y.a.pred[df.a$z == 0], col = 1)
lines(df.a$t[df.a$z == 1], y.a.pred[df.a$z == 1], col = 2)
```




```
sqrt(mean(fit.a$residuals^2))
```

```
## [1] 0.4610949
```

We have two ways of fitting our data; ignoring the time variable t, and incorporating it in an interaction term.

```
fit.simple <- function(df) {
    fit <- lm(y ~ z, data = df)
    rmse <- sqrt(mean(fit$residuals^2))
    return(list(
        fit = fit,
        rmse = rmse
    ))
}

fit.interaction <- function(df) {
    fit <- lm(y ~ z * t, data = df)
    rmse <- sqrt(mean(fit$residuals^2))
    treatment.effect.mean <- mean(t * fit$coefficients["z:t"] + fit$coefficients["z"])
    return(list(
        fit = fit,
        rmse = rmse,
        treatment.effect.mean = treatment.effect.mean
    ))
}

fit.simple(df.a)
```

```
## $fit
##
## Call:
## lm(formula = y ~ z, data = df)
##
## Coefficients:
## (Intercept)            z
##      0.9496       1.3906
##
##
## $rmse
## [1] 0.4670886
```

```
fit.interaction(df.a)
```

```
## $fit
##
## Call:
## lm(formula = y ~ z * t, data = df)
##
## Coefficients:
## (Intercept)            z            t          z:t
##      1.0362       1.1345      -0.1868       0.4981
##
##
## $rmse
## [1] 0.4610949
##
## $treatment.effect.mean
## [1] 1.38357
```

```r
fit.simple(df.b)
```

```
## $fit
##
## Call:
## lm(formula = y ~ z, data = df)
##
## Coefficients:
## (Intercept)            z
##      0.9493       1.3843
##
##
## $rmse
## [1] 0.5911895
```

```r
fit.interaction(df.b)
```

```
## $fit
##
## Call:
## lm(formula = y ~ z * t, data = df)
##
## Coefficients:
## (Intercept)            z            t          z:t
##      0.6476       1.4929       5.9724      -5.6037
##
##
## $rmse
## [1] 0.5813291
##
## $treatment.effect.mean
## [1] -1.308996
```

```r
fit.simple(df.c)
```

```
## $fit
##
## Call:
## lm(formula = y ~ z, data = df)
##
## Coefficients:
## (Intercept)            z
##      0.9609       1.5046
##
##
## $rmse
## [1] 0.4412264
```

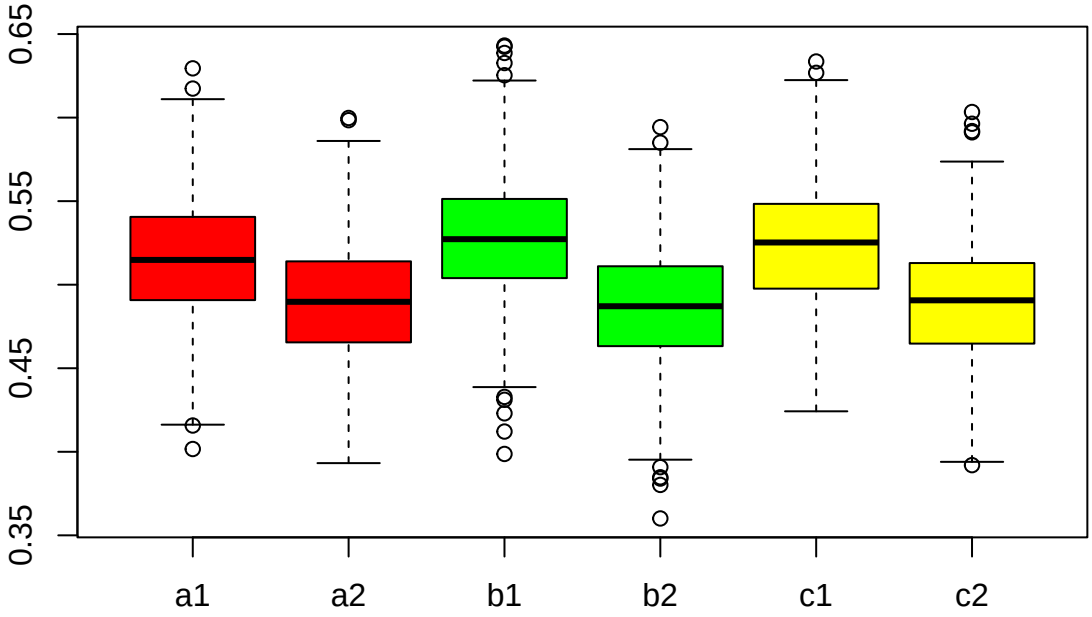
```r
fit.interaction(df.c)
```

```
## $fit
##
## Call:
## lm(formula = y ~ z * t, data = df)
##
## Coefficients:
```

```
## (Intercept)           z           t         z:t
##      1.0171      1.1121     -0.1317      0.7746
##
##
## $rmse
## [1] 0.41188
##
## $treatment.effect.mean
## [1] 1.499362
```

By running these for a 1000 times using each of the three methods, we can compare their RMSE's.

```
rmse <- function(method, reps = 1000) replicate(reps, {
    df <- method()
    f1 <- fit.simple(df)
    f2 <- fit.interaction(df)
    c(f1$rmse, f2$rmse)
})

rmse.all <- rbind(
    rmse(method.a),
    rmse(method.b),
    rmse(method.c)
)

boxplot(
    t(rmse.all),
    names = c("a1", "a2", "b1", "b2", "c1", "c2"),
    col = c("red", "red", "green", "green", "yellow", "yellow")
)
```



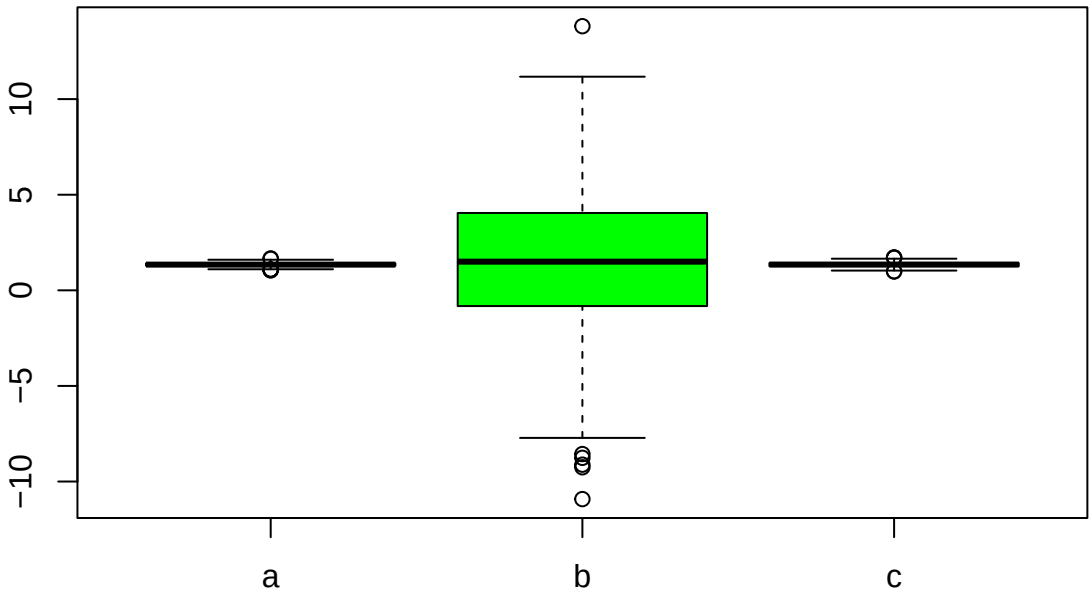
We can also compare the average treatment effect (averaged across 100 days) estimated by the second analysis method.

```
treatment.effect.mean <- function(method, reps = 1000) replicate(reps, {
    df <- method()
```

```
    f <- fit.interaction(df)
    f$treatment.effect.mean
})

tfm.all <- rbind(
    treatment.effect.mean(method.a),
    treatment.effect.mean(method.b),
    treatment.effect.mean(method.c)
)

rowMeans(tfm.all)
```

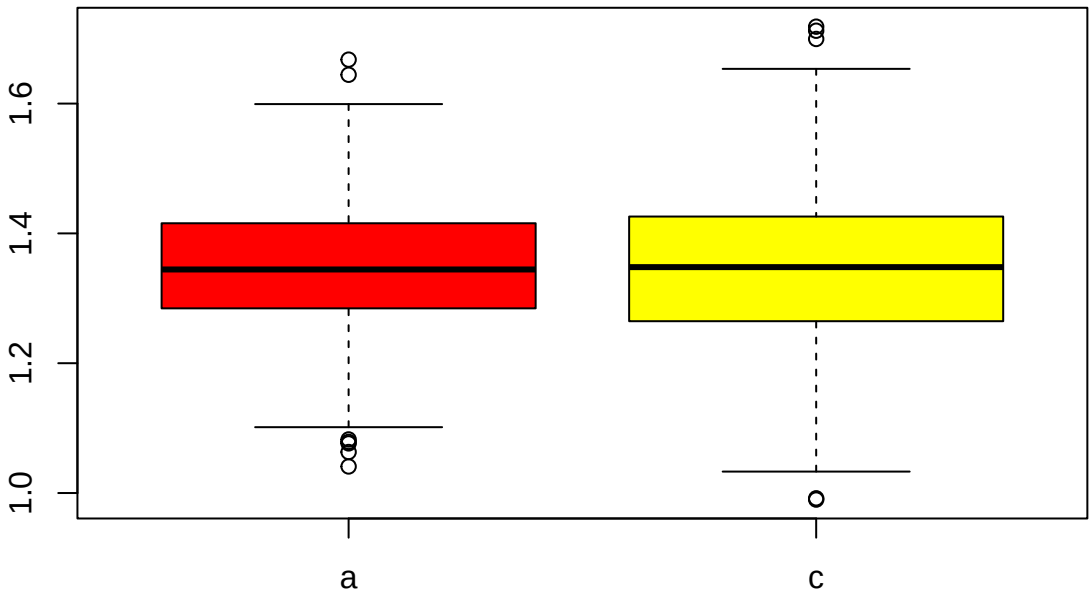
```
## [1] 1.348797 1.550780 1.346138
```

```
boxplot(
    t(tfm.all),
    names = c("a", "b", "c"),
    col = c("red", "green", "yellow")
)
```



Methods (a) and (c) estimate the average treatment effect (averaged across 100 days) much better than method (b); the true answer is $1.0 + 0.5 \times 0.7 = 1.35$. We can take a closer look at methods (a) and (c) below.

```
boxplot(
    t(tfm.all[c(1, 3), ]),
    names = c("a", "c"),
    col = c("red", "yellow")
)
```



It appears that method (a) has a *slightly* tighter spread than (c).