Tutorial 10: Filament model prediction comparison

3D printer

Revisit Tutorial 9 and download the 3D printer `filament1` data from Learn. We considered two linear models for these data, named A and B:

- Model A: $y_i = \theta_0 + \theta_1 x_{i,1} + \epsilon_i$
- Model B: $y_i = \theta_0 + \theta_1 x_{i,1} + \theta_2 x_{i,2} + \epsilon_i$

Recall that in this tutorial, predictions with models A and B and associated scores were obtained with

```
lm_prediction <- function(data, model, newdata) {
  if (model == "A") {
    fit0 <- lm(Actual_Weight ~ 1 + CAD_Weight, data = data)

  } else {
     fit0 <- lm(Actual_Weight ~ 1 + CAD_Weight + Material, data = data)
  }

  pred0 <- predict(fit0, newdata = newdata, se.fit = TRUE,
                   interval = "prediction", level = 0.95)

  mean =  pred0$fit[, "fit"]
  sd <- sqrt(pred0$se.fit^2 + summary(fit0)$sigma^2)

    q <- qt(1 - 0.05/2, df = Inf)
    lwr <- mean - q * sd
    upr <- mean + q * sd

  data.frame(mean = mean, sd = sd,
             lwr = lwr, upr = upr)
}

pred_A <- lm_prediction(data = filament1, model = "A", newdata = filament1)
pred_B <- lm_prediction(data = filament1, model = "B", newdata = filament1)

score_A <- cbind(pred_A, filament1) %>%
  mutate(
    se = mean((Actual_Weight - mean))^2,
    ds = (Actual_Weight - mean)^2/sd^2 + 2 * log(sd)
  )
score_B <- cbind(pred_B, filament1) %>%
  mutate(
    se = mean((Actual_Weight - mean)^2),
    ds = (Actual_Weight - mean)^2/sd^2 + 2 * log(sd)
  )
```

as well as with a 50% estimation/prediction data split.

Leave-one-out prediction

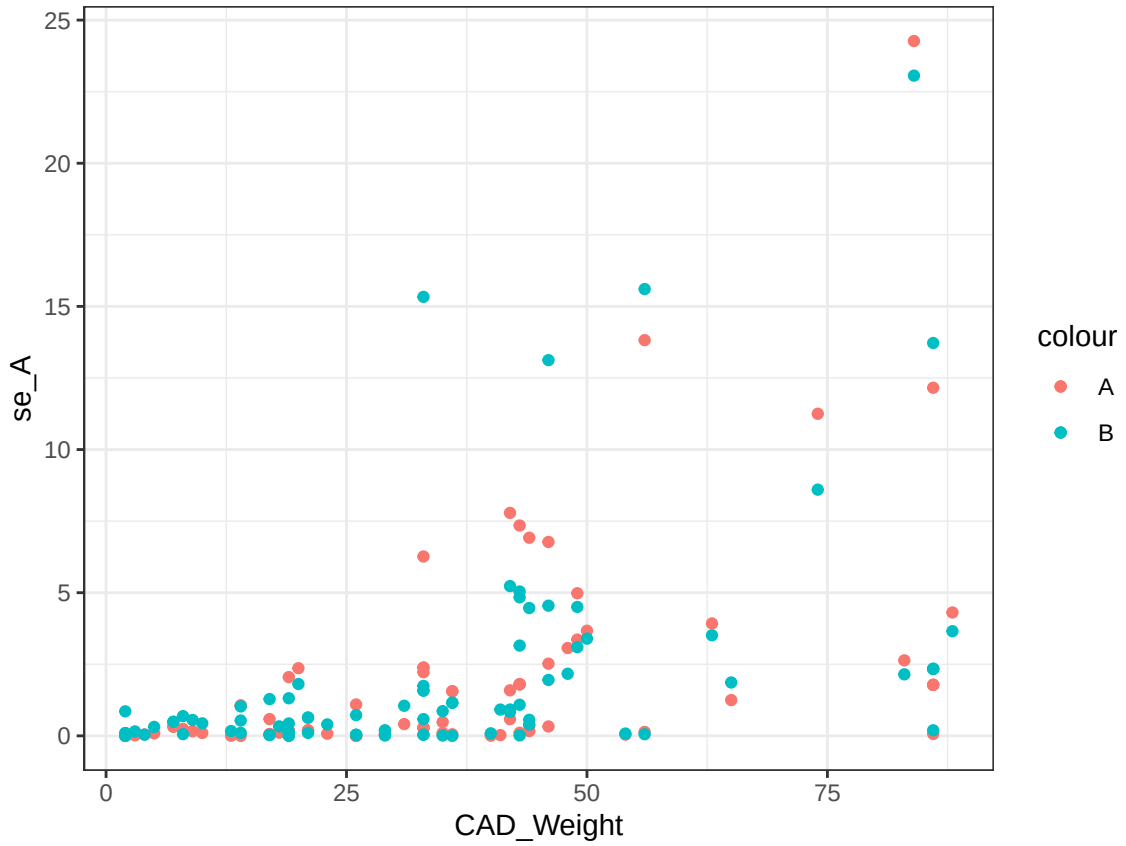
Write code to perform leave-one-out cross validation for both models and to compute the Squared Error and Dawid-Sebastiani scores.

Solution:

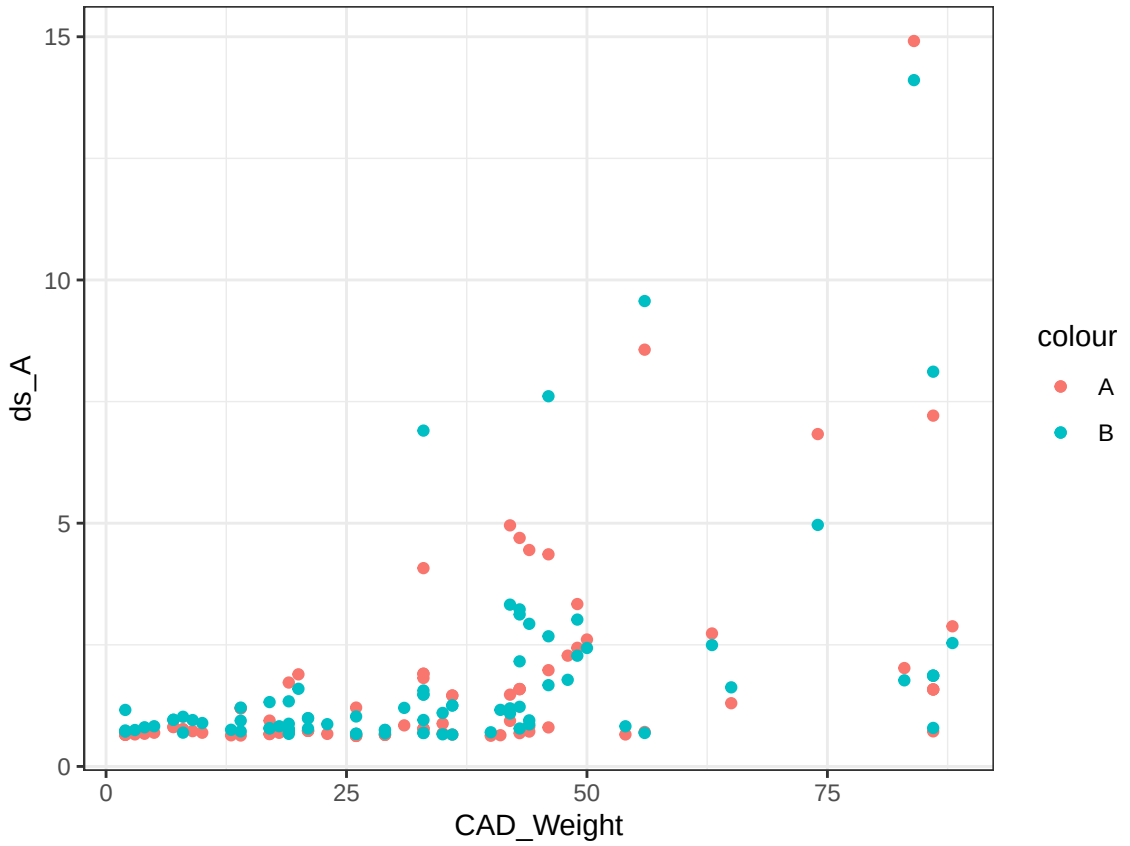
```
data <- filament1 %>% mutate(mean_A = NA_real_, sd_A = NA_real_,
                             mean_B = NA_real_, sd_B = NA_real_)

for (i in seq_len(nrow(filament1))) {
  fit_A <- lm(Actual_Weight ~ 1 + CAD_Weight, data = data[-i, , drop = FALSE])
  pred_A <- predict(fit_A, newdata = data[i,], se.fit = TRUE,
              interval = "prediction", level = 0.95)
  data[i, "mean_A"] =  pred_A$fit[, "fit"]
  data[i, "sd_A"] <- sqrt(pred_A$se.fit^2 + summary(fit_A)$sigma^2)

  fit_B <- lm(Actual_Weight ~ 1 + CAD_Weight + Material, data = data[-i, , drop = FALSE])
  pred_B <- predict(fit_B, newdata = data[i,], se.fit = TRUE,
              interval = "prediction", level = 0.95)
  data[i, "mean_B"] =  pred_B$fit[, "fit"]
  data[i, "sd_B"] <- sqrt(pred_B$se.fit^2 + summary(fit_B)$sigma^2)
}

data <- data %>%
  mutate(
    se_A = (Actual_Weight - mean_A)^2,
    ds_A = (Actual_Weight - mean_A)^2/sd_A^2 + 2 * log(sd_A),
    se_B = (Actual_Weight - mean_B)^2,
    ds_B = (Actual_Weight - mean_B)^2/sd_B^2 + 2 * log(sd_B)
  )
```

Using ggplot, plot each score versus the `CAD_weight` for both models and compare them.

```
ggplot() +
  geom_point(aes(CAD_Weight, se_A, colour = "A"), data = data) +
  geom_point(aes(CAD_Weight, se_B, colour = "B"), data = data)
```




```
ggplot() +
  geom_point(aes(CAD_Weight, ds_A, colour = "A"), data = data) +
  geom_point(aes(CAD_Weight, ds_B, colour = "B"), data = data)
```



Exchangeability test

Next, we want to investigate whether one model is better at predicting than the other. If they are equivalent, it should not matter, on average, if we randomly swap the A and B prediction scores within each leave-one-out score pair (S_i^A, S_i^B). Use the test statistic $\frac{1}{N}\sum_{i=1}^{N}(S_i^A - S_i^B)$, and make a Monte Carlo estimate of the p-value for a test of exchangeability between model predictions from A and from B against the alternative hypothesis that B is better than A. First compute the test statistic for the two prediction scores.

Hints: For this particular test statistic, one possible approach is to first compute the pairwise score differences and then generate randomisation samples by sampling random sign changes. Compute the test statistic for each randomly altered set of values and compare with the original test statistic. Start with $J = 1000$ iterations when testing your code. Increase to 10000 for more precise results.

Solution:

```
score_diff <- data.frame(se = data$se_A - data$se_B,
                         ds = data$ds_A - data$ds_B)
statistic0 <- score_diff %>% summarise(se = mean(se), ds = mean(ds))
J <- 10000
statistic <- data.frame(se = numeric(J),
                        ds = numeric(J))
for (loop in seq_len(J)) {
  random_sign <- sample(c(-1, 1), size = nrow(score_diff), replace = TRUE)
  statistic[loop, ] <- score_diff %>% summarise(se = mean(random_sign * se),
                                                ds = mean(random_sign * ds))
}
p_values <-
  statistic %>%
```

```
  summarise(se = mean(se > statistic0$se),
            ds = mean(ds > statistic0$ds))
# Estimates:
p_values
```

```
##       se     ds
## 1 0.8649 0.8445
```

We see that for both the Square Error and Dawid-Sebastiani scores, the two model predictions appear exchangeable.