Prediction Scores

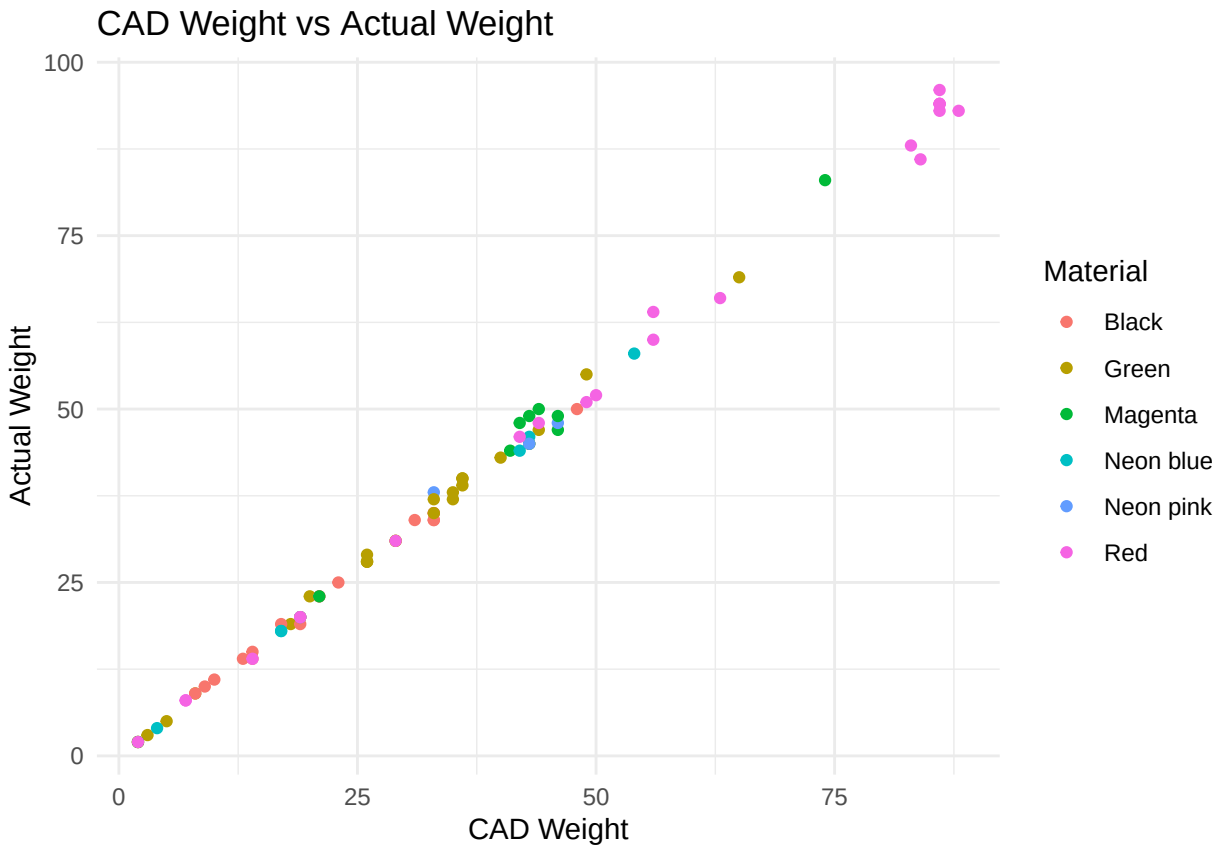
Qinqing Li

2024-09-07

```
filament1
```

```
## # A tibble: 86 x 5
##    Index Date       Material  CAD_Weight Actual_Weight
##    <dbl> <date>     <chr>          <dbl>         <dbl>
##  1     1 2017-12-29 Neon pink         33            38
##  2     2 2017-12-29 Green             40            43
##  3     3 2017-12-30 Green             20            23
##  4     4 2017-12-30 Green             44            47
##  5     5 2017-12-30 Green             18            19
##  6     6 2017-12-30 Green              3             3
##  7     7 2017-12-31 Green              5             5
##  8     8 2018-01-01 Green             35            37
##  9     9 2018-01-02 Green             19            20
## 10    10 2018-01-03 Green              8             9
## # ... with 76 more rows
```

```
ggplot(filament1, aes(x = CAD_Weight, y = Actual_Weight, colour = Material)) +
  geom_point() +
  labs(title = "CAD Weight vs Actual Weight",
       x = "CAD Weight",
       y = "Actual Weight") +
  theme_minimal()  # Optional: a clean theme
```



Estimate

Construct model A and B based on CAD_Weight, to estimate Actual_Weight:

```
#Note model can only take A or B
est_A <-filament1_estimate(filament1, "A")
est_B <-filament1_estimate(filament1, "B")
```

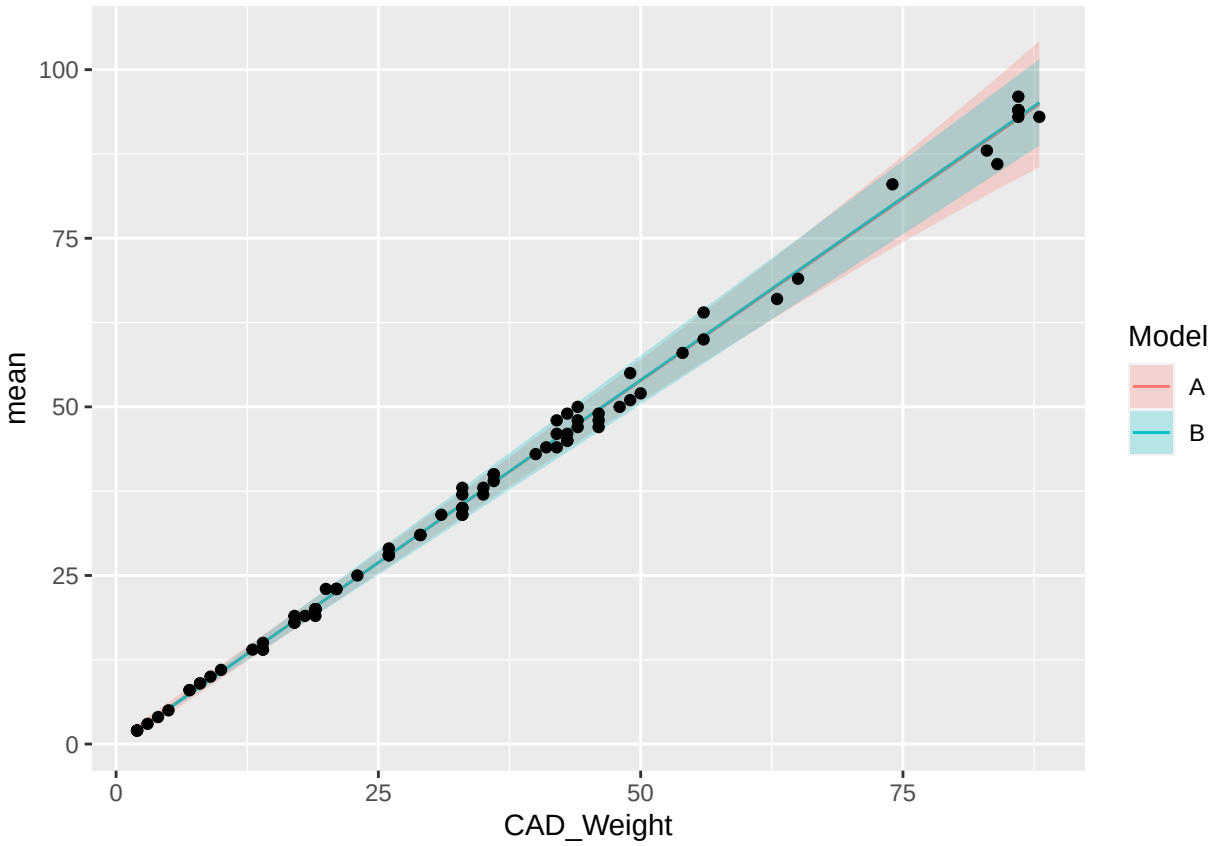
Predict

Predicted Actual_Weight using models A and B: (since it's predicted, slightly different than real data, I would like to see obs. lie inside the interval (lwr, upr) for good predictions)

```
pred_A<- filament1_predict(est_A, filament1)
pred_B<- filament1_predict(est_B, filament1)
```

Draw graph of data (points) and ribbon of prediction:

```
#First, combine all 4 datasets into 1 dataset: rbind: row bind, cbind: col bindl.
ggplot(rbind(cbind(pred_A, filament1, Model = "A"),
             cbind(pred_B, filament1, Model = "B")),
       mapping = aes(CAD_Weight)) +
  geom_line(aes(y = mean, col = Model)) +
  geom_ribbon(aes(ymin = lwr, ymax = upr, fill = Model), alpha = 0.25) +
  geom_point(aes(y = Actual_Weight), data = filament1)
```



Prediction Scores

A scoring rule provides evaluation metrics for predicted prob. distributions.

Proper scoring rules

A scoring rule S (a loss function) is proper relative to F (a predicted prob. distribution) if (assuming negative orientation) its expected score is minimised when the forecasted prob. distribution matches the distribution of the observation. It is strictly proper if it holds with equality if and only if F = Q. Q is distr. of observations.

```
score_A <-cbind(filament1,pred_A) %>%
  mutate(
    se = proper_score("se", Actual_Weight, mean = mean),
    ds = proper_score("ds", Actual_Weight, mean = mean, sd = sd),
    interval = proper_score("interval", Actual_Weight, lwr = lwr, upr = upr, alpha = 0.1)
)

score_B <-cbind(filament1,pred_B) %>%
  mutate(
    se = proper_score("se", Actual_Weight, mean = mean),
    ds = proper_score("ds", Actual_Weight, mean = mean, sd = sd),
    interval = proper_score("interval", Actual_Weight, lwr = lwr, upr = upr, alpha = 0.1)
)
```

```
# na.rm = TRUE means ignore missing values NA
average_A <- score_A %>%
  summarise(
```

```
    avg_se = mean(se, na.rm = TRUE),
    avg_ds = mean(ds, na.rm = TRUE),
    avg_interval = mean(interval, na.rm = TRUE)
  )

average_B <- score_B %>%
  summarise(
    avg_se = mean(se, na.rm = TRUE),
    avg_ds = mean(ds, na.rm = TRUE),
    avg_interval = mean(interval, na.rm = TRUE)
  )

averages_df <- rbind(
  cbind(Model = "A", average_A),
  cbind(Model = "B", average_B)
)

averages_df %>%
  kable(col.names = c("Model", "SE", "DS", "Interval"))
```

Model	SE	DS	Interval
A	1.783616	1.0428110	5.471989
B	1.790706	0.8874053	5.108026

Do the scores indicate that one of the models is better or worse than the other? Do the three score types agree with each other?

ans: our scores are negative orientated. On average model B has smaller DS and interval scores, so model B perform better than A. The squared error score (SE) doesn't really care about the difference between the two models, since it doesn't directly involve the variance model (the paramter estimates for the mean are different, but not by much).

Data splitting

Split the data into two parts, with 50% to be used for parameter estimation, and 50% to be used for prediction assessment.

Redo the previous analysis for the problem using this division of the data:

```
#training model A and B using data_estimation
est_A1 <-filament1_estimate(data_estimation, "A")
est_B1 <-filament1_estimate(data_estimation, "B")
#After training (the parameters), the models are applied to the new dataset filament1_pred. This functi
pred_A1<- filament1_predict(est_A1, data_prediction)
pred_B1<- filament1_predict(est_B1, data_prediction)


score_A1 <-cbind(data_prediction,pred_A1) %>%
  mutate(
    se = proper_score("se", Actual_Weight, mean = mean),
    ds = proper_score("ds", Actual_Weight, mean = mean, sd = sd),
    interval = proper_score("interval", Actual_Weight, lwr = lwr, upr = upr, alpha = 0.1)
)
```

```r
score_B1 <-cbind(data_prediction,pred_B1) %>%
  mutate(
    se = proper_score("se", Actual_Weight, mean = mean),
    ds = proper_score("ds", Actual_Weight, mean = mean, sd = sd),
    interval = proper_score("interval", Actual_Weight, lwr = lwr, upr = upr, alpha = 0.1)
)

average_A1 <- score_A1 %>%
  summarise(
    avg_se = mean(se, na.rm = TRUE),
    avg_ds = mean(ds, na.rm = TRUE),
    avg_interval = mean(interval, na.rm = TRUE)
  )

average_B1 <- score_B1 %>%
  summarise(
    avg_se = mean(se, na.rm = TRUE),
    avg_ds = mean(ds, na.rm = TRUE),
    avg_interval = mean(interval, na.rm = TRUE)
  )

averages_df1 <- rbind(
  cbind(Model = "A", average_A1),
  cbind(Model = "B", average_B1)
)

averages_df1 %>%
  kable(col.names = c("Model", "SE", "DS", "Interval"))
```

Model	SE	DS	Interval
A	2.143120	1.360013	6.328483
B	2.113181	1.053308	5.464981

Model B still performs better with lower DS and interval scores, and SE scores are similar, which agrees with the previous results.