DataMiningAndWrangling_FA4_Khafaji

Bone Mineral Density

In this exercise, we will be looking at a data set on spinal bone mineral density, a physiological indicator that increases during puberty when a child grows.

In this dataset, idnum is an identifier for each child and spnbmd represents the relative change in spinal bone mineral density between consecutive doctor's visits.

The goal is to learn about the typical trends of growth in bone mineral density during puberty for boys and girls.

Importing Data

first, let's import the dataset for bone mineral density

```
bmd_raw <- read_excel("bmd-data.xlsx")
bmd_raw
```

```
## # A tibble: 169 x 9
##    idnum   age sex   fracture weight_kg height_cm medication waiting_time spnbmd
##    <dbl> <dbl> <chr> <chr>        <dbl>     <dbl> <chr>             <dbl>  <dbl>
##  1   469  57.1 F     no frac~        64      156. Anticonvu~           18  0.879
##  2  8724  75.7 F     no frac~        78      162  No medica~           56  0.795
##  3  6736  70.8 M     no frac~        73      170. No medica~           10  0.907
##  4 24180  78.2 F     no frac~        60      148  No medica~           14  0.711
##  5 17072  54.2 M     no frac~        55      161  No medica~           20  0.791
##  6  3806  77.2 M     no frac~        65      168  No medica~            7  0.730
##  7 17106  56.2 M     no frac~        77      159  No medica~           26  1.01
##  8 23834  49.9 F     no frac~        59      150  No medica~            9  0.731
##  9  2454  68.4 M     no frac~        64      167  Glucocort~            6  0.689
## 10  2088  66.3 M     no frac~        72      160. No medica~           10  0.947
## # i 159 more rows
```

Tidy Data

At first glance, the data given looks tidy. All we really have to do is run through a few checks so that we are sure that there is data integrity:

```
bmd <- bmd_raw %>% dplyr::filter((age >=0) & (waiting_time >=0 ) ) %>% drop_na() %>%
  mutate(sex =case_when(
    sex == "F" ~ "Female",
    sex == "M" ~ "Male"
  )) %>%
  rename( spn_bmd_percentChange = spnbmd)
bmd
```

```
## # A tibble: 169 x 9
##    idnum   age sex    fracture    weight_kg height_cm medication    waiting_time
##    <dbl> <dbl> <chr>  <chr>           <dbl>     <dbl> <chr>                <dbl>
```

```
##  1   469  57.1 Female no fracture        64      156. Anticonvulsa~           18
##  2  8724  75.7 Female no fracture        78      162  No medication           56
##  3  6736  70.8 Male   no fracture        73      170. No medication           10
##  4 24180  78.2 Female no fracture        60      148  No medication           14
##  5 17072  54.2 Male   no fracture        55      161  No medication           20
##  6  3806  77.2 Male   no fracture        65      168  No medication            7
##  7 17106  56.2 Male   no fracture        77      159  No medication           26
##  8 23834  49.9 Female no fracture        59      150  No medication            9
##  9  2454  68.4 Male   no fracture        64      167  Glucocortico~            6
## 10  2088  66.3 Male   no fracture        72      160. No medication           10
## # i 159 more rows
## # i 1 more variable: spn_bmd_percentChange <dbl>
```

Explore

Let's explore the data more:

first, let's find out how many children are in the data set, how many children per gender, and their respective median ages.

```
bind_rows(
  bmd %>% dplyr::filter(age<=20) %>% group_by(sex) %>%
  summarise(Count = n(),
            Median_Age = median(age)
            ),
  bmd %>% dplyr::filter(age<=20) %>%
    summarise(
      sex = "Overall",
      Count = n(),
      Median_Age = median(age)
  )

)
```

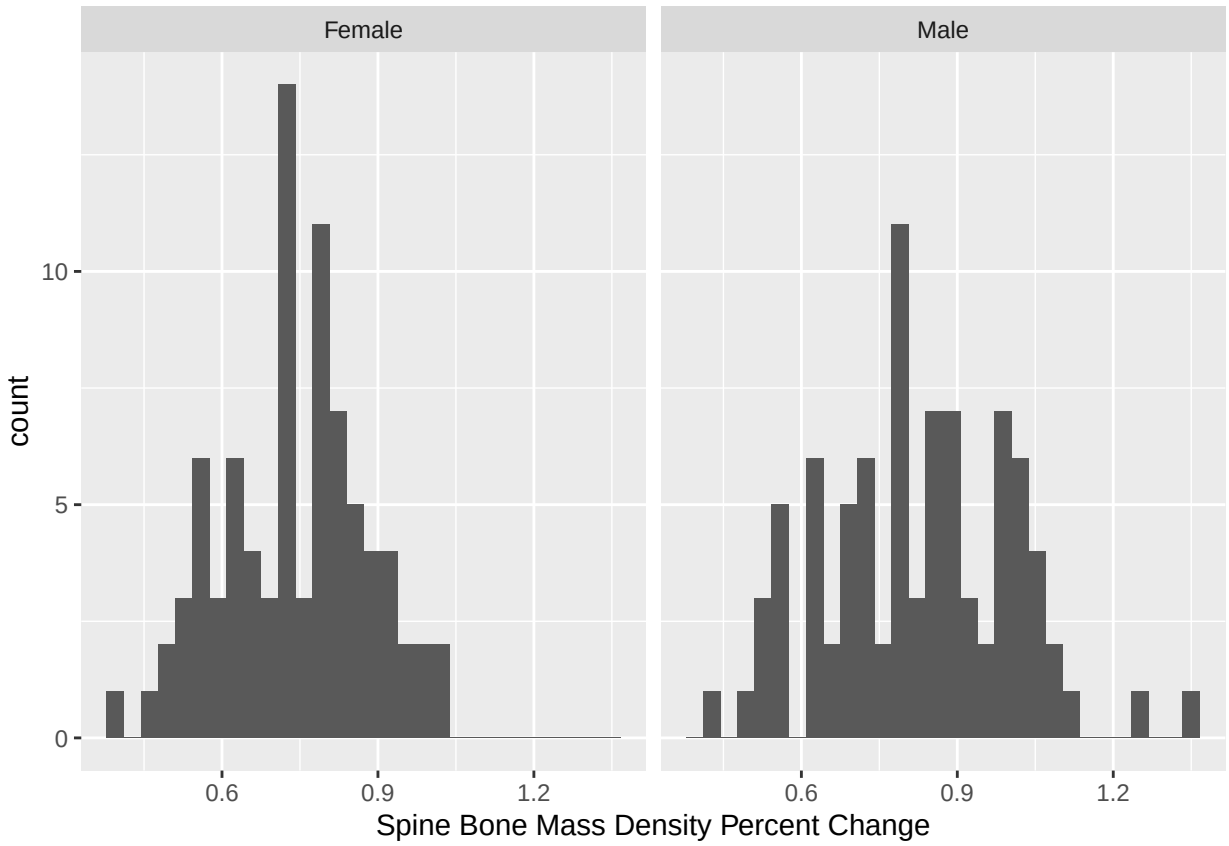
```
## # A tibble: 1 x 3
##   sex     Count Median_Age
##   <chr>   <int>      <dbl>
## 1 Overall     0         NA
```

There are no children in the datasets. The official definition of children is that of ages below 18. Here, we adjusted our filter to include the age of 20, but to no avail. Given this, we are unable to give a median age of children.

Now, we want to produce plots to compare the distributions of spnbmd and age between boys and girls (display these as two plots side by side, one for spnbmd and one for age). Are there apparent differences in either spnbmd or age between these two groups?

```
bmd %>% ggplot(aes(x=spn_bmd_percentChange)) +
  geom_histogram() +
  facet_wrap(~sex) +
  xlab("Spine Bone Mass Density Percent Change")
```

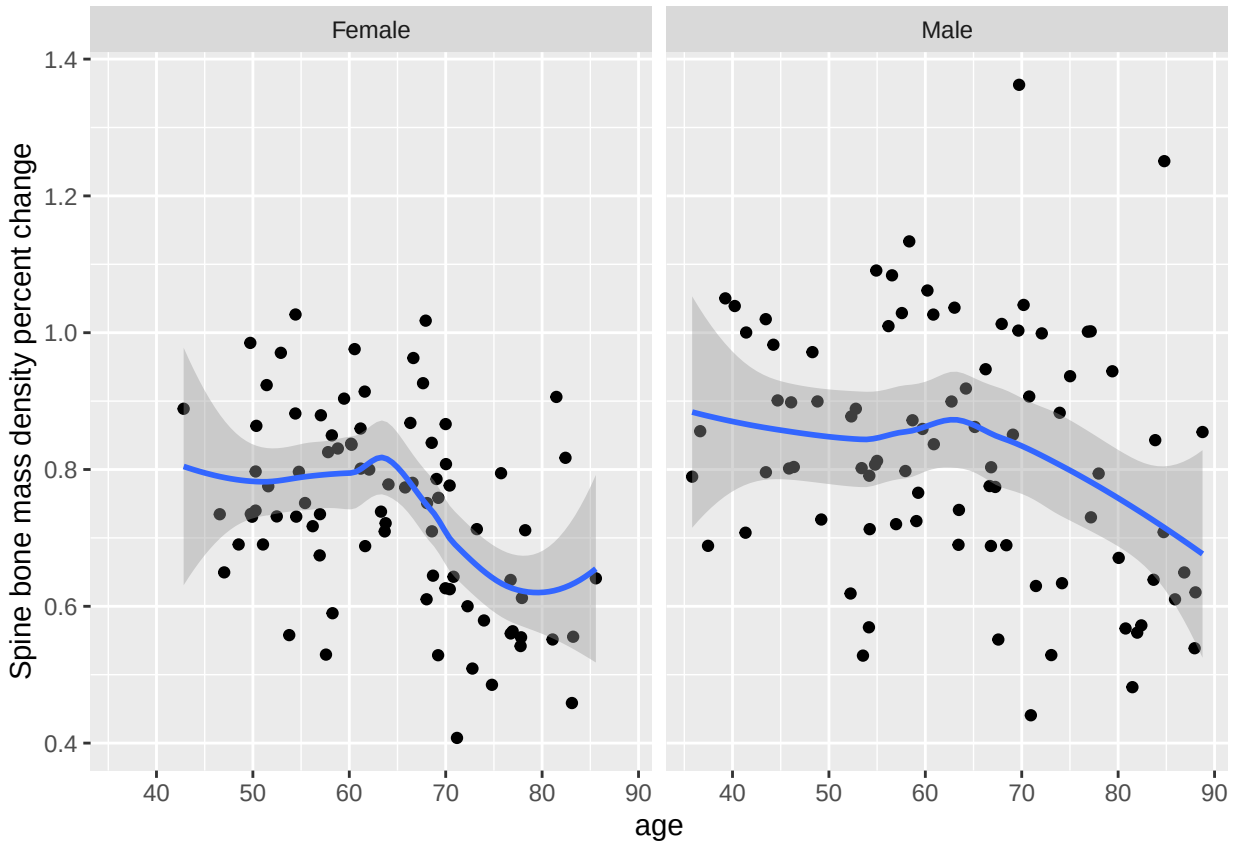
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can see that the relative change in spine bone mass density is less spread out, and is skewed to the right, more so than the spine bone mass density for males. This shows that the change in bone mass density for males has more variation. This shows that more males get higher bone mass density than the average in their group, than number of females getting higher bone density than the average in their group.

Create a scatter plot of spnbmd (y axis) versus age (x axis), faceting by gender. What trends do you see in this data?

```
bmd %>% ggplot(aes(x=age, y=spn_bmd_percentChange)) +
  geom_point()+
  facet_wrap(~sex)+
  stat_smooth(method = "loess",
              formula = y ~ x,
              geom = "smooth")+
  ylab("Spine bone mass density percent change")
```



As we can see from the scatter plot, age the change in spinal bone mass density seems to peak at ages around 60 to 65, the change then decreases past that range.

Model

Since there are some trends in the data, we can create a model to look at the data better.

Split

```
set.seed(5) # seed set for reproducibility (DO NOT CHANGE)
n <- nrow(bmd)
train_samples <- sample(1:n, round(0.8*n))

bmd_train <- bmd[train_samples, ]
bmd_train
```

```
## # A tibble: 135 x 9
##    idnum   age sex    fracture    weight_kg height_cm medication    waiting_time
##    <dbl> <dbl> <chr>  <chr>           <dbl>     <dbl> <chr>                <dbl>
##  1  8854  47.0 Female no fracture        47      148. Glucocortico~           22
##  2    77  77.9 Female no fracture        50      152  No medication           24
##  3   690  81.5 Male   fracture           54      154  Glucocortico~           12
##  4    55  73.2 Female no fracture        52      153  No medication           89
##  5 22168  73.9 Male   no fracture        75      164. Glucocortico~            8
##  6 24190  76.7 Female fracture           73      156. No medication           11
##  7  2458  82.4 Male   fracture           48      158  No medication           21
##  8  6726  62.7 Male   no fracture        78      161  No medication            8
```

```
##  9 17116  48.3 Male   no fracture        68      160  No medication           14
## 10    35  56.2 Female no fracture        68      150. No medication           18
## # i 125 more rows
## # i 1 more variable: spn_bmd_percentChange <dbl>
```

```
bmd_test <- bmd[-train_samples, ]
bmd_test
```

```
## # A tibble: 34 x 9
##    idnum   age sex    fracture    weight_kg height_cm medication    waiting_time
##    <dbl> <dbl> <chr>  <chr>           <dbl>     <dbl> <chr>                <dbl>
##  1   469  57.1 Female no fracture        64      156. Anticonvulsa~           18
##  2  2088  66.3 Male   no fracture        72      160. No medication           10
##  3  3047  64.2 Male   no fracture        90      175  Glucocortico~           28
##  4  5288  40.2 Male   no fracture        66      165  No medication            8
##  5   109  69.0 Female no fracture        72      154  No medication           11
##  6  4205  46.1 Male   no fracture        73      171  No medication           37
##  7   159  65.8 Female no fracture        74      145  No medication           14
##  8  6981  48.8 Male   no fracture        96      169  No medication           33
##  9  8704  49.8 Female no fracture        46      150  No medication           17
## 10   812  67.9 Male   no fracture        80      170  No medication           13
## # i 24 more rows
## # i 1 more variable: spn_bmd_percentChange <dbl>
```

Tune

Since, as we saw, the distribution for males and females are different, let's separate them.

```
bmd_train_female <- bmd_train %>% dplyr::filter(sex=="Female")
bmd_test_female <- bmd_test %>% dplyr::filter(sex=="Female")

bmd_train_male <- bmd_train %>% dplyr::filter(sex=="Male")
bmd_test_male <- bmd_test %>% dplyr::filter(sex=="Male")
```

Using cross_validate_spline from the stat471 R package, perform 10-fold cross-validation on bmd_train_male and bmd_train_female, trying degrees of freedom 1,2,. . . ,15. Display the two resulting CV plots side by side.

```
K <- 10

n_male <- nrow(bmd_train_male)
folds_male <- sample(rep(1:K, length.out = n_male))
bmd_train_male <- bmd_train_male %>%
  mutate(fold = folds_male)

n_female <- nrow(bmd_train_female)
folds_female <- sample(rep(1:K, length.out = n_female))
bmd_train_female <- bmd_train_female %>%
mutate(fold = folds_female)
```

Let's first do cross validation for males:

```
df_values <- 1:15

# Initialize an empty tibble to store results
cv_results_tibble_male <- tibble(df = integer(), cv_mean = numeric(), cv_se = numeric())
```

```r
for (df_val in df_values){
  # create a vector of out-of-fold predictions
  out_of_fold_predictions <- numeric(n_male)

  # iterate over folds
  for (current_fold in 1:K) {
    # out-of-fold data will be used for training
    out_of_fold_data <- bmd_train_male %>% dplyr::filter(fold != current_fold)

    # in-fold data will be used for validation
    in_fold_data <- bmd_train_male %>% dplyr::filter(fold == current_fold)
    out_of_fold_data
    in_fold_data

    # train on out-of-fold data
    spline_fit <-
      lm(spn_bmd_percentChange ~ ns(age, df = df_val),
         data = out_of_fold_data)

    # predict on in-fold data
    out_of_fold_predictions[folds_male == current_fold] <-
    predict(spline_fit, newdata = in_fold_data)
  }

  # add the out-of-fold predictions to the data frame
  results <- bmd_train_male %>%
  mutate(yhat = out_of_fold_predictions, y = spn_bmd_percentChange)

  # compute the CV estimate and standard error
  cv_results <- results %>%
  group_by(fold) %>%
  summarise(cv_fold = mean((yhat - y)^2)) %>% # CV estimates per fold
  summarise(
  cv_mean = mean(cv_fold),
  cv_se = sd(cv_fold) / sqrt(K)
  )

  cv_results_tibble_male <- bind_rows(
    cv_results_tibble_male, tibble(df = df_val, cv_mean = cv_results$cv_mean, cv_se = cv_results$cv_se))
}

cv_results_tibble_male <- cv_results_tibble_male %>% mutate(sex = "Male")
```

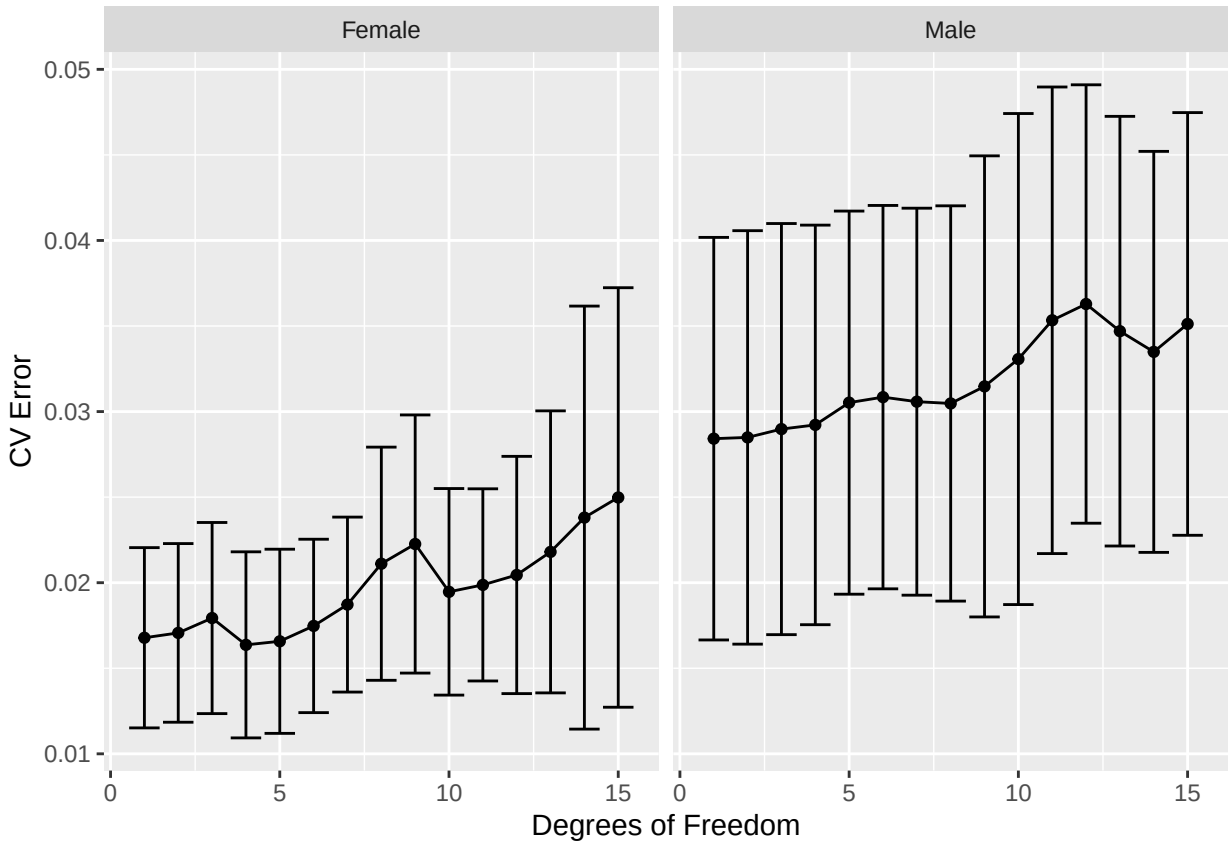
doing the same cross validation for females.

```r
df_values <- 1:15

# Initialize an empty tibble to store results
cv_results_tibble_female <- tibble(df = integer(), cv_mean = numeric(), cv_se = numeric())

for (df_val in df_values){
  # create a vector of out-of-fold predictions
  out_of_fold_predictions <- numeric(n_female)
```

```
  # iterate over folds
  for (current_fold in 1:K) {
    # out-of-fold data will be used for training
    out_of_fold_data <- bmd_train_female %>% dplyr::filter(fold != current_fold)

    # in-fold data will be used for validation
    in_fold_data <- bmd_train_female %>% dplyr::filter(fold == current_fold)
    out_of_fold_data
    in_fold_data

    # train on out-of-fold data
    spline_fit <-
      lm(spn_bmd_percentChange ~ ns(age, df = df_val),
         data = out_of_fold_data)

    # predict on in-fold data
    out_of_fold_predictions[folds_female == current_fold] <-
    predict(spline_fit, newdata = in_fold_data)
  }

  # add the out-of-fold predictions to the data frame
  results <- bmd_train_female %>%
  mutate(yhat = out_of_fold_predictions, y = spn_bmd_percentChange)

  # compute the CV estimate and standard error
  cv_results <- results %>%
  group_by(fold) %>%
  summarise(cv_fold = mean((yhat - y)^2)) %>% # CV estimates per fold
  summarise(
  cv_mean = mean(cv_fold),
  cv_se = sd(cv_fold) / sqrt(K)
  )

  cv_results_tibble_female <- bind_rows(
    cv_results_tibble_female, tibble(df = df_val, cv_mean = cv_results$cv_mean, cv_se = cv_results$cv_s
}

cv_results_tibble_female <- cv_results_tibble_female %>% mutate(sex = "Female")
```

Let's merge them together, and then create a side by side plot.

```
bmd_train_cv_merged <- bind_rows(cv_results_tibble_male, cv_results_tibble_female)
bmd_train_cv_merged %>% ggplot(aes(x=df, y=cv_mean))+
  geom_line()+
  geom_point()+
  geom_errorbar(
    aes(x=df,
        ymin = cv_mean - 1.96*cv_se,
        ymax = cv_mean + 1.96*cv_se)
  )+
  facet_wrap(~sex) +
  labs(x="Degrees of Freedom", y="CV Error")
```



For females, a degree of freedom of 4 provided the least CV error. For males, a degree of freedom of 1 yielded the best results.

It is worth noting that the CV error for the females, as well as the values captured within their 95% confidence range, are much smaller compared to those of the males.

For the females, the one standard error rule encompasses all models with degrees of freedom from 1 to 12 (min = 0.12, max=0.2), while the one standard error rule for the males encompasses all degrees of freedom from 1 to 15 (min= 0.2, max=0.04).

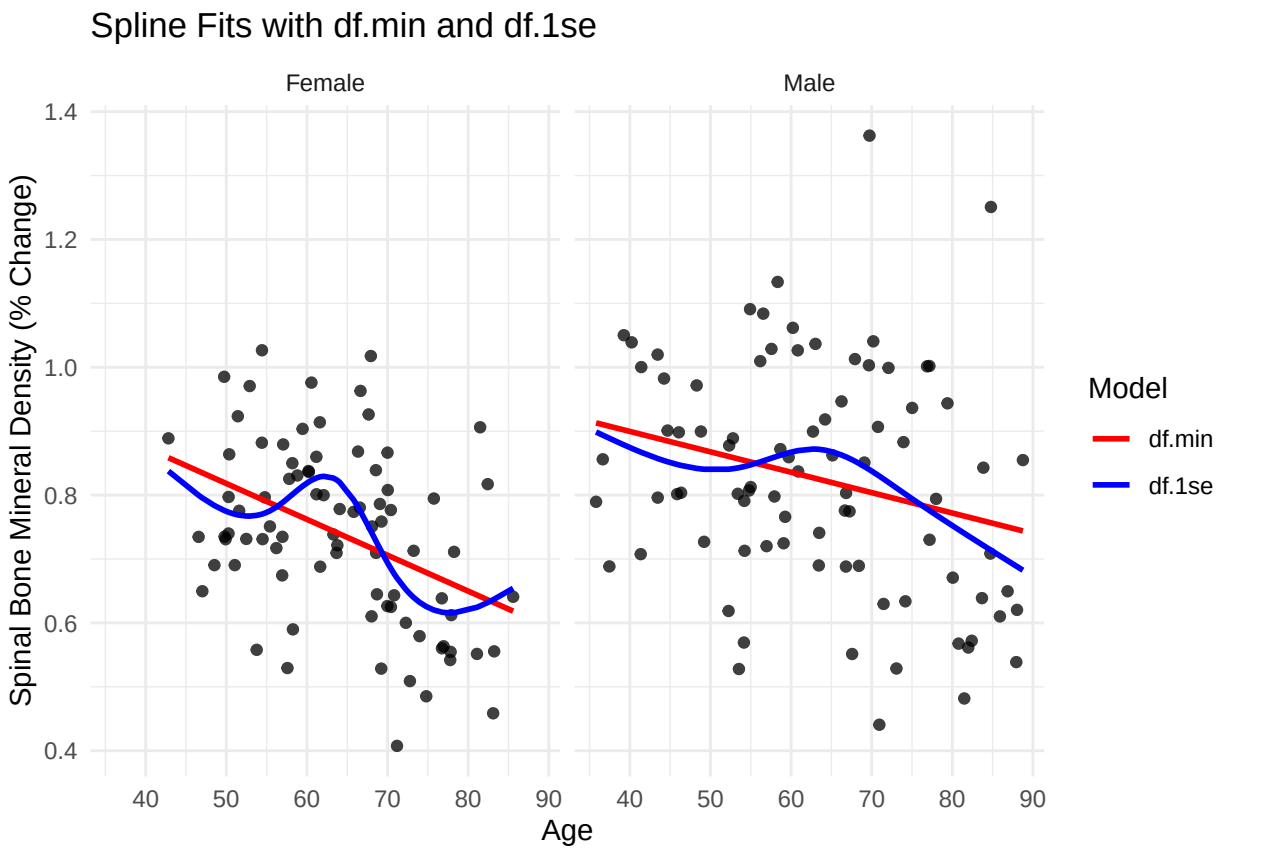
```
cv_summary <- bmd_train_cv_merged %>%
  group_by(sex) %>%
  summarize(
    df_min = df[which.min(cv_mean)],
    df_1se = df[which(cv_mean <= min(cv_mean) + cv_se[which.min(cv_mean)])][1]
  ) %>% ungroup()

df.min <- cv_summary %>% dplyr::filter(df_min == max(df_min)) %>% slice(1) %>% select(df_min)

df.min <- as.numeric(as.character(unlist(df.min)))


df.1se <- cv_summary %>% dplyr::filter(df_1se == max(df_1se)) %>% slice(1) %>% select(df_1se)

df.1se <- as.numeric(as.character(unlist(df.1se)))
```

```r
cat("df.min = ", df.min, ", df.1se= ", df.1se)
```

```
## df.min =  4 , df.1se=  1
```

```r
fit_male_df.min <- lm(spn_bmd_percentChange ~ ns(age, df = df.min), data = bmd %>% dplyr::filter(sex ==
fit_female_df.min <- lm(spn_bmd_percentChange ~ ns(age, df = df.min), data = bmd %>% dplyr::filter(sex =

fit_male_df.1se <- lm(spn_bmd_percentChange ~ ns(age, df = df.1se), data = bmd %>% dplyr::filter(sex ==
fit_female_df.1se <- lm(spn_bmd_percentChange ~ ns(age, df = df.1se), data = bmd %>% dplyr::filter(sex =


bmd <- bmd %>%
  mutate(
    pred_df.min = NA,  # Initialize the column with NA
    pred_df.1se = NA   # Initialize the column with NA
  )

bmd <- bmd %>%
  group_by(sex) %>%
  mutate(
    pred_df.min = if_else(
      sex == "Male",
      predict(fit_male_df.min, newdata = pick(everything())),
      predict(fit_female_df.min, newdata = pick(everything()))
    ),
    pred_df.1se = if_else(
      sex == "Male",
      predict(fit_male_df.1se, newdata = pick(everything())),
      predict(fit_female_df.1se, newdata = pick(everything()))
    )
  ) %>%
  ungroup()

bmd_withpred <- bmd %>%
  pivot_longer(
    cols = starts_with("pred_"),
    names_to = "df",
    values_to = "prediction"
  )

bmd_withpred %>% ggplot(aes(x = age, y = spn_bmd_percentChange)) +
  geom_point(alpha = 0.5) +  # Scatter plot of the original data
  geom_line(aes(y = prediction, color = df), linewidth = 1) +  # Lines for predictions
  facet_wrap(~ sex) +  # Facet by sex
  labs(
    title = "Spline Fits with df.min and df.1se",
    x = "Age",
    y = "Spinal Bone Mineral Density (% Change)",
    color = "Model"
  ) +
  scale_color_manual(
    values = c("pred_df.min" = "blue", "pred_df.1se" = "red"),  # Custom colors
    labels = c("df.min", "df.1se")  # Custom legend labels
  ) +
```

```
  theme_minimal()
```



For me, it seems like the spline using df.1se captures the trends better that the spline trained on df.min

Final Fit

Let's use df.min to train a spline for our training datasets.

```
fit_male_df.min <- lm(spn_bmd_percentChange ~ ns(age, df = df.min), data = bmd_train_male)
fit_male_df.1se <- lm(spn_bmd_percentChange ~ ns(age, df = df.1se), data = bmd_train_male)

bmd_train_male <- bmd_train_male %>%
  mutate(
    pred_df.min = predict(fit_male_df.min, newdata = bmd_train_male),
    pred_df.1se = predict(fit_male_df.1se, newdata = bmd_train_male)
  )
bmd_test_male <- bmd_test_male %>%
  mutate(
    pred_df.min = predict(fit_female_df.min, newdata = bmd_test_male),
    pred_df.1se = predict(fit_female_df.1se, newdata = bmd_test_male)
  )

fit_female_df.min <- lm(spn_bmd_percentChange ~ ns(age, df = df.min), data = bmd_train_female)
fit_female_df.1se <- lm(spn_bmd_percentChange ~ ns(age, df = df.1se), data = bmd_train_female)
bmd_train_female <- bmd_train_female %>%
  mutate(
    pred_df.min = predict(fit_female_df.min, newdata = bmd_train_female),
    pred_df.1se = predict(fit_female_df.1se, newdata = bmd_train_female)
```

```
  )
bmd_test_female <- bmd_test_female %>%
  mutate(
    pred_df.min = predict(fit_female_df.min, newdata = bmd_test_female),
    pred_df.1se = predict(fit_female_df.1se, newdata = bmd_test_female)
  )
```

Evaluate

```
# Calculate RMSE for males
# for df.min
train_rmse_male_df.min <- rmse(bmd_train_male$spn_bmd_percentChange, bmd_train_male$pred_df.min)
test_rmse_male_df.min <- rmse(bmd_test_male$spn_bmd_percentChange, bmd_test_male$pred_df.min)

# for df.1se
train_rmse_male_df.1se <- rmse(bmd_train_male$spn_bmd_percentChange, bmd_train_male$pred_df.1se)
test_rmse_male_df.1se <- rmse(bmd_test_male$spn_bmd_percentChange, bmd_test_male$pred_df.1se)


# Calculate training RMSE for females
# for df.min
train_rmse_female_df.min <- rmse(bmd_train_female$spn_bmd_percentChange, bmd_train_female$pred_df.min)
test_rmse_female_df.min <- rmse(bmd_test_female$spn_bmd_percentChange, bmd_test_female$pred_df.min)

# for df.1se
train_rmse_female_df.1se <- rmse(bmd_train_female$spn_bmd_percentChange, bmd_train_female$pred_df.1se)
test_rmse_female_df.1se <- rmse(bmd_test_female$spn_bmd_percentChange, bmd_test_female$pred_df.1se)
```

```
rmse_tibble <- tibble(sex = c("Male", "Male", "Female", "Female"),
       df_model = c("df.min", "df.1se", "df.min", "df.1se"),
       train_rmse = c(train_rmse_male_df.min, train_rmse_male_df.1se, train_rmse_female_df.min, train_rm
       test_rmse = c(test_rmse_male_df.min, test_rmse_male_df.1se, test_rmse_female_df.min, test_rmse_fe
       )

rmse_tibble
```

```
## # A tibble: 4 x 4
##   sex    df_model train_rmse test_rmse
##   <chr>  <chr>         <dbl>     <dbl>
## 1 Male   df.min        0.160     0.241
## 2 Male   df.1se        0.162     0.247
## 3 Female df.min        0.115     0.128
## 4 Female df.1se        0.123     0.133
```

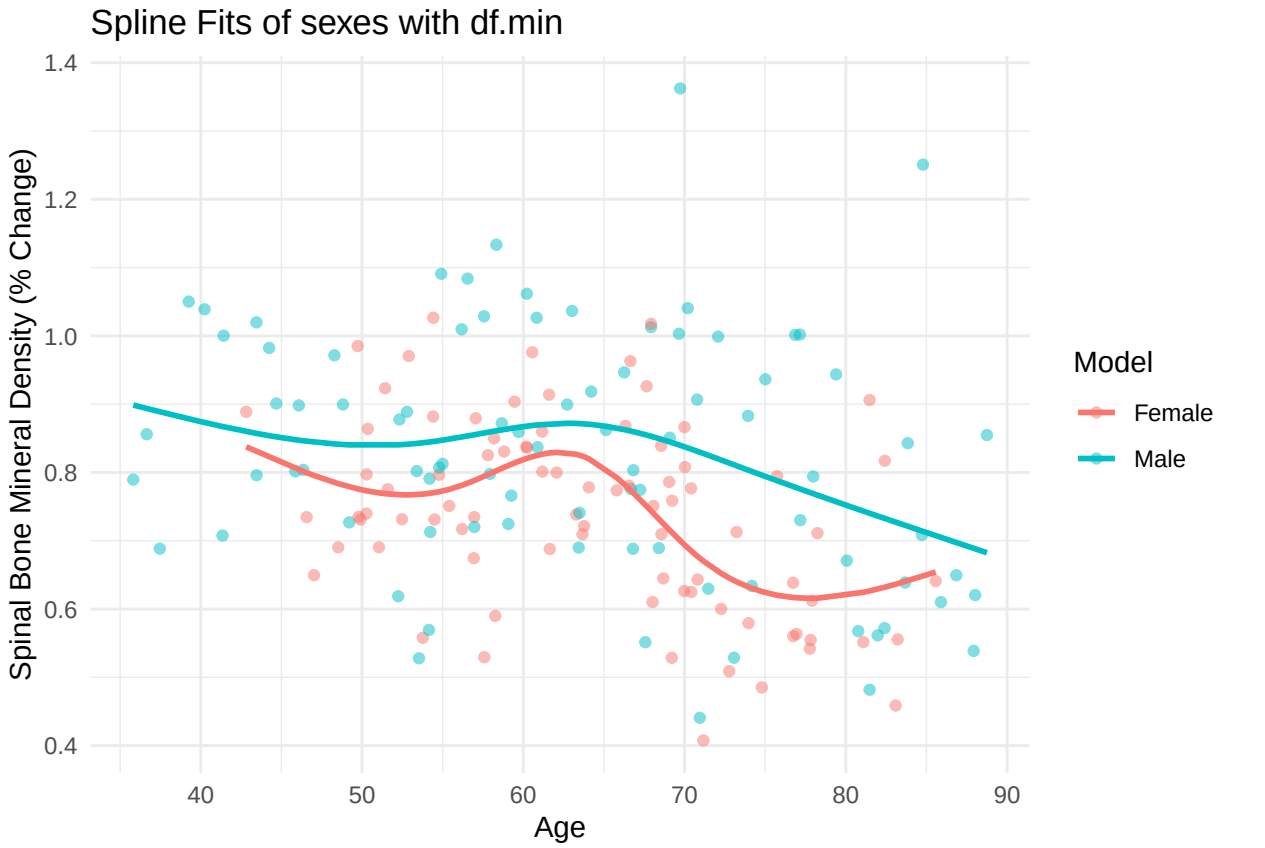
Given the size of our data set the training RMSE and the test RMSE obtained are both within acceptable levels. There is not much overfitting that has occured, given that the training RMSE and the test RMSE of both sexes are close to each other.

This also shows that the models that used df.min were the better model, although not by much.

Interpret

```
bmd %>% ggplot(aes(x = age, y = spn_bmd_percentChange, color = sex)) +
  geom_point(alpha = 0.5) +  # Scatter plot of the original data
```

```
geom_line(aes(y = pred_df.min), linewidth = 1) +  # Lines for predictions
labs(
  title = "Spline Fits of sexes with df.min",
  x = "Age",
  y = "Spinal Bone Mineral Density (% Change)",
  color = "Model"
) +
theme_minimal()
```



At what ages (approximately) do boys and girls reach the peaks of their growth spurts? The data does not include individuals with age less than 35. However, the model spline seems to trend that the change of spinal bone mass density is greater in the earlier ages.

At what ages does growth largely level off for boys and girls? Do these seem in the right ballpark? We can see from the spline that, between the ages of 60 and 65, there is a spike in relative change in spinal bone mass density. If we are to assume that all values in the column are in terms of absolute value, we could see this as a change in the negative direction, indicating weakening bones. Else, this shows that the spinal bone mass density actually increased at this range, and its change decreased from 65 to 75.

KNN and Bias-Variance tradeoff

You own a square apple orchard, measuring 200 meters on each side.

You have planted trees in a grid ten meters apart from each other. Last apple season, you measured the yield of each tree in your orchard (in average apples per week). You noticed that the yield of the different trees seems to be higher in some places of the orchard and lower in others, perhaps due to differences in sunlight and soil fertility across the orchard.

Unbeknownst to you, the yield Y of the tree planted E1 meters to the right and E2 meters up from the bottom left-hand corner of the orchard has distribution Y = f (E) + ϵ, where

$$f(E) = 50 + 0.001E_1^2 + 0.001E_2^2, \epsilon \sim N(0, \sigma^2), \sigma = 4$$

A simple rule to predict this season's yield (15 points)

This apple season is right around the corner, and you'd like to predict the yield of each tree. You come up with perhaps the simplest possible prediction rule: predict this year's yield for any given tree based on last year's yield from that same tree.

1. What is the training error of such a rule?

Solving for the training error gives:

$$\text{training error} = \frac{1}{N}\Sigma_{i=1}^{N}(Y^{\text{training}} - \hat{f}(E_i^{\text{training}}))$$

We can write $Y^{training}$ and $\hat{f}(E_i^{\text{training}})$ as:

$$\text{training error} = \frac{1}{N}\Sigma_{i=1}^{N}((f(E) + \epsilon) - f(E)$$

Due to them both having the same E_1 and E_2 values, differing only by some epsilon following the normal distribution.

$$\Rightarrow \frac{1}{N}\Sigma_{i=1}^{N}\epsilon = \frac{N}{N}\epsilon = \epsilon$$

Therefore, the training error for the rule is ϵ.

2. What is the mean squared bias, mean variance, and expected test error of this prediction rule?

From the previous answer, our expected test error is:

$$\text{ETE} = \frac{1}{N}\Sigma_{i=1}^{N}E[(f(E) + \epsilon) - f(E)]$$

$$\Rightarrow \frac{1}{N}\Sigma_{i=1}^{N}E[\epsilon] = \frac{1}{N}\Sigma_{i=1}^{N}(\sigma^2) = \frac{N}{N}(\sigma^2) = 16$$

We also know that Bias is:

$$\text{bias} = E[\hat{f}(E_i^{\text{test}})] - f(E_i^{\text{test}})$$

We also know that the expected value of $\hat{f}(E_i^{\text{test}})$ is $f(E_i^{\text{test}})$, i.e.

$$E[\hat{f}(E_i^{\text{test}})] = f(E_i^{\text{test}})$$

hence,

$$\text{bias} = f(E_i^{\text{test}}) - f(E_i^{\text{test}}) = 0$$

for the mean variance, since all predictions and observed values have the same distribution, apart from ϵ, which has a variance of 16.

$$\text{Mean Variance} = Var(Y) = 16$$

K-nearest neighbors regression (conceptual) (15 points)

As a second attempt to predict a yield for each tree, you average together last year's yields of the K trees closest to it (including itself, and breaking ties randomly if necessary). So if you choose K = 1, you get back the simple rule from the previous section. This more general rule is called K-nearest neighbors (KNN) regression (see ISLR p. 105).

KNN is not a parametric model like linear or logistic regression, so it is a little harder to pin down its degrees of freedom.

1. What happens to the model complexity as K increases? Why? The model gets more simple, as it considers more points around it and avoids overfitting around 1 data point
2. The degrees of freedom for KNN is sometimes considered n/K, where n is the training set size. Why might this be the case? [Hint: consider a situation where the data are clumped in groups of K.]

- Given n data points in a dataset, the data points are separated into K folds. Each fold's misclassification error is then averaged.

this means that the dataset is separated into K groups. Then, the amount of variables being considered for each model is n/K

3. Conceptually, why might increasing K tend to improve the prediction rule? What does this have to do with the bias-variance tradeoff?

- Increasing K increases the training set for each fold. This then decreases the variance for the model, allowing it to capture the overall trend.

4. Conceptually, why might increasing K tend to worsen the prediction rule? What does this have to do with the bias-variance tradeoff?

- However, Increasing K also increases the bias for the model, as the predicted value now takes in influence from other points, it will be harder for it to be precise and accurate for the actual value.

K-nearest neighbors regression (simulation) (25 points)

Now, we try KNN for several values of K. For each value of K, we use a numerical simulation to compute the bias and variance for every tree in the orchard. These results are contained in training_results_summary below.

```
training_results_summary <- readRDS("training_results_summary.rds")
training_results_summary
```

```
## # A tibble: 6,174 x 5
##        K    X1    X2    bias variance
##    <dbl> <dbl> <dbl>   <dbl>    <dbl>
##  1     1     0     0 -0.25       16.2
##  2     1     0    10  0.14       12.2
##  3     1     0    20 -0.523      20.4
##  4     1     0    30  0.109      15.6
##  5     1     0    40 -0.566      21.4
##  6     1     0    50 -0.336      15.9
##  7     1     0    60 -1.04       12.4
##  8     1     0    70 -0.0213     12.4
```

```
##  9     1     0    80  0.045      18.3
## 10     1     0    90 -0.312      14.7
## # i 6,164 more rows
```

1. Create a new tibble called overall_results the contains the mean squared bias, mean variance, and expected test error for each value of K. This tibble should have four columns: K, mean_sq_bias, mean_variance, and expected_test_error.

```
overall_results <- training_results_summary %>% group_by(K) %>%
  summarise(
    mean_sq_bias = mean(bias^2),
    mean_variance = mean(variance),
    expected_test_error = mean(bias^2) + mean(variance)
  )

overall_results
```

```
## # A tibble: 10 x 4
##        K mean_sq_bias mean_variance expected_test_error
##    <dbl>        <dbl>         <dbl>               <dbl>
##  1     1        0.280          17.5                17.8
##  2     2        0.256          17.4                17.6
##  3     3        0.227          17.7                17.9
##  4     4        0.264          17.7                18.0
##  5     5        0.255          17.8                18.0
##  6     6        0.252          17.3                17.6
##  7     7        0.244          17.7                18.0
##  8     8        0.232          17.6                17.8
##  9     9        0.232          17.6                17.9
## 10    10        0.240          17.6                17.9
```

2. Using overall_results, plot the mean squared bias, mean variance, and expected test error on the same axes as a function of K. Based on this plot, what is the optimal value of K?

```
overall_results %>% ggplot(aes(x=K)) +
  geom_point(aes(y=mean_sq_bias, color="mean sq bias"), size=1)+
  geom_point(aes(y=mean_variance, color="mean variance"), size=1)+
  geom_point(aes(y=expected_test_error, color="expected test error"), size=1)+
  labs(y="", x="K", title="Mean Square Bias, Mean Variance, and Expected Test Error, vs. K", color="Met
  scale_color_manual(values = c("mean sq bias" = "red", "mean variance" = "blue", "expected test error"
```

Basing on the expected test error, the best values for K is either 2 or 6.

3. We are used to the bias decreasing and the variance increasing when going from left to right in the plot. Here, the trend seems to be reversed. Why is this the case?

- Because in KNN, the smaller the K, the more complex the model, and vice versa. Reducing the complexity of the model decreases variance, while increasing bias. Since the model becomes less complex from left to right, i.e. K is increasing, we see the variance decreasing and the bias increasing.

4. The mean squared bias has a strange bump between K = 1 and K = 5, increasing from K = 1 to K = 2 but then decreasing from K = 2 to K = 5. Why does this bump occur? [Hint: Think about the rectangular grid configuration of the trees. So for a given tree, the closest tree is itself, and then the next closest four trees are the ones that are one tree up, down, left, and right from it.]

- The data actually does not fit this question. the mean squared bias decreases from K=1 to K=3, then increases at K=4 before decreasing again at K=5. But I would credit the decrease in bias in the earlier K values due to more data points being able to interpolate the position of the data point better.

5. Based on the information in training_results_summary, which tree and which value of K gives the overall highest absolute bias? Does the sign of the bias make sense? Why do this particular tree and this particular value of K give us the largest absolute bias?

```
training_results_summary %>% dplyr::filter(abs(bias) == max(abs(bias)))
```

```
## # A tibble: 1 x 5
##       K    X1    X2  bias variance
##   <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1     1     0    70 -2.06     13.9
```

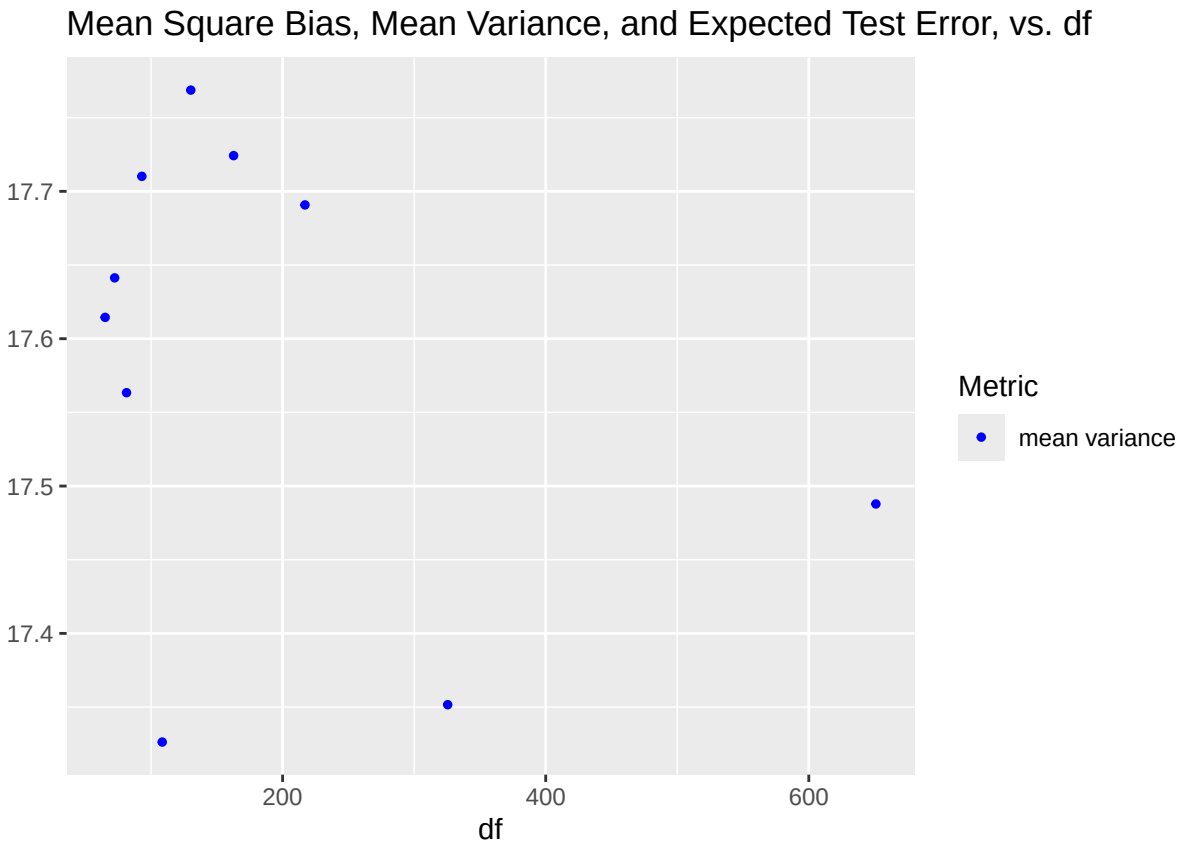
```
#
```

- K = 1, and (X1,X2)=(0,70) gives an absolute bias of 2.059342, the highest in the dataset.

Given by the value of the bias, the model seems to be underfitting, which is definitely odd as K=1 is supposed to have the opposite problem. However, if the datapoints close to X2 = 70 have higher weights, this might help explain the underfit in the region.

6. Redo the bias-variance plot from part 2, this time putting df = n/K on the x-axis. What do we notice about the variance as a function of df?

```
overall_results %>% ggplot(aes(x=nrow(training_results_summary%>%dplyr::filter(K==1))/K)) +
  #geom_point(aes(y=mean_sq_bias, color="mean sq bias"), size=1)+
  geom_point(aes(y=mean_variance, color="mean variance"), size=1)+
  #geom_point(aes(y=expected_test_error, color="expected test error"), size=1)+
  labs(y="", x="df", title="Mean Square Bias, Mean Variance, and Expected Test Error, vs. df", color="M
  scale_color_manual(values = c("mean sq bias" = "red", "mean variance" = "blue", "expected test error"
```



- A lower df usually indicated a lower mean variance than those with a df higher than it, except for a few outliers in the right. The lowest mean variance recorded has a df of around 150.

7. Derive a formula for the KNN mean variance. [Hint: First, write down an expression for the KNN prediction for a given tree. Then, compute the variance of this quantity using the fact that the variance of the average of N independent random variables each with variance s2 is s2/N . Finally, compute the mean variance by averaging over trees.]

$$\hat{p}(X^{test}) = \frac{1}{K}\Sigma_{i \in N_K} I(Y^{train} = 1)$$

$$VAR(\hat{p}(X^{test})) = \frac{\sigma^2}{k}$$

suppose that there are B trees. then:

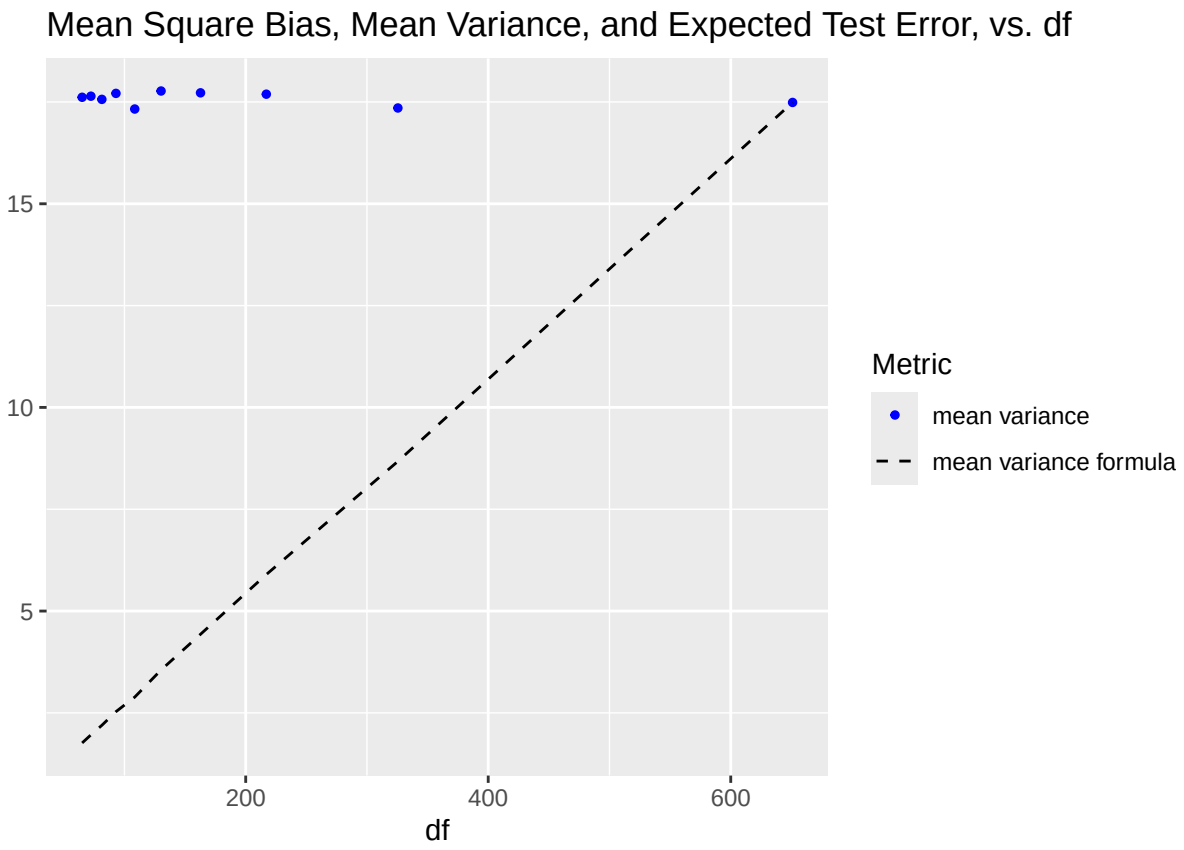
$$F(X) = \frac{1}{B}\Sigma_{b=1}^{B} p(X_b^{test})$$

then:

$$VAR(F(X)) = \frac{\sigma^2}{kB}$$

8. Create a plot like that in part 6, but with the mean variance formula from part 7 superimposed as a dashed curve. Do these two variance curves match?

```
overall_results %>% ggplot(aes(x=nrow(training_results_summary%>%dplyr::filter(K==1))/K)) +
  #geom_point(aes(y=mean_sq_bias, color="mean sq bias"), size=1)+
  geom_point(aes(y=mean_variance, color="mean variance"), size=1)+
  geom_line(aes(y=(mean_variance/K), color="mean variance formula"),  linetype="dashed")+
  #geom_point(aes(y=expected_test_error, color="expected test error"), size=1)+
  labs(y="", x="df", title="Mean Square Bias, Mean Variance, and Expected Test Error, vs. df", color="M
  scale_color_manual(values = c("mean sq bias" = "red", "mean variance" = "blue", "expected test error"
```



no it does not capture the curve