Homework 5: ANOVA and Intervals

Ruonan Zhao

October 03, 2024

Acknowledgements: This homework was adopted from material by Dr. Mario Giacomazzo.

Instructions:

The purpose of this homework assignment is to look more into understanding the ANOVA table and how intervals enhance our predictions. Follow the instructions closely, ensuring that your code produces the exact output requested. Avoid modifying data or plots unless instructed. Don't sort the data unless you are told to sort the data. Pay attention to the comments and make sure to remove any unnecessary # signs before adding your code. If a code block contains #DO NOT CHANGE, follow the instructions accordingly. You should include all code used (do not hide code cells). **You should knit your RMD file to a PDF after you answer every question.**

After you are done, knit the RMarkdown file to PDF and submit the PDF to Gradescope under Homework 5.

Questions

Q1 (9 Points)

The table below is an ANOVA table from a simple linear regression fitted to a dataset with 50 observations. First knit the document to see what the output looks like in the Markdown table below:

Source	d.f.	Sum of Squares	Mean Square	F	P-value
Model	NA	NA	250	NA	NA
Residual	NA	400	NA		
Total	NA	NA			

Fill-in the NA values of this ANOVA table based on the information in the table. You can put your work for finding the NA values in the code chunk below. You should show all work for the values added to the table. **Carry your answers to 4 decimal places when performing calculations but round your final values in the table to two decimal places.**

You need to know that in R, `5e-6` is equivalent to scientific notation 5×10^{-6}.

```
n <- 50
SSModel <- 250
SSE <- 400
SSTotal <- SSModel + SSE
```

```r
model_df <- 1
residual_df <- n-2
Total_df <- n-1
MSModel <- 250
MSE <- SSE/(n-2)
MSE <- round(MSE, digits = 4)
f_statistic <- MSModel/MSE
p_value <- 1 - pf(f_statistic, df1 = model_df, df2 = residual_df)


cat("\n", "n", n,"\n", "SSModel", SSModel, "\n", "SSE", SSE, "\n","SSTotal", SSTotal, "\n", "Model d.f.
```

```
##
##  n 50
##  SSModel 250
##  SSE 400
##  SSTotal 650
##  Model d.f. 1
##  Residual d.f. 48
##  Total d.f. 49
##  MSModel 250
##  MSE 8.3333
##  F 30.00012
##  P_Value 1.559845e-06
```

Source	d.f.	Sum of Squares	Mean Square	F	P-value
Model	1	250	250	30	0.00
Residual	48	400	8.33		
Total	49	650			

Q2 (4 Points)

The dataset **Pines** comes from the **Stat2Data** package. Start by looking at the help documentation to understand the contents of this dataset. After loading the package, `?Pines` can be useful to get descriptions of all the variables in the dataset. Run the code `?Pines` temporarily, then erase so that it is not present when you knit the document.

Create a new dataframe called **Pines2** that only contains the two variables named *Hgt96* and *Diam97* and only contains observations that have a non-missing measurements for these two variables. Considering using the `na.omit()` function which removes all observations where there is at least one missing value in one of the columns.

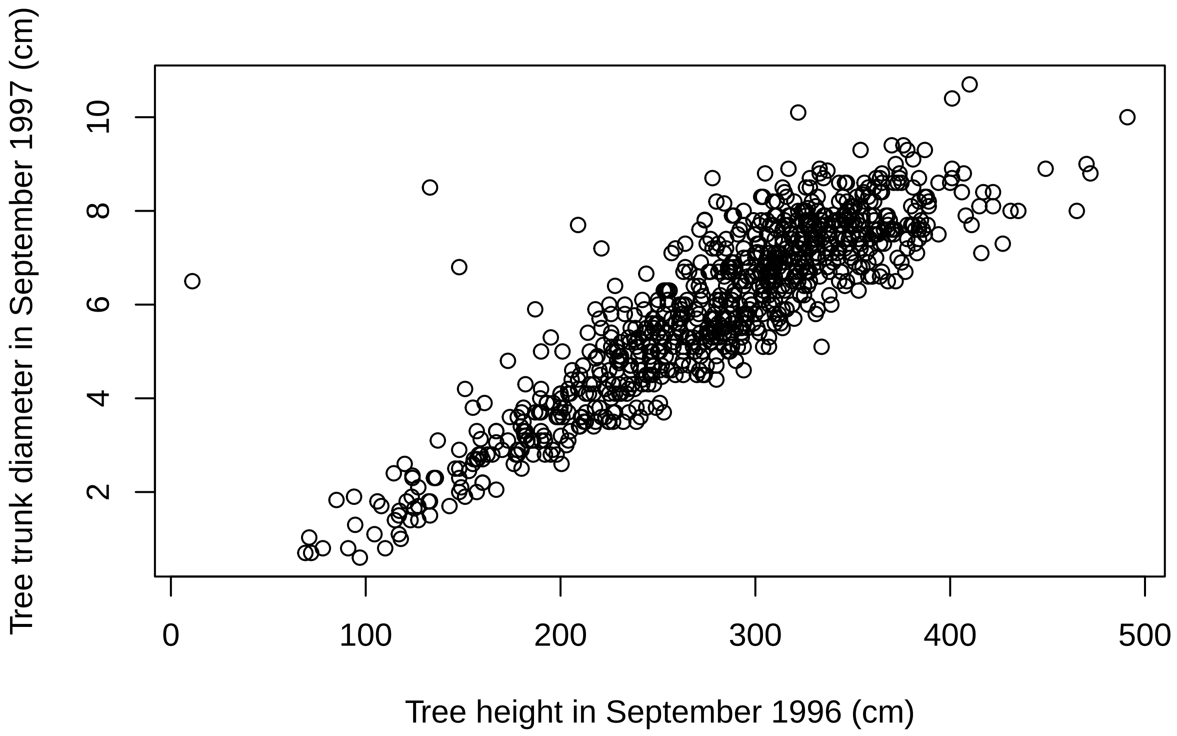
Use the `str()` function to print out a preview of **Pines2** and use the `plot()` function to visualize the relationship of *Diam97* versus *Hgt96*. The output from these two functions should be your only output.

```r
data("Pines") #DO NOT CHANGE
Pines2 <- Pines[, c("Hgt96", "Diam97")]
Pines2 <- na.omit(Pines2)

str(Pines2)
```

```
## 'data.frame':    851 obs. of  2 variables:
##  $ Hgt96 : num  284 387 294 310 318 328 157 282 251 201 ...
##  $ Diam97: num  6.6 9.3 7 6.9 7.6 7.7 2 6.1 3.9 5 ...
##  - attr(*, "na.action")= 'omit' Named int [1:149] 1 4 17 19 22 37 38 50 52 59 ...
##   ..- attr(*, "names")= chr [1:149] "1" "4" "17" "19" ...
```

```r
plot(Diam97 ~ Hgt96, Pines2,
     xlab = "Tree height in September 1996 (cm)",
     ylab = "Tree trunk diameter in September 1997 (cm)")
```

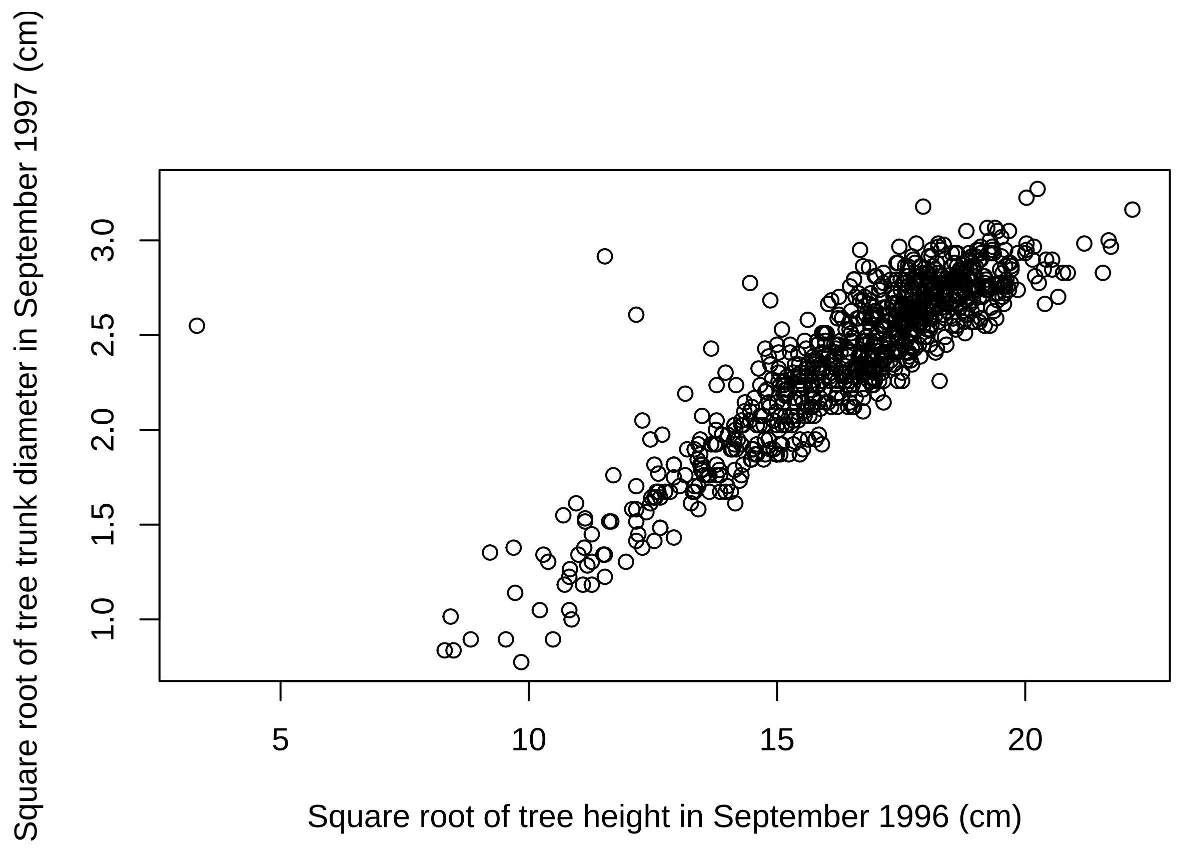


Q3 (2 Points)

Add variables to **Pines2** named *sqH96* and *sqD97* that are square roots of the original variables *Hgt96* and *Diam97*, respectively. Then, create the same scatterplot from the previous question with the square roots of the variables replacing the original variables.

```r
Pines2$sqH96 <- sqrt(Pines2$Hgt96)
Pines2$sqD97 <- sqrt(Pines2$Diam97)

plot(sqD97 ~ sqH96, Pines2,
     xlab = "Square root of tree height in September 1996 (cm)",
     ylab = "Square root of tree trunk diameter in September 1997 (cm)")
```



Q4 (2 Points)

There is one data point that has high leverage and is clearly outside the pattern of the rest of the data more than all the other points. Identify this point and create a data frame called **Pines3** that contains all the data from **Pines2** except for this problematic point which we are assuming is an error/mistake.

The `which.min()` and `which.max()` functions can be useful for identifying the location of a minimum or maximum in a vector. The first two lines of code show how these functions work. Using one of these functions may be more useful then scrolling through the data to find the exact row where the problem exists.

Also, when we are subsetting using the bracket notation, you can go from selecting observation(s) to removing observation(s) by converting positive integers to negative integers. The next two lines of code illustrate how this works on a vector.

I recommend learning how these two ideas can be combined to identify the exact row that contains the bad observation. Use the `str()` function on **Pines3** to confirm that this observation is removed.

```r
which.min(c(4,9,5,3,2,5,6,2,3,1,8)) #DO NOT CHANGE
```

```
## [1] 10
```

```r
which.max(c(4,9,5,3,2,5,6,2,3,1,8)) #DO NOT CHANGE
```

```
## [1] 2
```

```r
c(3,4,5,6,7,8)[c(1,3)]  #DO NOT CHANGE
```

```
## [1] 3 5
```

```r
c(3,4,5,6,7,8)[-c(1,3)]  #DO NOT CHANGE
```

```
## [1] 4 6 7 8
```

```r
avg.x = mean(Pines2$sqH96)
dev_squared = (Pines2$sqH96 - avg.x)^2
leverage = 1/nrow(Pines2) + dev_squared / sum(dev_squared)

Pines2$Leverage = leverage

outline <- which.max(Pines2$Leverage)

Pines3 <- Pines2[-outline, ]
str(Pines3)
```

```
## 'data.frame':    850 obs. of  5 variables:
##  $ Hgt96   : num  284 387 294 310 318 328 157 282 251 201 ...
##  $ Diam97  : num  6.6 9.3 7 6.9 7.6 7.7 2 6.1 3.9 5 ...
##  $ sqH96   : num  16.9 19.7 17.1 17.6 17.8 ...
##  $ sqD97   : num  2.57 3.05 2.65 2.63 2.76 ...
##  $ Leverage: num  0.00119 0.00329 0.00124 0.0014 0.00152 ...
##  - attr(*, "na.action")= 'omit' Named int [1:149] 1 4 17 19 22 37 38 50 52 59 ...
##   ..- attr(*, "names")= chr [1:149] "1" "4" "17" "19" ...
```

Q5 (6 Points)

Fit a linear model for the relationship *sqD97* vs *sqH96* based on the data in **Pines3**. Print out the ANOVA table from this model and xalculate the standard error of the regression and the r-squared statistic only using the numbers in this table. In your output, I should see the ANOVA table and the final calculations of the two statistics mentioned above based on the numbers in that table.

After this, round your numerical values for the standard error and r-squared to two decimal places, and write them in the appropriate space below the code chunk. Don't round until you get your final answer.

```r
lmod <- lm(sqD97 ~ sqH96, Pines3)

anova_lmod <- anova(lmod)
anova_lmod
```

```
## Analysis of Variance Table
##
## Response: sqD97
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## sqH96       1 128.871 128.871    4085 < 2.2e-16 ***
## Residuals 848  26.752   0.032
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
SSModel <- anova_lmod$`Sum Sq`[1]
SSE <- anova_lmod$`Sum Sq`[2]
SSTotal <- SSModel + SSE

residual_df <- anova_lmod$Df[2]

standard_error <- sqrt(SSE / residual_df)
r2 <- SSModel/SSTotal

cat("standard error:", standard_error, "\n")
```

```
## standard error: 0.1776152
```

```r
cat("r-squared:", r2, "\n")
```

```
## r-squared: 0.8280973
```

Standard Error of Regression: 0.18

r-squared: 0.83

Q6 (2 Points)

Perform the appropriate t-test for the correlation between *sqD97* and *sqH96*. Also, use the `summary()` function on your model. Compare the p-values from the t-test for the slope and the t-test for the correlation. Notice they are not identically presented, but they are identical. This is just due to different rounding rules used in the two outputs.

I only want to see the output from the correlation test and the `summary()` function.

```r
cor.test(x=Pines3$sqH96,y=Pines3$sqD97)
```

```
##
##  Pearson's product-moment correlation
##
## data:  x and y
## t = 63.914, df = 848, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8976857 0.9208913
## sample estimates:
##       cor
## 0.9099985
```

```r
summary(lmod)
```

```
##
## Call:
## lm(formula = sqD97 ~ sqH96, data = Pines3)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.45496 -0.11903 -0.00831  0.10305  1.39446
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.475363   0.045396  -10.47   <2e-16 ***
## sqH96        0.173108   0.002708   63.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1776 on 848 degrees of freedom
## Multiple R-squared:  0.8281, Adjusted R-squared:  0.8279
## F-statistic:  4085 on 1 and 848 DF,  p-value: < 2.2e-16
```

Q7 (4 Points)

Suppose there were two Pine trees that were not in the dataset **Pines3**. One of these, pine trees had a height of 300cm and the other had a height of 410cm in 1996. I want to generate two intervals for the purpose of predicting the actual tree trunk diameter in centimeters for each of these trees in 1997.

To do this follow these steps:

1. Create a data.frame called **newx** that only contains the tree heights for these two trees in 1996. Use the same original variable name from **Pines**.
2. Create a variable of the same name of a variable used before that is the square root of these heights. Add this variable to **newx**.
3. Use the `predict()` function appropriately to apply the fitted model to the data in **newx**. Make sure you obtain the appropriate interval for this situation. Save the output from the `predict()` function into an object called **predictx**.
4. Remember that are model predicts the square root of the diameter and not the actual diameter. Therefore, run the code `predictx^2` to untransform all the numbers in the table back to their original units. This should be the only output from this code.

This table shows the predicted diameters in cm for these two trees in 1997 along with the appropriate prediction intervals. Look at the original scatter plot and think about if your final values make sense. Read the textbook about how to appropriately interpret and use these prediction intervals for both of these trees.

```r
#1
newx <- data.frame(Hgt96 = c(300, 410))
#2
newx$sqH96 <- sqrt(newx$Hgt96)
#3
predictx <- predict(lmod, newdata = newx, interval = "prediction")
#4
predictx <- predictx^2

print(predictx)
```

```
##        fit      lwr       upr
## 1 6.365342 4.726799  8.247267
## 2 9.179782 7.184852 11.418814
```