ECON 4101 Econometrics
CM05 Homework

Pranav Singh

Jan 29, 2017

Goal:

Use simple linear regression and analysis of variance to test the hypothesis that reported hectares of corn (soybeans) are explained by the number of pixels of corn (soybeans) in sample segment within county, from satellite data.

Data:

Survey and satellite data for 37 observations of corn and soy beans in 12 Iowa counties, obtained from the 1978 June Enumerative Survey of the U.S. Department of Agriculture and from land observatory satellites (LANDSAT) during the 1978 growing season.

county: county number
cornhec: hectares of corn
soyhec: hectares of soybeans
cornpix: satellite pixels of corn
soypix: satellite pixels of soybeans

```
library(xlsx)
library(data.table)
temp <- tempfile()
download.file("http://evansresearch.us/DSC/Spring2017/ECMT/Data_Woolridge/corn.xls",
    temp)
data <- setDT(read.xlsx2(temp, 1, colClasses = c("character", rep("numeric", 4))))
unlink(temp)
```

Problem 1

```
colnames(data) <- c("county", "cornhec", "soyhec", "cornpix", "soypix")
n <- nrow(data)
print(paste0("Number of observations: ", n))
```

```
## [1] "Number of observations: 36"
```

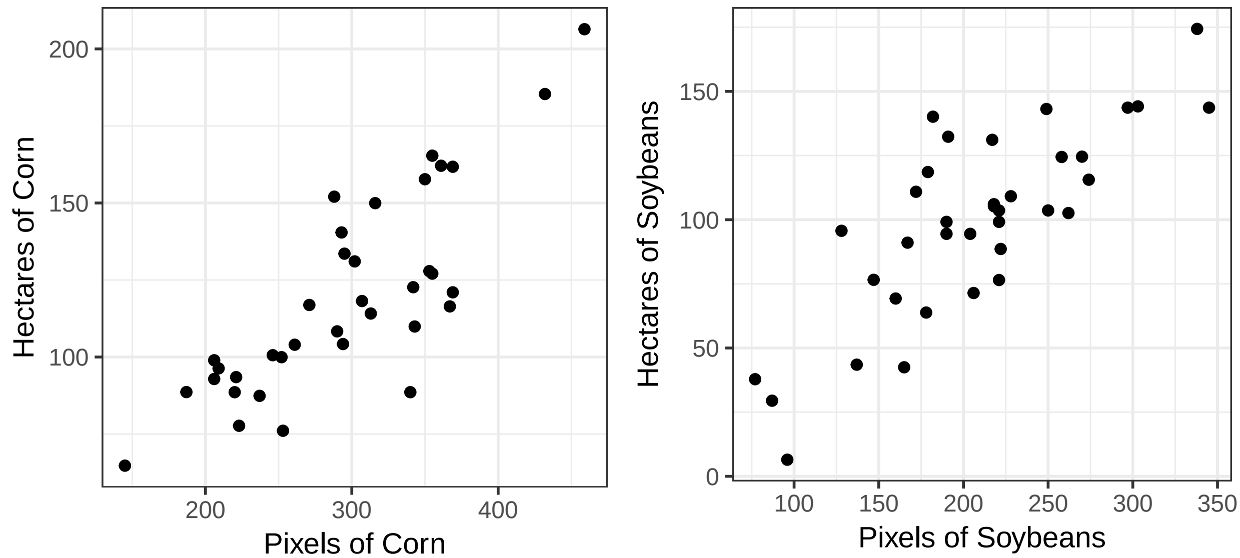
```
sapply(data[, !"county"], function(x) c(summary(x), `Standard Deviation` = sd(x),
    `Coefficient of Variance` = sd(x)/mean(x)))
```

```
##                             cornhec      soyhec     cornpix      soypix
## Min.                     64.7500000   6.4700000 145.0000000  77.0000000
## 1st Qu.                  95.6100000  76.5500000 243.8000000 170.8000000
## Median                  115.3000000 103.1000000 294.5000000 211.5000000
## Mean                    119.2000000  98.8000000 295.3000000 207.4000000
```

```
## 3rd Qu.                  135.3000000 124.5000000 350.8000000 249.2000000
## Max.                     206.4000000 174.3000000 459.0000000 345.0000000
## Standard Deviation        32.1964214  36.7839390  70.1254432  63.4847313
## Coefficient of Variance    0.2701646   0.3723165   0.2374897   0.3060324
```

Problem 2

```r
g1 <- ggplot(data = data, aes(x = cornpix, y = cornhec)) + geom_point() + labs(y = "Hectares of Corn",
    x = "Pixels of Corn")
g2 <- ggplot(data = data, aes(x = soypix, y = soyhec)) + geom_point() + labs(y = "Hectares of Soybeans"
    x = "Pixels of Soybeans")
grid.arrange(g1, g2, ncol = 2)
```



Problem 3

```r
lm.corn <- lm(data = data, cornhec ~ cornpix)
lm.soy <- lm(data = data, soyhec ~ soypix)

cornhec.hat <- predict(lm.corn)
soyhec.hat <- predict(lm.soy)

cornhec.bar <- mean(data$cornhec)
soyhec.bar <- mean(data$soyhec)
sse.corn <- sum((data$cornhec - cornhec.hat)^2)
sse.soy <- sum((data$soyhec - soyhec.hat)^2)
ssr.corn <- sum((cornhec.bar - cornhec.hat)^2)
ssr.soy <- sum((soyhec.bar - soyhec.hat)^2)
tss.corn <- ssr.corn + sse.corn
tss.soy <- ssr.soy + sse.soy
r2.corn <- ssr.corn/tss.corn
r2.soy <- ssr.soy/tss.soy
```

```
df.regression <- 1
df.error <- n - df.regression - 1

msr.corn <- ssr.corn/df.regression
msr.soy <- ssr.soy/df.regression
mse.corn <- sse.corn/df.error
mse.soy <- sse.soy/df.error

fstat.corn <- msr.corn/mse.corn
fstat.soy <- msr.soy/mse.soy
```

```
options(knitr.kable.NA = "")
anova.rownames <- c("Regression", "Error", "Total")
anova.corn <- data.frame(`Degrees of Freedom` = c(df.regression, df.error, df.regression +
    df.error), `Sum of Squares` = c(ssr.corn, sse.corn, tss.corn), `Mean Sum of Squares` = c(msr.corn,
    mse.corn, NA), `F Statistic` = c(fstat.corn, NA, NA))
rownames(anova.corn) <- anova.rownames
anova.soy <- data.frame(`Degrees of Freedom` = c(df.regression, df.error, df.regression +
    df.error), `Sum of Squares` = c(ssr.soy, sse.soy, tss.soy), `Mean Sum of Squares` = c(msr.soy,
    mse.soy, NA), `F Statistic` = c(fstat.soy, NA, NA))
rownames(anova.soy) <- anova.rownames

kable(anova.corn, digits = 4, caption = "ANOVA: CornHec ~ CornPix")
```

Table 1: ANOVA: CornHec ~ CornPix

	Degrees.of.Freedom	Sum.of.Squares	Mean.Sum.of.Squares	F.Statistic
Regression	1	24270.05	24270.0473	68.7005
Error	34	12011.29	353.2731	
Total	35	36281.33		

```
kable(anova.soy, digits = 4, caption = "ANOVA: SoyHec ~ SoyPix")
```

Table 2: ANOVA: SoyHec ~ SoyPix

	Degrees.of.Freedom	Sum.of.Squares	Mean.Sum.of.Squares	F.Statistic
Regression	1	30592.41	30592.4124	62.0439
Error	34	16764.62	493.0772	
Total	35	47357.04		

```
see.corn <- sqrt(mse.corn)
see.soy <- sqrt(mse.soy)

fcrit <- qf(0.95, df.regression, df.error)
pf.corn <- pf(fstat.corn, df.regression, df.error, lower.tail = F)
pf.soy <- pf(fstat.soy, df.regression, df.error, lower.tail = F)

print(paste0("Corn: Standard Error of Estimate = ", see.corn))
```

```
## [1] "Corn: Standard Error of Estimate = 18.7955618381817"
```

```
print(paste0("Soybeans: Standard Error of Estimate = ", see.soy))
```

```
## [1] "Soybeans: Standard Error of Estimate = 22.2053408535331"
```

```
print(paste0("Critical value of F at .05 significance level: ", fcrit))
```

```
## [1] "Critical value of F at .05 significance level: 4.13001774565201"
```

```
print(paste0("Corn: (F-statistic, p-value) = (", fstat.corn, ", ", pf.corn, ")"))
```

```
## [1] "Corn: (F-statistic, p-value) = (68.7005158266815, 0.00000000112961941756451)"
```

```
print(paste0("Soybeans: (F-statistic, p-value) = (", fstat.soy, ", ", pf.soy, ")"))
```

```
## [1] "Soybeans: (F-statistic, p-value) = (62.0438638902307, 0.00000000358642239029152)"
```

Since the F-test statistic for both models is greater than the critical F-value, we conclude that, at the 5% significance level, the predictors do help explain more of the variance of their corresponding responses than does the null (intercept-only) model. That is, when modeling hectares of corn/soybeans, the simple linear regression model that takes into account satellite pixels of corn/soybeans has a better fit than the null model that only includes an intercept term.

```
aes.near.topleft <- aes(x = -Inf, y = Inf, hjust = -0.1, vjust = 2)
g1 <- g1 + geom_line(aes(y = cornhec.hat)) + geom_text(aes.near.topleft, label = lm_eqn(lm.corn),
    parse = T, size = 4)
g2 <- g2 + geom_line(aes(y = soyhec.hat)) + geom_text(aes.near.topleft, label = lm_eqn(lm.soy),
    parse = T, size = 4)
grid.arrange(g1, g2, nrow = 2)
```