Regression

Getting some data

- We already know how to load .csv data in to R.

```
hassy<-read.csv("avocado.csv")
colnames(hassy)[3]<-"price" #rename for convenience
colnames(hassy)[4]<-"sales" #rename for convenience
colnames(hassy)
```

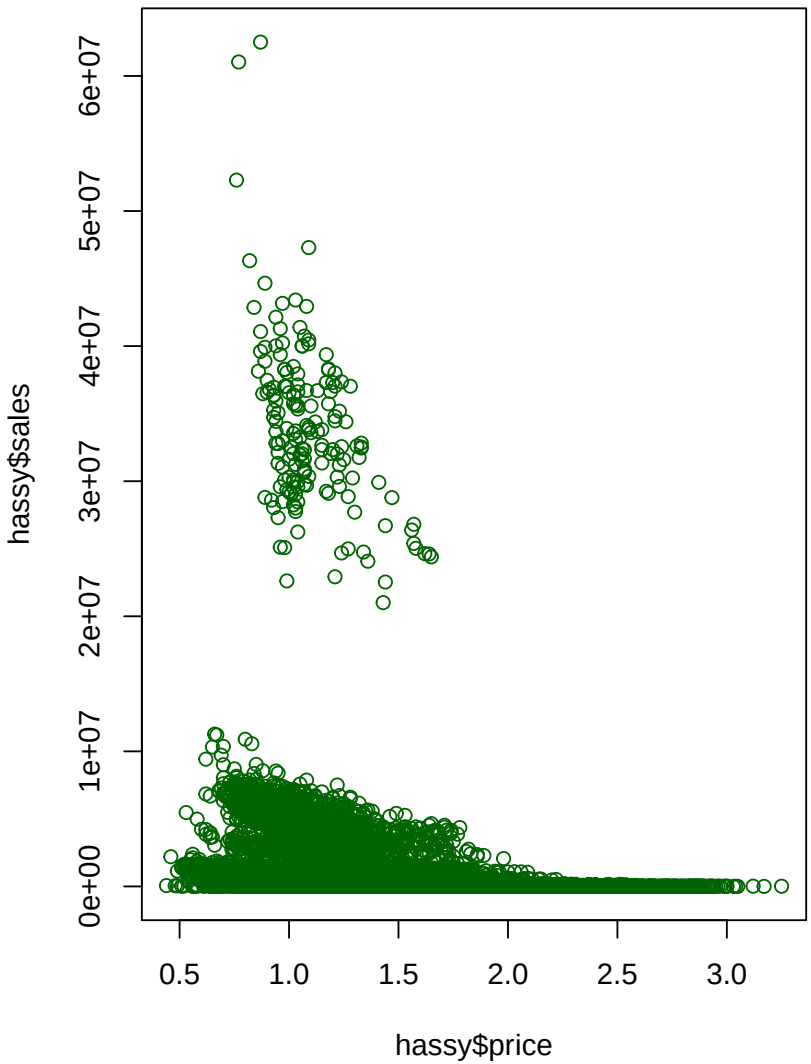
```
##  [1] "X"            "Date"         "price"        "sales"        "X4046"
##  [6] "X4225"        "X4770"        "Total.Bags"   "Small.Bags"   "Large.Bags"
## [11] "XLarge.Bags" "type"         "year"         "region"
```

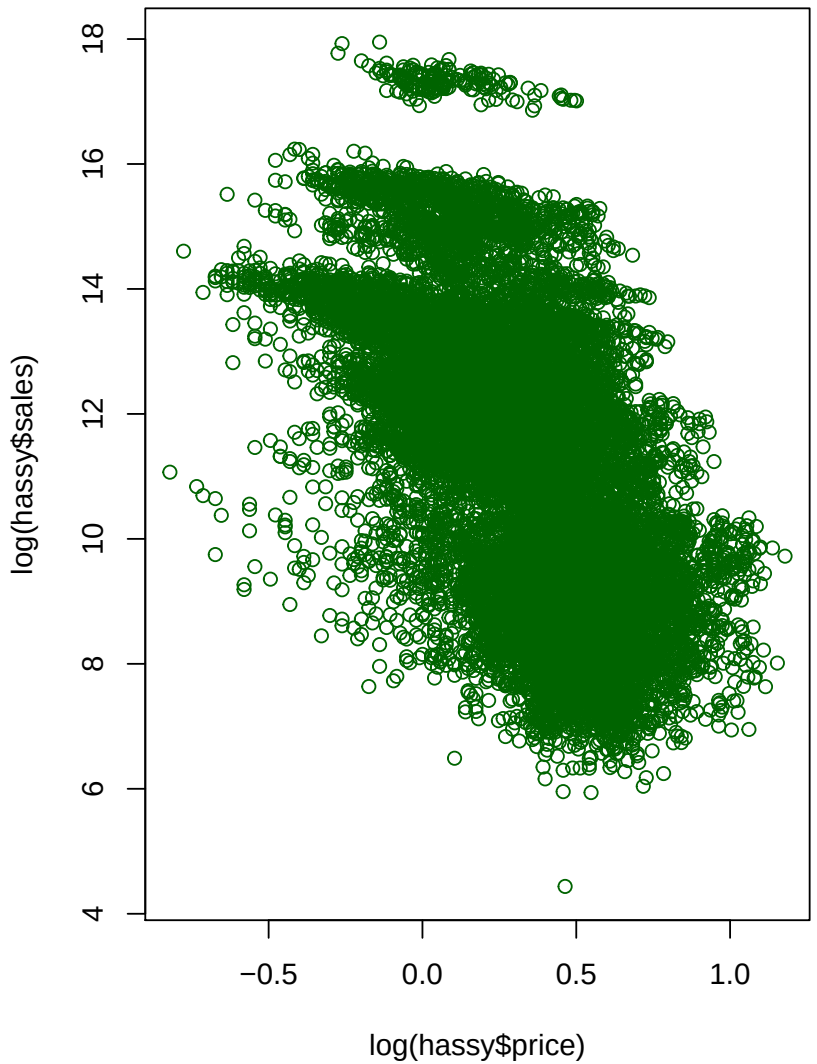
```
str(hassy)
```

Simple Linear Regression

- We will start by looking at log(prices) and log(sales), here is why.

```
par(mfrow = c(1,2))
plot(hassy$price, hassy$sales, col = "darkgreen")
plot(log(hassy$price), log(hassy$sales), col = "darkgreen")
```





Simple Linear Regression

- Recall the logarithm definition:

$$log(a) = b \Leftrightarrow a = e^b$$

where $e \approx 2.72$ is Euler's number.

- So in the following equation, b_1 is added to $log(y)$ for each unity in x.

$$log(y) = b_0 + b1x$$

- Recall that $e^{(a+b)} = e^a e^b$, therefore:

$$y = e^{(b_0+b_1x)} = e^{b_0} e^{b_1x}$$

- Now let's make:

$$y^* = e^{b_0} e^{b_1(x+1)} = e^{b_0} e^{b_1x+b_1} = e^{b_0+b_1x} e^{b_1} = ye^{b_1}$$

Simple Linear Regression

- Therefore, each unit increase in x leads y to be multiplied by the factor e^{b_1}.
- How do you know that variables should be modeled as changing multiplicatively? One indicator is everyday language. Be on the lookout for variables whose change is usually expressed in percentage rather than absolute terms.
- Another common scenario models against each other two variables that both move multiplicatively. This will be called a log-log model, which suits for our avocado data. In this case we have:

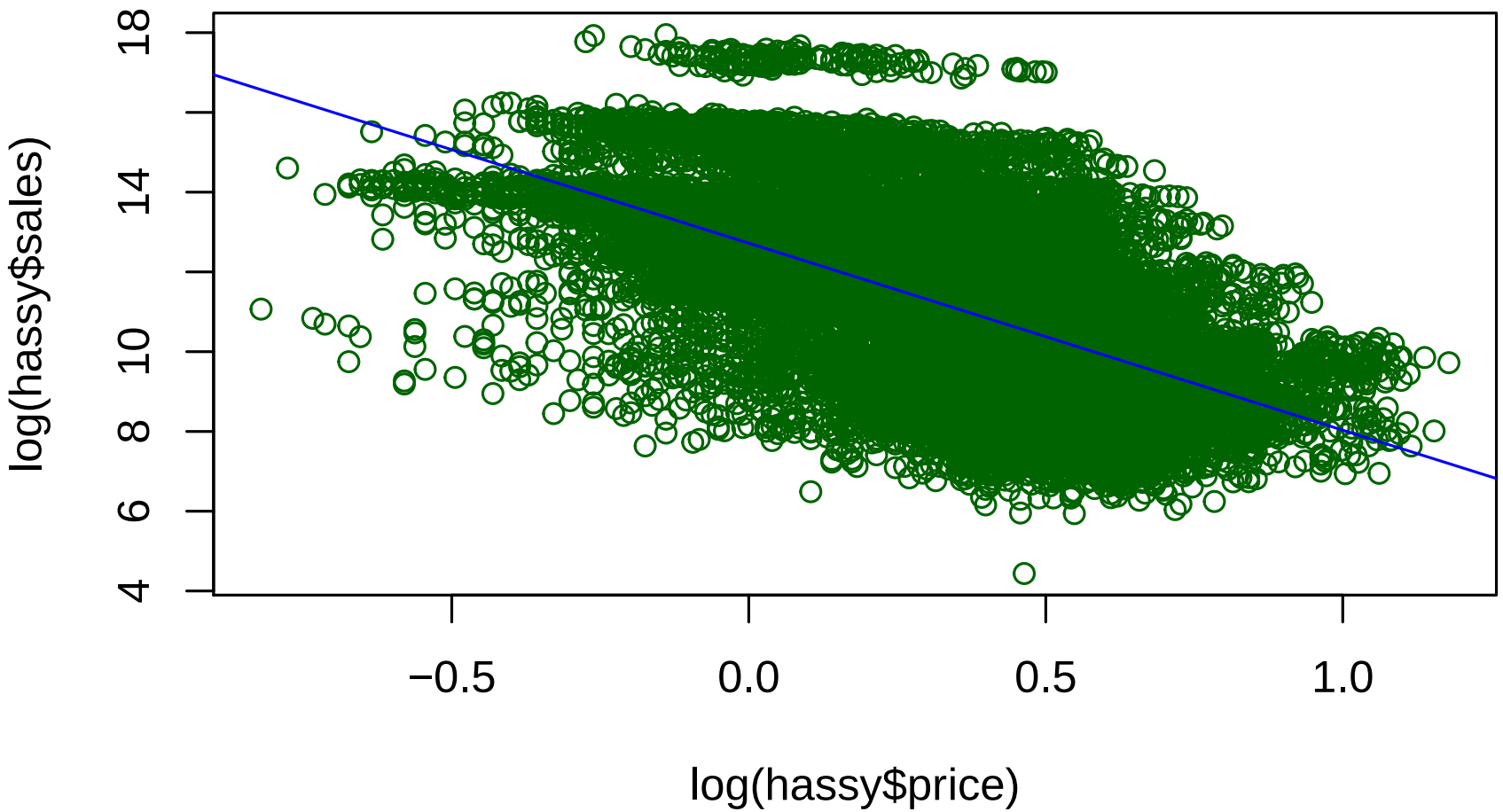
$$log(sales) = \beta_0 + \beta_1 log(price) + \varepsilon$$

- The interpretation is very convenient. Sales increase by $\beta_1\%$ for every 1% increase in price.

Simple Linear Regression

- We are ready to start running some regressions. You can run regressions in R with the **lm** (linear model) and the **glm** (generalized linear model) commands.
- If you don't tell R the type of generalized linear model that you want, glm an lm will do the same thing.

```
fit = lm(log(sales) ~ log(price), data = hassy)
plot(log(hassy$price), log(hassy$sales), col = "darkgreen")
abline(fit, col = "blue")
```



Simple Linear Regression

- The commands **summary(fit)** and **coef(fit)** extract information from the model.
- The command **predict(fit,newdata)** is for predictions. The argument newdata contains the right side variables used in the prediction. It must have the **same format** as the data used to estimate the model. Same column names, same factor levels, etc (for the variables used in the model).

```
coef(fit)
```

```
## (Intercept)  log(price)
##   12.720763   -4.688243
```

```
predict(fit, newdata=data.frame(price=1.5))
```

```
##        1
## 10.81984
```

Simple Linear Regression

```
summary(fit)
```

```
##
## Call:
## lm(formula = log(sales) ~ log(price), data = hassy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.2488 -1.2830 -0.0161  1.1999  6.6370
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.72076    0.01968  646.31   <2e-16 ***
## log(price)  -4.68824    0.04723  -99.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.848 on 18247 degrees of freedom
## Multiple R-squared:  0.3506, Adjusted R-squared:  0.3506
## F-statistic:  9853 on 1 and 18247 DF,  p-value: < 2.2e-16
```

Simple Linear Regression

- If we just look at y we have the marginal distribution of sales.
- However, we are looking at $y|x$, which is the conditional distribution of y on x.
- Moreover, the Ordinary Least Squares (OLS) looks at the mean of the conditional distribution:

$$E[y|x] = \beta_0 + \beta_1 x$$

Linear Models

- The previous example used only one variable (price) as predictor. That is why it is called a simple linear regression.
- We might want to add more variables:

$$E[y|X] = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = X\beta$$

where X is the matrix where each column is a variable and β is the vector with all β_is.

- When fitting a linear regression, we make some assumptions on the conditional distribution $y|x$. The most usual is to assume that it follows a *Gaussian* (Normal) distribution.

$$y|X \sim N(X\beta, \sigma^2)$$

Thus, we can write the linear regression model as:

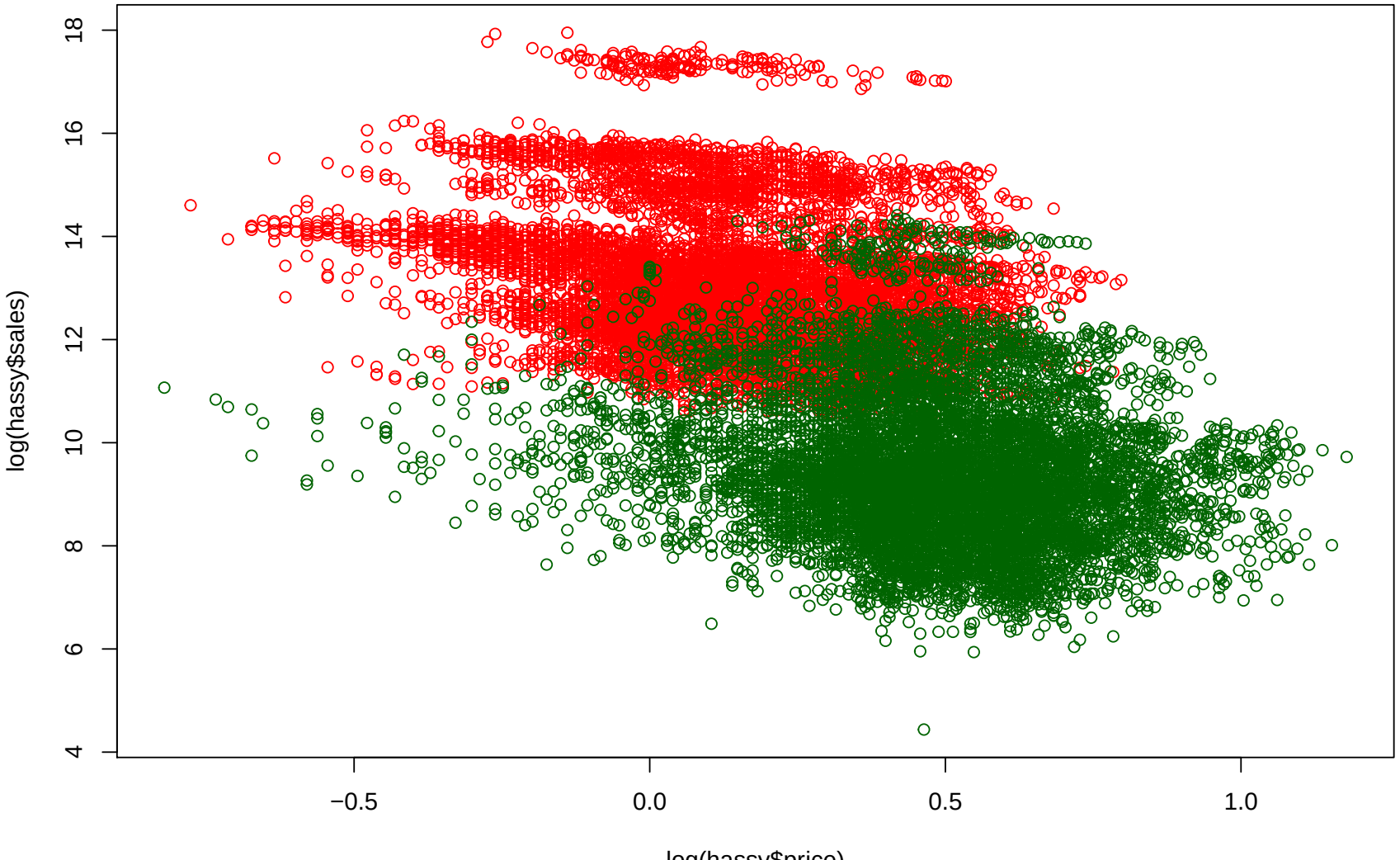
$$y = X\beta + \varepsilon, \qquad \varepsilon \sim N(0, \sigma^2)$$

where ε are the populational errors.

Linear Models

- Let's look at some different variables. The variable type tells if the avocado is conventional or organic.

```
plot(log(hassy$price), log(hassy$sales),
     col = ifelse(hassy$type=="conventional","red", "darkgreen"))
```



Linear Models

```
fit2 = lm(log(sales) ~ log(price) + type, data = hassy)
summary(fit2)
```

```
##
## Call:
## lm(formula = log(sales) ~ log(price) + type, data = hassy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0707 -0.9661 -0.1823  0.7308  4.8455
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.29818    0.01515  877.54   <2e-16 ***
## log(price)  -1.28678    0.04402  -29.23   <2e-16 ***
## typeorganic -3.19332    0.02550 -125.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.356 on 18246 degrees of freedom
## Multiple R-squared:  0.6507, Adjusted R-squared:  0.6507
## F-statistic: 1.7e+04 on 2 and 18246 DF,  p-value: < 2.2e-16
```

Linear Models

```
plot(log(hassy$price), log(hassy$sales),
     col = ifelse(hassy$type=="conventional","red", "darkgreen"))
abline(coef(fit2)[1], coef(fit2)[2], col = "blue")
abline(coef(fit2)[1] + coef(fit2)[3] , coef(fit2)[2], col = "black")
```