Linear Regression

Jeremy Oldfather

October 11, 2016

Review

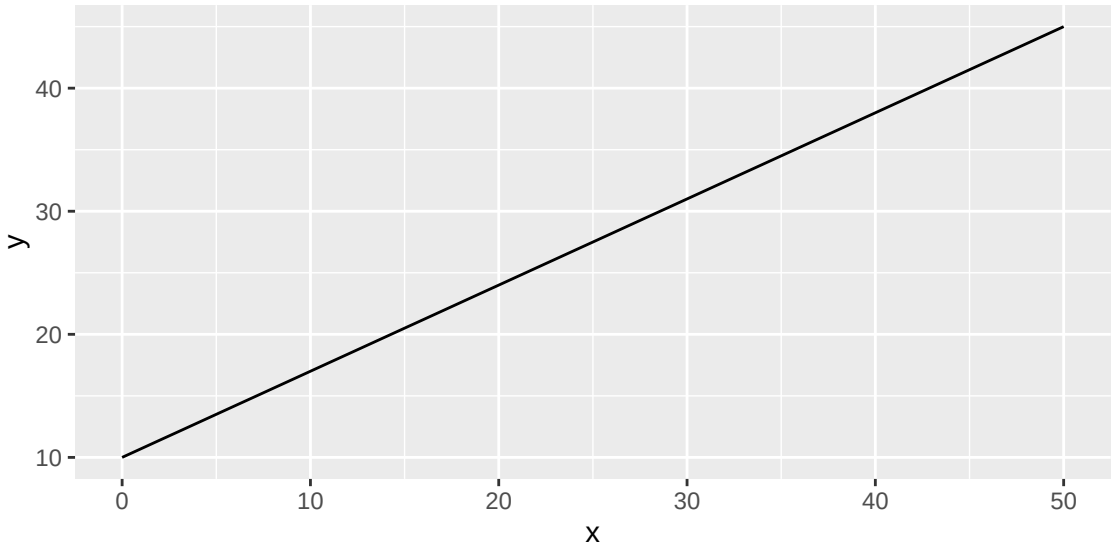
What is linear regression?

Start with a line.

$$y = mx + b \tag{1}$$

We know how this appears visually. The line crosses the y-axis at the height of b (intercept) and for each unit we move left or right, we move up or down by m (slope) units. For example, with $b = 10$ and $m = .7$, we get the following line:

```
b<-10
m<-.7
x<-0:50
y<-m*x + b
qplot(x,y,geom="line")
```



Since the line has two parameters, m and b, we could take any **two** distinct points, draw a line between them, and solve for the parameters. If we only have **one** point, we could draw infinitely many lines through the point.

Let's take this idea to some data. I have some data on the sales of DC properties. We can conceptualize each property as a point. Here is the ***sale_price*** and ***living_sqft*** of the first two properties.

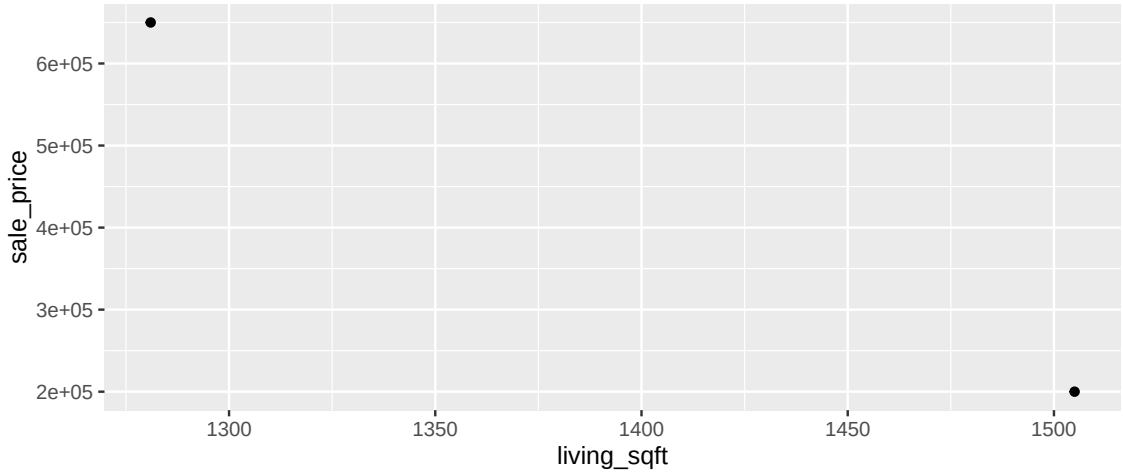
```
# see: https://github.com/theoldfather/DC_Properties
sales<-read_csv("~/Projects/DC_Properties/data/sales_clean.csv")
features<-read_csv("~/Projects/DC_Properties/data/features_clean.csv")
details<-read_csv("~/Projects/DC_Properties/data/details_clean.csv")
```

```
d<-sales %>%
  filter(sale_price>0) %>%
  left_join(features,by="ssl") %>%
  filter(!is.na(living_sqft) & living_sqft>0) %>%
  select(sale_price,living_sqft)
head(d,2) %>% data.frame()

##   sale_price living_sqft
## 1     200000        1505
## 2     650000        1281
```

Plotting these two points, we get the following scatterplot.

```
ggplot(d[1:2,],aes(x=living_sqft,y=sale_price)) + geom_point()
```



We could draw a segment between the lines by changing ***geom_point()*** to ***geom_line()***, but let's do it by finding the slope and intercept, and then plotting the line through the points.

If we subscript equation 1 with i to represent each observation, we can write it in a condensed notation as

$$y_i = mx_i + b \tag{2}$$

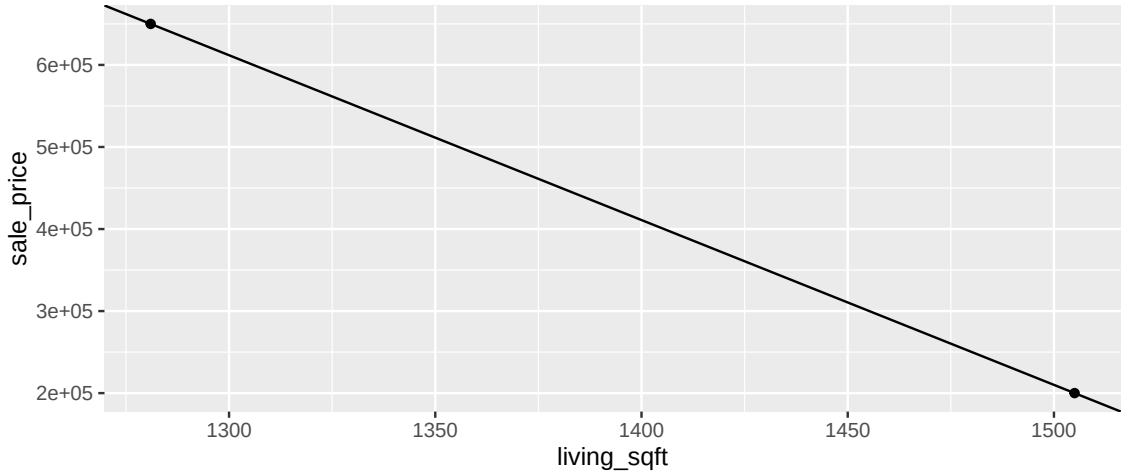
and then expand it for $i \in \{1, 2\}$.

$$y_1 = mx_1 + b \tag{3}$$
$$y_2 = mx_2 + b \tag{4}$$

Solving for m and b we get

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{5}$$
$$b = y_1 - mx_1 \tag{6}$$

```
fit_line<-function(y,x){
  m <- (y[2]-y[1])/(x[2]-x[1])
  b <- y[1] - m*x[1]
  return(c(b,m))
}
beta<-fit_line(d$sale_price[1:2],d$living_sqft[1:2])
ggplot(d[1:2,],aes(x=living_sqft,y=sale_price)) +
  geom_point() + geom_abline(intercept=beta[1],slope=beta[2])
```

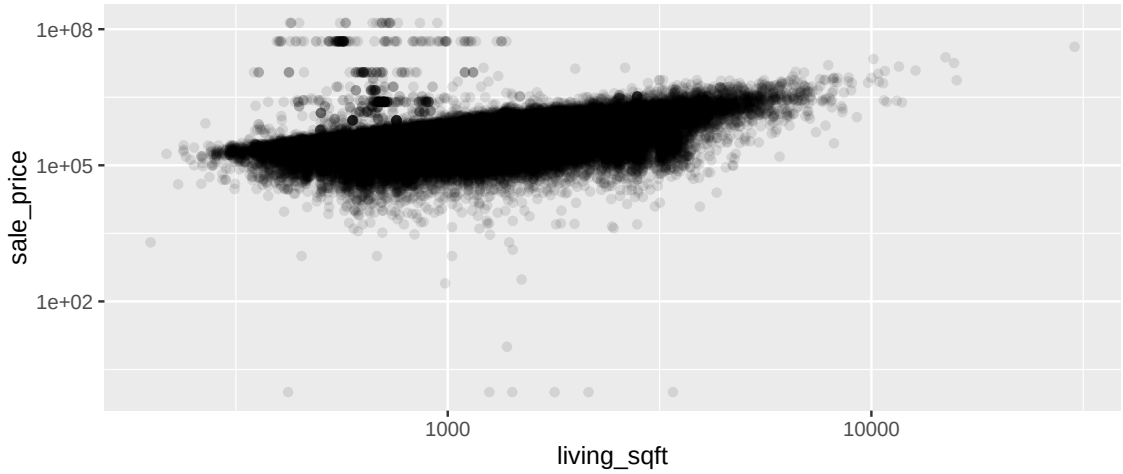


Do you believe this is an accurate model of the relationship between sales price and the living area of a home? $b = 3.2234375 \times 10^6$ implies that a house with no space would be worth \$3 million. And $m = -2008.9285714$ implies that increasing the living area of a home by 1 sqft would decrease its values by \$2,008.

Thankfully, we have data on more than two homes. But what does it mean to draw a line through **more than two** points? We know our model above is very bad—so not every point will fall on the same line.

Here is the rest of our data plotted on log-log scale (we will talk about log-log scale later).

```
ggplot(d,aes(x=living_sqft,y=sale_price)) +
  geom_point(alpha=.1) + scale_y_log10() + scale_x_log10()
```



This scatterplot seems promising–it visually re-enforces our beliefs that people place more value on larger homes. But we still want to quantify these beliefs to inform decision about buying, selling, making home improvements, etc.

Even though there is not a single line that can pass through each point, we can still find an "average line" that best-fits the data. This is the essence of regression.

Simple Linear Regression

Let's expand our general formula for a line above (Equation 2) to express the fact that our line does not fit exactly, but rather has some slack distance between it and each point.

$$y_i = mx_i + b + \epsilon_i \tag{7}$$

The slack or **residual** term, ϵ_i, also gets a subscript i because we cannot assume that the line will be the same distance (vertically) from every point. Or in other words, our model will explain some home prices better than others.

Let's also formalize the notation a bit more and replace b with β_0 and m with β_1. Many papers use beta (β) to represent the parameters of the model, but sometimes you will also find the intercept represented as alpha (α). This is simply different notation and has the same meaning.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{8}$$

Next, we need some way of defining what it means to be the line of best-fit. What we want conceptually is the model that explains our data most accurately, and on the flip-side, the model that is the least inaccurate. Let's use the latter conceptualization and rewrite the equation above in terms of ϵ_i

$$\epsilon_i = (y_i - (\beta_0 + \beta_1 x_i)) \tag{9}$$

But minimizing a single residual is not helpful. Residuals can be negative, so the further we shift the fitted line above our points, the more and more negative each residual will become. We could keep shifting the line higher and higher and always produce a more negative residual. What if we square the residual—ϵ_i^2? Now a point that is 6 units below the fitted line will be as equally bad as a point 6 units above the line.

$$\epsilon_i^2 = (y_i - (\beta_0 + \beta_1 x_i))^2 \tag{10}$$

And we want to consider the value of all the squared residuals as once so that each point plays a role is how our line is fit. Summing over the squared residuals leads to the following formulation:

$$\sum_i \epsilon_i^2 = \sum_i (y_i - (\beta_0 + \beta_1 x_i))^2 \tag{11}$$

This is the intuition behind the loss function (the function to be minimized) in simple linear regression called the **Sum of Squared Residuals (SSR)**.

$$SSR(\beta_0, \beta_1) = \sum_i (y_i - (\beta_0 + \beta_1 x_i))^2 \tag{12}$$

By minimizing the SSR, we can find unique solutions for the coefficients β_0 and β_1. One way to do this is by taking the derivative of $SSR(\beta_0, \beta_1)$ with respect to each β, setting the derivatives equal to zero, and then solving for the coefficients. The following solutions are exactly analogous to the solutions for the intercept and slope of the line in Equations (6) and (5).

$$\beta_1 = \frac{\sum_i y_i x_i - \frac{\sum_i y_i \sum_i x_i}{n}}{\sum_i x_i^2 - \frac{(\sum_i x_i)^2}{n}} \tag{13}$$

$$\beta_0 = \sum_i y_i - \beta_1 \sum_i x_i \tag{14}$$

This might look messy at first, but notice there are really only 4 statistics we need to compute:

$$\sum_i y_i x_i \tag{15}$$

$$\sum_i y_i \tag{16}$$

$$\sum_i x_i \tag{17}$$

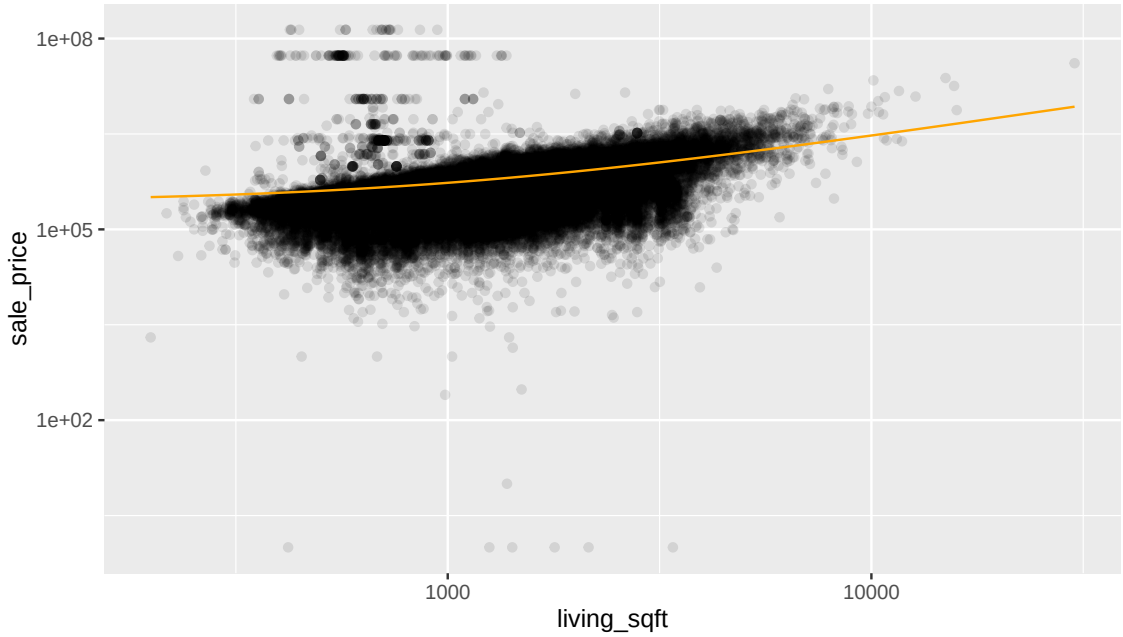
$$\sum_i x_i^2 \tag{18}$$

Using this idea, let's write a function to estimate the betas in R.

```r
fit_simple_lm<-function(y,x){
  n<-length(y)
  syx<-sum(y*x)
  sy<-sum(y)
  sx<-sum(x)
  sxx<-sum(x^2)
  b1<- (syx - sy*sx/n)/(sxx - sx^2/n)
  b0<- (sy - b1*sx)/n
  return(c(b0,b1))
}
```

Now we can use our function to estimate the betas–the coefficients of our simple linear model.

```r
# find our betas
beta<-fit_simple_lm(y=d$sale_price,x=d$living_sqft)
# use betas to create a data frame of fitted sales prices
E<-data.frame(sale_price=beta[1]+beta[2]*d$living_sqft,
                  living_sqft=d$living_sqft)
# plot our data and the fitted line
ggplot(d,aes(x=living_sqft,y=sale_price)) +
  geom_point(alpha=.1) +
  geom_line(data=E,colour="orange") +
  scale_y_log10() + scale_x_log10()
```



The fitted line is beginning to look more intuitive and it is certainly a better model than the one constructed from two points. But why is it curved? I thought we fitted a "line"?

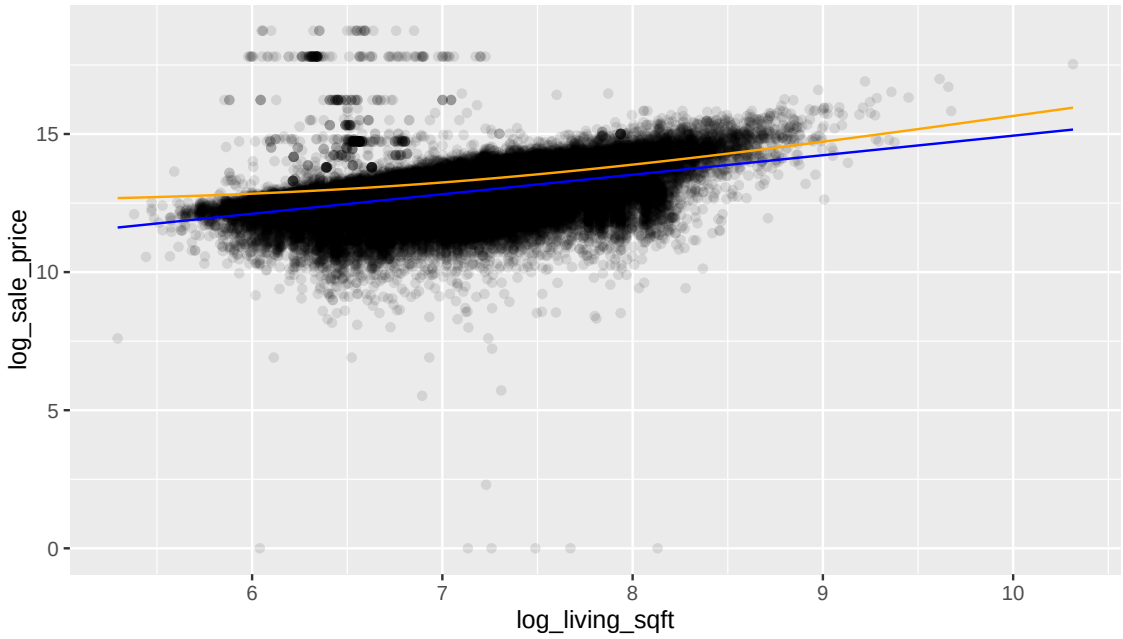
The answer is that the order in which we apply convex/concave functions to our data and fitted model matters! We can get this right by referencing **Jensen's Inequality**, which states that when

we apply a concave function (think of a hill that could "cave" in) to an expected value, its output will always be greater than or equal to the expected values of the same function applied to the input. If $f(x)$ is the concave function, the inequality is:

$$f(E(x)) \geq E(f(x)) \tag{19}$$

In our case, $log(x)$ is the concave function and $E(x)$ is our fitted line, $\hat{y}_i = \beta_0 + \beta_1 x_i$. We can visualize the difference this order of operations makes by plotting both alternatives. The **orange** line is what we plotted before, $log(E(\beta_0 + \beta_1 x_i))$, and the **blue** one is the model fitted on the log data, $E(log(\beta_0 + \beta_1 x_i))$.

```
# log scale both y and x manually
logd <- d %>%
  mutate(log_sale_price=log(sale_price),
         log_living_sqft=log(living_sqft))
# estimate betas again based on log-log model
beta<-fit_simple_lm(y=logd$log_sale_price,x=logd$log_living_sqft)
# E(log(x))
E_log<-data.frame(log_sale_price=beta[1]+beta[2]*logd$log_living_sqft,
                  log_living_sqft=logd$log_living_sqft)
# log(E(x))
log_E<-E %>% mutate(log_sale_price = log(sale_price),
                    log_living_sqft = log(living_sqft))
# plot the log-log data manually (no scale_y_log10() or scale_x_log10() )
ggplot(logd,aes(x=log_living_sqft,y=log_sale_price)) +
  geom_point(alpha=.1) +
  geom_line(data=log_E,colour="orange") +
  geom_line(data=E_log,colour="blue")
```




```
beta
```

```
## [1] 7.8751859 0.7063513
```

Interpretting a Linear Regression

Now that we finally have the fitted line that we expected to see (the blue one), let's step back and think about how we should interpret the coefficients that parameterize it: $\beta_1 = 0.7063513$ and $\beta_0 = 7.8751859$.

level-level

With a level-level model, we have the raw levels of variables on each side of the regression.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Since this model has the form of a line, we can interpret β_1 as the slope, $\Delta y / \Delta x$. To prove this to ourselves, we only need to take the derivative of the model with respect to x.

$$\frac{dy}{dx} = \beta_1 \tag{20}$$

$$\beta_1 = \frac{dy}{dx} \tag{21}$$

So we can interpret $\beta 1$ as the **marginal effect** since a 1 unit increase in x leads to a β_1 unit change in y.

log-level

A log-level model is not as intuitive, but we can apply the same analysis to arrive at an interpretation.

$$log(y_i) = \beta_0 + \beta_1 x_i + \epsilon_i \tag{22}$$
$$y_i = e^{\beta_0 + \beta_1 x_i + \epsilon_i} \tag{23}$$

Taking the derivative with respect to x:

$$\frac{dy}{dx} = \beta_1 e^{\beta_0 + \beta_1 x_i + \epsilon_i} \tag{24}$$
$$= \beta_1 y \tag{25}$$

Solving for β_1:

$$\beta_1 = \frac{dy}{dx}\frac{1}{y} \tag{26}$$

The **marginal effect**, $dy/dx = \beta_1 y$, implies that a change in x will impact y differently depending on the level of y. For instance, if $\beta_1 = .001$, and the current value of my home is \$100,000 and I decide to build an extension that increases its area by 200 sqft., then I would expect its change in value from the improvement to be

$$.001 \cdot 100,000 \cdot 200 = 20,000$$

But if my home is currently worth \$200,000, then I expect the improvement to increase it by

$$.001 \cdot 200,000 \cdot 200 = 40,000$$

log-log

For a log-log model (our example for this lession), we can apply the same analysis as the previous two models. Starting from the model equation:

$$log(y_i) = \beta_0 + \beta_1 log(x_i) + \epsilon_i \tag{27}$$
$$y_i = e^{\beta_0 + \beta_1 log(x_i) + \epsilon_i} \tag{28}$$

Take the derivative with respect to x:

$$\frac{dy}{dx} = \beta_1 e^{\beta_0 + \beta_1 x_i + \epsilon_i} \cdot \frac{1}{x_i} \tag{29}$$
$$= \beta_1 \frac{y}{x} \tag{30}$$

Solving for β_1:

$$\beta_1 = \frac{dy}{dx}\frac{x}{y} \tag{31}$$

The **marginal effect**, $dy/dx = \beta_1 y/x$, implies that a change in x will impact y differently depending on the level of y relative to x. For instance, let's say $\beta_1 = .001$, the current value of my home is \$100,000, and its living area is 1000 sqft. Now if I decide to build an extension that increases its area by 200 sqft., then I would expect its change in value from the improvement to be

$$.001 \cdot 100,000/1000 \cdot 200 = 20$$

But if my home is currently worth \$200,000, then I expect the improvement to increase its value by

$$.001 \cdot 200,000/1000 \cdot 200 = 40$$

Since the model we have been using is in log-log form, let's plug its coefficient for living area into my hypothetical home renovation scenerio.

$$0.7063513 \cdot 200,000/1000 \cdot 200 = 14,127.03$$

Simple Linear Regression in R

Thankfully, R has some built-in regression tools–we don't need to write our own each time we want to fit a model. All we need is a data frame and model formula. First, let's create the data frame.

```
# subset some data for a regression
# and make sure both sale_price and living_sqft are log-scaled
data.model<-sales %>%
  filter(sale_price>0) %>%
  left_join(features,by="ssl") %>%
  left_join(select(details,-sale_price,-neighborhood),by="ssl") %>%
  filter(!is.na(living_sqft) & living_sqft>0) %>%
  mutate(log_sale_price=log(sale_price),
         log_living_sqft=log(living_sqft))
```

Next, let's write our model as an R formula. Our model is:

$$log(sale\_price_i) = \beta_0 + \beta_1 log(living\_sqft_i) + \epsilon_i$$

In R, we simply write

$$log\_sale\_price \sim log\_living\_sqft$$

Then we give both pieces of information to **lm()**, R's linear model function.

```
lm(log_sale_price ~ log_living_sqft, data=data.model)
```

```
##
## Call:
## lm(formula = log_sale_price ~ log_living_sqft, data = data.model)
##
## Coefficients:
##     (Intercept)  log_living_sqft
##          7.8752           0.7064
```

Notice that in the formula, we do not need to specify the implicit **1** for β_0. R assumes we want an intercept by default. But we could specify this if we really want a reminder that it is there.

```
lm(log_sale_price ~ 1 + log_living_sqft, data=data.model)

##
## Call:
## lm(formula = log_sale_price ~ 1 + log_living_sqft, data = data.model)
##
## Coefficients:
##     (Intercept)  log_living_sqft
##          7.8752           0.7064
```

And if we wanted a model without an intercept, we could simply subtract **1** from the formula. This forces our fitted line to be drawn through the origin.

```
lm(log_sale_price ~ -1 + log_living_sqft, data=data.model)

##
## Call:
## lm(formula = log_sale_price ~ -1 + log_living_sqft, data = data.model)
##
## Coefficients:
## log_living_sqft
##           1.813
```

Summarizing the regression

The **lm()** function returns an object that contains information about the regression that was run. If we do not assign the object to a variable, it will print the estimated coefficients to the screen. However, it is better to keep our results so we can summarize the regression in a paper, use the estimated coefficients for prediction, or analyze the residuals.

So let's run the previous regression and assign to the variable **fit**.

```
fit<-lm(log_sale_price ~ log_living_sqft, data=data.model)
```

The fitted lm object contains too much information to print here, but you can inspect it with the **str()** function or by clicking on it in your Environment panel in Rstudio. The **summary()** function will print a prettier version of the results to screen, similar to the output in Stata.

```
summary(fit)

##
## Call:
## lm(formula = log_sale_price ~ log_living_sqft, data = data.model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.6189  -0.3601   0.1332   0.4313   6.5902
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     7.875186   0.033900   232.3   <2e-16 ***
## log_living_sqft 0.706351   0.004778   147.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7196 on 85295 degrees of freedom
## Multiple R-squared:  0.204,Adjusted R-squared:  0.204
## F-statistic: 2.186e+04 on 1 and 85295 DF,  p-value: < 2.2e-16
```

And to get latex output of the table, we can use the **stargazer** and print the results straight into our PDF.

```
library(stargazer) # you will need to install this package
stargazer(fit,type="latex",style="qje")
```

Table 1:

	log_sale_price
log_living_sqft	0.706***
	(0.005)
Constant	7.875***
	(0.034)
N	85,297
R^2	0.204
Adjusted R^2	0.204
Residual Std. Error	0.720 (df = 85295)
F Statistic	21,859.330*** (df = 1; 85295)
Notes:	***Significant at the 1 percent level.
	**Significant at the 5 percent level.
	*Significant at the 10 percent level.

Extracting Model Components

As we have seen, the fit object returned by **lm()** contains the information about the regression that created the object. A benefit of holding on to the object is that we can extract components from the model and analyze them and/or compare them to those from other models.

Coefficients (β_k). The betas of the model are stored as a named vector in the fit object and can be extracted by either explicitly referencing them,

```
fit$coefficients

##     (Intercept) log_living_sqft
##       7.8751859       0.7063513
```

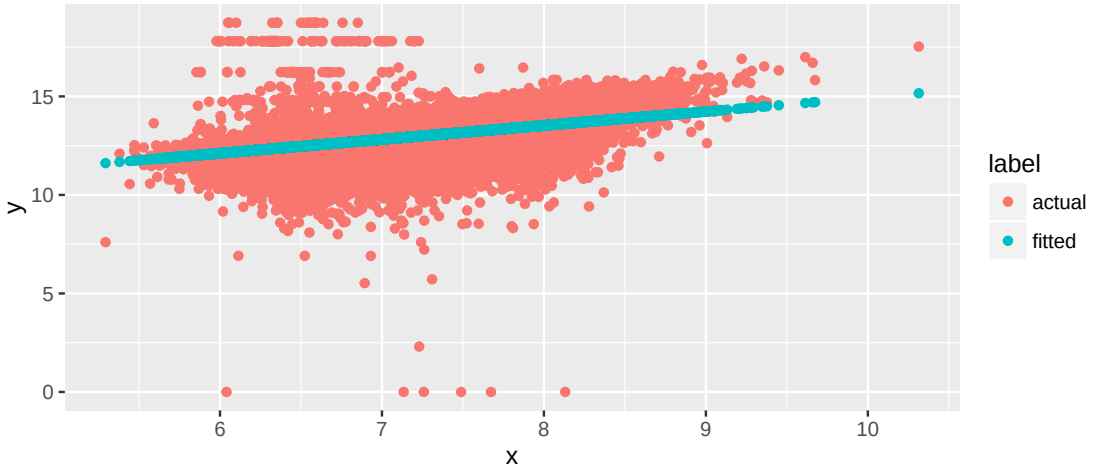
or by applying the **coef()** function to the object.

```
coef(fit)

##     (Intercept) log_living_sqft
##       7.8751859       0.7063513
```

Fitted Values ($\hat{y}$). The fitted values, or simply $\hat{y}$, can be extracted by applying the **fitted()** function to the object.

```
rbind(data.frame(y=data.model$log_sale_price,x=data.model$log_living_sqft,label="actual"),
      data.frame(y=fitted(fit),x=data.model$log_living_sqft,label="fitted")
      ) %>%
  ggplot(aes(x,y,group=label,colour=label)) + geom_point()
```



From our model, $\hat{y}_i = \beta_0 + \beta_1 x_i$ and $\epsilon_i = y_i - \hat{y}_i$. So in the scatterplot above, every $\hat{y}_i$ has a corresponding y_i either directly above or below it. And the residual ϵ_i is the difference between them vertically. We can check this:

```
all.equal(resid(fit),(data.model$log_sale_price - fitted(fit)))

## [1] TRUE
```

Residuals (ϵ_i). In you missed it in the last step, we can also extract the residuals, ϵ_i, by applying the **resid()** function to the fit object.

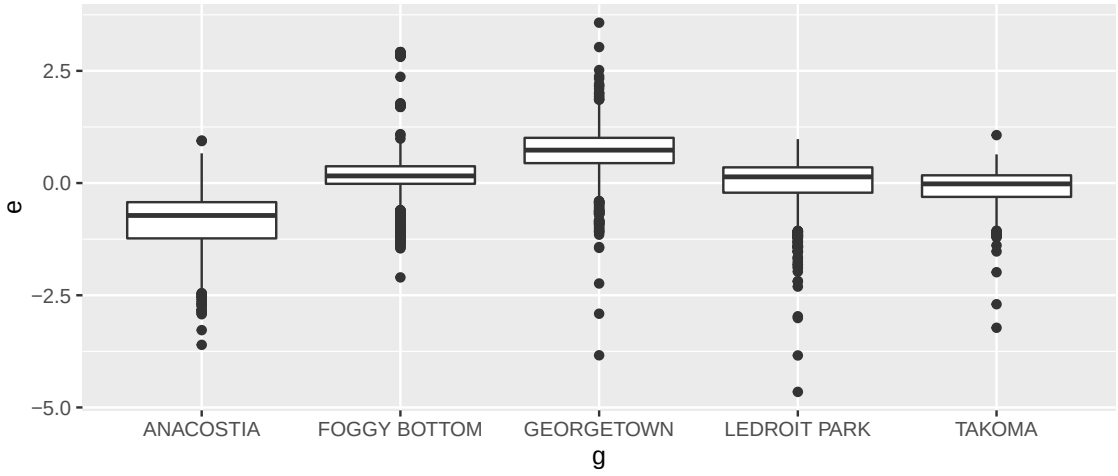
```
resid(fit) %>% summary()

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -13.6200  -0.3601   0.1332   0.0000   0.4313   6.5900
```

Residuals play a key role in our analysis of the goodness-of-fit of our model and will provide motivation for introducing **multiple linear regression** in the next section.

Multiple Linear Regression

```
data.frame(e=resid(fit),g=data.model$neighborhood) %>%
  filter(g %in% c("FOGGY BOTTOM","ANACOSTIA","LEDROIT PARK","TAKOMA","GEORGETOWN")) %>%
  ggplot(aes(y=e,x=g)) + geom_boxplot()
```



1 Appendix

1.1 Closed Form Solution for β_0 and β_1

Let's denote $SSR(\beta_0, \beta_1)$ as our loss function and call it $L(\beta_0, \beta_1)$.

$$L(\beta_0, \beta_1) = \sum_i (y_i - (\beta_0 + \beta_1 x_i))^2 \tag{32}$$

Take the partial derivatives of L with respect to both β_0 and β_1 and set them equal to zero.

$$\frac{\partial L}{\partial \beta_0} = \sum_i 2(y_i - \beta_0 - \beta_1 x_i)(-1) = 0 \tag{33}$$

$$\frac{\partial L}{\partial \beta_1} = \sum_i 2(y_i - \beta_0 - \beta_1 x_i)(-x_i) = 0 \tag{34}$$

We can pull out -2 from the sum and divide them through without changing the meaning of either equation.

$$\sum_i (y_i - \beta_0 - \beta_1 x_i) = 0 \tag{35}$$

$$\sum_i (y_i - \beta_0 - \beta_1 x_i)(x_i) = 0 \tag{36}$$

Distribute x_i through the second equation.

$$\sum_i (y_i - \beta_0 - \beta_1 x_i) = 0 \tag{37}$$

$$\sum_i (y_i x_i - \beta_0 x_i - \beta_1 x_i^2) = 0 \tag{38}$$

Distribute the sums. Notice $\sum_i \beta_0 = \beta_0 \sum_i 1 = n\beta_0$ because we have n observations.

$$\sum_i y_i - n\beta_0 - \beta_1 \sum_i x_i = 0 \tag{39}$$

$$\sum_i y_i x_i - \beta_0 \sum_i x_i - \beta_1 \sum_i x_i^2 = 0 \tag{40}$$

Solve for β_0 in the first equation.

$$\beta_0 = \frac{\sum_i y_i - \beta_1 \sum_i x_i}{n} \tag{41}$$

$$= \frac{\sum_i y_i}{n} - \beta_1 \frac{\sum_i x_i}{n} \tag{42}$$

Plug it into the second equation.

$$\sum_i y_i x_i - (\frac{\sum_i y_i}{n} - \beta_1 \frac{\sum_i x_i}{n}) \sum_i x_i - \beta_1 \sum_i x_i^2 = 0 \tag{43}$$

Solve for β_1.

$$\beta_1 = \frac{\sum_i y_i x_i - \frac{\sum_i y_i \sum_i x_i}{n}}{\sum_i x_i x_i - \frac{\sum_i x_i \sum_i x_i}{n}} \tag{44}$$

$$= \frac{\sum_i y_i x_i - \frac{\sum_i y_i \sum_i x_i}{n}}{\sum_i x_i^2 - \frac{(\sum_i x_i)^2}{n}} \tag{45}$$

Then written together.

$$\beta_1 = \frac{\sum_i y_i x_i - \frac{\sum_i y_i \sum_i x_i}{n}}{\sum_i x_i^2 - \frac{(\sum_i x_i)^2}{n}} \tag{46}$$

$$\beta_0 = \frac{\sum_i y_i}{n} - \beta_1 \frac{\sum_i x_i}{n} \tag{47}$$