Introducing Time

Mike Bader

June 24, 2016

Contents

Meet Time

You might be thinking that you have ended up in the wrong class. After all, you took this class to learn about growth modeling and we have described means and regression of cross-sectional data. Now, we can meet Father Time and introduce him into our framework.

Variation Around a Mean over Time

Write Models and Create Data Using Time

Rather than think about variation across our unit of analysis, now we can think about variation *over time* within the same unit. Let's start with what we have already discussed. Let's say we want to know the average median home value per square foot over the past 12 months. This might help us eliminate the volatility of sales month-over-month and get a sense of our best (relatively naive) guess of prices at any point duing the year. We would simply take the mean:

$$y_t = \mu_x + \epsilon_t$$

You will notice the index, t, marks the time at which we observed prices. In this example there will be twelve values of t since we are estimating the monthly average price over a year. You will also notice that we indexed our outcomes, y_t, and our error, ϵ_t, but not the mean. All of the observations share the same mean. This should look really familiar since we did it before to look at the mean (logged) price across metros.

Let's make up our own New York metro over the past twelve months and suppose that the mean price over those twelve months equaled \$175. Formally writing the model to generate this data, we would write:

$$\ln(price_t) = \ln(175) + \epsilon_t$$

We expect the (logged) median home value in the New York metro in month t to equal ln(\$175) plus (or minus) some error. We expect these errors to have a mean of zero (the definition of the mean) and be normally distributed with a variance of some amount, let's say 0.01, which means that prices fluctuate by one percent from month to month. Just as we did before, we can create a dataset representing this fake reality:

```
set.seed(5831209)
N       <- 12
mu      <- log(175)
sigma   <- 0.01

metro.12mo <- data.frame(t=c(1:N))
metro.12mo$price_t <- rlnorm(N, meanlog=mu, sdlog=sigma)
metro.12mo$lnprice_t <- log(metro.12mo$price_t)
metro.12mo
```

```
##     t  price_t lnprice_t
## 1   1 177.8149  5.180743
## 2   2 175.7523  5.169076
## 3   3 176.5230  5.173451
## 4   4 174.5605  5.162272
## 5   5 176.2093  5.171672
## 6   6 174.3462  5.161043
## 7   7 175.1082  5.165404
## 8   8 176.7124  5.174524
## 9   9 176.7317  5.174633
## 10 10 177.7049  5.180125
## 11 11 175.6499  5.168493
## 12 12 174.2125  5.160276
```

Analyze Our Generated Data

Now let's analyze the data that we just generated by adding a column representing the mean to the data (that will be the same for each month) and also calculate the error by subtracting the mean from the value of $\ln(price)$. We then summarize the residuals for the data:

```
metro.12mo$mu <- mean(metro.12mo$lnprice_t)
metro.12mo$e_t <- metro.12mo$lnprice_t - metro.12mo$mu
metro.12mo
```

```
##     t  price_t lnprice_t       mu          e_t
## 1   1 177.8149  5.180743 5.170143  0.010600530
## 2   2 175.7523  5.169076 5.170143 -0.001066884
## 3   3 176.5230  5.173451 5.170143  0.003308876
## 4   4 174.5605  5.162272 5.170143 -0.007871020
## 5   5 176.2093  5.171672 5.170143  0.001529900
## 6   6 174.3462  5.161043 5.170143 -0.009099718
## 7   7 175.1082  5.165404 5.170143 -0.004738689
## 8   8 176.7124  5.174524 5.170143  0.004381230
## 9   9 176.7317  5.174633 5.170143  0.004490030
## 10 10 177.7049  5.180125 5.170143  0.009981997
## 11 11 175.6499  5.168493 5.170143 -0.001649448
## 12 12 174.2125  5.160276 5.170143 -0.009866803
```

And we see that the expectation of the errors equals zero and the standard deviation equals 0.01, just as we programmed it to.

```r
round(c(mean(metro.12mo$e_t),sd(metro.12mo$e_t)),2)
```

```
## [1] 0.00 0.01
```

Variation around a Trend

Let's think about the model we just constructed. We created a world in which the error in January would be just as likely to be above or below the 12-month mean as the error in June or December. That, however, doesn't make a whole lot of sense. Generally, prices rise or fall over time. Let's say that prices rise over the year. Let us say that prices rose at a pretty consistent level over the entire year. If that is the case, then the January, February, and March prices will be lower more likely to be lower than the overall 12-month mean than the prices in October, November, and December. The prices in those last three months will likely be higher than the overall mean.

Write Models and Create Trend Data

We have described a very simple growth model: we described a process where prices grow over time. A general form of this model would look like the following:

$$\ln\left(price_t\right) = \beta_0 + \beta_1\left(month_t\right) + \epsilon_t$$

The price in month t equals the initial price, β_0 plus the monthly level of growth, β_1 times the number of months since we first started observing prices. If you notice, the equation above is nothing more than a regression equation with *time* being the independent variable that improves our guess of our outcome over simplyl taking the mean.

Let us model this type of process by generating our own data. In a world (cue Don LaFontaine), we imagine that real estate prices across our fake New York metro grew by half a percent monthly. Let's say that the mean we found above represented the mid-year price. We could express this by setting our slope term, β_1, to equal 0.01 (i.e., $\beta_1 = 0.01$).

But what would our intercept be? Let us say that the mean that we found before represented the mid-year price on a consistent linear trend. We would subtract six months of growth from the average value. That is our intercept (β_0) equals:

$$\ln(price_{jan}) = \ln(price_{jul}) - \beta_1 \times 6 = \ln(\$175) - 0.01 \times 6 = 5.10$$

That means that we can generate fake New York City real estate price trends with the equation:

$$\ln\left(price_t\right) = 5.10 + 0.01 \times month_t + \epsilon_t$$

And we will assume that prices bounce around that monthly trend line with a standard deviation of about 0.5% (i.e., $\sigma = 0.005$) and create these data from our model (after first renaming our old variables so we don't get confused and subtracting one from each value of `t` since we want to index our first month to 0, not 1):

```r
suppressMessages(library(dplyr))
metro.12mo$t <- metro.12mo$t - 1 #Indexes the first month to zero rather than one
cols <- which(names(metro.12mo) %in% c('price_t','lnprice_t','e_t','mu'))
names(metro.12mo)[cols] <- c('price_t.old','lnprice_t.old','e_t.old','mu.old')

beta_0 <- 5.10
beta_1 <- 0.01
```

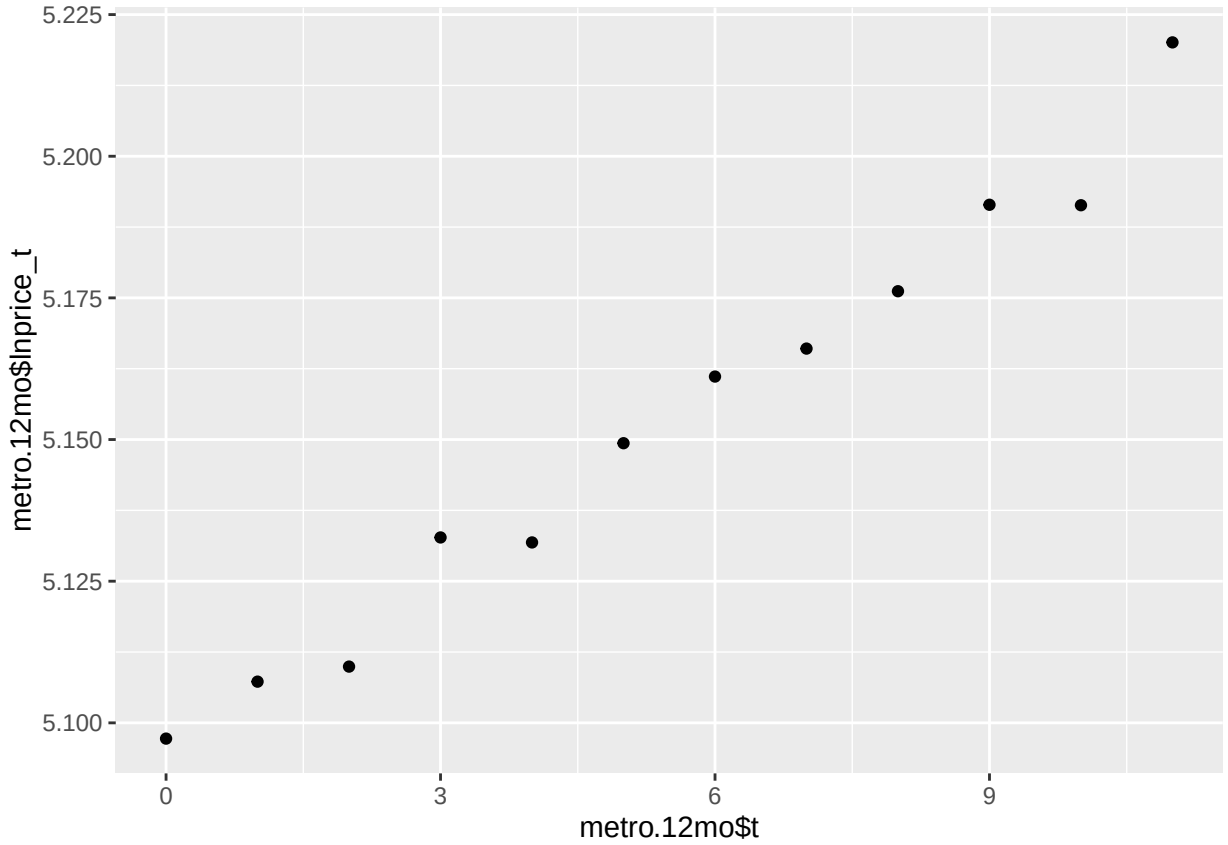
```
sigma  <- 0.005

metro.12mo <- mutate(metro.12mo
                     , lnprice_t = beta_0 + beta_1*t + rnorm(12,mean=0,sd=sigma)
                     , price_t = exp(lnprice_t))
```

And now we can look at what the data in our fake New York metro look like:

```
metro.12mo
```

```
##     t price_t.old lnprice_t.old  e_t.old       mu.old lnprice_t  price_t
## 1   0    177.8149      5.180743 5.170143  0.010600530  5.097222 163.5668
## 2   1    175.7523      5.169076 5.170143 -0.001066884  5.107273 165.2191
## 3   2    176.5230      5.173451 5.170143  0.003308876  5.109929 165.6586
## 4   3    174.5605      5.162272 5.170143 -0.007871020  5.132720 169.4775
## 5   4    176.2093      5.171672 5.170143  0.001529900  5.131841 169.3286
## 6   5    174.3462      5.161043 5.170143 -0.009099718  5.149355 172.3203
## 7   6    175.1082      5.165404 5.170143 -0.004738689  5.161116 174.3590
## 8   7    176.7124      5.174524 5.170143  0.004381230  5.166052 175.2217
## 9   8    176.7317      5.174633 5.170143  0.004490030  5.176175 177.0045
## 10  9    177.7049      5.180125 5.170143  0.009981997  5.191436 179.7264
## 11 10    175.6499      5.168493 5.170143 -0.001649448  5.191353 179.7115
## 12 11    174.2125      5.160276 5.170143 -0.009866803  5.220095 184.9517
```

```
qplot(metro.12mo$t, metro.12mo$lnprice_t)
```



Analyze Fake Price Trend Data

Now we can estimate the data generating process by using a regression with time as our only predictor:

```
fake.trend.model <- lm(lnprice_t ~ t,data=metro.12mo)
summary(fake.trend.model)
```

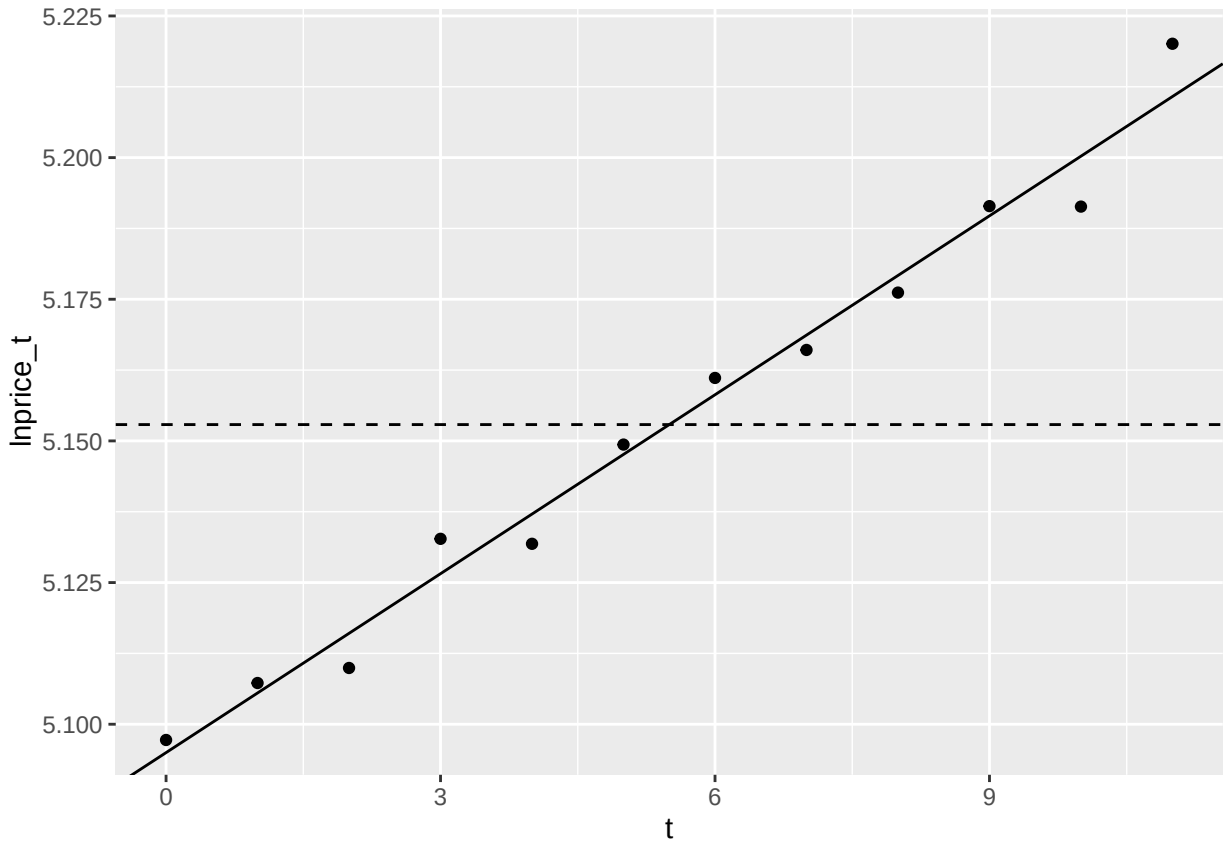
```
##
## Call:
## lm(formula = lnprice_t ~ t, data = metro.12mo)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.008896 -0.003579  0.001725  0.002420  0.009319
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.0949855  0.0030119  1691.6  < 2e-16 ***
## t           0.0105264  0.0004638    22.7 6.22e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005547 on 10 degrees of freedom
## Multiple R-squared:  0.981,  Adjusted R-squared:  0.979
## F-statistic:   515 on 1 and 10 DF,  p-value: 6.217e-10
```

And we find that our model correctly estimated the data generating process (which, it should, we know *exactly* how the data were generated). But let's take a closer look at a couple of things about the model. First, let's plot the trend line and the overall mean[1]:

```
## TO DO: Rescale axis
coef <- fake.trend.model$coefficients
plt <- qplot(t, lnprice_t,data=metro.12mo)
plt +  geom_abline(intercept=coef[1],slope=coef[2]) +
       geom_hline(yintercept=mean(metro.12mo$lnprice_t),lty=2)
```

[1] Commands without using `ggplot2`:

```
plot(metro.12mo$t, metro.12mo$lnprice_t)
abline(coef)
abline(h=mean(metro.12mo$lnprice_t),lty=2)
```



Now let's take a look at the residuals of the model. In addition to creating the residual from the trend (which I will call `ehat`), I also want to create a column showing the residual from the overall mean of the twelve months of data.

```
metro.12mo <- mutate(metro.12mo
                     , beta_0 = coef[1]
                     , beta_1 = coef[2]
                     , pred.lnprice_t = beta_0 + beta_1*t
                     , ehat = lnprice_t - pred.lnprice_t
                     , mu = mean(lnprice_t)
                     , e_mu = lnprice_t - mean(lnprice_t)
)
metro.12mo[,c('t','beta_0','beta_1','ehat','mu','e_mu')]
```

```
##      t   beta_0     beta_1         ehat       mu         e_mu
## 1    0 5.094986 0.01052636  0.002236133 5.152881 -0.055658841
## 2    1 5.094986 0.01052636  0.001760796 5.152881 -0.045607819
## 3    2 5.094986 0.01052636 -0.006109274 5.152881 -0.042951531
## 4    3 5.094986 0.01052636  0.006155823 5.152881 -0.020160074
## 5    4 5.094986 0.01052636 -0.005250036 5.152881 -0.021039574
## 6    5 5.094986 0.01052636  0.001737334 5.152881 -0.003525845
## 7    6 5.094986 0.01052636  0.002972717 5.152881  0.008235897
## 8    7 5.094986 0.01052636 -0.002617864 5.152881  0.013171675
## 9    8 5.094986 0.01052636 -0.003021371 5.152881  0.023294527
## 10   9 5.094986 0.01052636  0.001712969 5.152881  0.038555226
## 11  10 5.094986 0.01052636 -0.008896257 5.152881  0.038472358
## 12  11 5.094986 0.01052636  0.009319028 5.152881  0.067214002
```

```
round(c(mean(metro.12mo$ehat),sd(metro.12mo$ehat)),3)
```

```
## [1] 0.000 0.005
```

```
round(c(mean(metro.12mo$e_mu),sd(metro.12mo$e_mu)),3)
```

```
## [1] 0.000 0.038
```

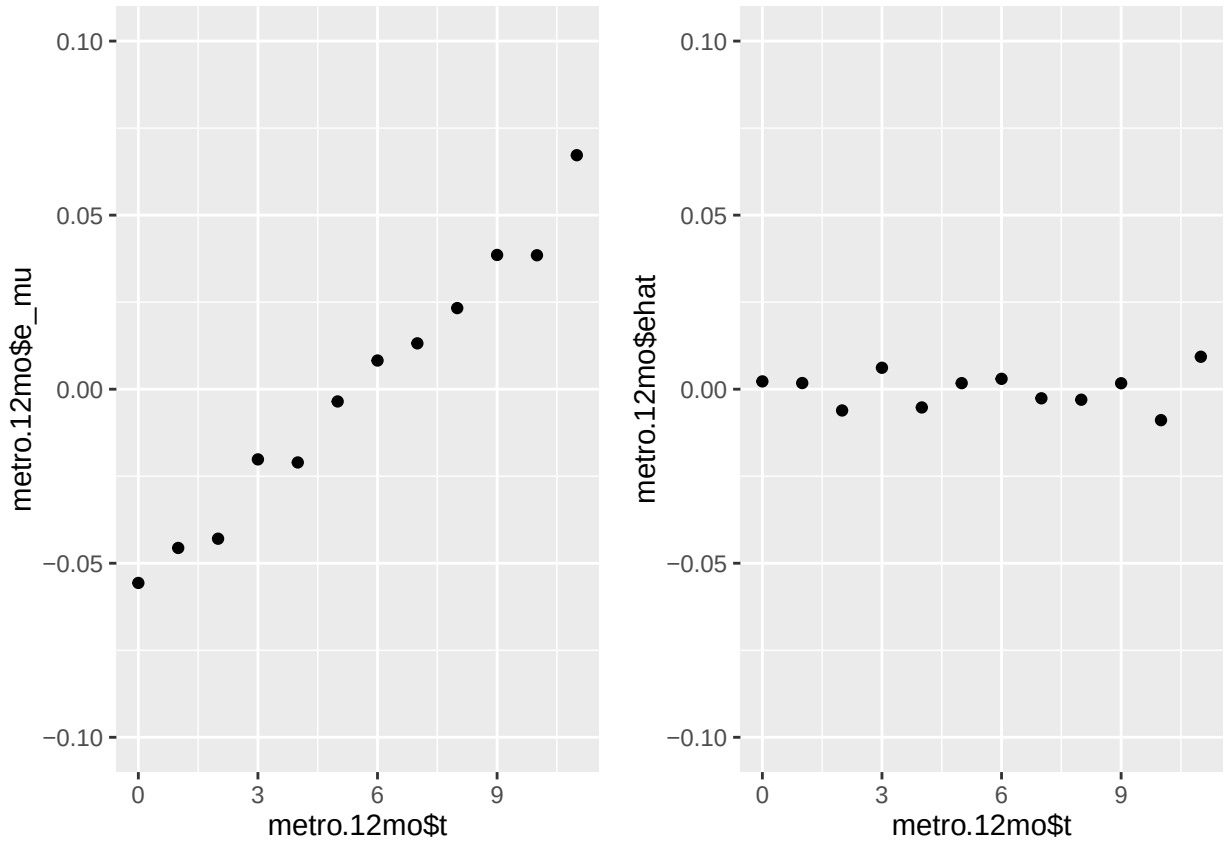
You can see the data structure of this trend model when we print the data above. All of the units (which are months, remember) have the same values of `beta_0` and `beta_1`. The errors of the trend model are distributed around a mean of zero and a standard deviation of 0.0053, just as we programmed above.

Comparing Analysis of the Mean to Analysis of the Trend

Now let's take a look at how we improve over our naive estimate of the mean. The mean, as we constructed it to be, came out to be just a little less than 5.17 (which is the natural logarithm of $175). The mean of the residuals away from `mu` (the 12-month mean) away from this mean equals zero. As we would expect, the standard deviation is higher as well because our estimates are not as good as using the trend data, 0.0383 compared to 0.0053 from above. We can use this data to see how much variance we can explain by including the trend in our model: $1 - \frac{\sigma^2_{trend}}{\sigma^2_{\mu}} = 0.98$. Accounting for the trend reduced the residuals off of our best estimate by 98% (this would not happen in cases in real analysis; remember we made this world so that the only variation in it was time).

More importantly, however, the residuals also correlate with time:

```
p1 <- qplot(metro.12mo$t,metro.12mo$e_mu,ylim=c(-.1,.1))
p2 <- qplot(metro.12mo$t,metro.12mo$ehat,ylim=c(-.1,.1))
plot_grid(p1,p2,ncol=2)
```




```
cor(metro.12mo$t,metro.12mo$e_mu)
```

```
## [1] 0.9904313
```

You *do not* want to see obvious patterns in residuals. Remember that one of the assumptions of regression is that your residuals should be heterskedastic, meaning that they vary randomly. This pattern obviously violates that pattern. The plot on the left shows the residuals away from the mean while the one on the right shows the residuals away form the trend. By not accounting for time, we do not explain the pattern evident in this world very well. Of course, we know that because we generated the world represented by these dots. Most of the time we don't know the process that generated the data and do not know the *omitted variables* that might lurk.

Specialness of Time

Due to the nature of time, the values β_0 and β_1 take on special meanings in a regression using time as a predictor. Since β_0 equals the value when the x value equals zero, this takes on a special meaning when time is our x value: *it is the outcome when we mark the beginning of time.* Time is also special because it can only march forward. No matter how much you might wish to do so, you cannot go back in time. Time represents a unidirectional relationship. Because β_1 represents the difference in the outcome for a one unit change in the x value and that change must go in only one direction, when we use time as our x variable β_1 also represents something special: *it is the growth rate (or rate of decline).*

Centering Time

Remember how we figured out the value we wanted the intercept to be based on subtracting growth from the overall average that we assigned to happen in our fake New York metropolitan area? Well, just as we

centered our across-metro intercept at the price of real estate in the New York metro to help us interpret the data, we can do the same with time. As you can see from how we entered the data, for all R cares time is just represents another number. We could set t to equal zero in July by subtracting six.[2] Let's see what result that gives us:

```
fake.trend.model.centered <- lm(lnprice_t ~ I(t-6),data=metro.12mo)
summary(fake.trend.model.centered)
```

```
##
## Call:
## lm(formula = lnprice_t ~ I(t - 6), data = metro.12mo)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.008896 -0.003579  0.001725  0.002420  0.009319
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.1581437  0.0016179  3188.3  < 2e-16 ***
## I(t - 6)    0.0105264  0.0004638    22.7 6.22e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005547 on 10 degrees of freedom
## Multiple R-squared:  0.981,  Adjusted R-squared:  0.979
## F-statistic:   515 on 1 and 10 DF,  p-value: 6.217e-10
```

The value of the intercept changed because we moved the y-axis six months into the future. Rather than t equalling zero in January, it now equals -6. The value of the intercept, β_0, equals the mean price, μ that we calculated (i.e., $\beta_0 = \mu_x = 5.16$. Just as before, this value represents the **conditional grand mean** of the data. But, because of the specialness of time, the interpretation takes on a special meaning which is the value at the mid-point of the trend you measure. Once again, it's helpful to think about what the value of the intercept *means* and make sure that it's something interpretable.

EXERCISE: I want you to write out a model of linear change for a process in which you are interested in exploring. Be sure to specify the unit you believe will be changing over time. You should decide a) what you think that initial level will be, b) how you will measure time, c) what you think the rate of change will be, and d) how much variation you think that there will be off of the trend for each unit.

Then, using the code above, I want you to simulate the model that you wrote out. You should be able to follow step-for-step from the code above.

Variation Around the Real Home Value Trend

For this analysis, we are going to examine the trend of the Zillow median home value index per square foot from January to December 2015. This will exactly mimic the fake data that we generated above in structure.

Gather 2015 New York Metro Home Value Data to Analyze

Let's first grab the data and plot what it looks like.

[2] Remember the months are indexed starting at zero, so $t = 0$ in January, $t = 1$ in February, and so on. Therefore $t = 6$ in July.

```
price.sq.ft.url  # Assign the following value to this variable to download the data
```

```
## [1] "../Data/Metro_MedianValuePerSqft_AllHomes.csv"
```

```
                # directly from Zillow:
                # 'http://files.zillowstatic.com/research/public/Metro/Metro_MedianValuePerSqft_AllHom
zillow <- read.csv(price.sq.ft.url,header=TRUE)
varnames.2015 <- paste0("X2015.",sprintf("%02.0f",1:12))
ny15 <- zillow[zillow$RegionName=="New York, NY",
            c("RegionID","RegionName",varnames.2015)
            ]
ny15
```

```
##   RegionID   RegionName X2015.01 X2015.02 X2015.03 X2015.04 X2015.05
## 1   394913 New York, NY      229      230      231      232      231
##   X2015.06 X2015.07 X2015.08 X2015.09 X2015.10 X2015.11 X2015.12
## 1      231      231      231      231      232      233      234
```

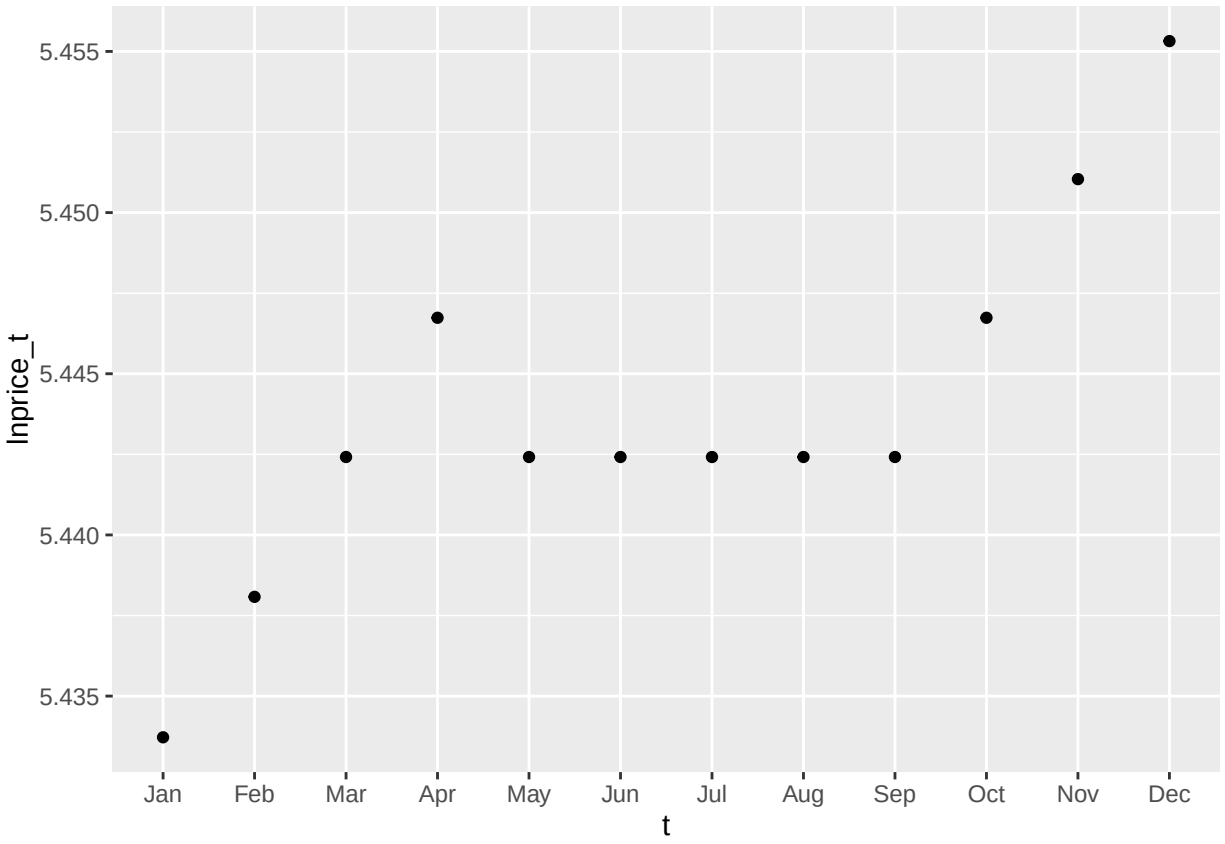
Uh oh! The data don't look like what we had before. We have a column for every month rather than a row for every month. Let's try to fix that:

```
ny15.long <- reshape(ny15,idvar="RegionID",timevar="t",direction="long",varying=varnames.2015)
ny15.long$t <- ny15.long$t - 1  # Indexes time to equal zero in January
ny15.long$price_t <- ny15.long$X2015
ny15.long
```

```
##           RegionID   RegionName  t X2015 price_t
## 394913.1    394913 New York, NY  0   229     229
## 394913.2    394913 New York, NY  1   230     230
## 394913.3    394913 New York, NY  2   231     231
## 394913.4    394913 New York, NY  3   232     232
## 394913.5    394913 New York, NY  4   231     231
## 394913.6    394913 New York, NY  5   231     231
## 394913.7    394913 New York, NY  6   231     231
## 394913.8    394913 New York, NY  7   231     231
## 394913.9    394913 New York, NY  8   231     231
## 394913.10   394913 New York, NY  9   232     232
## 394913.11   394913 New York, NY 10   233     233
## 394913.12   394913 New York, NY 11   234     234
```

That looks more like what we're used to! Now, let's take the logarithm of price and plot the data:

```
ny15.long$lnprice_t <- log(ny15.long$price_t)
ny15.plt <- qplot(t,lnprice_t,data=ny15.long) +
            scale_x_continuous(breaks=ny15.long$t,labels=month.abb,minor_breaks=NULL)
ny15.plt
```



The data look roughly linear, acknowledging that some patterns might emerge which otherwise might not given a relatively small sample of twelve.

Analyze New York Metro Home Value Trend in 2015

Since the data do not appear to be wildly non-linear, let's proceed to analyzing the linear trend. We will go ahead and run a linear model and measure the error based on the coefficients. Let's center our results to be the conditional average (logged) price per square foot over the entire year:

```
ny15.trend.model <- lm(lnprice_t ~ I(t-5.5), data=ny15.long)
summary(ny15.trend.model)
```

```
##
## Call:
## lm(formula = lnprice_t ~ I(t - 5.5), data = ny15.long)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0045993 -0.0023326 -0.0004248  0.0018662  0.0060641
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 5.4438452  0.0009913 5491.421   <2e-16 ***
## I(t - 5.5)  0.0012687  0.0002872    4.418   0.0013 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.003434 on 10 degrees of freedom
## Multiple R-squared:  0.6612, Adjusted R-squared:  0.6274
## F-statistic: 19.52 on 1 and 10 DF,  p-value: 0.001298
```

```
coef <- ny15.trend.model$coefficients
```

Interpret New York Metro Home Value Trend in 2015

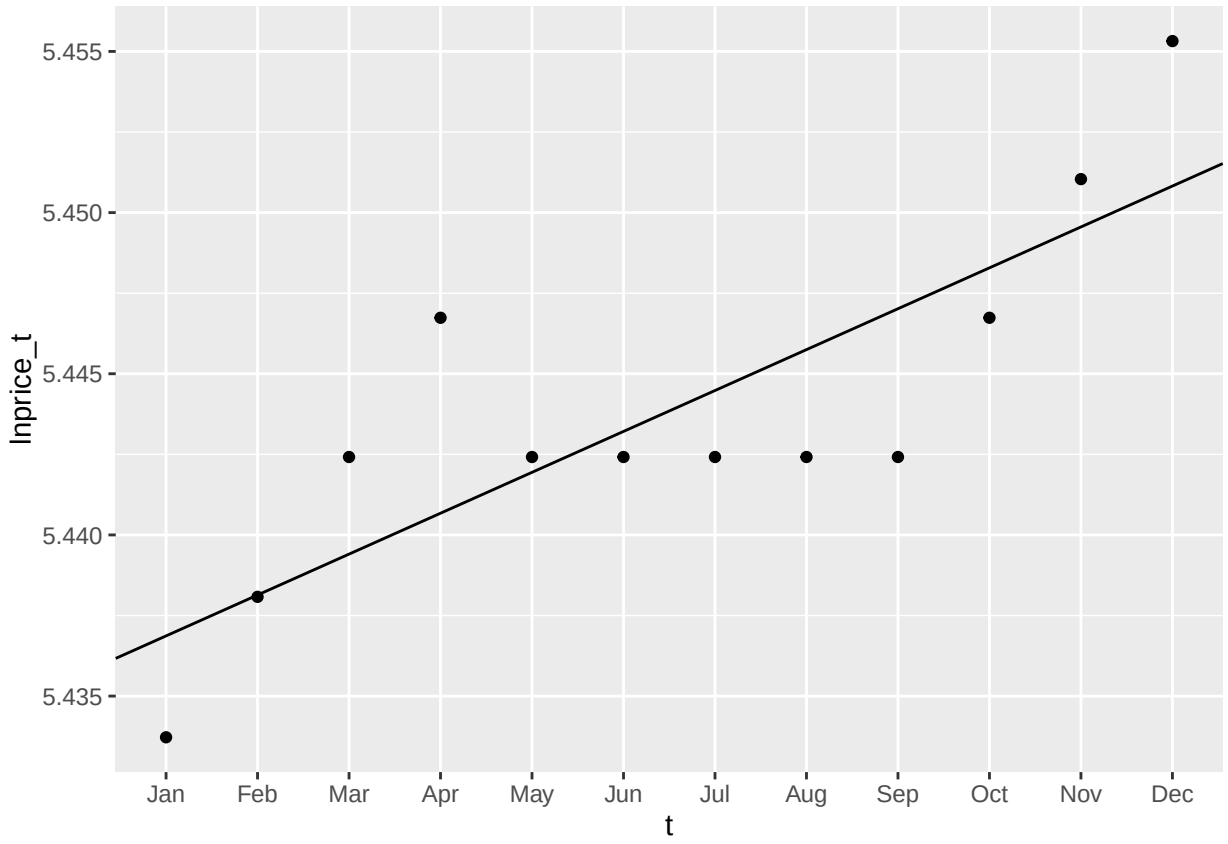
The results indicate that the conditional average of the median home value during the course of 2015 was $e^{\beta_0} = e^{5.44} = \$231.33$; and, indeed, when we take the mean value of the monthly observations `exp(mean(ny15.long$lnprice_t))` we get... drumroll, please... $231.33!

The estimate of β_1 indicates that median home values grew by 0.13% per month. There are 11 segments of time over which prices could grow so that, with compounding, would would estimate that prices grew -101% over 2015. The initial price, the price predicted in January, equals e raised to the intercept minus 5.5 times the rate of change: $e^{\beta_0 - \beta_1 \times 5.5} = e^{5.44 - 0.0013 \times 5.5} \approx \$229.72$. The final price that we predicted equals e raised to the sum of natural logarithm of 229.72 and 11 times the rate of change: $e^{5.44 + 0.0013 \times 11} = \$233$.

```
ny15.long <- mutate(ny15.long
                , beta_0 = coef[1]
                , beta_1 = coef[2]
                , pred.lnprice_t = beta_0 + beta_1*t
                , ehat = lnprice_t - pred.lnprice_t
                )
round(c(mean(ny15.long$ehat),sd(ny15.long$ehat)),4)
```

```
## [1] -0.0070  0.0033
```

```
ny15.plt + geom_abline(intercept=coef[1]-coef[2]*5.5,slope=coef[2])
```




```
round(ny15.long[,-1:-3],3)  # Subsetting removes character columns
```

```
##    X2015 price_t lnprice_t beta_0 beta_1 pred.lnprice_t   ehat
## 1    229     229     5.434  5.444  0.001          5.444 -0.010
## 2    230     230     5.438  5.444  0.001          5.445 -0.007
## 3    231     231     5.442  5.444  0.001          5.446 -0.004
## 4    232     232     5.447  5.444  0.001          5.448 -0.001
## 5    231     231     5.442  5.444  0.001          5.449 -0.007
## 6    231     231     5.442  5.444  0.001          5.450 -0.008
## 7    231     231     5.442  5.444  0.001          5.451 -0.009
## 8    231     231     5.442  5.444  0.001          5.453 -0.010
## 9    231     231     5.442  5.444  0.001          5.454 -0.012
## 10   232     232     5.447  5.444  0.001          5.455 -0.009
## 11   233     233     5.451  5.444  0.001          5.457 -0.005
## 12   234     234     5.455  5.444  0.001          5.458 -0.002
```

We estimated the standard deviation of the errors to have been 0.003, meaning that the month-to-month price flucuated off of the trend by about 0.3% in any given month. I printed the dataset of change so you can see what the data look like and then plotted the predicted trend over the scatterplot.

And that's it! We just ran a real model of metropolitan real estate value growth over an entire year. Not too bad!!!