Common Transformations

Irfan Kanat

July 4, 2017

In this learning activity we will learn about when and why transforming data may be justified as well as some common transformations.

Going back to our discussion of normal distribution, mean and standard deviation in Descriptive Statistics learning activity, sometimes the distribution of variables, or the sheer scale (number of rooms 1-10, versus the price of a house 100000-3000000) makes comparisons difficult. Especially if you want to compare the effect sizes in statistical models some sort of transformation is useful.

Another reason for transformation is that the variable is not normally distributed and the broad variance may obfuscate an otherwise legitimate relation.

Introducing the Data

You may remember the **ZRI Summary: Multifamily, SFR, Condo/Co-op (Current Month)** dataset from Assignment 1. I will use the same dataset for demonstration purposes.

```
zillow <- read.csv("data/zillow.csv")
View(zillow)
```

```
## Warning in View(zillow): X cannot set locale modifiers
```

Another dataset we will use today is SAT and ACT scores from the psych package (you do know how to obtain and activate packages).

```
data(sat.act)
?sat.act
```

Transformations for Non-Normal Data

The methods discussed below will bring your data closer to a normal distribution. But you should be aware of the caveat. The coefficients estimated in models with transformed variables do not mean the same thing. So take heed in presenting results of transformed data. What needs to be done depends on the type of transformation.

Right Skewed Data

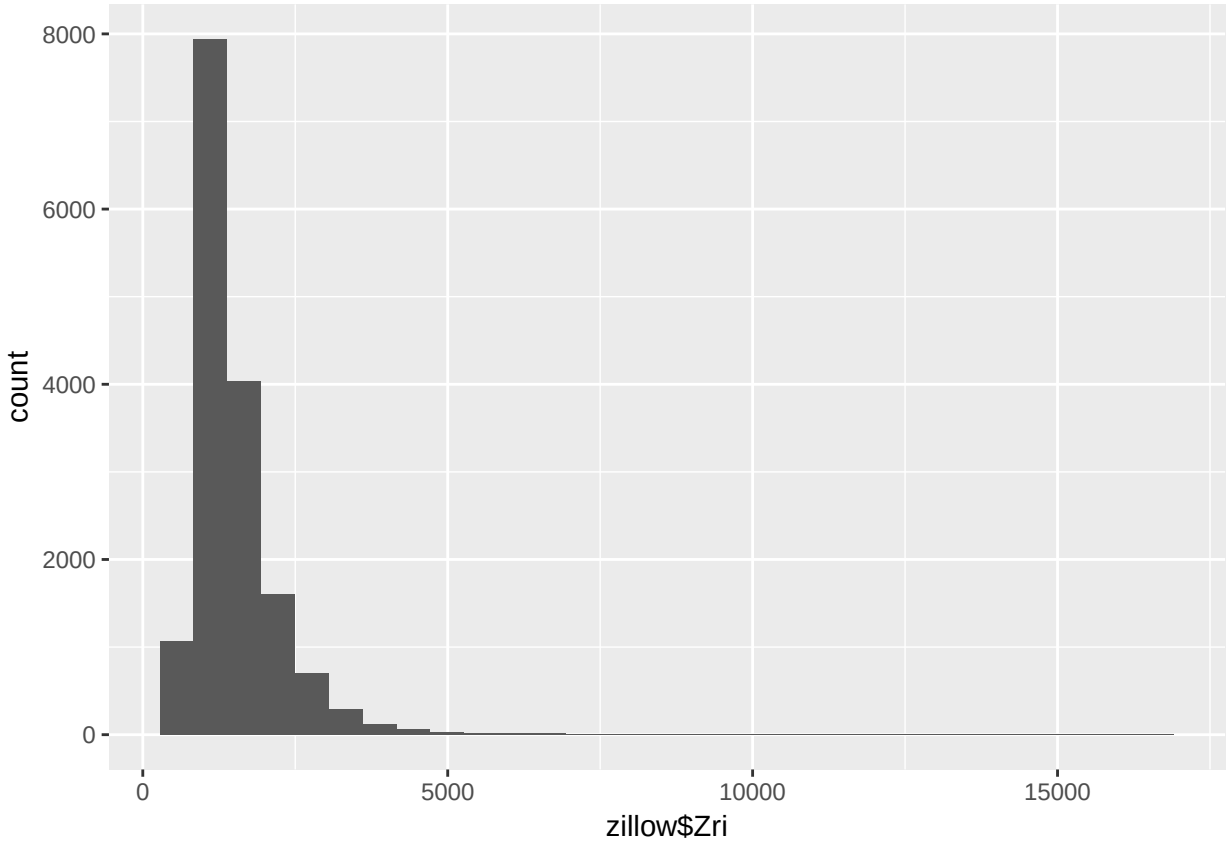
Prices, wages, wealth or other monetary figures often follow a power law distribution. This means you will have lots of observations on the left hand side of the scale and a handful of very large amounts on the right hand side. To help you visualize imagine how many people live on minimum wage, that is the left hand side. Now imagine how many people earn over 5 million dollars, that is the far right end.

If your goal is prediction the independent variable not being normally distributed is not an issue at all, but in some cases an otherwise significant relation may show up non-significant. Hence it is a good idea to transform these variables (or take the outliers out somehow).

Let us look at the distribution of zillow rent index. This is not a perfect example, yet it is sufficiently similar for our purposes. By now you know how qplot works. I will take some shortcuts and skip a few optional parameters.

```
qplot(zillow$Zri)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Most places in the country have rents that are below $2000, so you see high, easy to see bars on the left. There is virtually 0 free homes, so the x axis starts somewhere around $500. What you don't see is why the right hand side of the plot keeps going up to $17000.

There are two counties in California where the rents are above $15000. That push the boundary way out into $16600 range.

Here is a little exercise for you. Using the filtering skills we reviewed in Descriptive Statistics learning activity, find out which counties have rents indexes higher than $15000.

Log Transformation

A common transformation used in powerlaw distribution is log transformations. You basically take the natural logarithm of the variable. To understand why it would work, observe the examples below.

```
log(500)   # logarithm of $500 is 6.2
```

```
## [1] 6.214608
```

```
log(1500)  # $1500 is 7.3
```

```
## [1] 7.31322
```

```
log(15000) # $15000 is 9.6
```

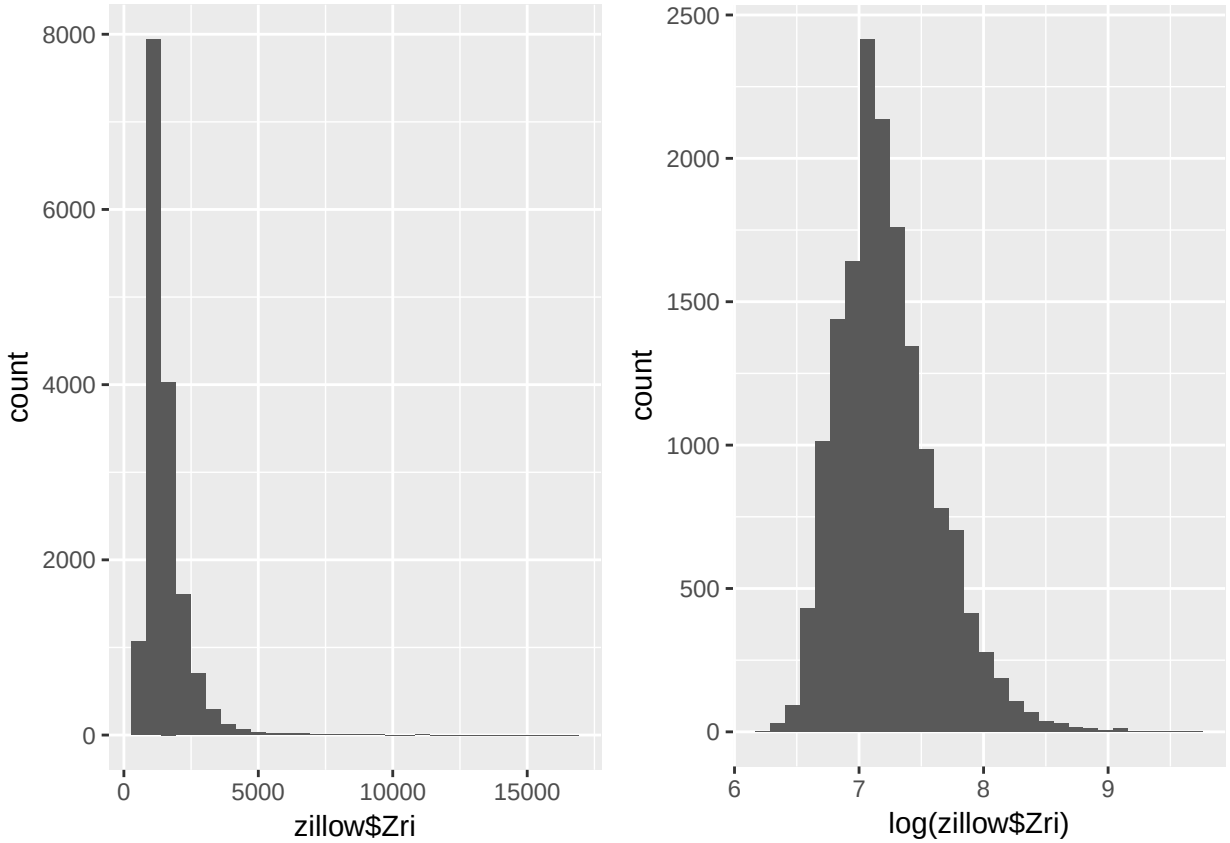
```
## [1] 9.615805
```

The log transformations shrinks the higher values more than it does the lower values, hence pulling those extreme observations back into the fold. In a sense, it gets rid of outliers without sacrificing data points.

Let us see what happens in a visual example.

```
p1 <- qplot(zillow$Zri)
p2 <- qplot(log(zillow$Zri))
grid.arrange(p1, p2, ncol = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Note the differences on scale in both x and y axes. While not perfect, log transformed zillow rent index looks a lot more normal compared to untransformed version.

You can always use log() function in models, but if you want to store transformed variable you can do it as follows:

```
# Create a new column and store transformed values in this column.
zillow$logZri <- log(zillow$Zri)
head(zillow[, c("logZri", "Zri")])
```

```
##     logZri  Zri
## 1 8.170469 3535
## 2 7.536364 1875
## 3 8.236685 3777
## 4 7.652546 2106
## 5 6.899723  992
## 6 8.194506 3621
```

When data is transformed this way, you need to take heed in interpreting coefficients. The beta coefficient no longer stands for a 1 unit change in ZRI.

Another issue you should be aware of is that log transformation only works with positive values.

```
log(0)
```

```
## [1] -Inf
```

```
log(-5)
```

```
## Warning in log(-5): NaNs produced
```

```
## [1] NaN
```

If your data has zeros you can simply add 1 to all observations, for negative values cube rooting instead of log transformation may help.

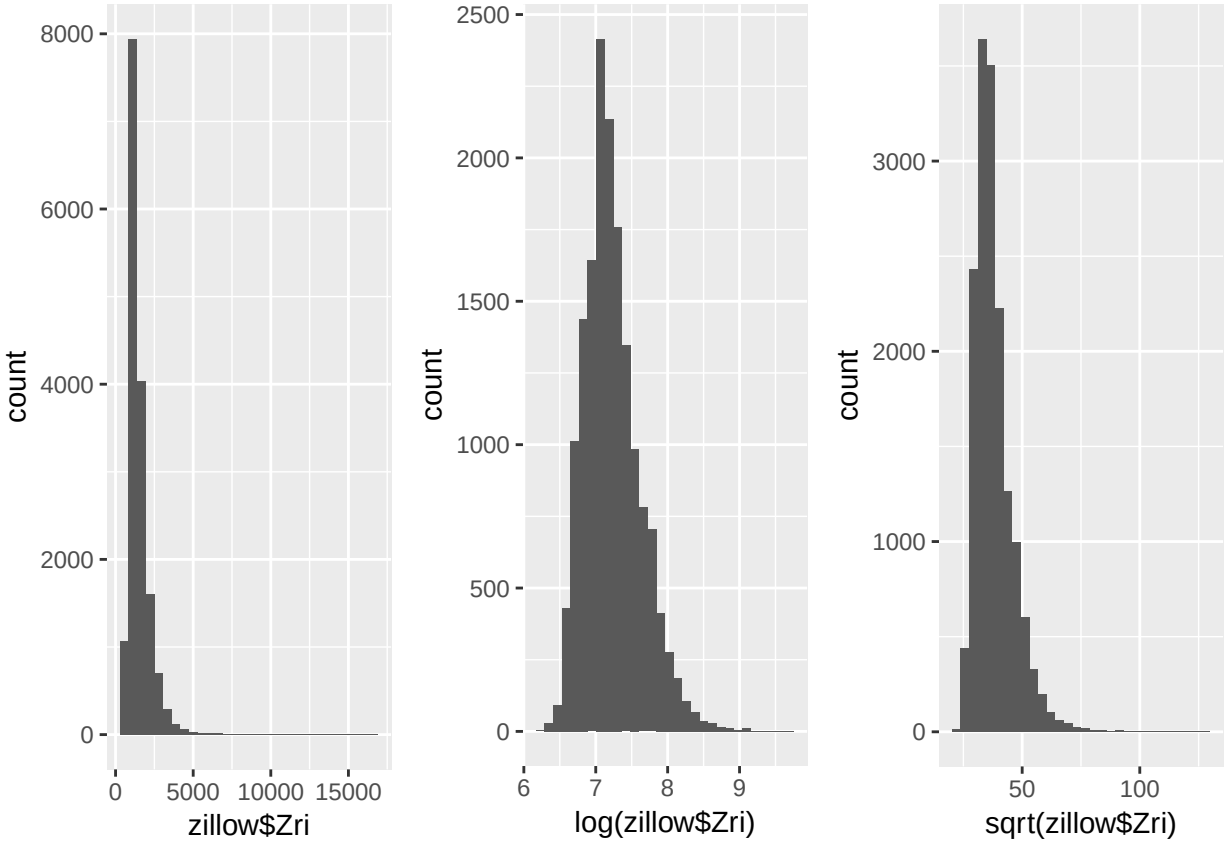
Square Root Transformation

An alternative to log transformation is taking square root of the variable. It works the same way as log transformation, in the sense that it shrinks the larger values more, hence brings them closer to the rest of the data. Depending on your data a square root may provide better results (in this case it doesn't).

Let us see visually.

```
p3 <- qplot(sqrt(zillow$Zri))
grid.arrange(p1, p2, p3, ncol = 3)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Again, square root transformation does not work with negative values, but it works with 0s.

```
sqrt(0)
```

```
## [1] 0
```

```
sqrt(-5)
```

```
## Warning in sqrt(-5): NaNs produced
```

```
## [1] NaN
```

Here is a simple exercise for you, calculate square root transformed ZRI and store it in a new column called sqrtZri.

Left Skewed Data

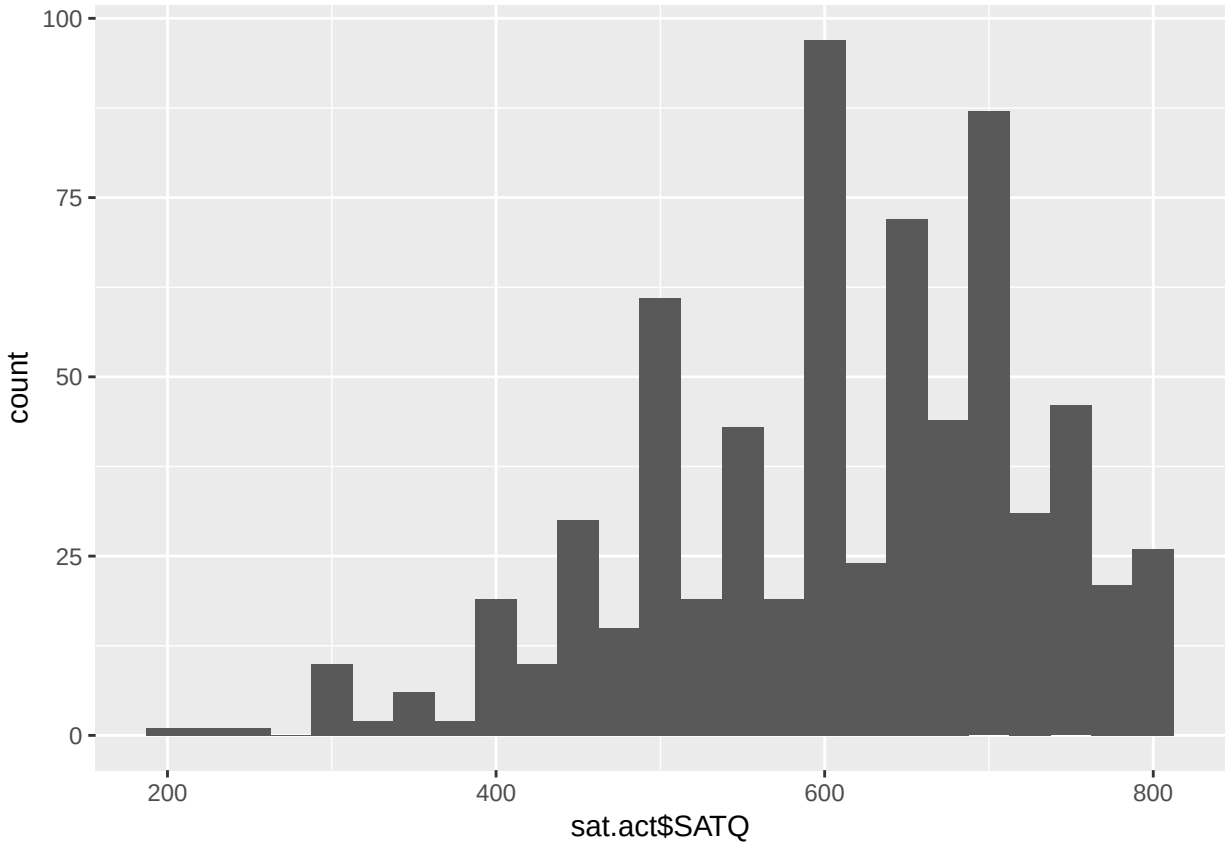
Some values such as aptitude scores are left skewed, with majority of the observations on the right hand side and just a few observations on the left.

SAT scores are traditionally of this kind.

Let us see:

```
qplot(sat.act$SATQ, binwidth = 25)
```

```
## Warning: Removed 13 rows containing non-finite values (stat_bin).
```

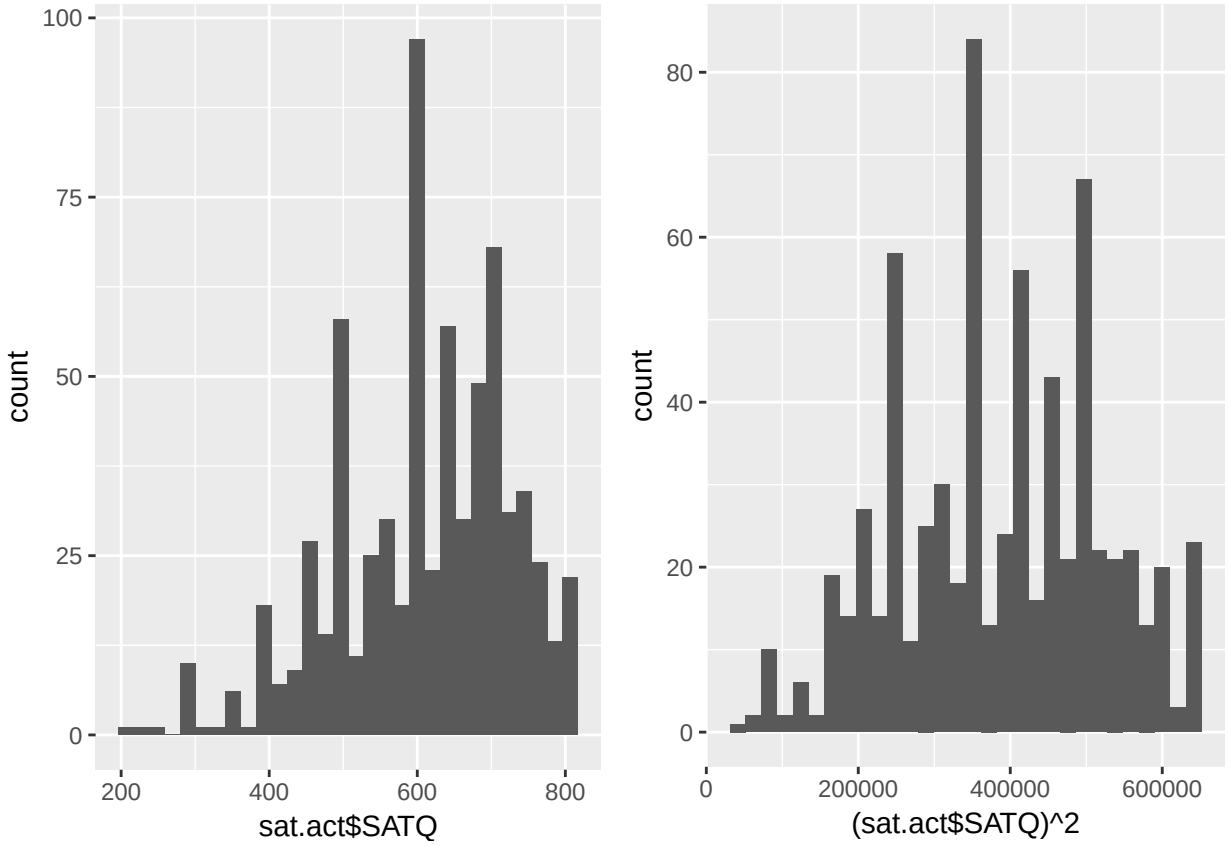


Square Transformation

If your data is left skewed taking the square of the values may help a bit. Let us see.

```
p1 <- qplot(sat.act$SATQ)
p2 <- qplot((sat.act$SATQ)^2)
options(scipen = 99)
grid.arrange(p1, p2, ncol = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 13 rows containing non-finite values (stat_bin).
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 13 rows containing non-finite values (stat_bin).
```



You can see the transformed values look a *bit* more normal. You should not expect miracles.

Here is another exercise, square transform SATV scores and store them in a new column called SATV2.

Box-Cox Transformation

A more advanced version of log transformation is box-cox transformation. It finds the exact base to transform the data, so will generally produce distributions closer to normal. Another advantage is that it will work with both left and right skewed data. I have never seen a big enough difference between common transformations and box-cox transformations, so I will spare you the details.

Normalization / Standardization

There may be certain cases where you may want to normalize data. Certain types of analysis (Principal Component Analysis) would be sensitive to differences in scale for example. Another case would be to

compare coefficients. The coefficients would mean one thing for a variable that ranges between 1000000 and 5000000, and another for a variable that ranges between 1 and 5.

Standardization in our case is substracting the mean value from each observation and dividing the remainder by sd. The result is the variable will have a mean of 0 and a standard deviation of 1. If all variables are standardized, the coefficients will mean exactly the same thing for each variable.

Let us standardize the Zillow Rent Index in zillow dataset and store it in a variable called stdZri.

```
zillow$stdZri <- scale(zillow$Zri)
# Let us verify that the data is indeed standardized
mean(zillow$stdZri) # Looks good
```

```
## [1] 0.00000000000000005954604
```

```
sd(zillow$stdZri) # Looks good
```

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

```
# Let us see how the results look compared to non transformed values.
head(zillow[, c("Zri", "stdZri")])
```

```
##    Zri     stdZri
## 1 3535  2.5934850
## 2 1875  0.4681786
## 3 3777  2.9033189
## 4 2106  0.7639291
## 5  992 -0.6623307
## 6 3621  2.7035913
```

Remember the mean Zri is 1509. So values higher than the mean Zri are positive whereas values lower than Zri are negative.

Solution to Exercises

1. Find out which counties have rents indexes higher than $15000.

```
zillow[zillow$Zri > 15000, ]
```

```
##             Date RegionName State                          Metro
## 4911  2017-05-31      90210    CA Los Angeles-Long Beach-Anaheim
## 10828 2017-05-31      94027    CA                  San Francisco
##            County          City SizeRank   Zri          MoM          QoQ
## 4911  Los Angeles Beverly Hills     4910 15736 -0.003987594 -0.012797992
## 10828   San Mateo      Atherton    10827 16599  0.005025430  0.006793231
##                YoY ZriRecordCnt   logZri   stdZri
## 4911  -0.04520357         8415 9.663706 18.21449
## 10828 -0.05928025         2670 9.717098 19.31939
```

2. Calculate square root transformed ZRI and store it in a new column called sqrtZri.

```
zillow$sqrtZri <- sqrt(zillow$Zri)
```

3. Square transform SATV scores and store them in a new column called SATV2.

```
sat.act$SATV2 <- (sat.act$SATV)^2
```