Introduction to Data Science: Data Transformations

Héctor Corrada Bravo

University of Maryland, College Park, USA

2020-03-11

EDA: Data Transformations

How is data distributed?

- visual EDA
- quantitative summaries

EDA: Data Transformations

How is data distributed?

- visual EDA
- quantitative summaries

Now consider transformations of attributes:

- help interpretation of data analyses
- help application statistical and machine learning models

Centering and scaling

A very common and important transformation is to scale data to a common unit-less scale.

Transforming variables from whatever units they are measured (e.g., diamond depth percentage)

into "standard deviations away from the mean" units (*standard units*, or z -score).

Centering and scaling

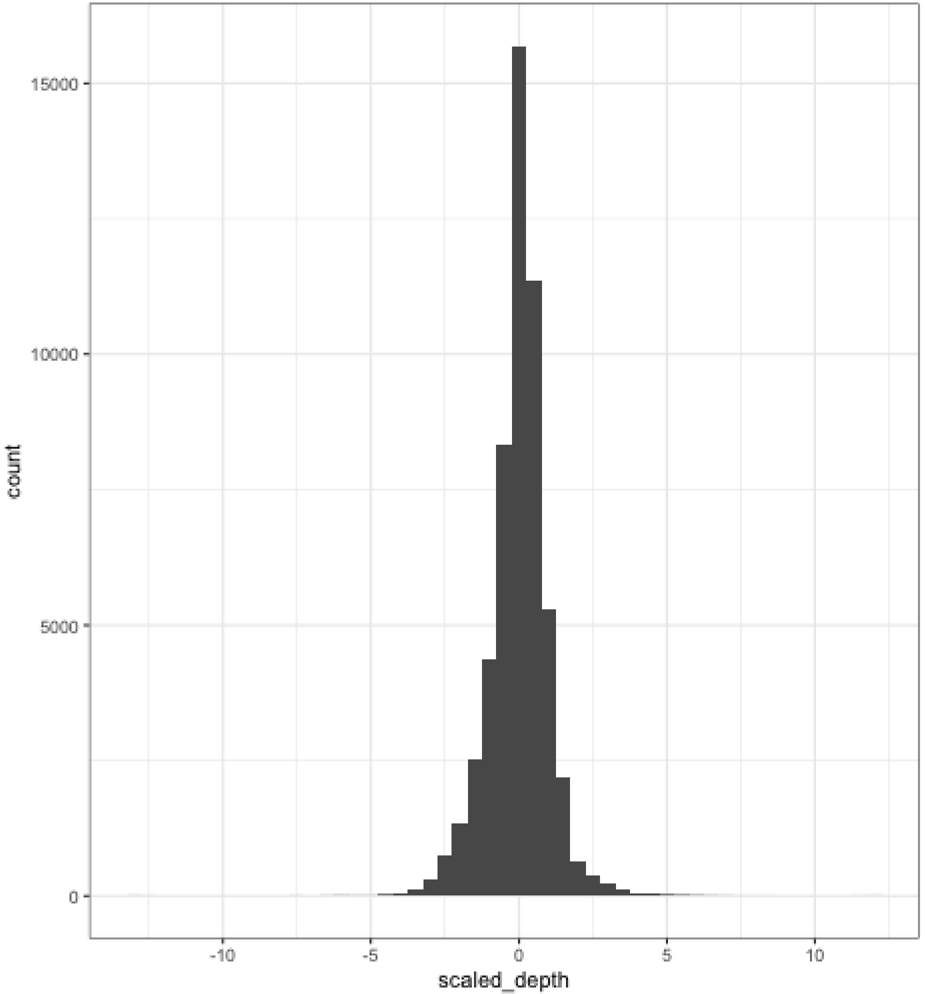
Given data $x = x_1, x_2, \ldots, x_n$, the transformation applied to obtain centered and scaled variable z is:

$$z_i = \frac{(x_i - \overline{x})}{\mathrm{sd}(x)}$$

where $\overline{x}$ is the mean of data x, and $\mathrm{sd}(x)$ is its standard deviation.

Centering and scaling

```
diamonds %>%
  mutate(scaled_depth = (depth - mean(depth)
  ggplot(aes(x=scaled_depth)) +
    geom_histogram(binwidth=.5)
```



Centering and scaling

Another name for this transformation is to *standardize* a variable.

Centering and scaling

Another name for this transformation is to *standardize* a variable.

After transformation:

- all variables in a dataset are in the same, and thus comparable, units.
- all variables have the same mean and variance

Centering and scaling

Another name for this transformation is to *standardize* a variable.

After transformation:

- all variables in a dataset are in the same, and thus comparable, units.
- all variables have the same mean and variance

This is very helpful for *multivariate* statistical and ML analyses

Centering and scaling

On occasion, you will have use to apply transformations that only *center* (but not scale) data:

$$z_i = (x_i - \overline{x})$$

Centering and scaling

Or, apply transformations that only *scale* (but not center) data:

$$z_i = \frac{x_i}{\mathrm{sd}(x)}$$

Treating categorical variables as numeric

Many modeling algorithms work strictly on numeric measurements.

For example:

- linear regression, or
- support vector machines

Strictly defined for numeric measurements.

Treating categorical variables as numeric

Many modeling algorithms work strictly on numeric measurements.

For example:

- linear regression, or
- support vector machines

Strictly defined for numeric measurements.

In this case, need to transform categorical variables into something that we can treat as numeric.

Treating categorical variables as numeric

Let's see a couple of important guidelines for *binary* variables:

categorical variables that only take two values, e.g.

- `health_insurance` Yes/No
- `cat_picture` Yes/No

Treating categorical variables as numeric

One option is to encode one value of the variable as 1 and the other as 0. For instance:

```
Wage %>%
  mutate(numeric_insurance = ifelse(health_ins == "1. Yes", 1, 0)) %>%
  select(year, age, health_ins, numeric_insurance) %>%
  head()
```

```
##   year age health_ins numeric_insurance
## 1 2006  18      2. No                 0
## 2 2004  24      2. No                 0
## 3 2003  45     1. Yes                 1
```

Treating categorical variables as numeric

Another option is to encode one value as 1 and the other as -1:

```
Wage %>%
  mutate(numeric_insurance = ifelse(health_ins == "1. Yes", 1, -1)) %>%
  select(year, age, health_ins, numeric_insurance) %>%
  head()
```

```
##   year age health_ins numeric_insurance
## 1 2006  18      2. No                -1
## 2 2004  24      2. No                -1
## 3 2003  45     1. Yes                 1
## 4 2003  43     1. Yes                 1
```

Treating categorical variables as numeric

The decision of which of these two transformations to use is based on the method to use or the goal of your analysis.

Treating categorical variables as numeric

The decision of which of these two transformations to use is based on the method to use or the goal of your analysis.

E.g, predict `wage` based on `health insurance status` (coded as 0/1) let's us make statements like: "on average, wage increases by `$XX` if a person has health insurance".

Treating categorical variables as numeric

The decision of which of these two transformations to use is based on the method to use or the goal of your analysis.

E.g, predict `wage` based on `health insurance status` (coded as 0/1) let's us make statements like: "on average, wage increases by $XX if a person has health insurance".

But, to predict `health insurance status` based on other attributes, a Support Vector Machine requires `health insurance status` to be coded as +1/-1

Treating categorical variables as numeric

For categorical attributes with more than two values, we extend this idea and encode *each* value of the categorical variable as a 0/1 column.

You will see this referred to as *one-hot-encoding*.

Treating categorical variables as numeric

```
Wage %>%
  mutate(race_white = ifelse(race == "1. White", 1, 0),
         race_black = ifelse(race == "2. Black", 1, 0),
         race_asian = ifelse(race == "3. Asian", 1, 0),
         race_other = ifelse(race == "4. Other", 1, 0)) %>%
  select(starts_with("race")) %>%
  sample_n(5)
```

```
##       race race_white race_black race_asian race_other
## 1 2. Black          0          1          0          0
## 2 1. White          1          0          0          0
## 3 2. Black          0          1          0          0
```

Discretizing continuous values.

How about transforming data in the other direction, from continuous to discrete values.

This can make it easier to compare differences related to continuous measurements:

Do doctors prescribe a certain medication to older kids more often? Is there a difference in wage based on age?

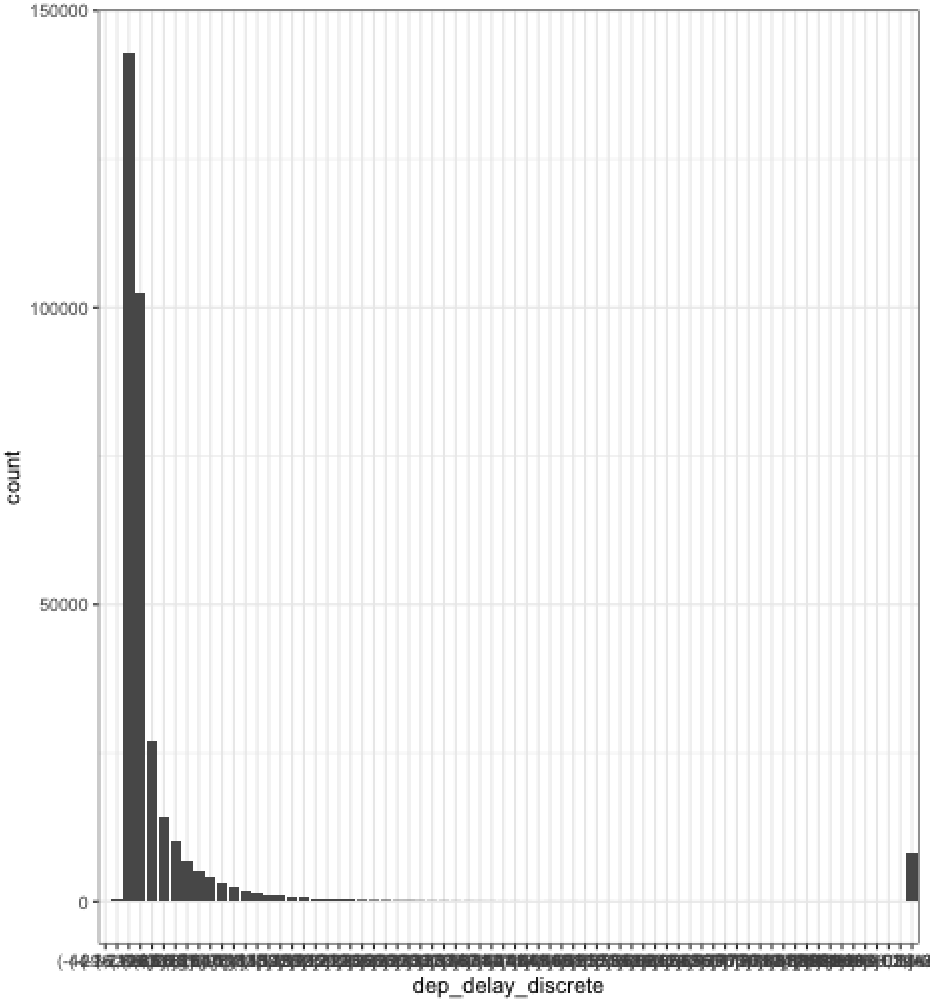
Discretizing continuous values.

It is also a useful way of capturing non-linear relationships in data: we will see this in our regression and prediction unit.

Two standard methods used for discretization are to use **equal-length** bins, where variable range is divided into bins *regardless* of the data distribution.

Discretizing continuous values.

```
flights %>%
  mutate(dep_delay_discrete = cut(dep_delay,
  ggplot(aes(x=dep_delay_discrete)) +
  geom_bar()
```

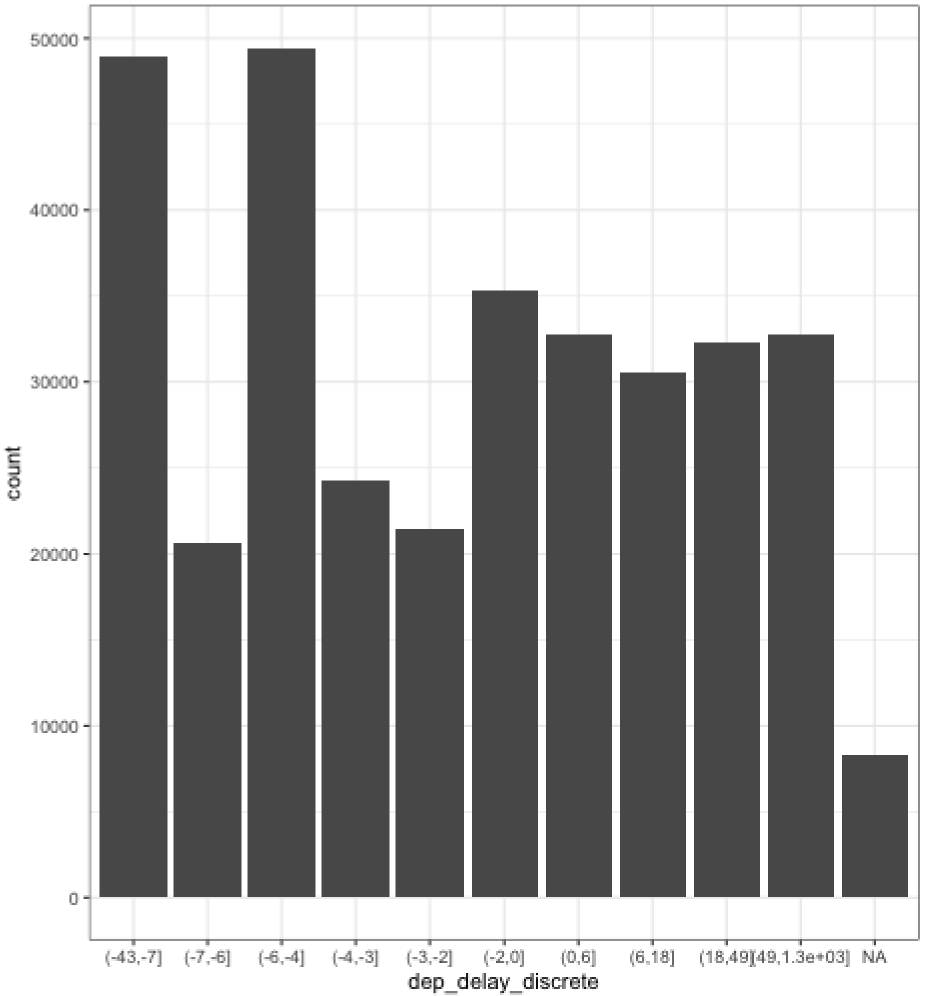


Discretizing continuous values.

The second approach uses **equal-sized** bins, where the range is divided into bins *based* on data distribution

Discretizing continuous values.

```
flights %>%
  mutate(dep_delay_discrete = cut(dep_delay,
          breaks=quantile(dep_delay, probs=s
  ggplot(aes(x=dep_delay_discrete)) +
  geom_bar()
```



Skewed Data

In many data analysis, variables will have a *skewed* distribution over their range.

In the last section we saw one way of defining skew using quartiles and median.

Variables with skewed distributions can be hard to incorporate into some modeling procedures, especially in the presence of other variables that are not skewed.

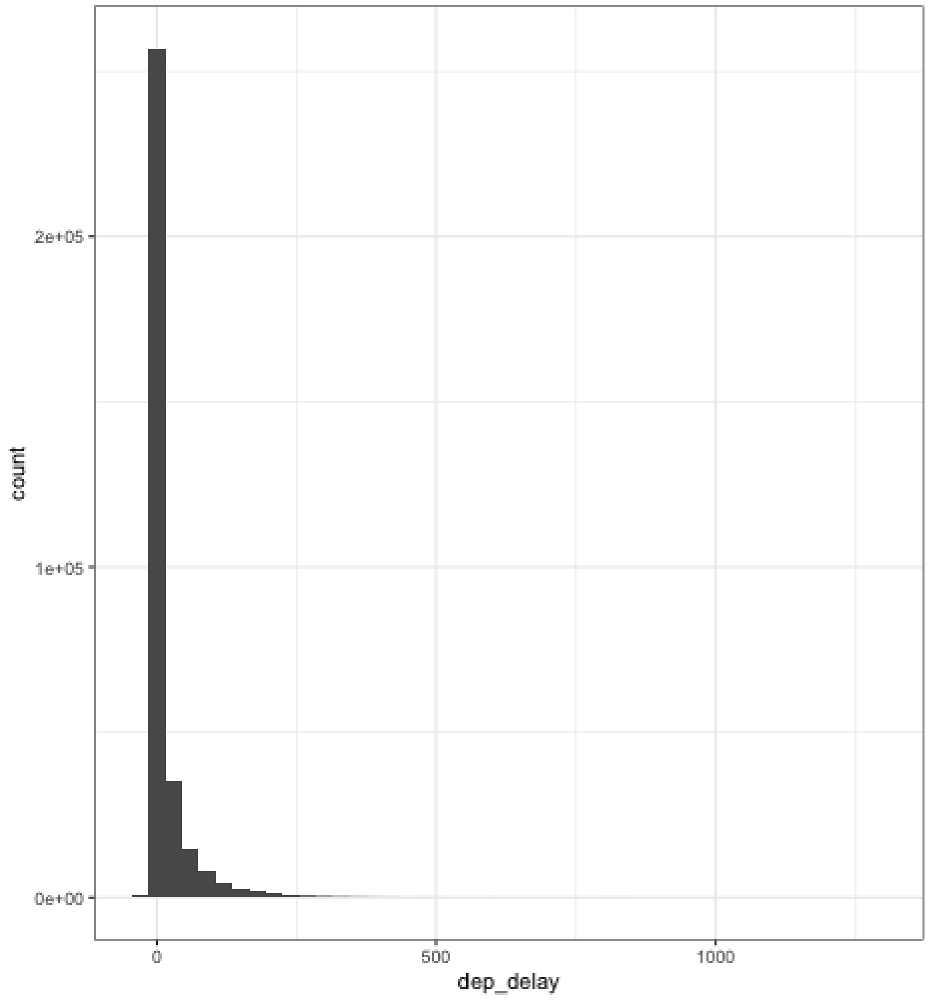
Skewed Data

Skewed data may arise when measuring *multiplicative* processes. In this case, interpretation of data may be more intiuitive after a transformation.

We have seen an example of skewed data previously when we looked at departure delays in our flights dataset.

Skewed Data

```
flights %>% ggplot(aes(x=dep_delay)) +
  geom_histogram(binwidth=30)
```



Skewed Data

In many cases a logarithmic transform is an appropriate transformation to reduce data skew:

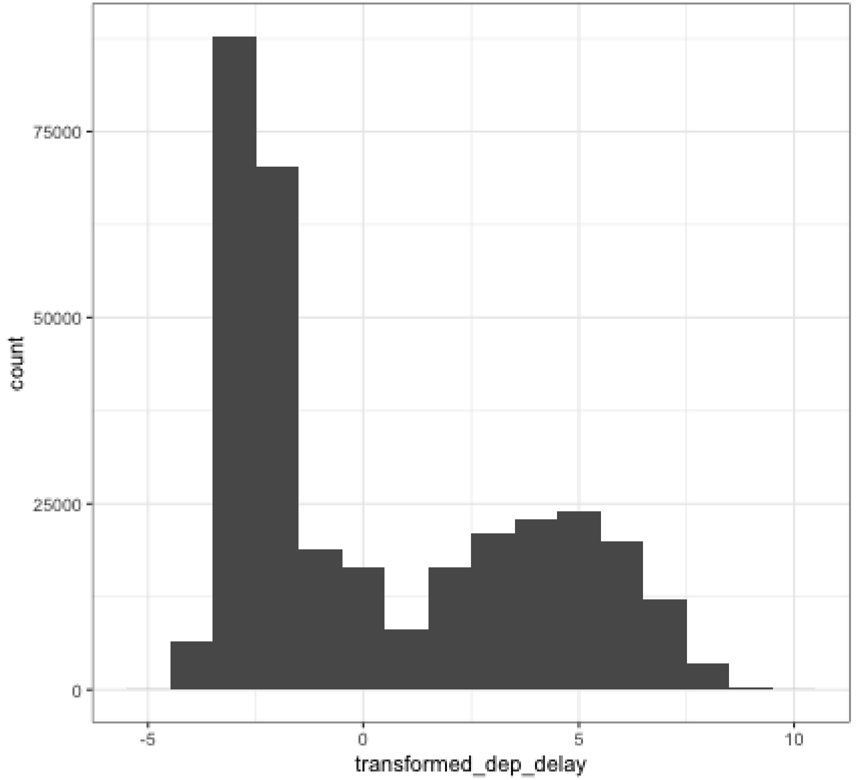
- If values are all positive: apply `log2` transform
- If some values are negative, two options
  - Started Log: shift all values so they are positive, apply `log2`
  - Signed Log: $sign(x) \times log2(abs(x) + 1)$.

Skewed Data

Here is a signed log transformation of departure delay data:

```
transformed_flights <- flights %>%
  mutate(transformed_dep_delay = sign(dep_de

transformed_flights %>%
  ggplot(aes(x=transformed_dep_delay)) +
    geom_histogram(binwidth=1)
```



Summary

Given what we learn from EDA (visually and statistically), we can guide decisions on data transformations

- Change data types continuous <-> numeric
- Standardization
- Log-transforms (reduce skew, also variance stabilization)