ggplot basics

Oleksii Yehorchenkov

2023-11-28

```
library(tidyverse)
library(ggrepel) # repel overlapping text labels
```

```
## Warning: package 'ggrepel' was built under R version 4.3.2
```

This tutorial is based on R graphics with ggplot2 workshop notes (https://rpubs.com/phela51/471711)

Points (scatterplot)

Now that we know about geometric objects and aesthetic mapping, we can make a `ggplot()`. `geom_point()` requires mappings for x and y, all others are optional.

Example data: housing prices

Let’s look at housing prices.

```
housing <- read_csv("./data/landdata-states.csv")
```

```
## Rows: 7803 Columns: 11
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): State, region
## dbl (9): Date, Home_Value, Structure_Cost, Land_Value, Land_Share_Pct, Home_...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(housing[1:5]) # view first 5 columns
```

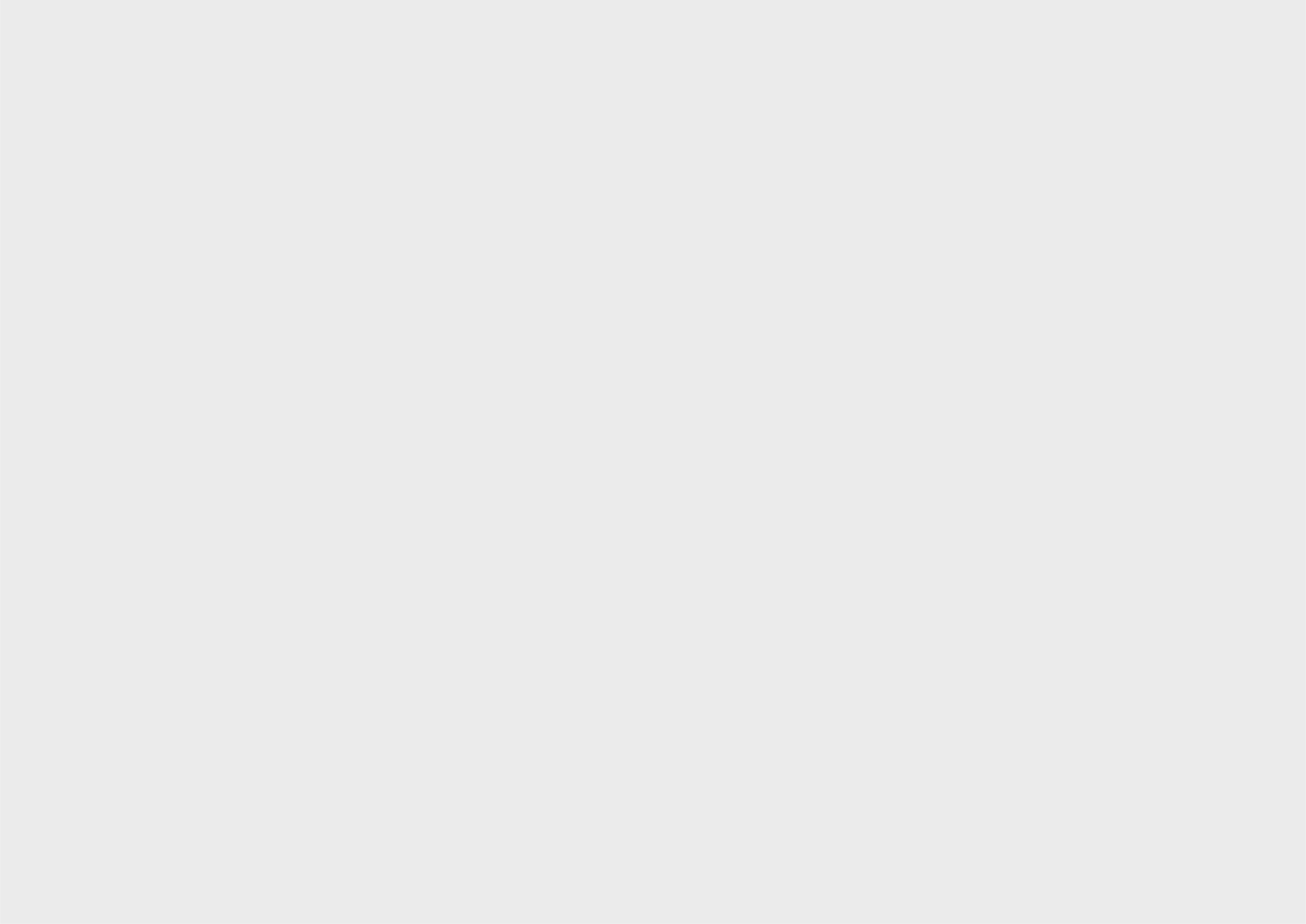
```
## # A tibble: 6 × 5
##   State region  Date Home_Value Structure_Cost
##   <chr> <chr>  <dbl>      <dbl>          <dbl>
## 1 AK    West   2010.     224952         160599
## 2 AK    West   2010.     225511         160252
## 3 AK    West   2010.     225820         163791
## 4 AK    West   2010      224994         161787
## 5 AK    West   2008      234590         155400
## 6 AK    West   2008.     233714         157458
```

create a subset for 1st quarter 2001

```
hp2001Q1 <- housing |> filter(Date == 2001.25)
```

Step 1: create a blank canvas by specifying data:

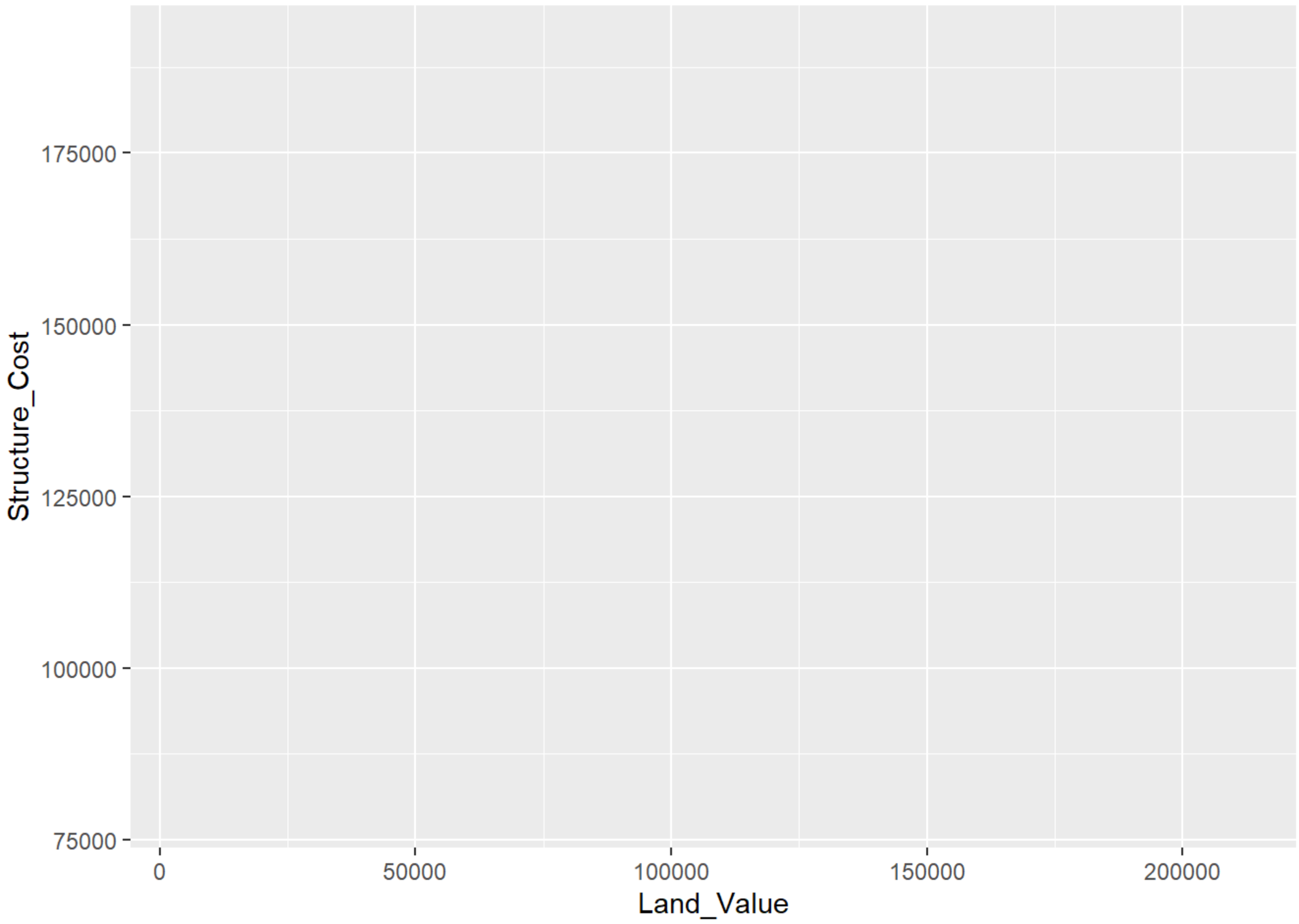
```
ggplot(data = hp2001Q1)
```



Step 2: specify aesthetic mappings (how you want to map variables to visual aspects):

here we map `Land_Value` and `Structure_Cost` to the x- and y-axes.

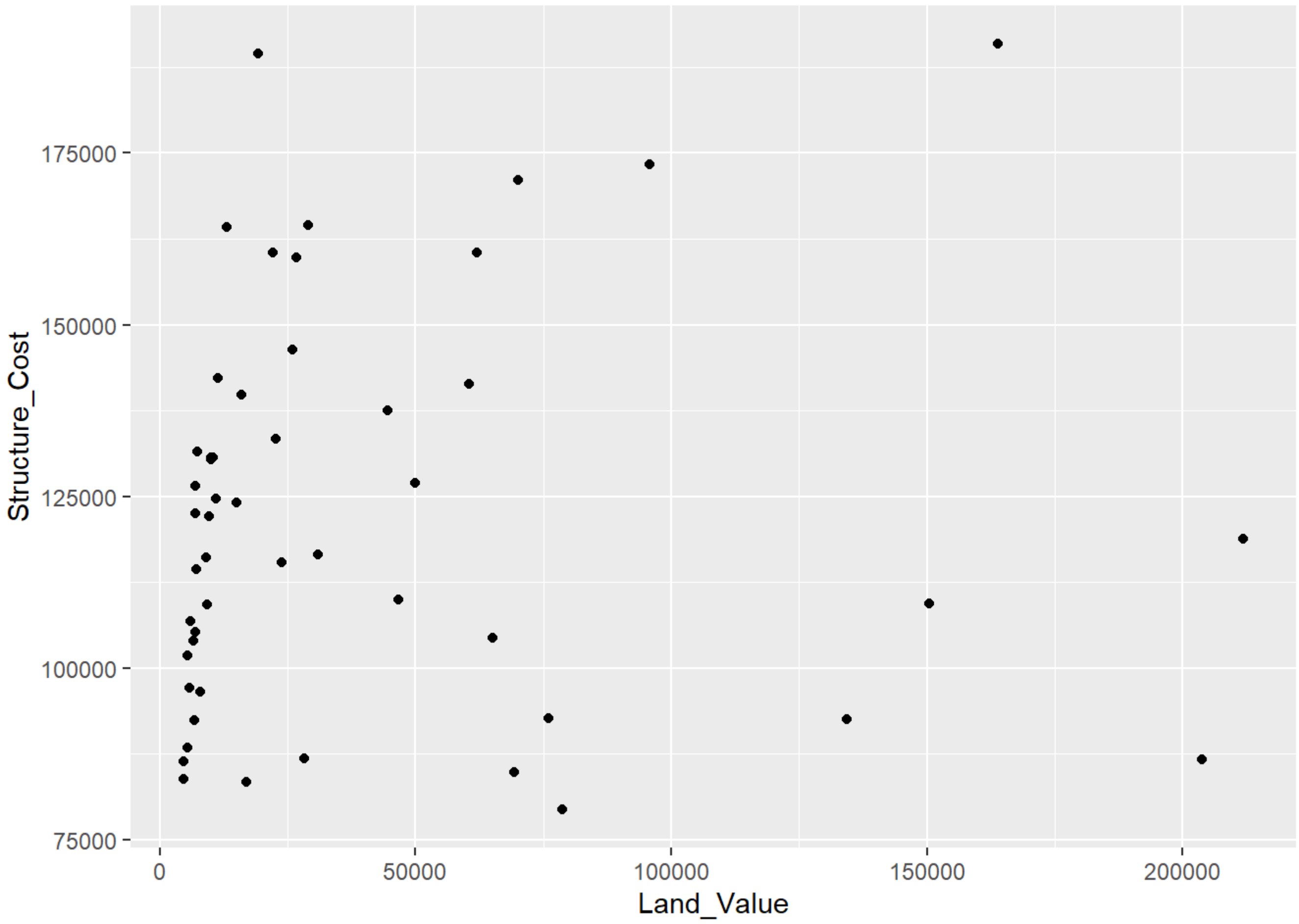
```
ggplot(data = hp2001Q1,
       mapping = aes(x = Land_Value, y = Structure_Cost))
```



Step 3: add new layers of geometric objects that will show up on the plot:

here we use `geom_point()` to add a layer with point (dot) elements as the geometric shapes to represent the data.

```
ggplot(data = hp2001Q1, aes(x = Land_Value, y = Structure_Cost)) +
    geom_point()
```



Lines (prediction line)

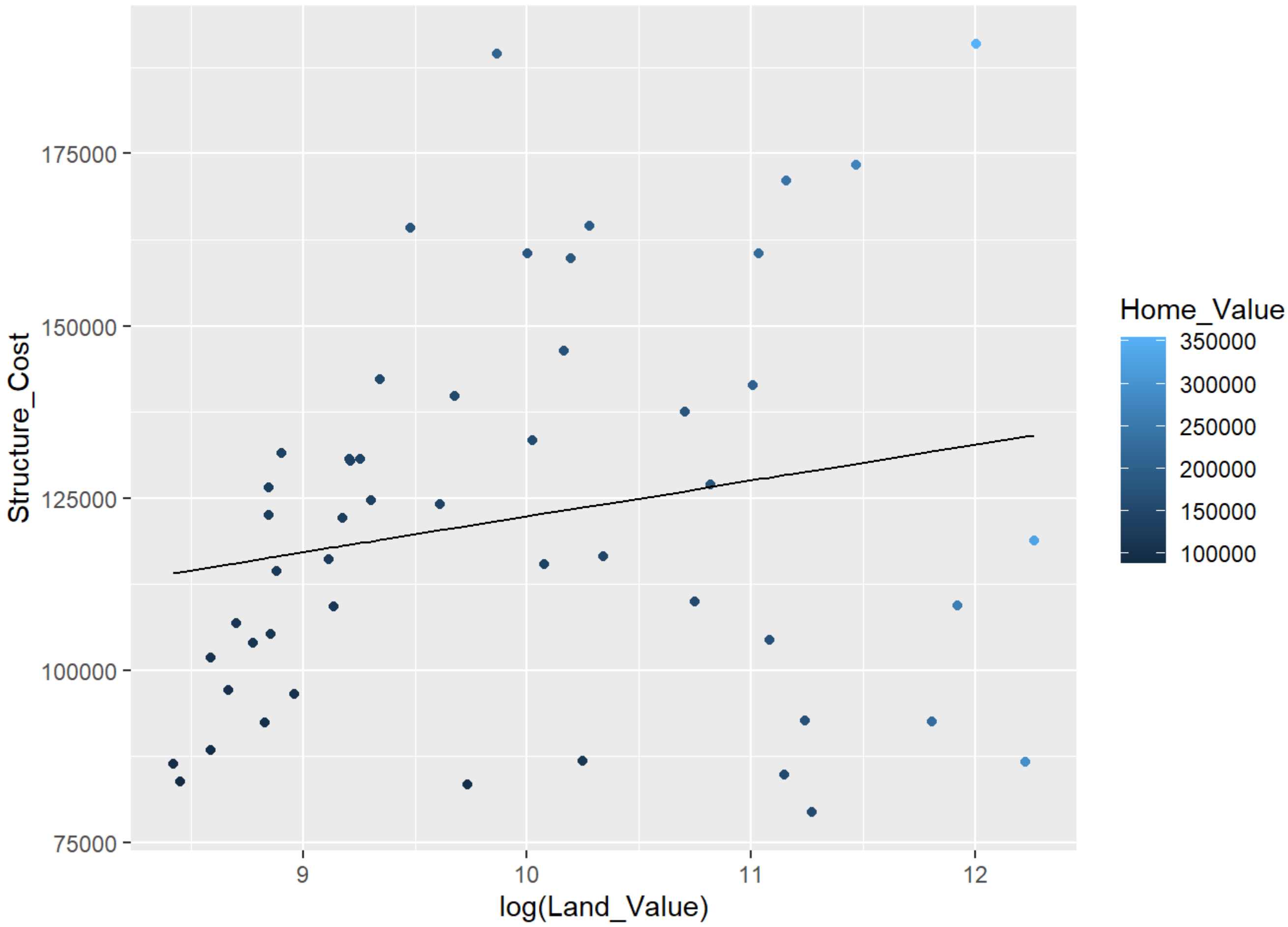
A plot constructed with `ggplot()` can have more than one geom. In that case the mappings established in the `ggplot()` call are plot defaults that can be added to or overridden — this is referred to as **aesthetic inheritance**. Our plot could use a regression line:

get predicted values from a linear regression

```
hp2001Q1$pred_SC <- lm(Structure_Cost ~ log(Land_Value),
                       data = hp2001Q1) |> predict()

p1 <- ggplot(hp2001Q1, aes(x = log(Land_Value), y = Structure_Cost))

p1 + geom_point(aes(color = Home_Value)) + # values for x and y are inherited from the ggplot
() call above
    geom_line(aes(y = pred_SC)) # add predicted values to the plot overriding the y values fr
om the ggplot() call above
```

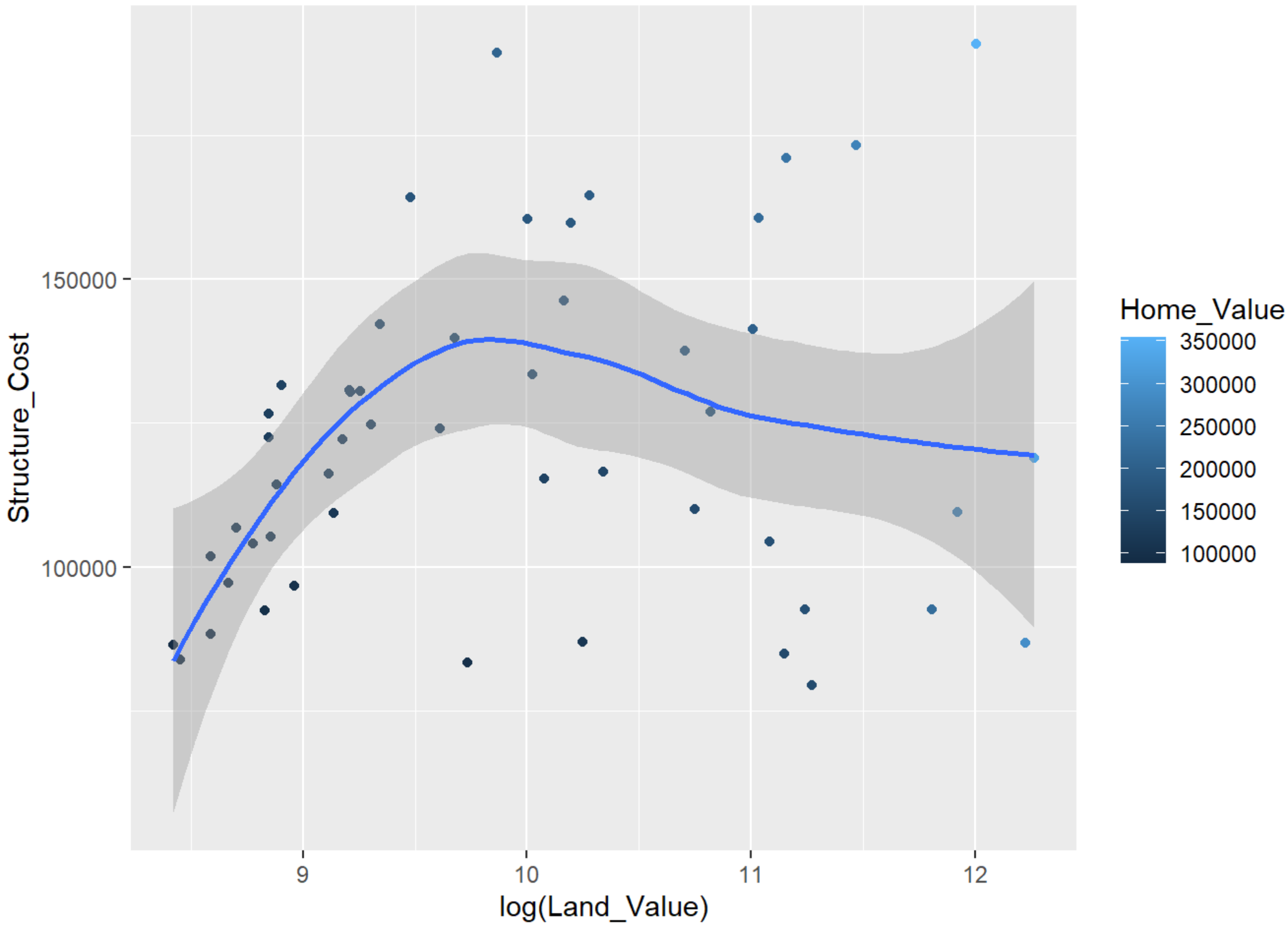


Smoothers

Not all geometric objects are simple shapes; the smooth geom includes a line and a ribbon.

```
p1 +
    geom_point(aes(color = Home_Value)) +
    geom_smooth()
```

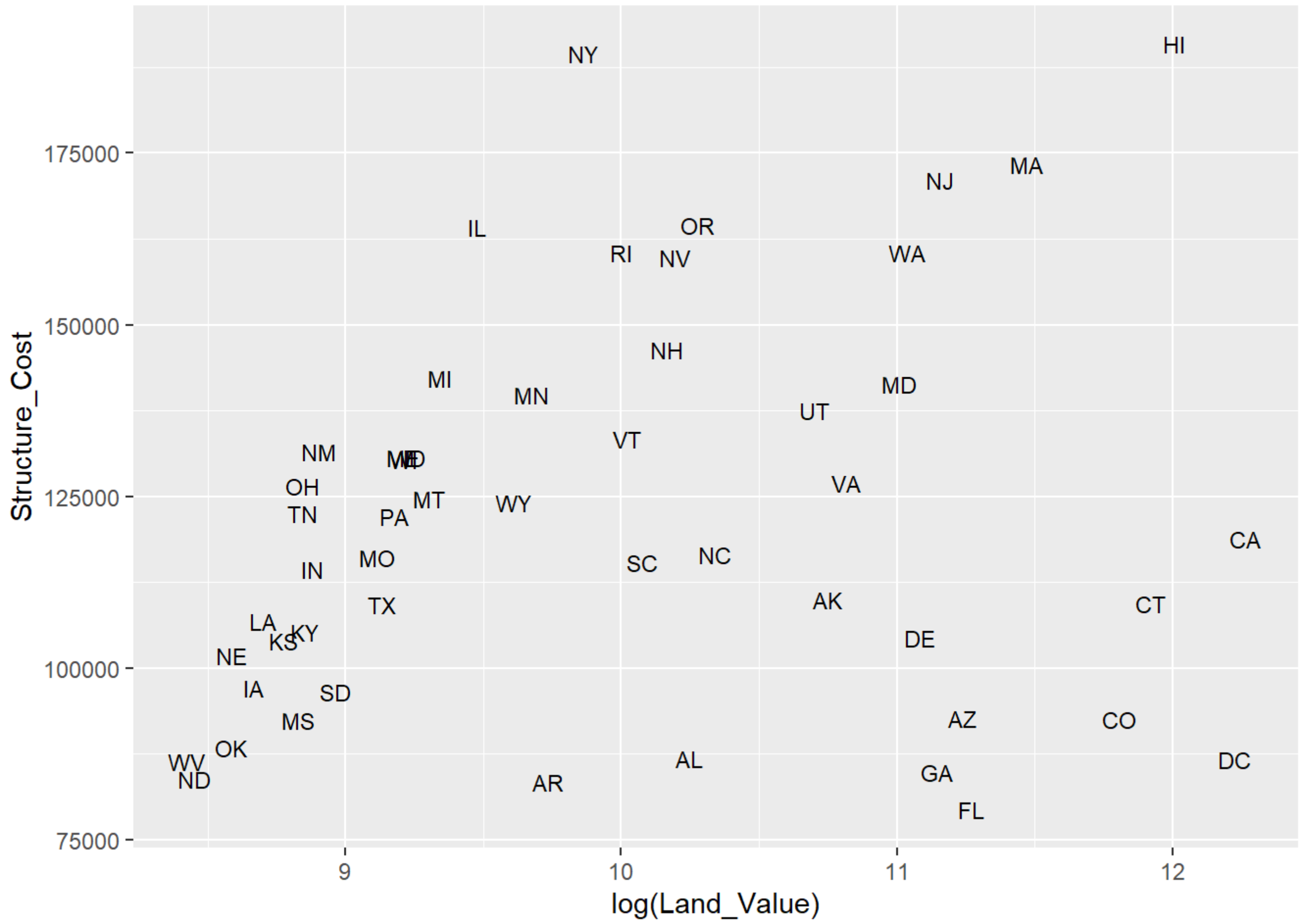
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Text (label points)

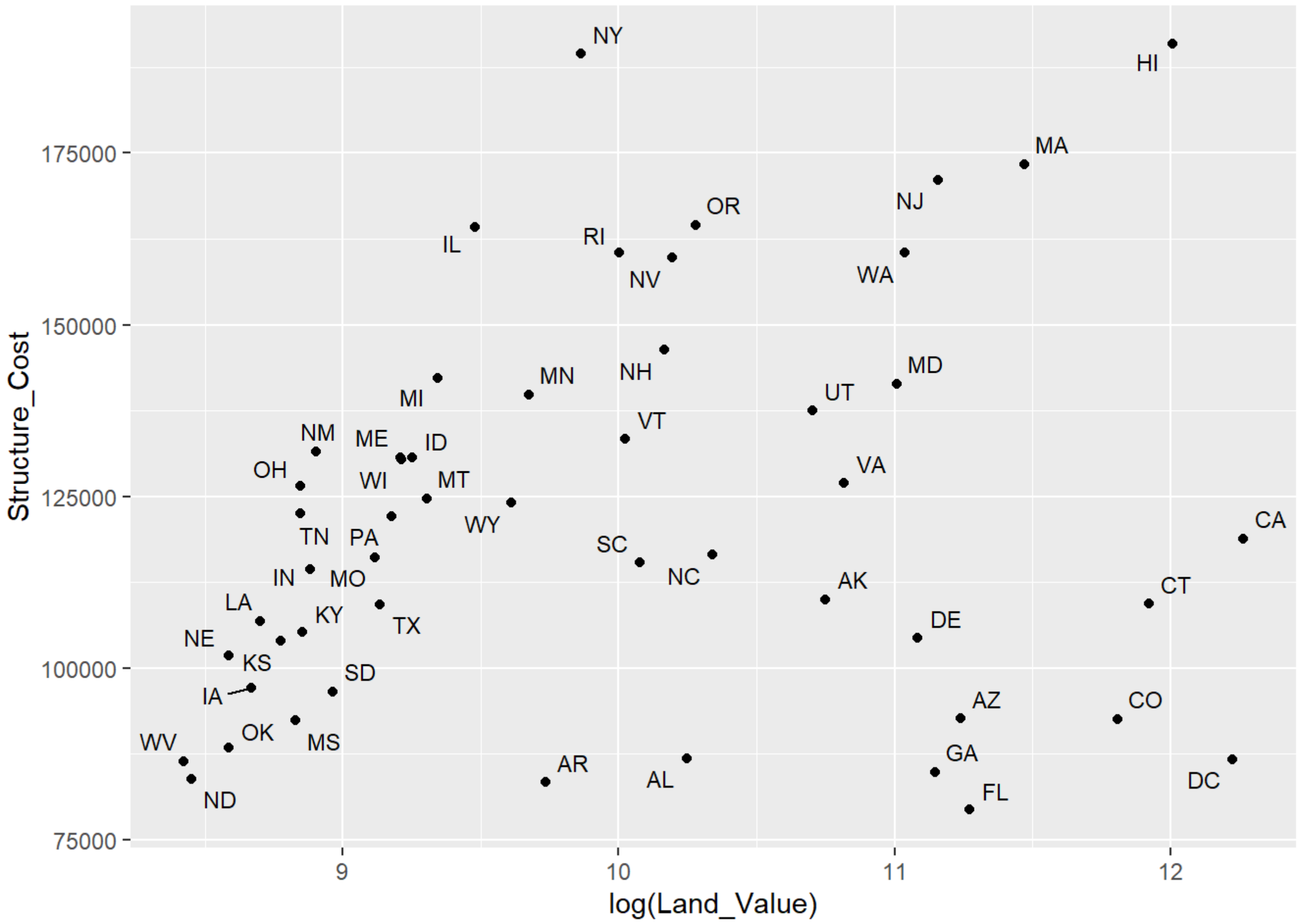
Each geom accepts a particular set of mappings; for example `geom_text()` accepts a `label` mapping.

```
p1 +
    geom_text(aes(label = State), size = 3)
```



But what if we want to include points and labels? We can use `geom_text_repel()` to keep labels from overlapping the points and each other.

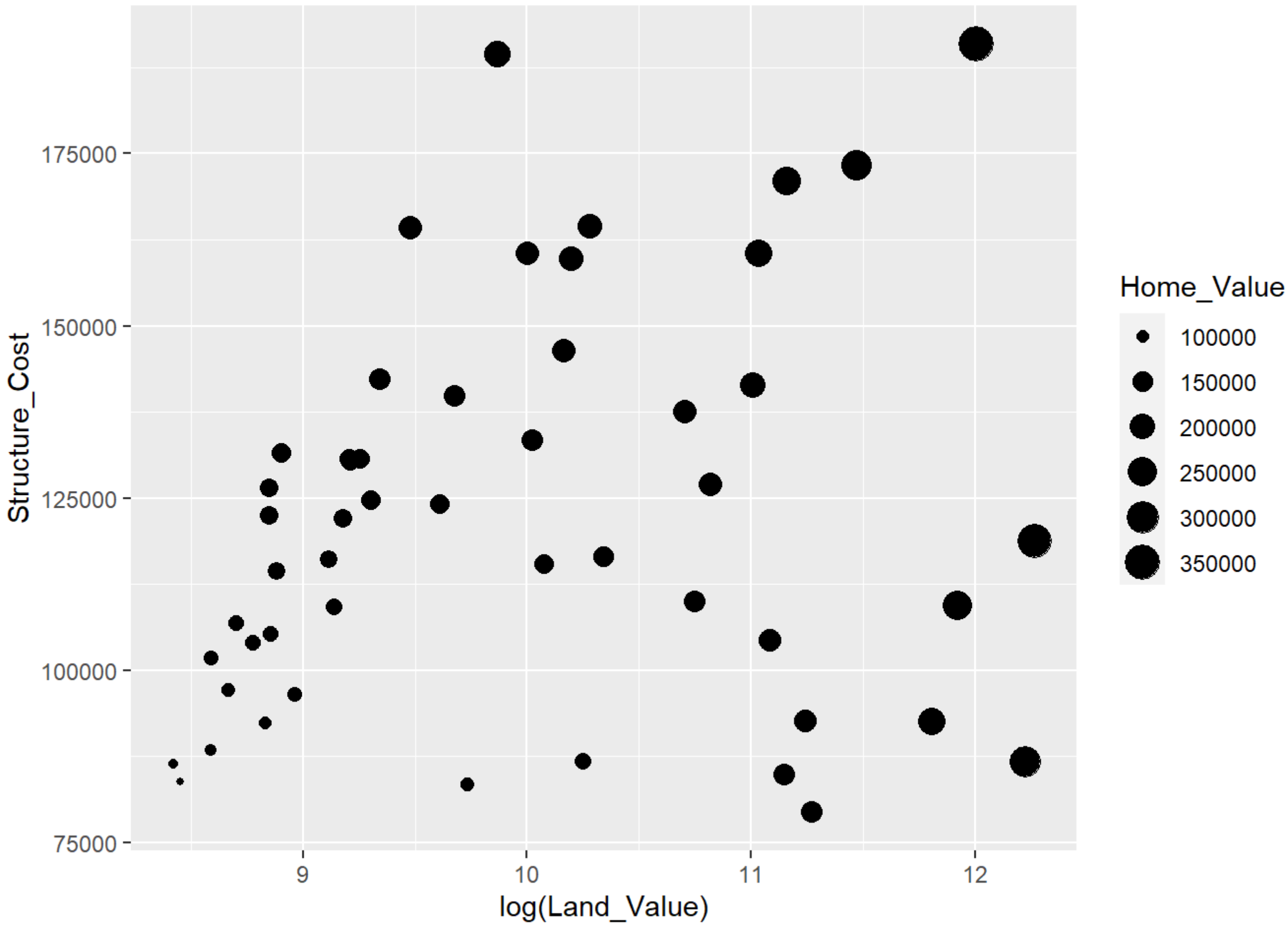
```
p1 +
    geom_point() +
    geom_text_repel(aes(label = State), size = 3)
```



Aesthetic mapping VS assignment

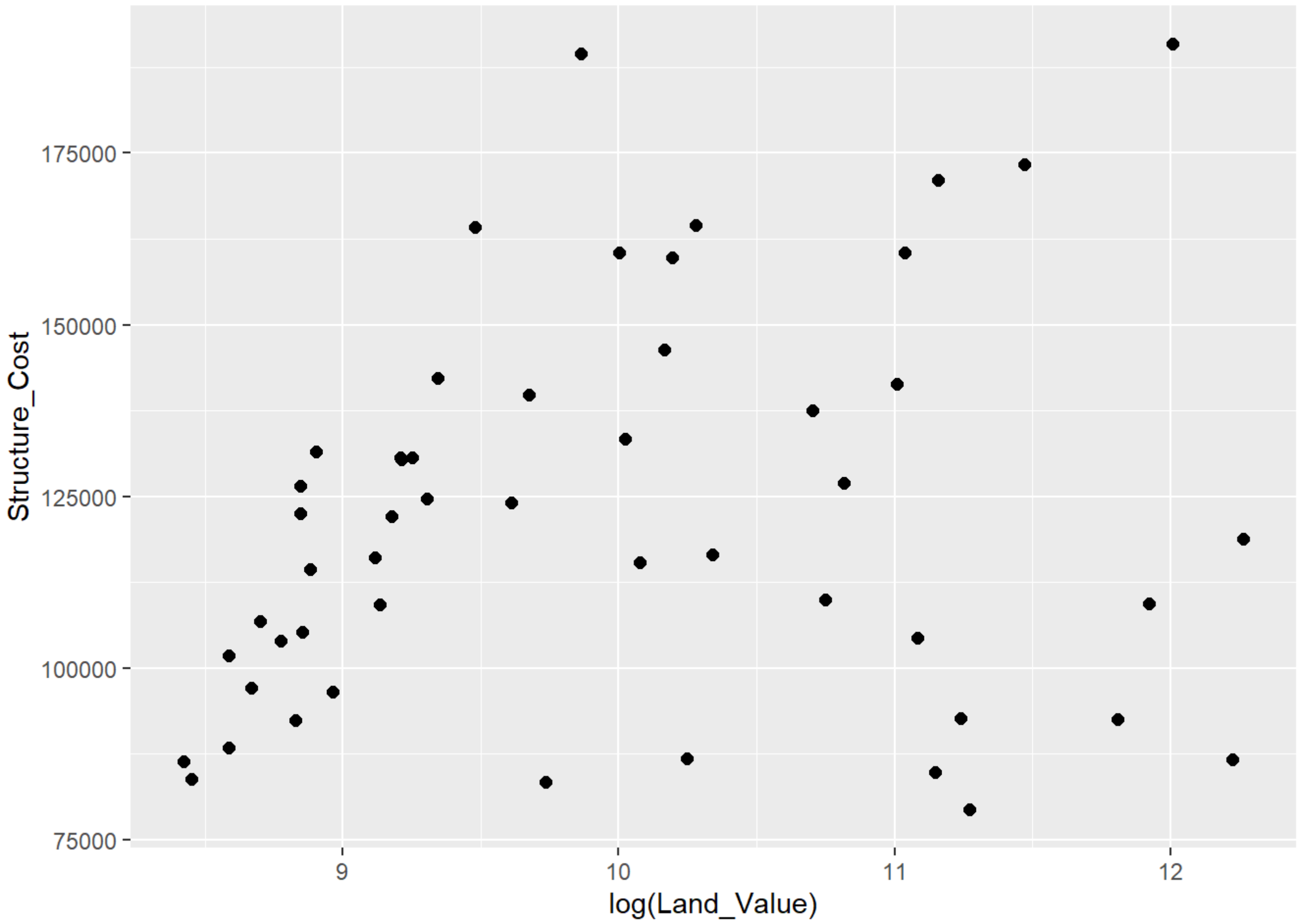
1. Variables are **mapped** to aesthetics inside the `aes()` function.

```
p1 +
    geom_point(aes(size = Home_Value))
```



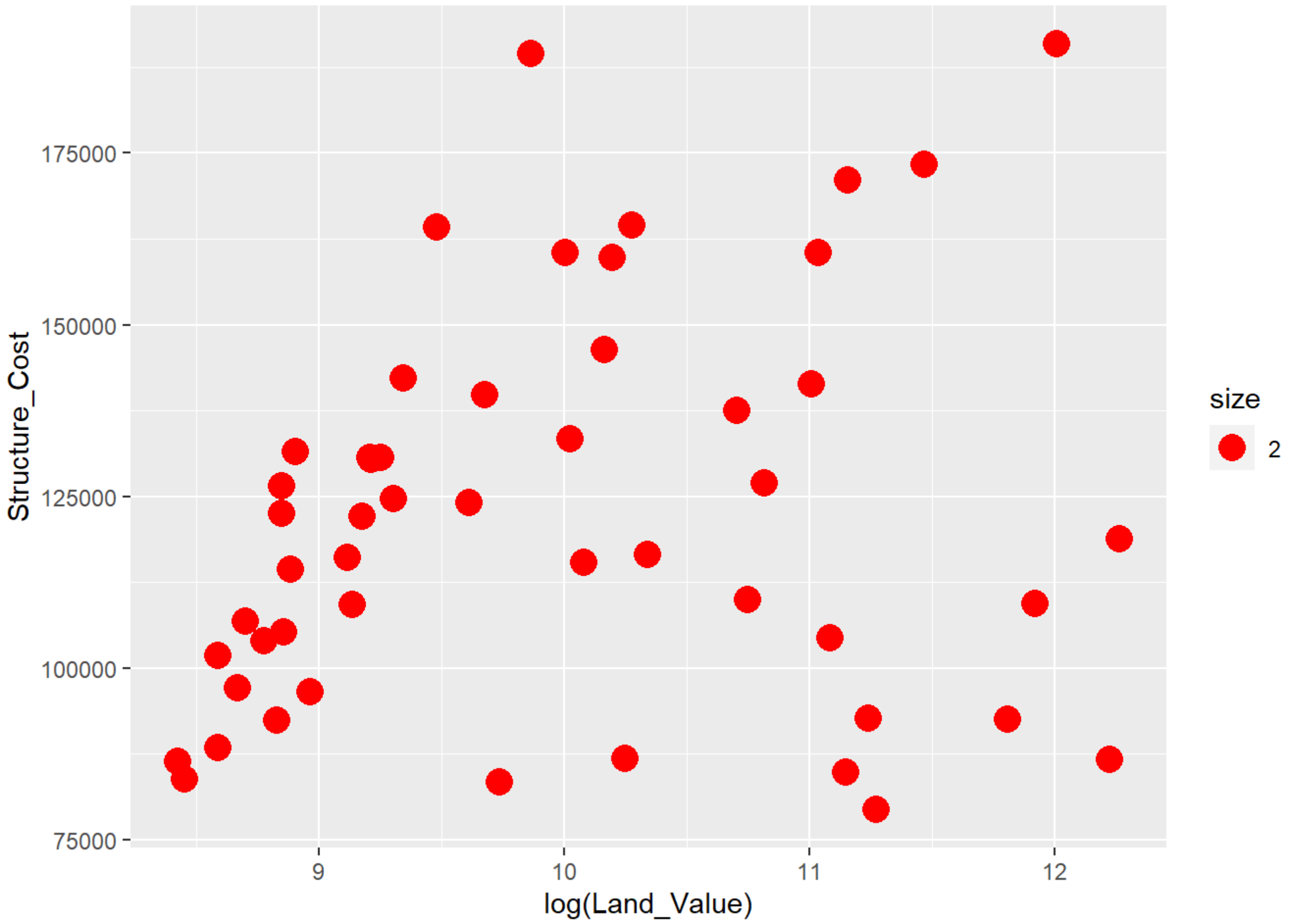
2. Constants are **assigned** to aesthetics outside the `aes()` call

```
p1 +
    geom_point(size = 2)
```



This sometimes leads to confusion, as in this example:

```
p1 +
    geom_point(aes(size = 2),# incorrect! 2 is not a variable
               color="red") # this is fine -- all points red
```

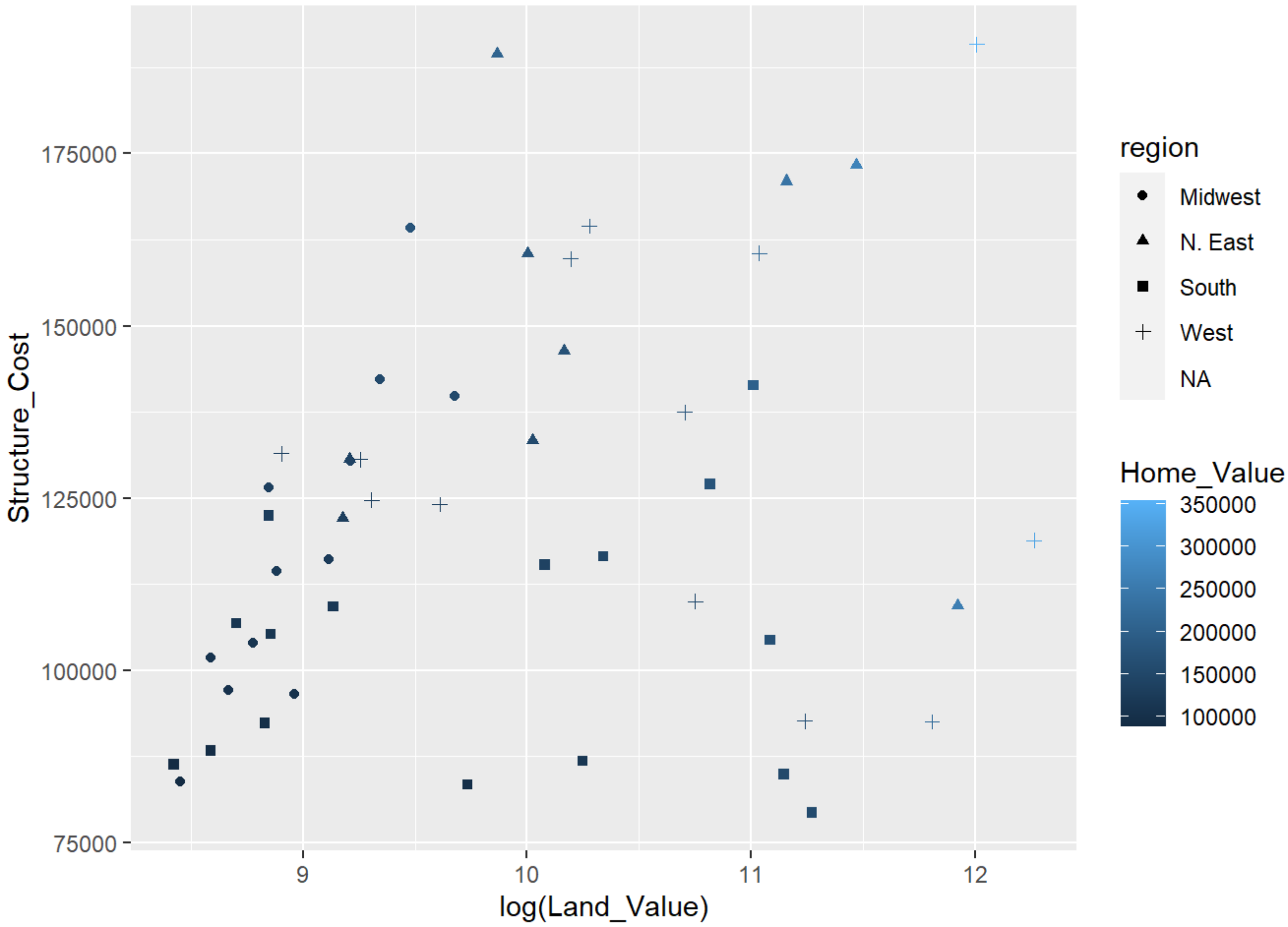


Mapping variables to other aesthetics

Other aesthetics are mapped in the same way as x and y in the previous example.

```
p1 +
    geom_point(aes(color = Home_Value, shape = region))
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```

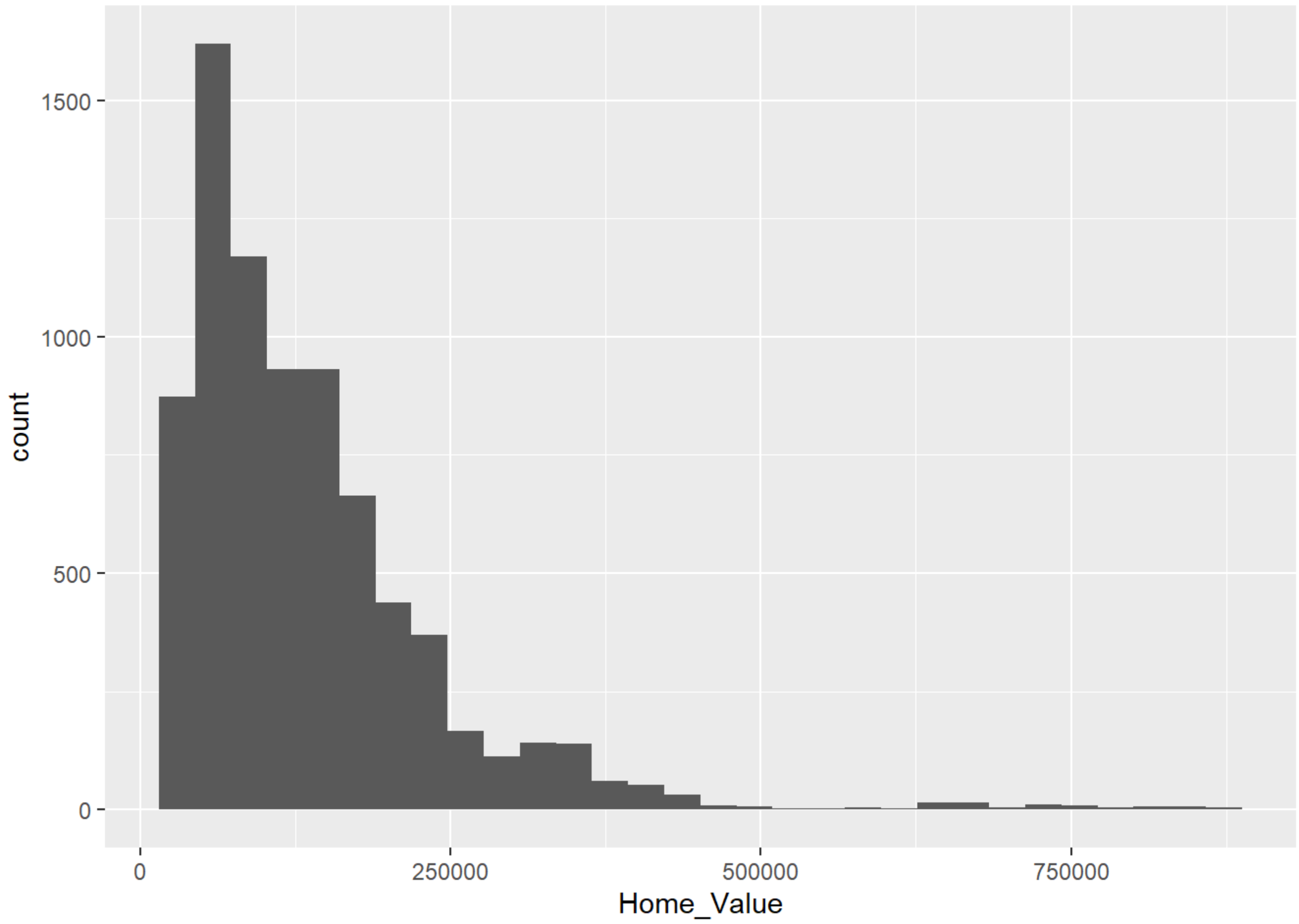


Other visualizations

Histogramm

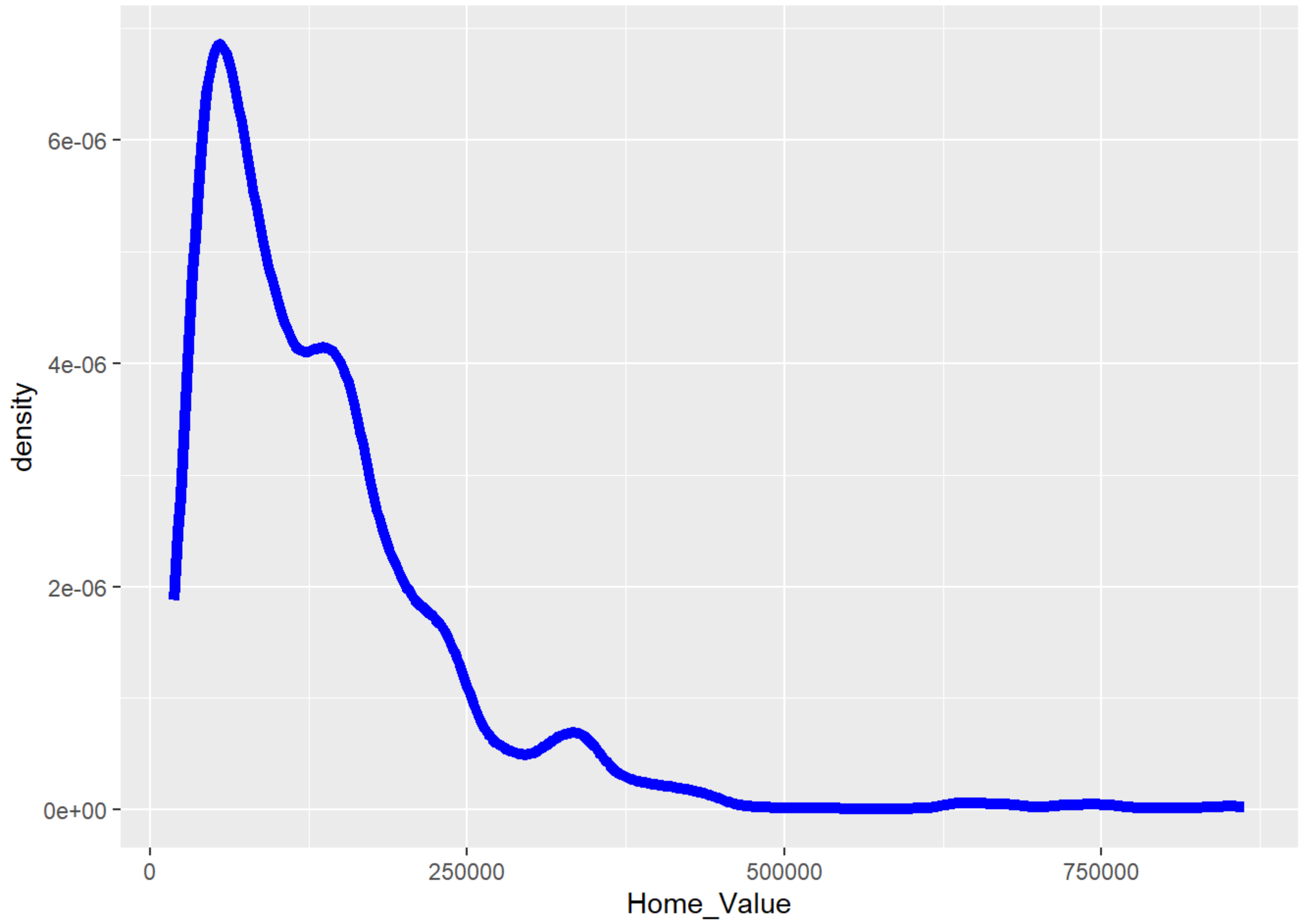
A histogram is an approximate representation of the distribution of numerical data. To construct a histogram, the first step is to “bin” (or “bucket”) the range of values— divide the entire range of values into a series of intervals—and then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent and are often (but not required to be) of equal size.

```
p2 <- ggplot(housing, aes(x = Home_Value))
p2 + geom_histogram(bins=30)
```



We can use a density estimate, which is a smoothed version of the histogram. This is a useful alternative to the histogram for continuous data that comes from an underlying smooth distribution

```
p2 <- ggplot(housing, aes(x = Home_Value))
p2 + geom_density(color="blue", linewidth=2)
```



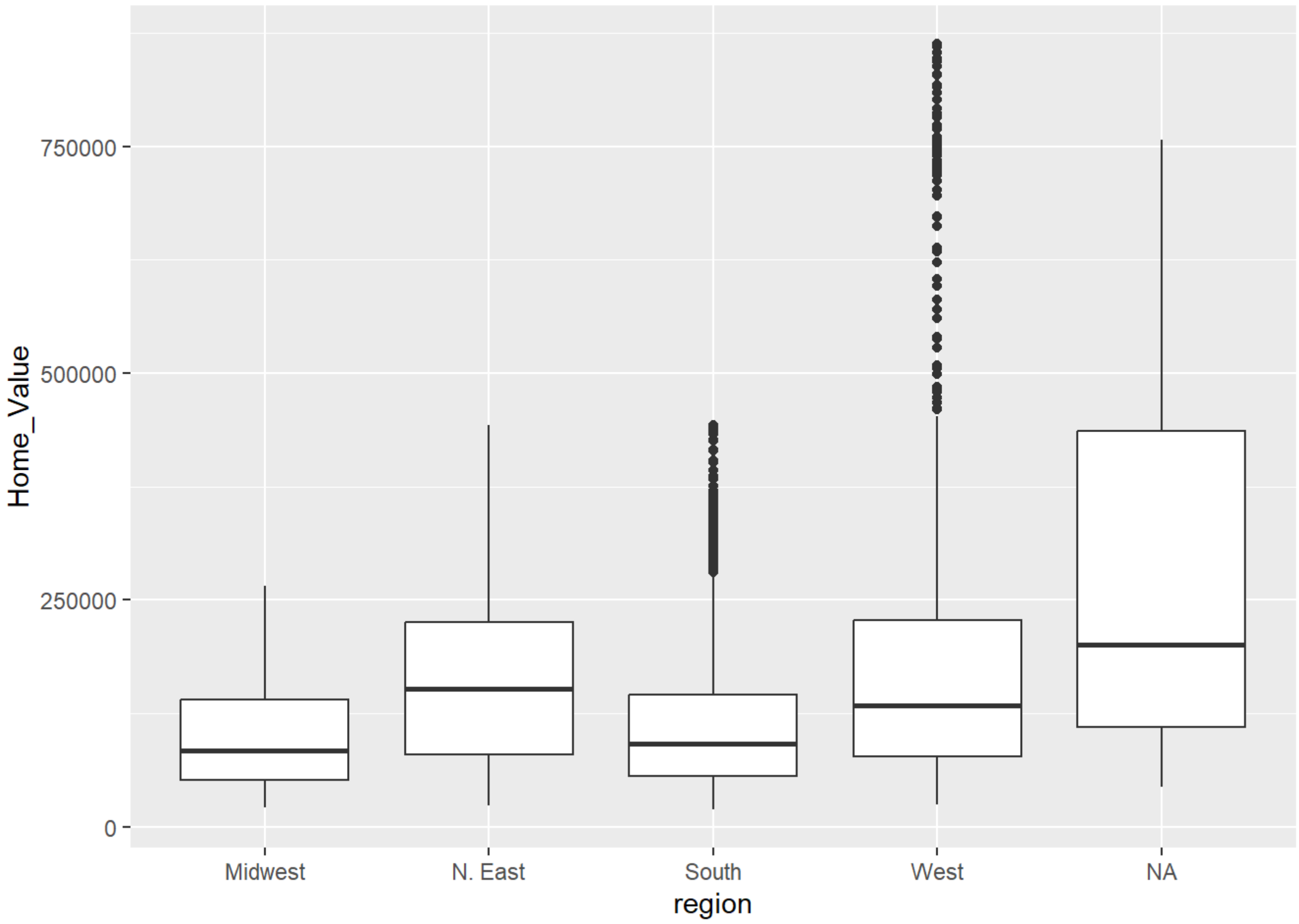
Boxplot

Boxlots are a type of data visualization that shows summary statistics for data.

More specifically, boxplots visualize what we call the “five number summary.” The five number summary is a set of values that includes:

- the minimum (-1.5 * IRQ)
- the first quartile (25th percentile)
- the median
- the third quartile (75th percentile)
- the maximum (+1.5 * IRQ)
- outliers

```
p3 <- ggplot(housing, aes(region, Home_Value))
p3 + geom_boxplot()
```


Home_Value
750000
500000
250000
0
Midwest
N. East
South
West
NA
region