post01

Michael Zhang

10/31/2017

An Introduction to ggplot2 Package

I. Introduction

This post is about different functions of ggplot2 package in R. I chose this topic because: first, this package is a lot more complicated than other packages that we've learnt; second, this is the part I'm struggling with.

Advantages of ggplot2

- It allows the user to manipulate all the plotting blocks individually.
- The theme of the graph is highly adjustable.
- High flexibility.

However, there are some shortages too:

- It can't generate 3-D graphs.
- It can't generate graphs with nodes or edges layout.
- It can't generate interactive graphics.

II. Functions and Their Inputs

In this section, we are going to learn different functions and their inputs in ggplot2 package.

1. Aesthetic Mapping

Basically, `aes` or aesthetic in ggplot2 means all the parameters that we can "see". For example:

- the position of axes
- outside color
- fill color
- shape of the points
- types of lines
- size of points/lines

2. Geometric Objects

`geom` or geometric objects are the things on a plot, such as:

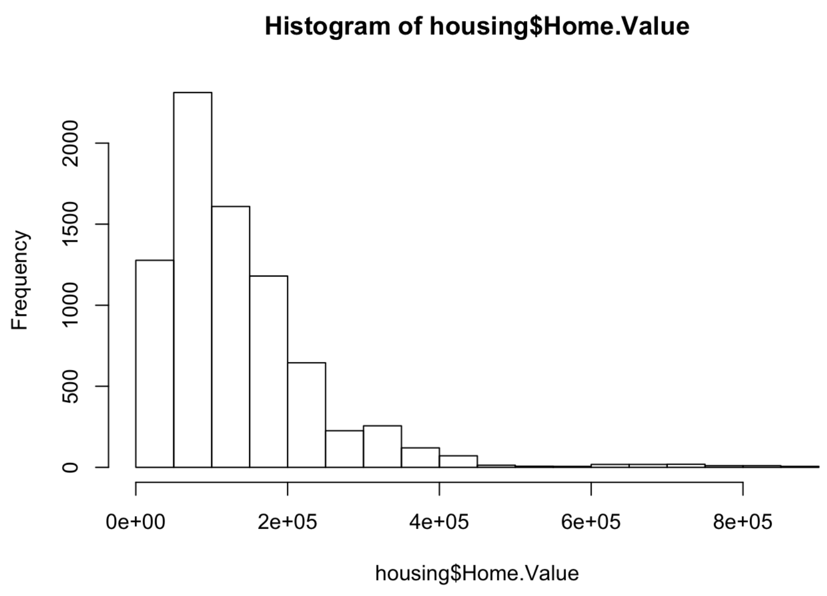
- lines (`geom_line`)
- points (`geom_point`)
- boxplot (`geom_boxplot`)

In order to have something to show on the graph, a graph must have at least one `geom` function.

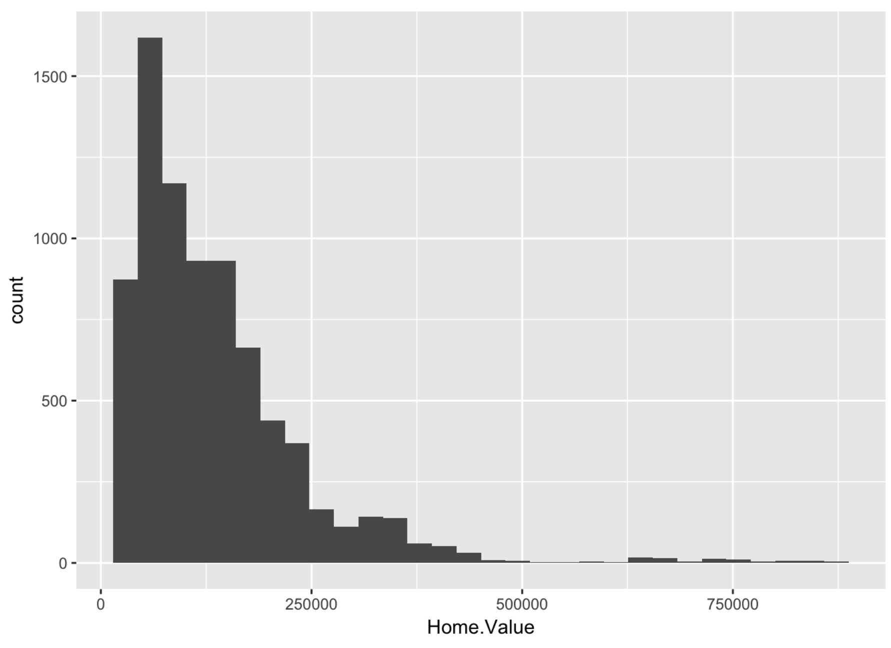
4. Prediction Line

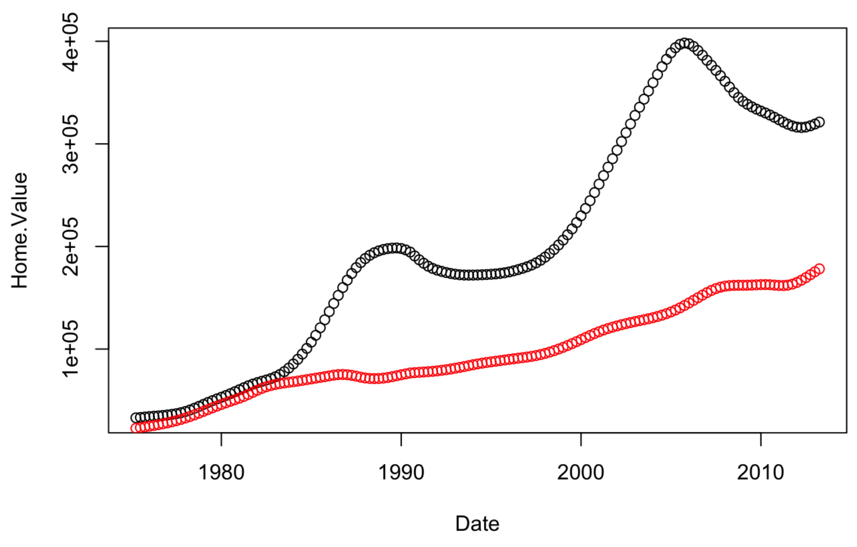
A plot constructed with `ggplot` can have more than one geom. In that case the mappings established in the `ggplot()` call are plot defaults that can be added to or overridden. Our plot could use a regression line:

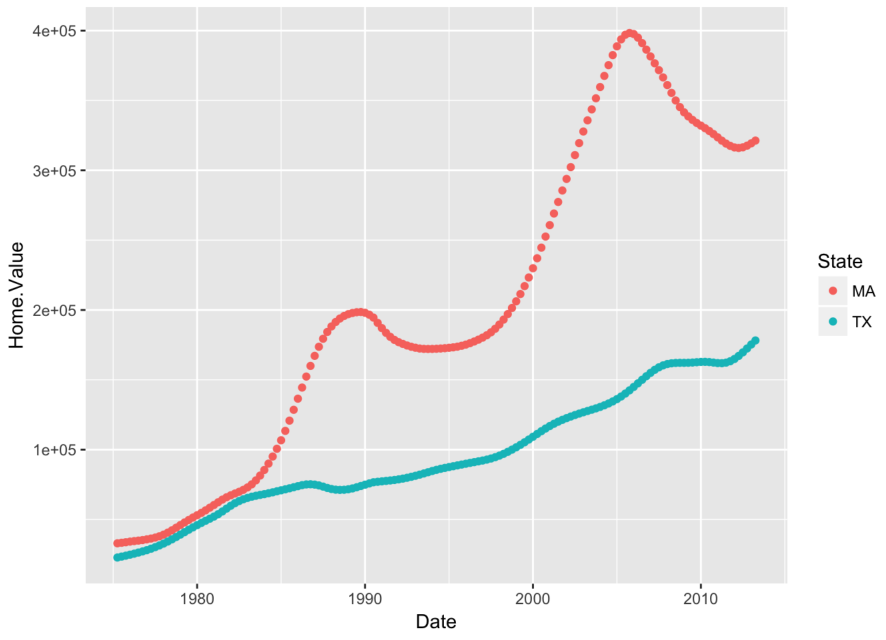
```
##   State region    Date Home.Value Structure.Cost
## 1    AK   West 2010.25     224952         160599
## 2    AK   West 2010.50     225511         160252
## 3    AK   West 2009.75     225820         163791
## 4    AK   West 2010.00     224994         161787
## 5    AK   West 2008.00     234590         155400
## 6    AK   West 2008.25     233714         157458
```

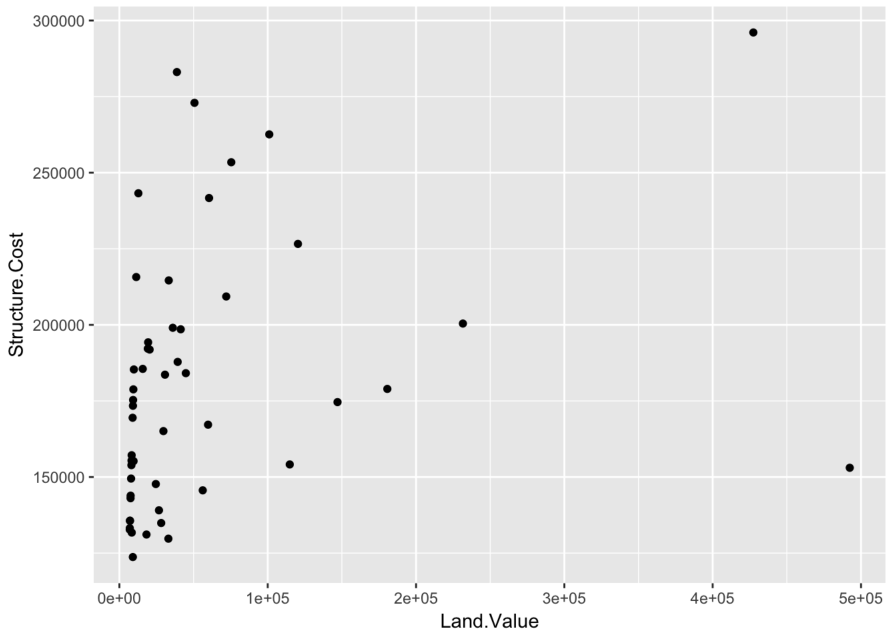


```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

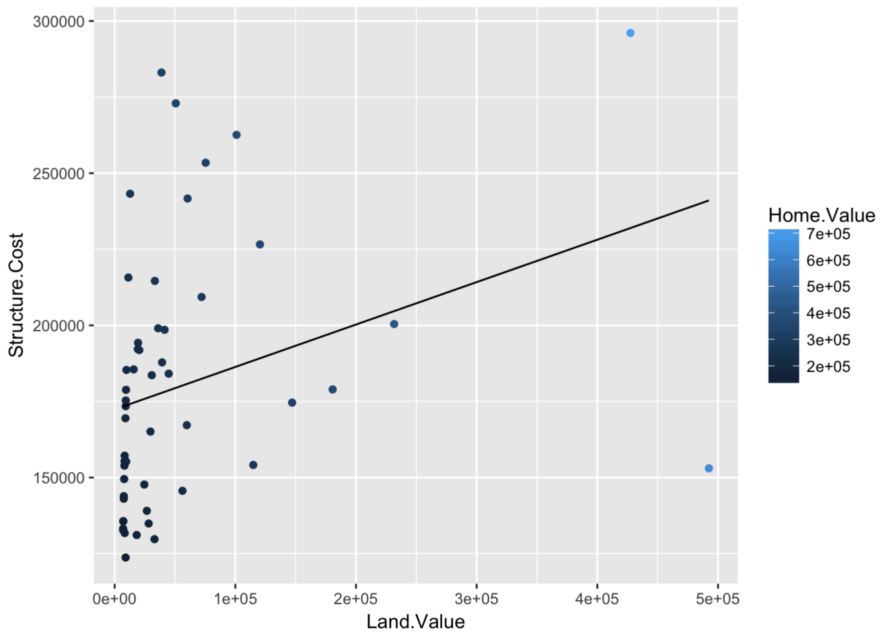








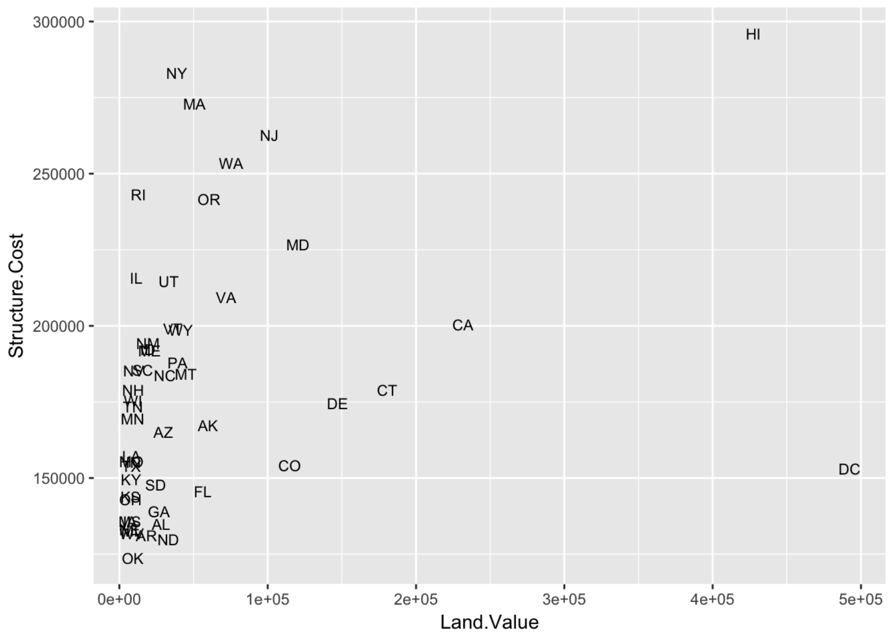
```r
library(ggplot2)
hp2001Q1$pred.SC <- predict(lm(Structure.Cost ~ Land.Value, data = hp2001Q1))

p1 <- ggplot(hp2001Q1, aes(x = Land.Value, y = Structure.Cost))

p1 + geom_point(aes(color = Home.Value)) +
  geom_line(aes(y = pred.SC))
```



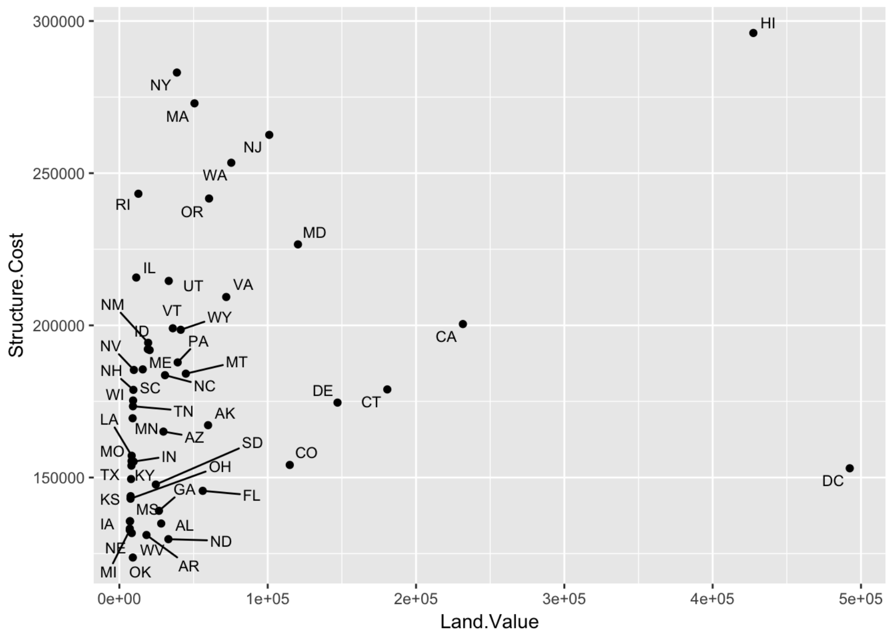
5. Label Points

Each `geom` accepts a particualar set of mappings–for example `geom_text()` accepts a `labels` mapping.

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



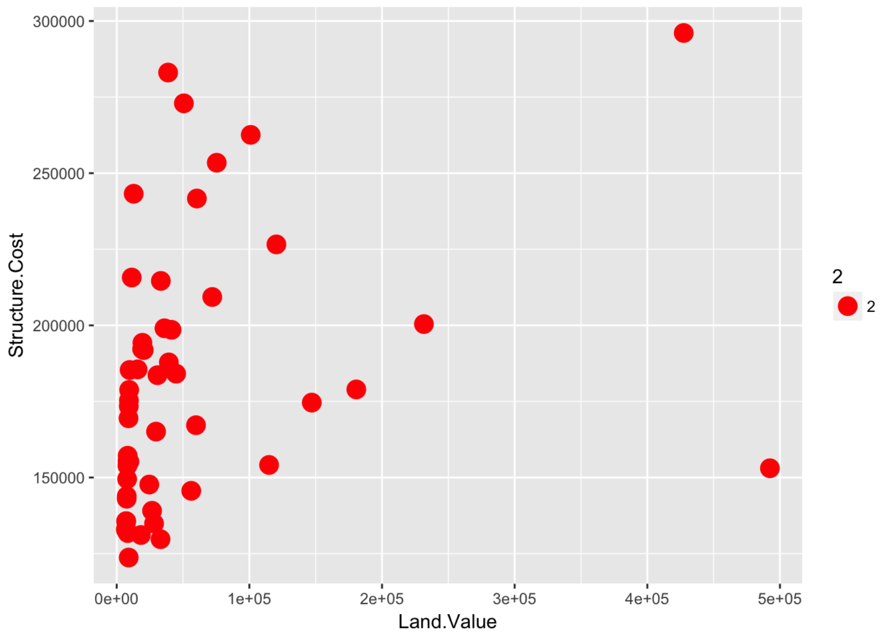
```
# install.packages("ggrepel")
library("ggrepel")
p1 +
  geom_point() +
  geom_text_repel(aes(label=State), size = 3)
```



Aesthetic Mapping VS Assignment

Note that variables are mapped to aesthetics with the `aes()` function, while fixed aesthetics are set outside the `aes()` call. 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
```

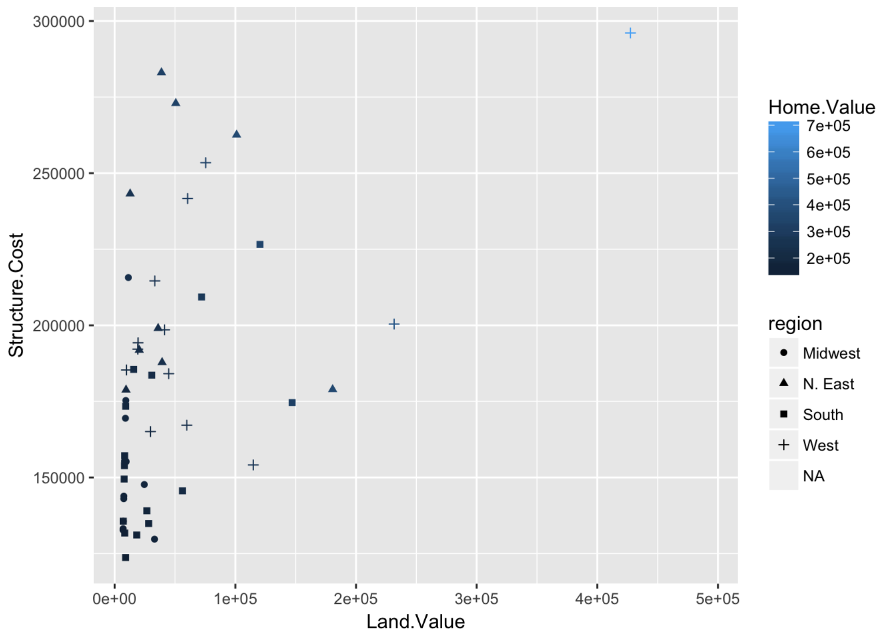


Mapping Variables To Other Aesthetics

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).
```



III. Statistical Transformations

1. Statistical Transformations

Sometimes, some plot, such as scatterplot, do not require data transformation. However, others, such as, such as boxplots, histograms, prediction lines and so on, require statistical transformations (i.e. changine axes' coordinates):

- for a boxplot the y values must be transformed to the median and 1.5(IQR)
- for a smoother smother the y values must be transformed into predicted values

Each `geom` has a default statistic, but these can be changed. For example, the default statistic for `geom_bar` is `stat_count`:

```
args(geom_histogram)
```

```
## function (mapping = NULL, data = NULL, stat = "bin", position = "stack",
##     ..., binwidth = NULL, bins = NULL, na.rm = FALSE, show.legend = NA,
##     inherit.aes = TRUE)
## NULL
```

```
args(stat_bin)
```

```
## function (mapping = NULL, data = NULL, geom = "bar", position = "stack",
##     ..., binwidth = NULL, bins = NULL, center = NULL, boundary = NULL,
##     breaks = NULL, closed = c("right", "left"), pad = FALSE,
##     na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
## NULL
```

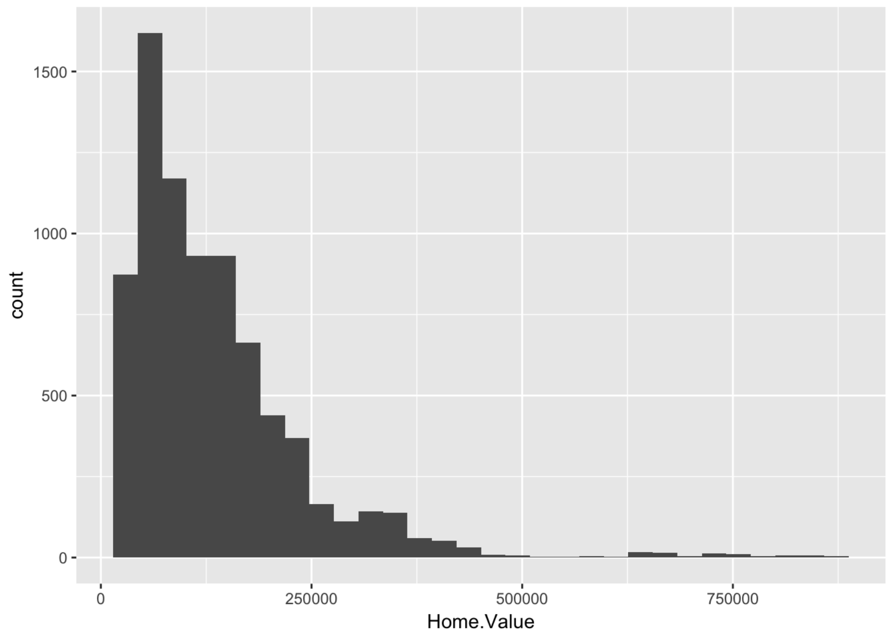
2. Setting Statistical Transformation Arguments

In order to change it you have to first determine which stat the geom uses, then determine the arguments to that stat.

For example, here is the default histogram of Home.Value:

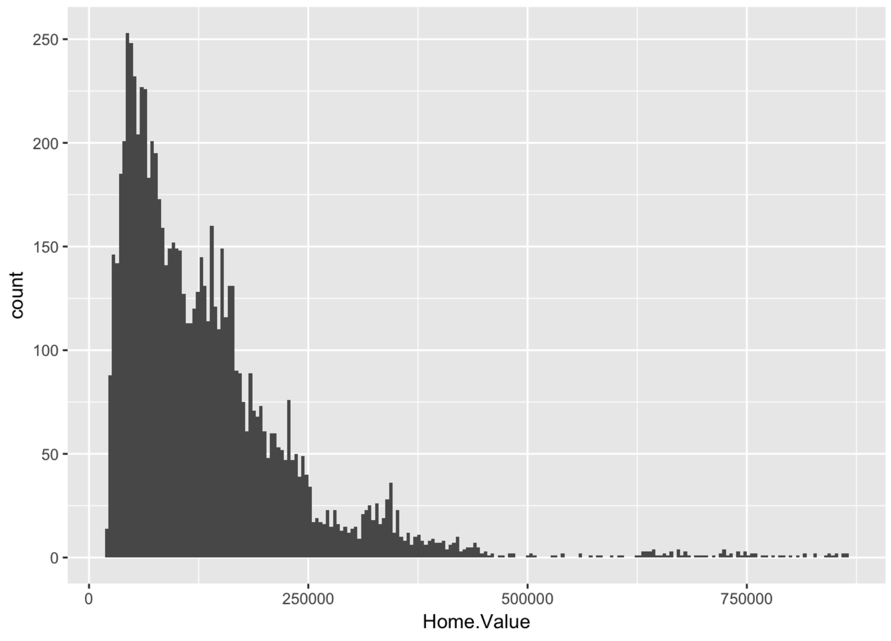
```
p2 <- ggplot(housing, aes(x = Home.Value))
p2 + geom_histogram()
```

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



We can change the binwidth by passing the `binwidth` argument to the `stat_bin` function:

```
p2 + geom_histogram(stat = "bin", binwidth=4000)
```



IV. Scales

1. Controlling Aesthetic Mapping

Aesthetic mapping (i.e., with `aes()`) only says that a variable should be mapped to an aesthetic. It doesn’t say *how* that should happy.

For example, when mapping a variable to *shape* with `aes(shape` x)= you don’t say *what* shapes should be used. Similarly, `aes(color` z)= doesn’t say *what* colors should be used. Describing what colors/shapes/sizes etc. to use is done by modifying the corresponding *scale*. In `ggplot2` scales include

- position
- color and fill
- size
- shape
- line type

Scales are modified with a series of functions using a `scale_<aesthetic>_<type>` naming scheme. Try typing `scale_<tab>` to see a list of scale modification functions.

2. Common Scale Arguments

The following arguments are common to most scales in ggplot2:

- name: the first argument gives the axis or legend title
- limits: the minimum and maximum of the scale
- breaks: the points along the scale where labels should appear

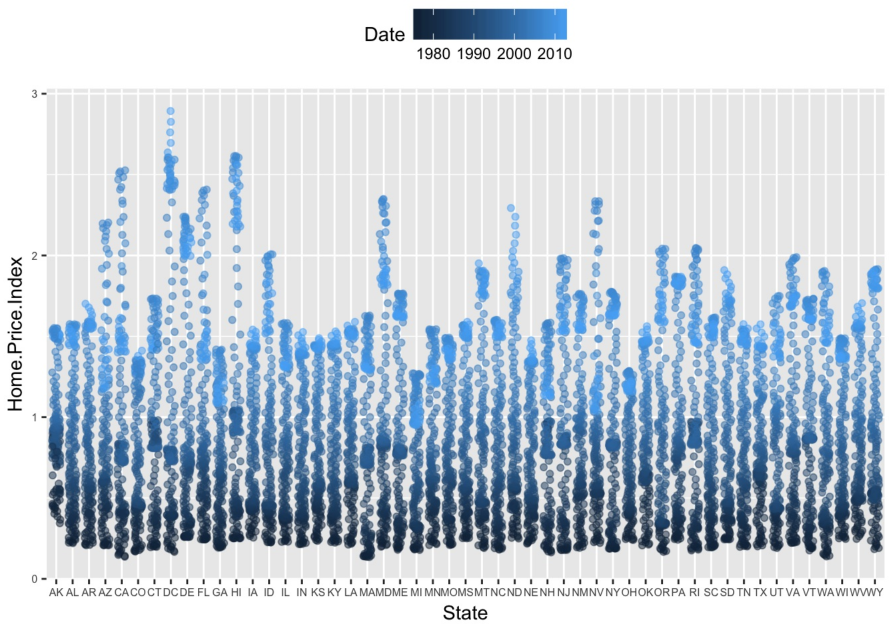
- labels: the labels that appear at each break

Specific scale functions may have additional arguments; for example, the `scale_color_continuous` function has arguments `low` and `high` for setting the colors at the low and high end of the scale.

3. Scale Modification Examples

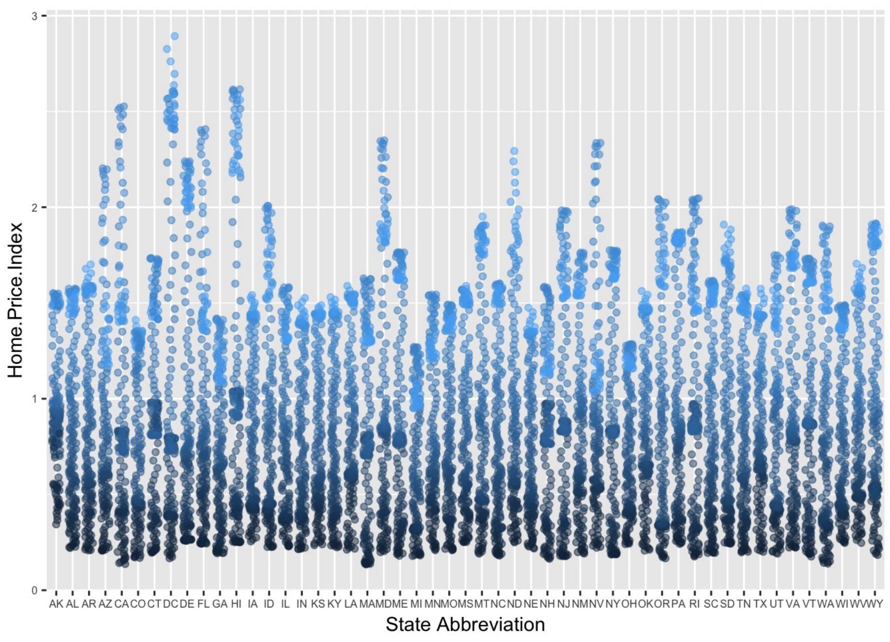
Start by constructing a dotplot showing the distribution of home values by Date and State.

```
p3 <- ggplot(housing,
             aes(x = State,
                 y = Home.Price.Index)) +
        theme(legend.position="top",
              axis.text=element_text(size = 6))
(p4 <- p3 + geom_point(aes(color = Date),
                       alpha = 0.5,
                       size = 1.5,
                       position = position_jitter(width = 0.25, height = 0)))
```



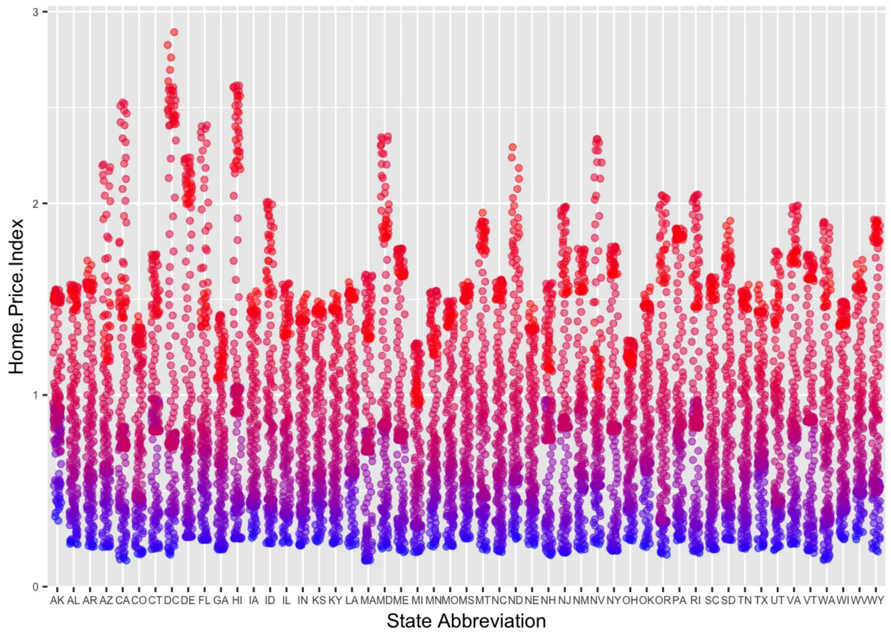
Now modify the breaks and labels for the x axis and color scales

```
p4 + scale_x_discrete(name="State Abbreviation") +
  scale_color_continuous(name="",
                         breaks = c(19751, 19941, 20131),
                         labels = c(1971, 1994, 2013))
```

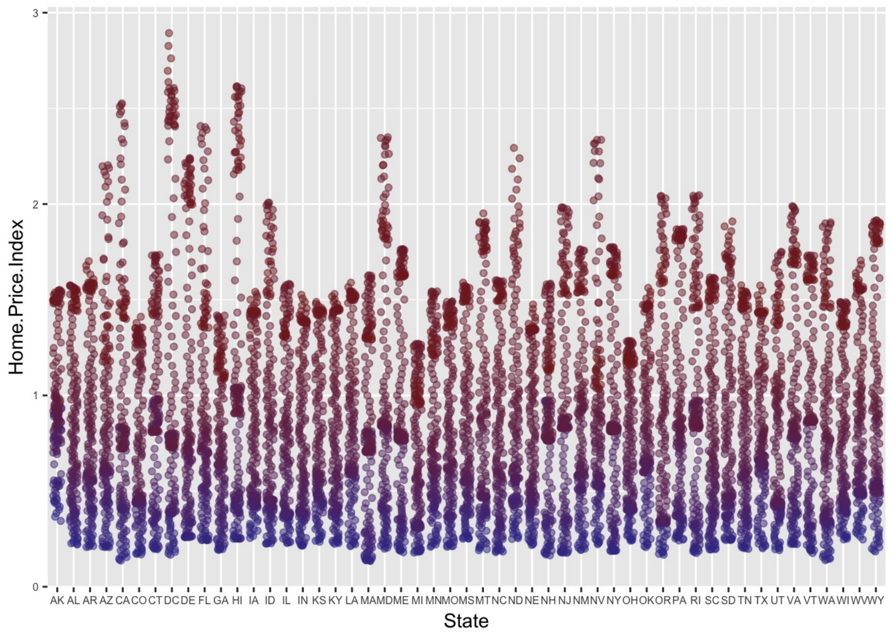


Next change the low and high values to blue and red:

```
p4 +
  scale_x_discrete(name="State Abbreviation") +
  scale_color_continuous(name="",
                         breaks = c(19751, 19941, 20131),
                         labels = c(1971, 1994, 2013),
                         low = "blue", high = "red")
```



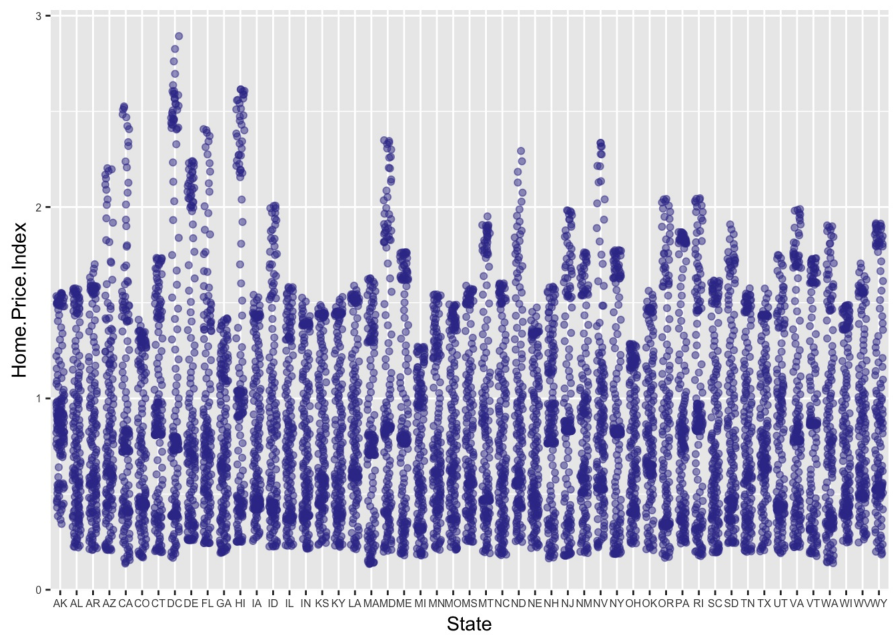
```
library(scales)
p4 +
  scale_color_continuous(name="",
                         breaks = c(19751, 19941, 20131),
                         labels = c(1971, 1994, 2013),
                         low = muted("blue"), high = muted("red"))
```



4. Color Scales

ggplot2 has a wide variety of color scales; here is an example using `scale_color_gradient2` to interpolate between three different colors.

```
p4 +
  scale_color_gradient2(name="",
                        breaks = c(19751, 19941, 20131),
                        labels = c(1971, 1994, 2013),
                        low = muted("blue"),
                        high = muted("red"),
                        mid = "gray60",
                        midpoint = 19941)
```



5. Available Scales

- Partial combination matrix of available scales

Scale	Types	Examples
scale$_{\text{color\_}}$	identity	scale$_{\text{fillcontinuous}}$
scale$_{\text{fill\_}}$	manual	scale$_{\text{colordiscrete}}$
scale$_{\text{size\_}}$	continuous	scale$_{\text{sizemanual}}$
	discrete	scale$_{\text{sizediscrete}}$
scale$_{\text{shape\_}}$	discrete	scale$_{\text{shapediscrete}}$
scale$_{\text{linetype\_}}$	identity	scale$_{\text{shapemanual}}$
	manual	scale$_{\text{linetypediscrete}}$
scale$_{\text{x\_}}$	continuous	scale$_{\text{xcontinuous}}$
scale$_{\text{y\_}}$	discrete	scale$_{\text{ydiscrete}}$
	reverse	scale$_{\text{xlog}}$
	log	scale$_{\text{yreverse}}$
	date	scale$_{\text{xdate}}$
	datetime	scale$_{\text{ydatetime}}$

Note that in RStudio you can type `scale_` followed by TAB to get the whole list of available scales.

V. Faceting

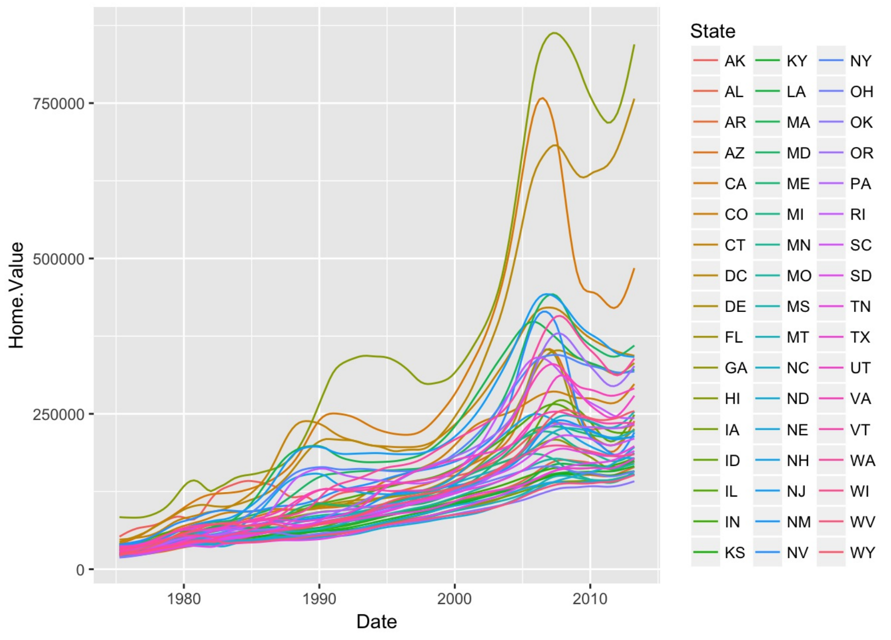
1. Faceting

- Faceting is `ggplot2` parlance for **small multiples**
- The idea is to create separate graphs for subsets of data
- `ggplot2` offers two functions for creating small multiples:
    1. `facet_wrap()` : define subsets as the levels of a single grouping variable
    2. `facet_grid()` : define subsets as the crossing of two grouping variables
- Facilitates comparison among plots, not just of geoms within a plot

2. What is the trend in housing prices in each state?

- Start by using a technique we already know–map State to color:

```
p5 <- ggplot(housing, aes(x = Date, y = Home.Value))
p5 + geom_line(aes(color = State))
```

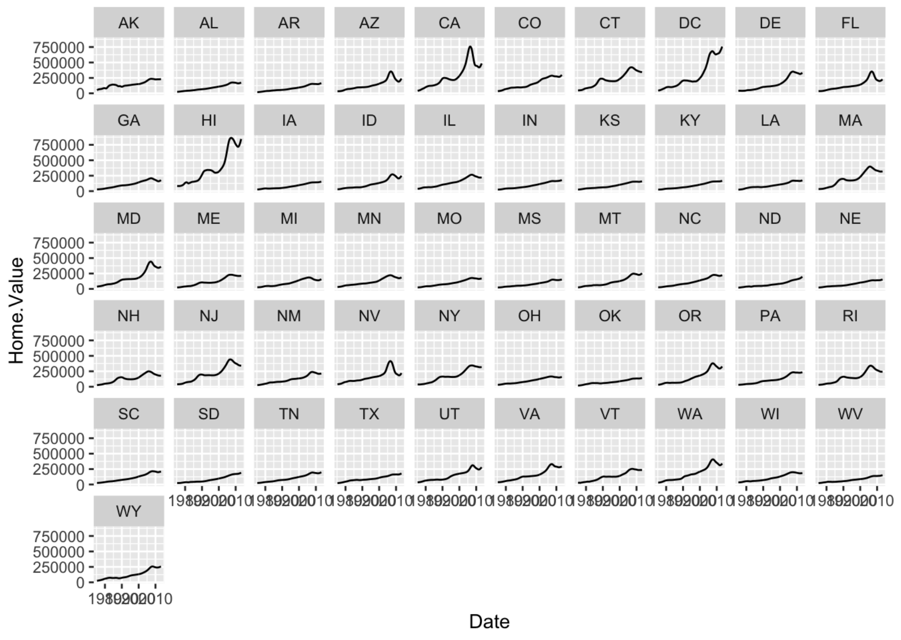


There are two problems here–there are too many states to distinguish each one by color, and the lines obscure one another.

3. Faceting to the rescue

We can remedy the deficiencies of the previous plot by faceting by state rather than mapping state to color.

```
(p5 <- p5 + geom_line() +
   facet_wrap(~State, ncol = 10))
```



There is also a `facet_grid()` function for faceting in two dimensions.

VI. Themes

1. Themes

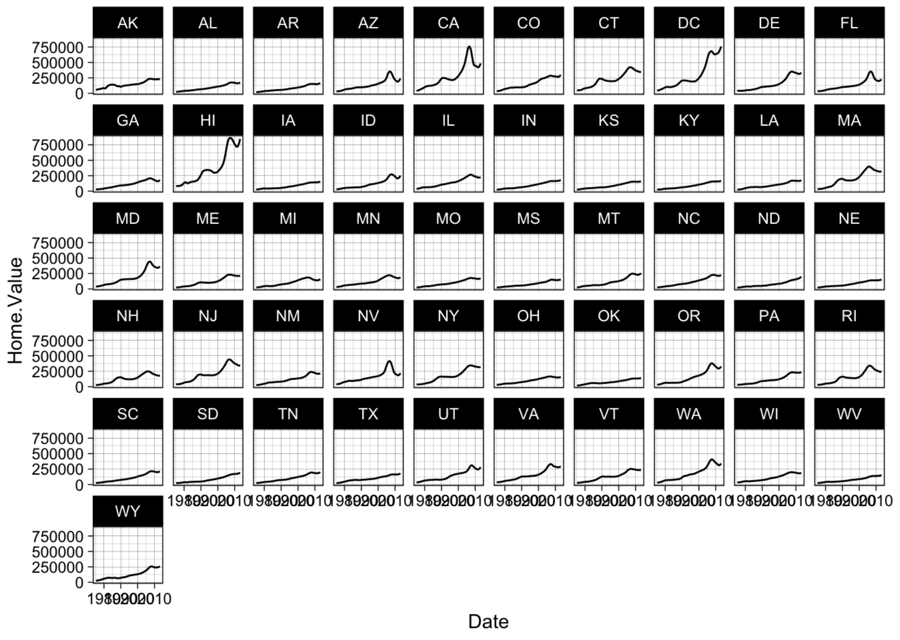
The `ggplot2` theme system handles non-data plot elements such as

- Axis labels
- Plot background
- Facet label backround
- Legend appearance

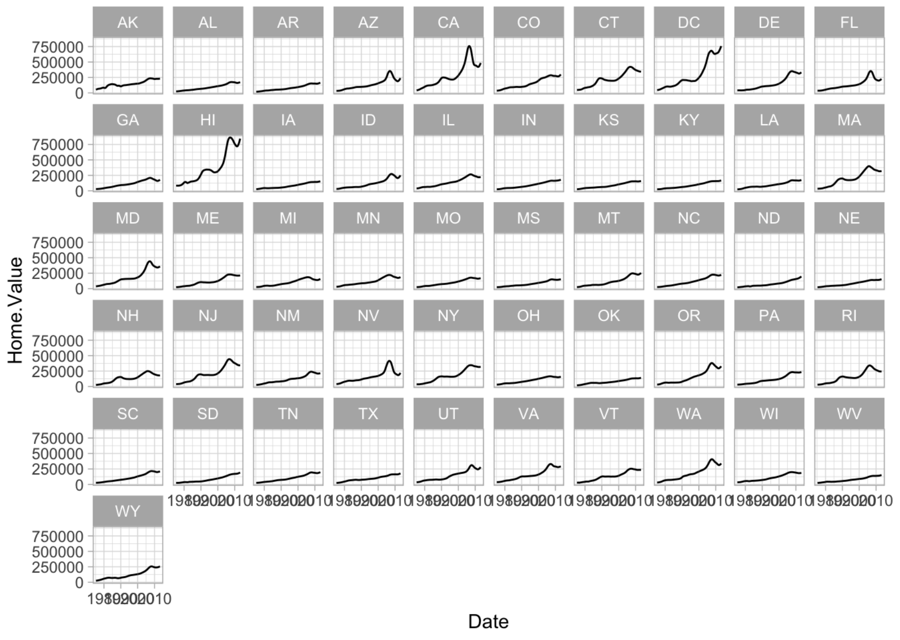
Built-in themes include:

- `theme_gray()` (default)
- `theme_bw()`
- `theme_classc()`

```
p5 + theme_linedraw()
```



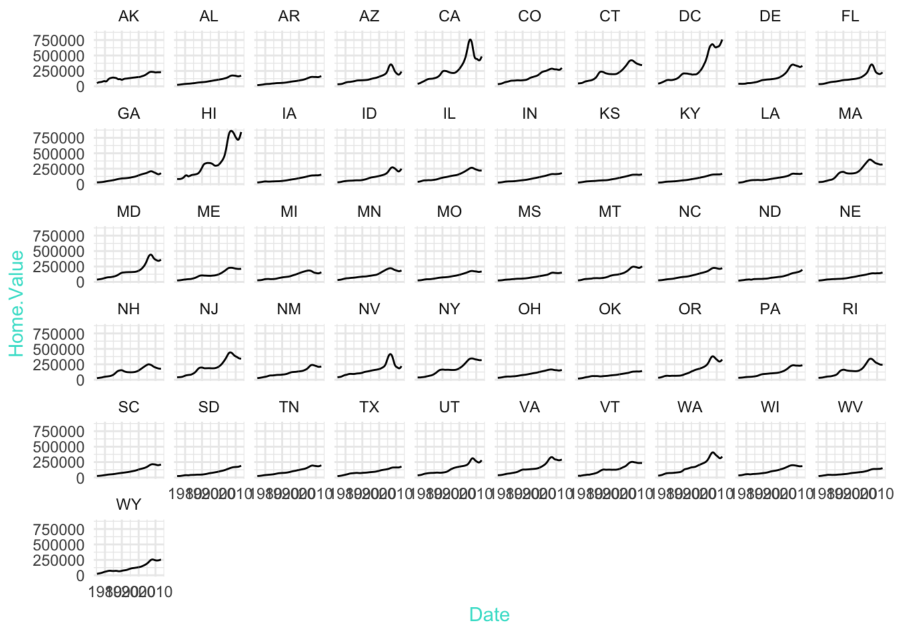
```
p5 + theme_light()
```



2. Overriding theme defaults

Specific theme elements can be overridden using `theme()`. For example:

```
p5 + theme_minimal() +
  theme(text = element_text(color = "turquoise"))
```



All theme options are documented in `?theme`.

VII. **Take Home Message**

- In general ggplot2 is a very powerful tool to create sophisticated graphs.
- There are several main parts of ggplot2:
  - Geometric Objects
  - Aesthetics
  - Statistical Transformations
  - Scales
  - Faceting
  - Themes
- By changing those parts, we can tweak the graph as desired. If not, it's time to consider other methods.

VIII. References

- ggplot2 resources
  - Datasets: http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html#introduction
  - Mailing list: http://groups.google.com/group/ggplot2
  - Wiki: https://github.com/hadley/ggplot2/wiki
  - Website: http://had.co.nz/ggplot2/
  - StackOverflow: http://stackoverflow.com/questions/tagged/ggplot
- IQSS resources
  - Research technology consulting: http://projects.iq.harvard.edu/rtc
  - Workshops: http://projects.iq.harvard.edu/rtc/filter_by/workshops