Lab 8

Stat C131A

In this lab, we will learn about heatmaps and PCA. The section on plotting in `ggplot` is optional.

Places rated dataset

The data were taken from the *Places Rated Almanac* which rates cities according to nine criteria. For all but two of the criteria, the higher the score, the better. For Housing and Crime, the lower the score the better. The following are descriptions of the criteria:

- Climate & Terrain: very hot and very cold months, seasonal temperature variation, heating- and cooling-degree days, freezing days, zero-degree days, ninety-degree days.
- Housing: utility bills, property taxes, mortgage payments.
- Health Care & Environment: per capita physicians, teaching hospitals, medical schools, cardiac rehabilitation centers, comprehensive cancer treatment centers, hospices, insurance/hospitalization costs index, flouridation of drinking water, air pollution.
- Crime: violent crime rate, property crime rate.
- Transportation: daily commute, public transportation, Interstate highways, air service, passenger rail service.
- Education: pupil/teacher ratio in the public K-12 system, effort index in K-12, accademic options in higher education.
- The Arts: museums, fine arts and public radio stations, public television stations, universities offering a degree or degrees in the arts, symphony orchestras, theatres, opera companies, dance companies, public libraries.
- Recreation: good restaurants, public golf courses, certified lanes for tenpin bowling, movie theatres, zoos, aquariums, family theme parks, sanctioned automobile race tracks, pari-mutuel betting attractions, major- and minor- league professional sports teams, NCAA Division I football and basketball teams, miles of ocean or Great Lakes coastline, inland water, national forests, national parks, or national wildlife refuges, Consolidated Metropolitan Statistical Area access.
- Economics: average household income adjusted for taxes and living costs, income growth, job growth.

Read Data.

```
place_rated <- read.csv("place.csv", stringsAsFactors = FALSE)
place_rated[, 1:9] <- scale(place_rated[, 1:9])
CA_NY_cities <- which(place_rated$state %in% c("CA", "NY"))
example_cities <- CA_NY_cities[c(10, 13, 15, 22, 24, 25, 26)]
row.names(place_rated) <- paste0(place_rated$city, place_rated$state)
```

Heatmap

Exercise 1

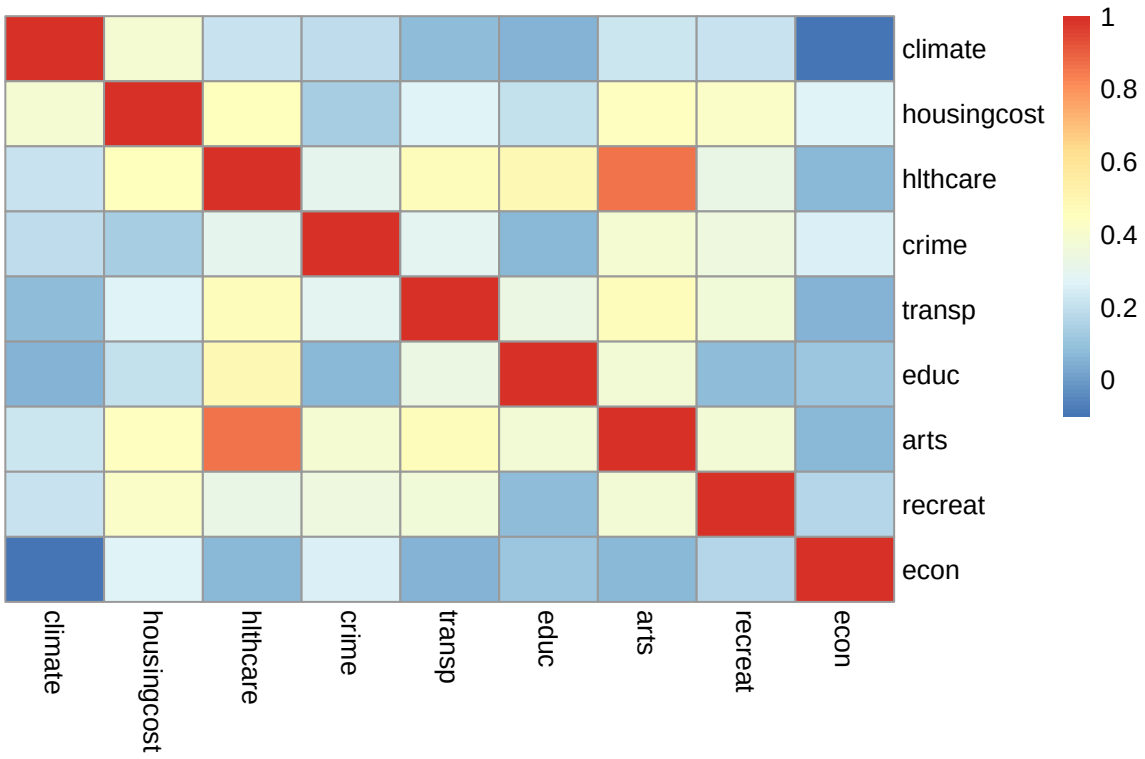
Heatmaps map numeric data to colors and are usually used to visualize correlation matrices (and other matrices)

(a) Calculate the correlation matrix between nine rating criteria using function `cor`. What is the correlation between arts and education?

```
# insert your code here and save the
# correlation matrix as `place_rated.cor`
# place_rated.cor <-
numeric.which <- which(sapply(place_rated, is.numeric))
place_rated.cor <- cor(place_rated[, 1:9])
```

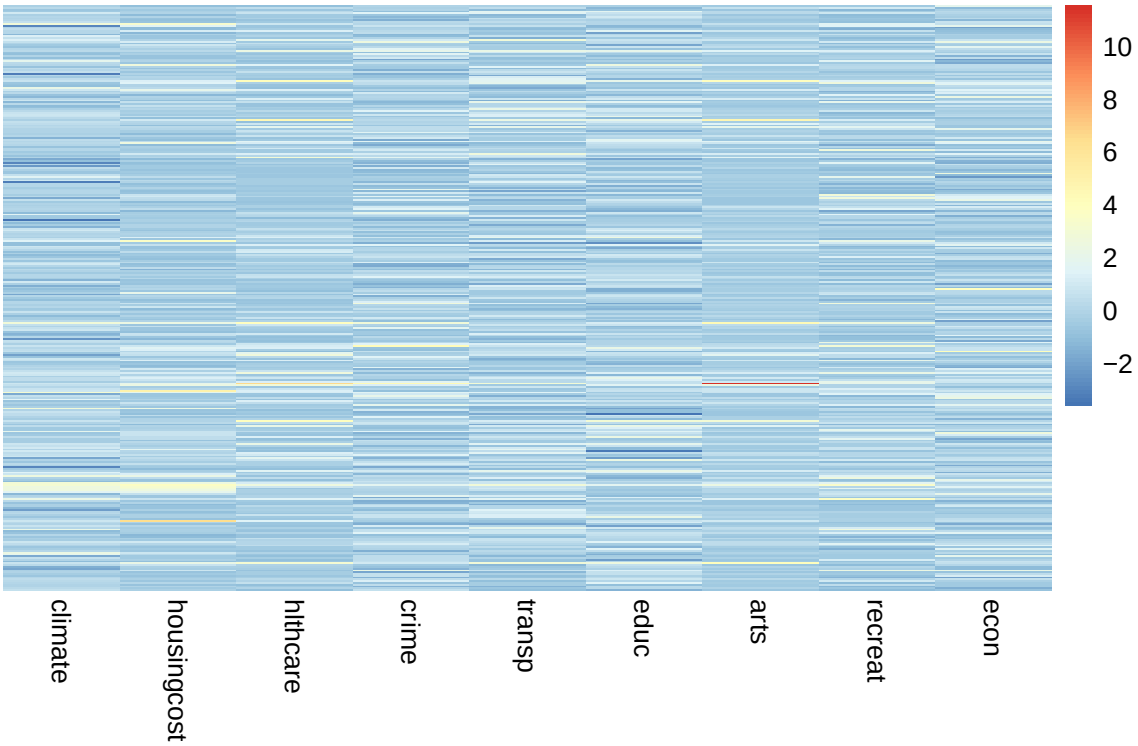
(b) Plot the heatmap for the correlation matrix using the `pheatmap()` functiomn from `pheatmap` package. If you are going to divide the nine rating criteria into two categories based on similarity, how would you split the variables?

```
# insert your code here for heatmap
library(pheatmap)
pheatmap(place_rated.cor,
         cluster_rows = F,
         cluster_cols = F)
```



(c) Plot the heatmap for data matrix.

```
# insert your code here for the heatmap
pheatmap(place_rated[, numeric.which],
         cluster_rows = F,
         cluster_cols = F,
        show_rownames = F)
```



Principal components analysis

Exercise 2

Principal components analysis allows you to convert a set of correlated variables into a set of linearly uncorrelated principal components. Each principal component is a linear combination of the original variables. The first principal component has the largest variance and the last principal component the least. You can think that the first principal component carries most information of the data (explains the largest proportion of data variance). That is why we usually use the first and the second PC to do visualization.

(a) Run PCA on the nine rating criteria using the `prcomp()` function. Save the result to `place.pca`.

```
# insert your code here to run pca,
# save your result as `place.pca`
place.pca <- prcomp(place_rated[, numeric.which])
place.pca
```

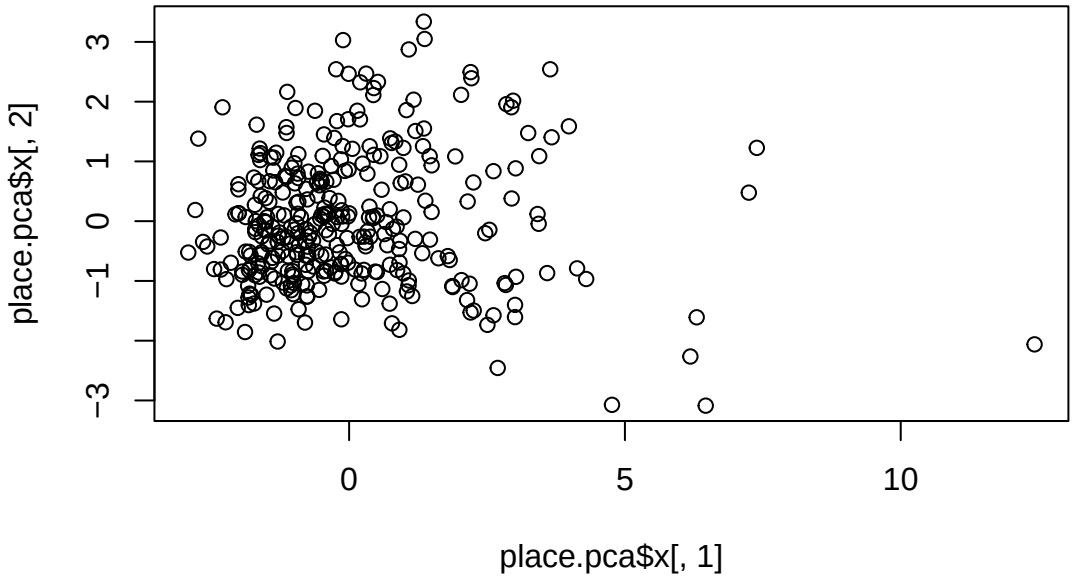
```
## Standard deviations (1, .., p=9):
## [1] 1.8461560 1.1018059 1.0684003 0.9596446 0.8679199 0.7940793 0.7021736
## [8] 0.5639490 0.3469900
##
## Rotation (n x k) = (9 x 9):
##                   PC1        PC2          PC3         PC4         PC5
## climate     0.2064140  0.2178353 -0.689955982  0.13732125 -0.3691499
## housingcost 0.3565216  0.2506240 -0.208172230  0.51182871  0.2334878
## hlthcare    0.4602146 -0.2994653 -0.007324926  0.01470183 -0.1032405
## crime       0.2812984  0.3553423  0.185104981 -0.53905047 -0.5239397
## transp      0.3511508 -0.1796045  0.146376283 -0.30290371  0.4043485
## educ        0.2752926 -0.4833821  0.229702548  0.33541103 -0.2088191
## arts        0.4630545 -0.1947899 -0.026484298 -0.10108039 -0.1050976
## recreat     0.3278879  0.3844746 -0.050852640 -0.18980082  0.5295406
```

```
## econ        0.1354123  0.4712833  0.607314475  0.42176994 -0.1596201
##                    PC6         PC7         PC8           PC9
## climate      0.37460469 -0.08470577 -0.36230833  0.0013913515
## housingcost -0.14163983 -0.23063862  0.61385513  0.0136003402
## hlthcare    -0.37384804  0.01386761 -0.18567612 -0.7163548935
## crime        0.08092329  0.01860646  0.43002477 -0.0586084614
## transp       0.46759180 -0.58339097 -0.09359866  0.0036294527
## educ         0.50216981  0.42618186  0.18866756  0.1108401911
## arts        -0.46188072 -0.02152515 -0.20398969  0.6857582127
## recreat      0.08991578  0.62787789 -0.15059597 -0.0255062915
## econ         0.03260813 -0.14974066 -0.40480926  0.0004377942
```

(b) Create a scatter plot of the first principal component versus the second principal component.

Hint. Similar to `lm`, `prcomp` gives you a list of objects. They are `sdev`, `rotation`, `center`, `scale` and `x`. You can access them using the dollar sign `$`. For now, you do not need to worry about what they are except `x`, which gives you a matrix with 9 columns (PCs).

```
# insert your code here for the scatter plot
plot(place.pca$x[,1],
     place.pca$x[,2])
```



For this data, we may not be able to observe separation of groups when plotting the first two PCs. However, there is more information we can examine by using the `biplot()` function. Besides creating scatter plots between PCs, the `biplot()` function also plots the weights of the linear combinations (loadings) when calculating principal components. We marked several cities in the output as follows.

```
# # Uncomment the code below after you get `place.pca`
# text_names = place_rated$city
# text_names[-example_cities] = 'o'
# biplot(place.pca, cex = 0.5, xlabs = text_names, xlim = c(-0.15, 0.4), ylim = c(-0.2, 0.2))
```

(c) Use `summary` function on your `place.pca` object. How many PCs will you choose in order to explain at least 80% of the data variance?

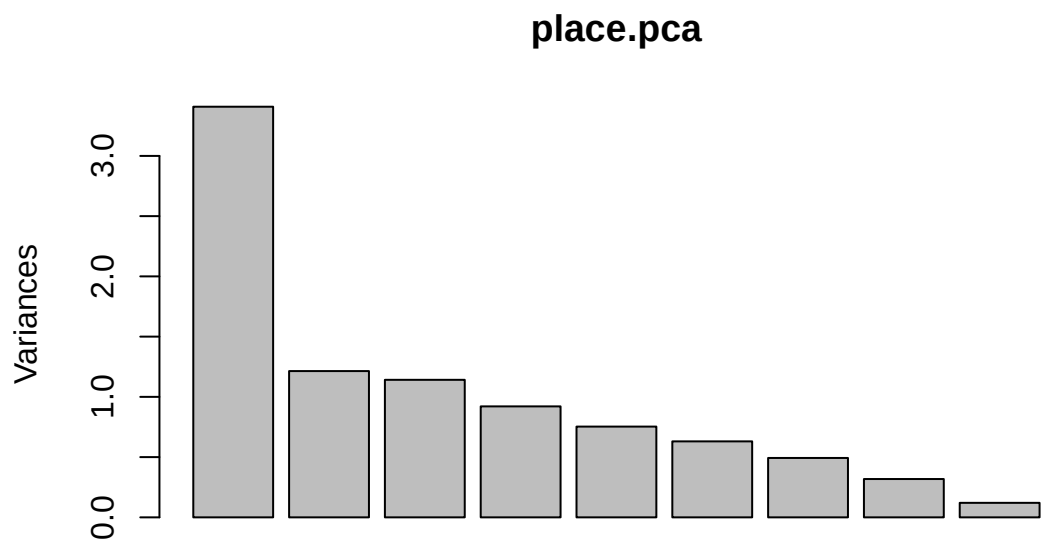
```
# insert your code here to summary your PCs
summary(place.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5     PC6     PC7
## Standard deviation     1.8462 1.1018 1.0684 0.9596 0.8679 0.79408 0.70217
## Proportion of Variance 0.3787 0.1349 0.1268 0.1023 0.0837 0.07006 0.05478
## Cumulative Proportion  0.3787 0.5136 0.6404 0.7427 0.8264 0.89650 0.95128
##                           PC8     PC9
## Standard deviation     0.56395 0.34699
## Proportion of Variance 0.03534 0.01338
## Cumulative Proportion  0.98662 1.00000
```

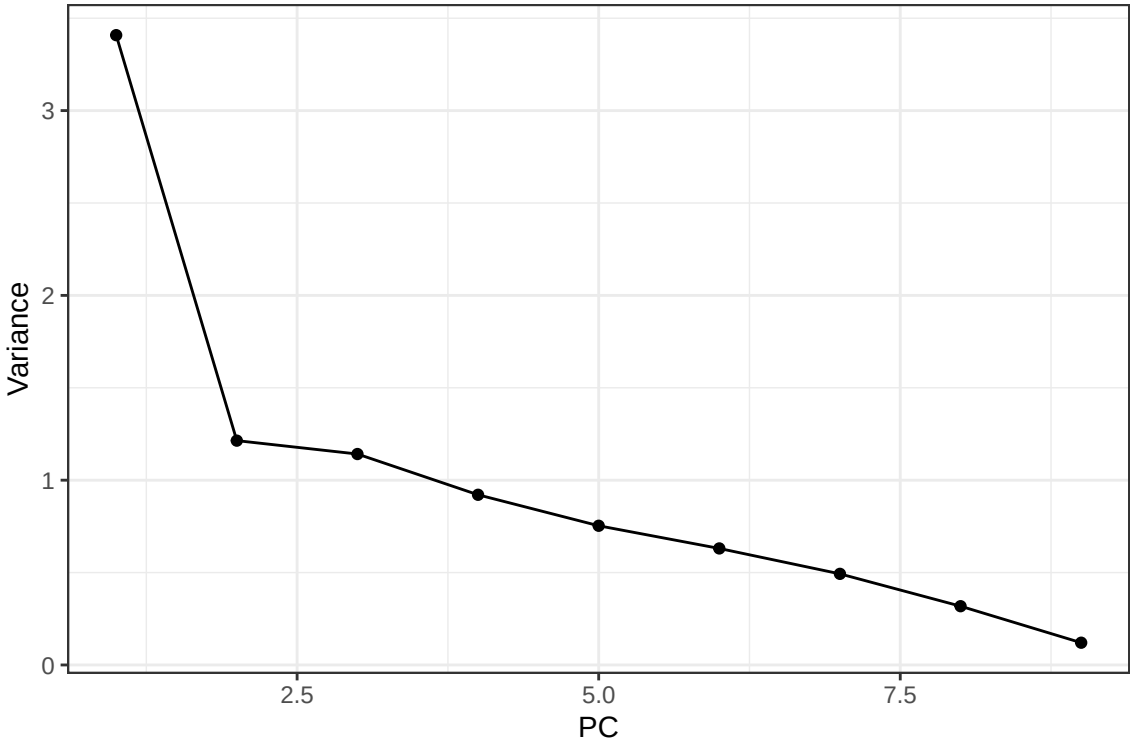
```
# How many PCs will you choose in order to explain at least 80% of the data variance?
# Please uncomment your answer
# nPC <- 2
# nPC <- 4
nPC <- 5
# nPC <- 8
```

(d) We can plot the scree plot or elbow plot simply by using the `plot()` function on our PCA object.

```
plot(place.pca)
```




```
suppressMessages(library(tidyverse))
data.frame(
  "PC" = 1:9,
  Variance = place.pca$sdev^2
) %>% ggplot(aes(x = PC, y = Variance))+geom_point()+geom_line()+theme_bw()
```


Variance
3
2
1
0
2.5
5.0
7.5
PC

A review of ggplot (optional)

ggplot is probably the most widely used plotting package in R. It creates publication-quality graphics with much cleaner and explicit arguments (compare to functions in the base package). It is especially user-friendly when dealing with data with categorical variables. ggplot allows you to create plots layer by layer, making it possible to create sophisticated plots.

In this section, we will create several plots using ggplot with our familiar craigslist dataset. These are very simple examples of what ggplot can do. If you are interested, you can refer to the documentation for ggplot(http://docs.ggplot2.org/current/).

```
library(ggplot2)
craigslist <- read.csv("craigslist.csv")
craigslist <- craigslist[-which(craigslist$size >= 30000), ]
```

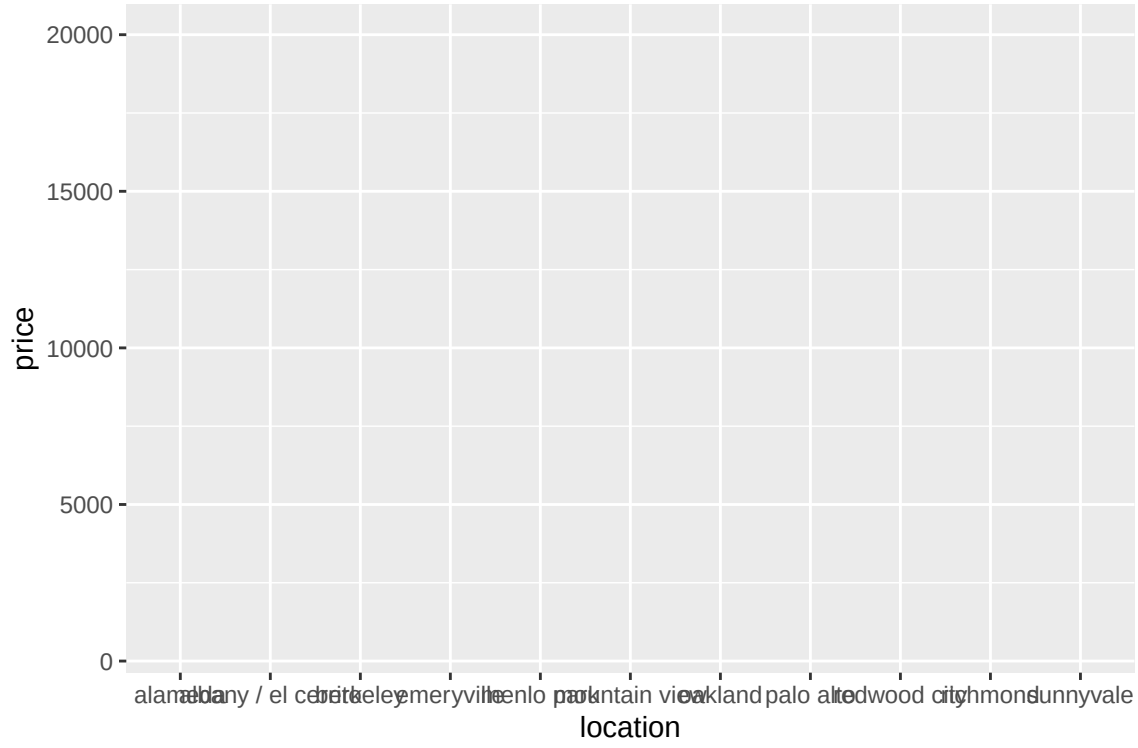
Example 1: boxplot and violin plots

First, we create a plot using ggplot().

```
p <- ggplot(
    craigslist, # specify the data frame
    aes(x = location, # variable in the x-axis, a categorical variable in craigslist
            y = price)) # variable in the y-axis, a continuous variable in craigslist
```

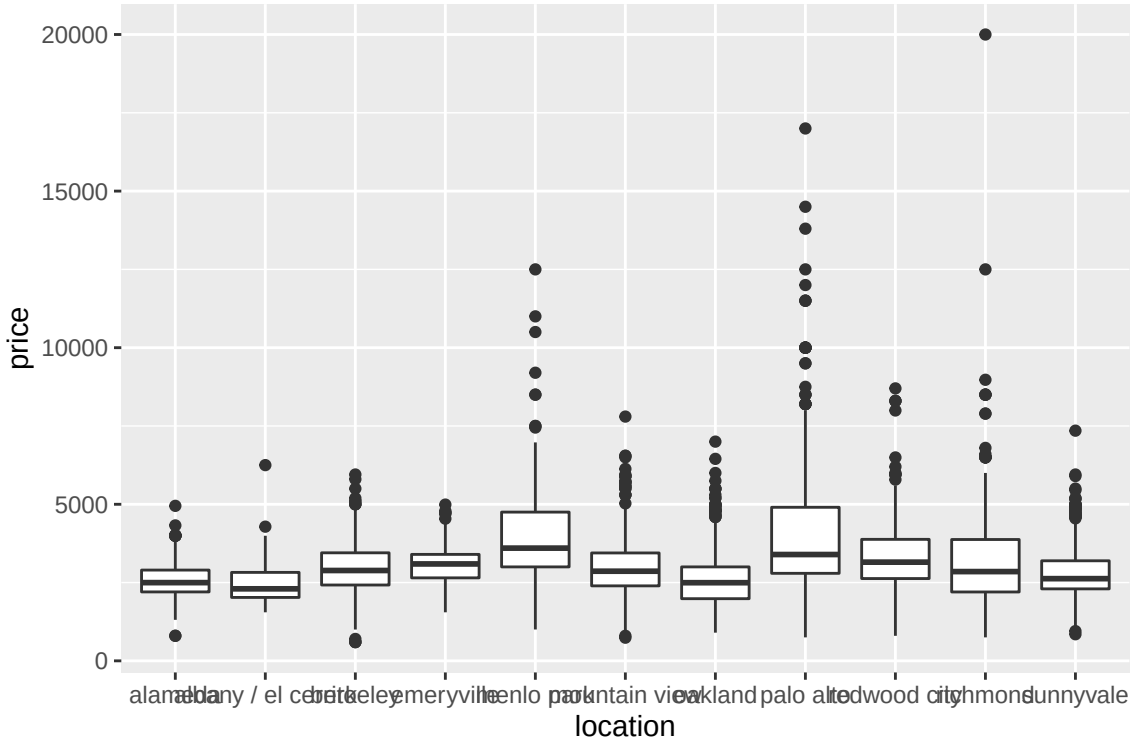
This first step initializes the plot. If you print p, you will see nothing other than the axis and the grid! This is because you haven't told ggplot what to plot.

```
p
```



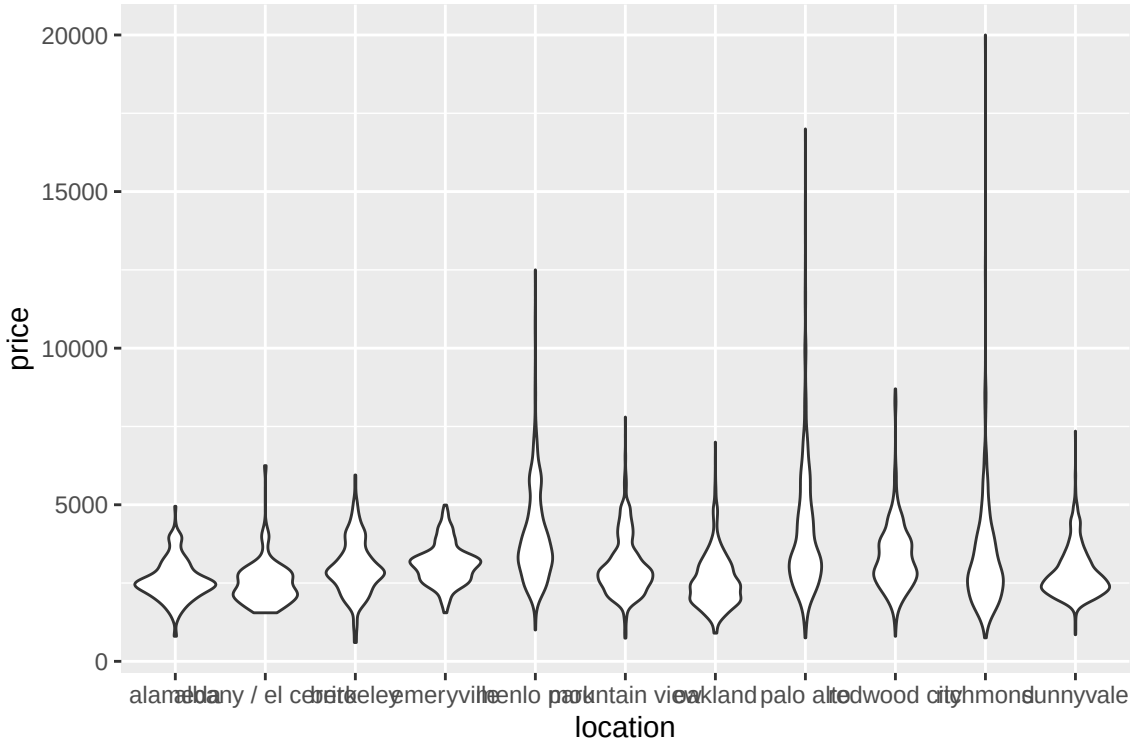
Once you initialized a plot, you are now able to add layers using various geoms. For example, to add boxplots to p, we use geom_boxplot().

```
p + geom_boxplot()
```



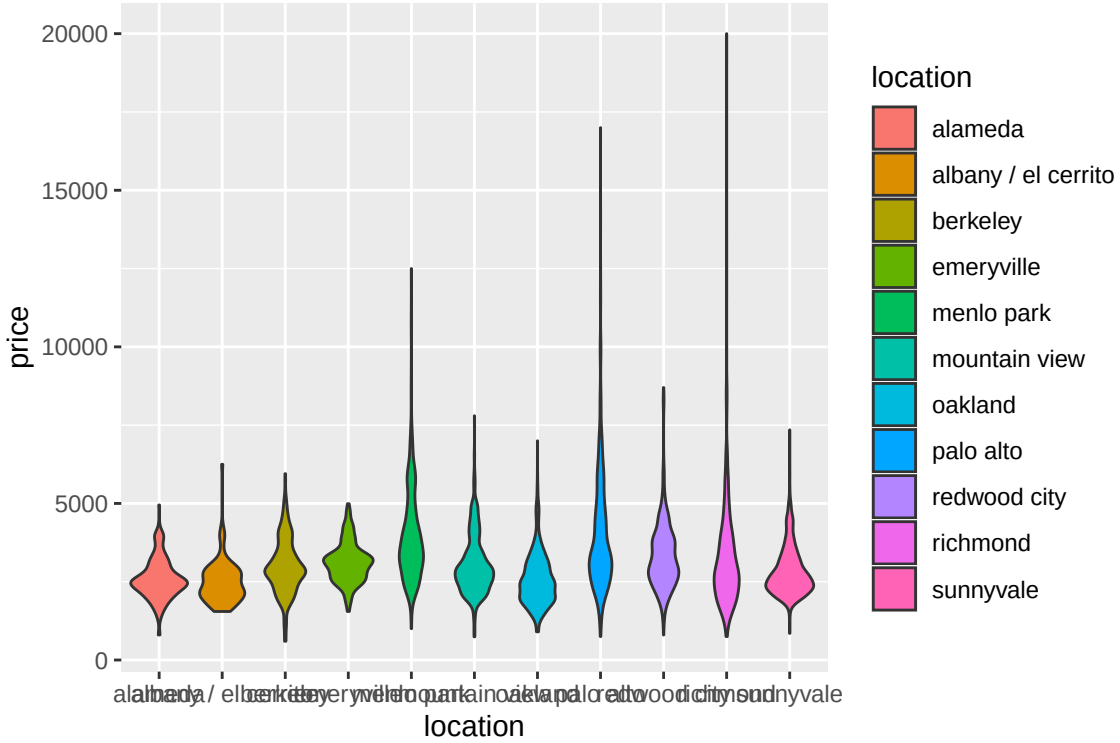
To add violin plots to `p`, we would use `geom_violin()`.

```
p + geom_violin()
```



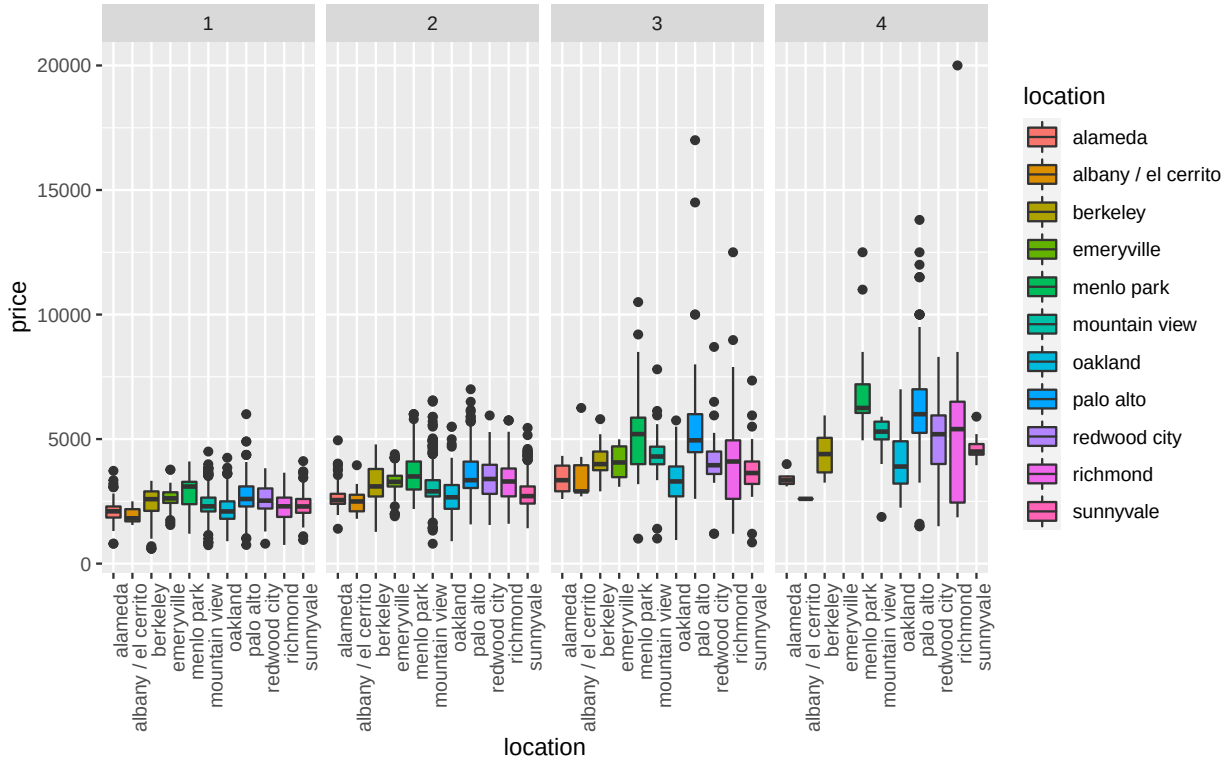
You can specify the fill color using the `fill` argument; `ggplot` will automatically create the legend for you.

```
p + geom_violin(aes(fill = location))
```



`ggplot` makes life easier when there are multiple categories.

```
p + geom_boxplot(aes(fill = location)) +
    facet_grid(.~brs) + # use facet by number of bedrooms
    # you can do facet_grid(brs~.) as well, which gives you 4 rows of plots instead of 4 columns
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) # rotate the x labels
```



Example 2: histogram and density plots

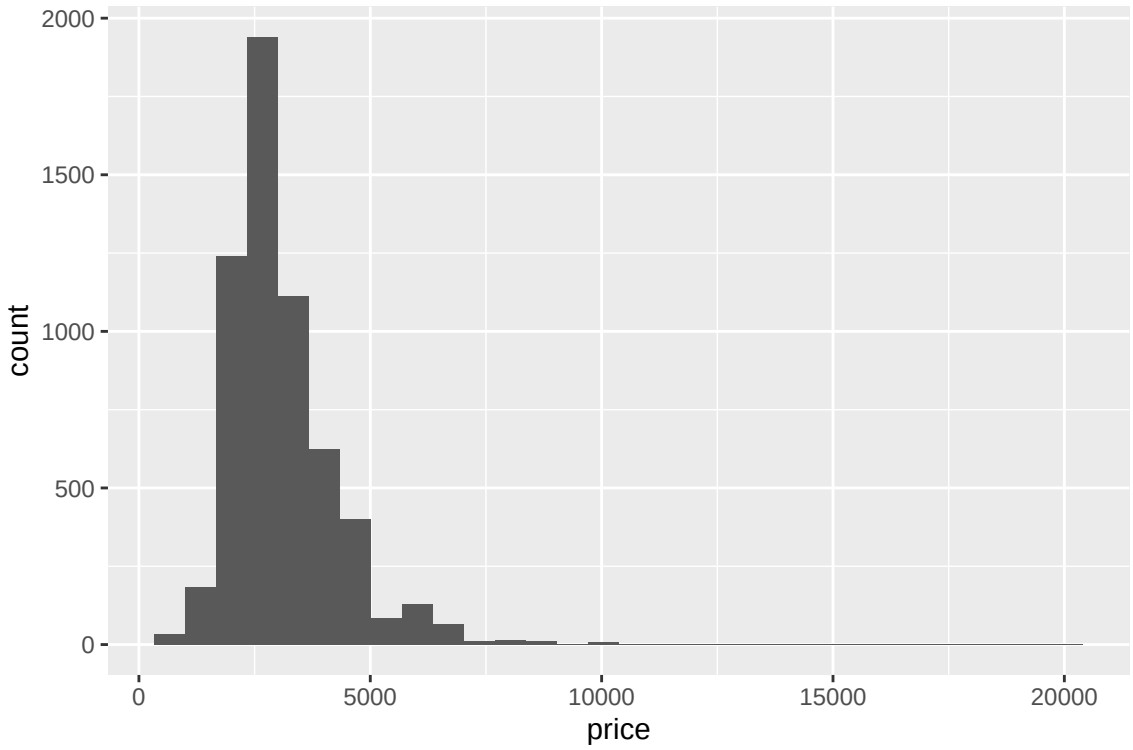
In histograms and density plots, the variable on the horizontal axis is from the data frame and that on the vertical axis is the counts or density. In such cases, we only need to specify argument `x` of `aes()` when initializing a plot with `ggplot`

```
p2 <- ggplot(craigslist,
             aes(x = price))
```

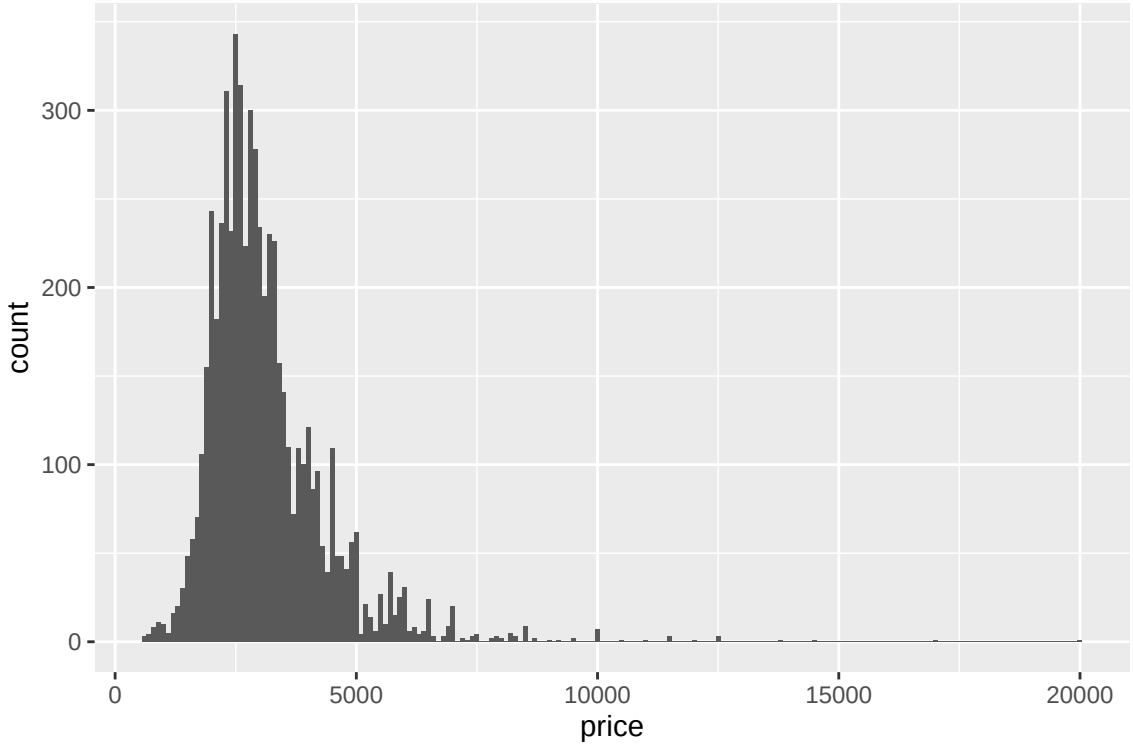
To add a histogram to `p2`, we would use `geom_histogram()`.

```
# A frequency histogram with the default bin width.
p2 + geom_histogram()
```

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

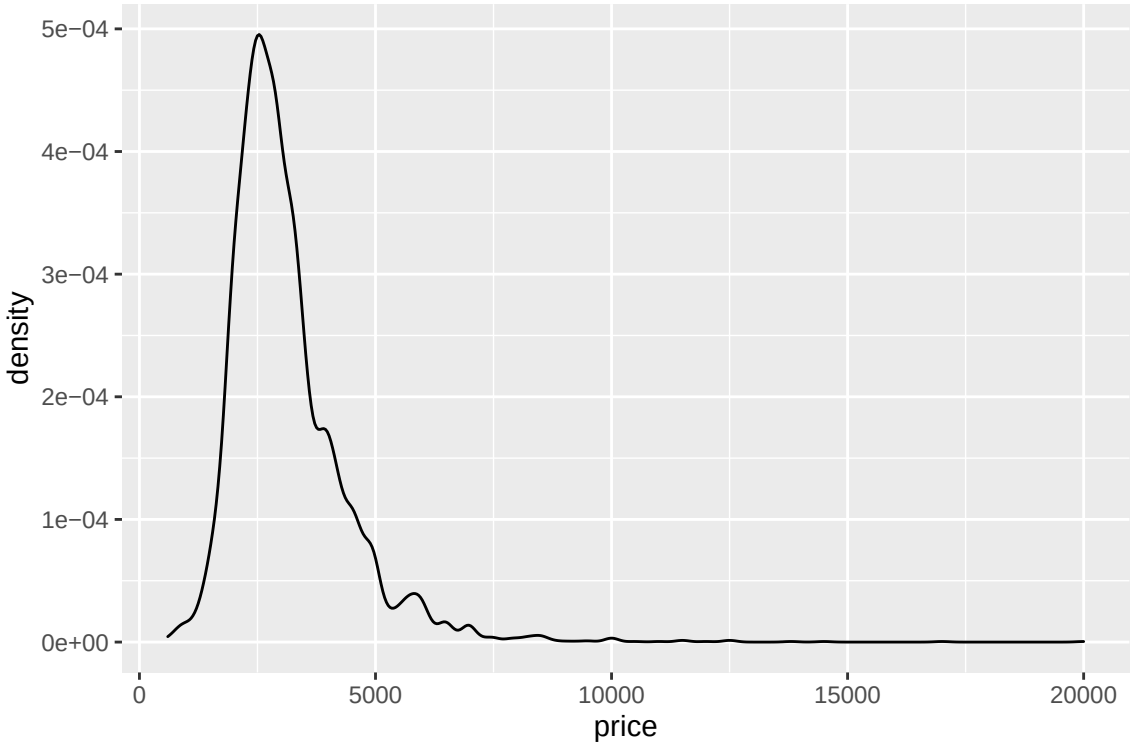



```
# A frequency histogram with bin width equal to 100
p2 + geom_histogram(binwidth = 100)
```



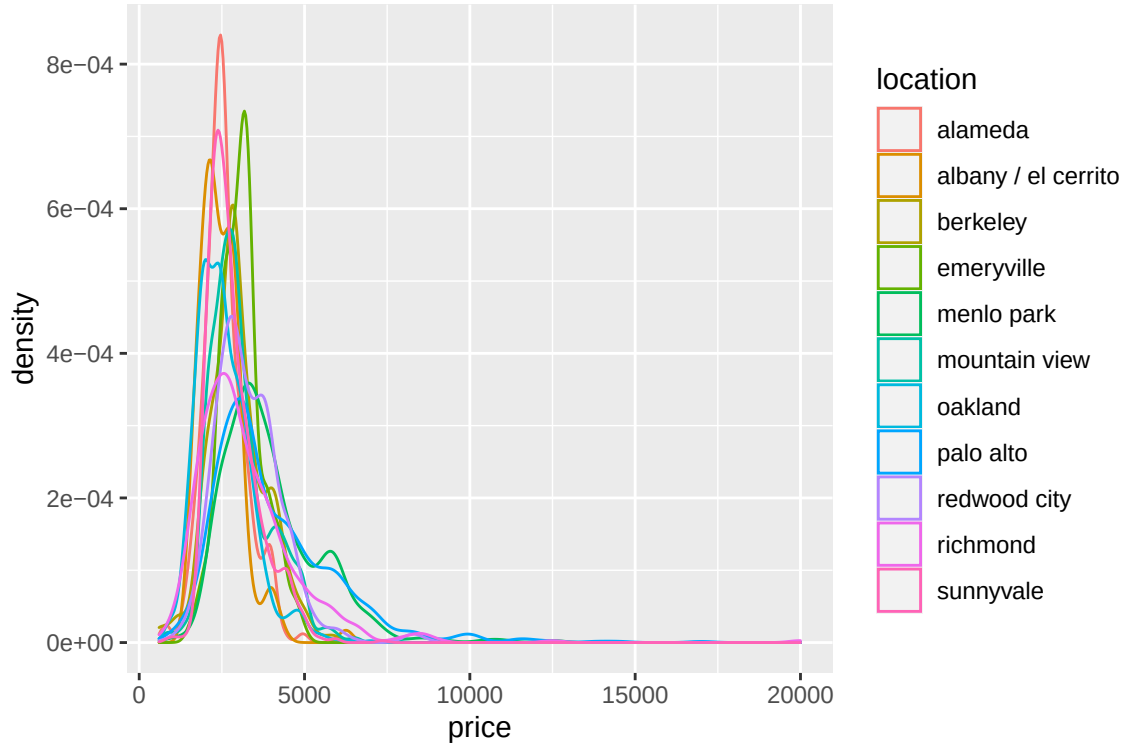
To add a density plot to `p2`, we would use `geom_density()`.

```
p2 + geom_density()
```



We could also plot densities by group.

```
p2 + geom_density(aes(colour = location))
```



Example 3: scatter plot and smooth curves

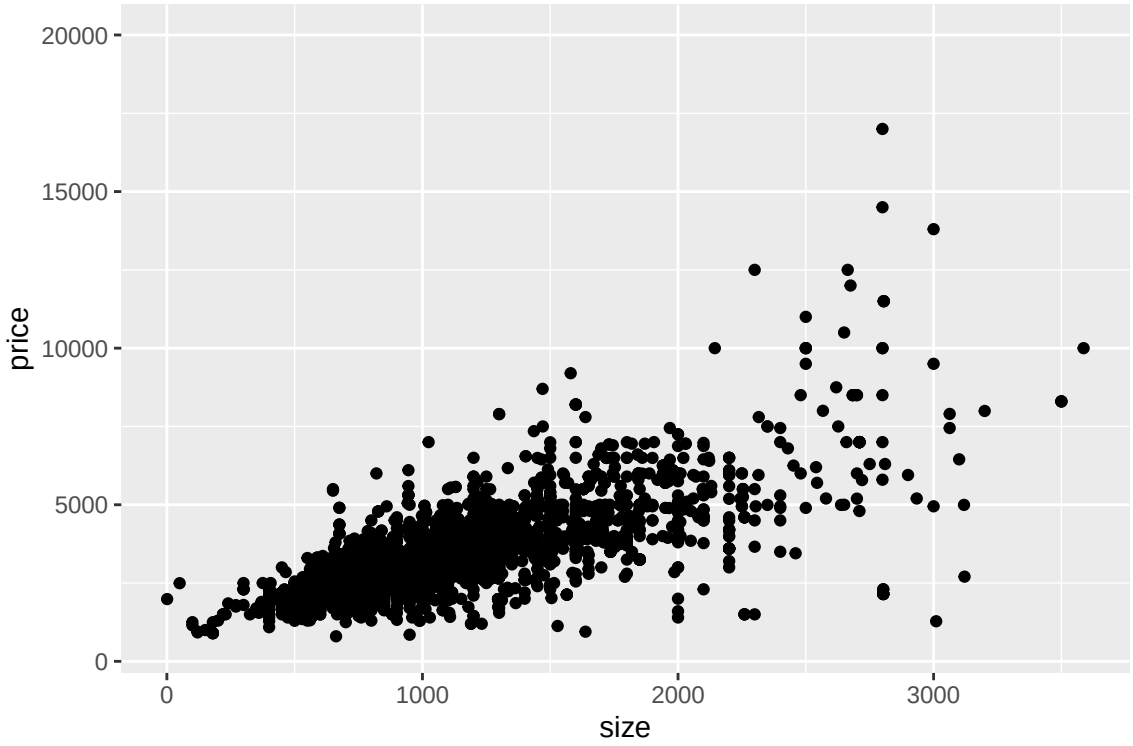
As usual, we initialize our plot by setting `x = size` and `y = price`.

```
p3 <- ggplot(craigslist, aes(x = size, y= price))
```

Add a scatter plot layer. There are missing values in the variable `size`. `ggplot` will ignore rows with missing values, but may print a warning.

```
p3 + geom_point()
```

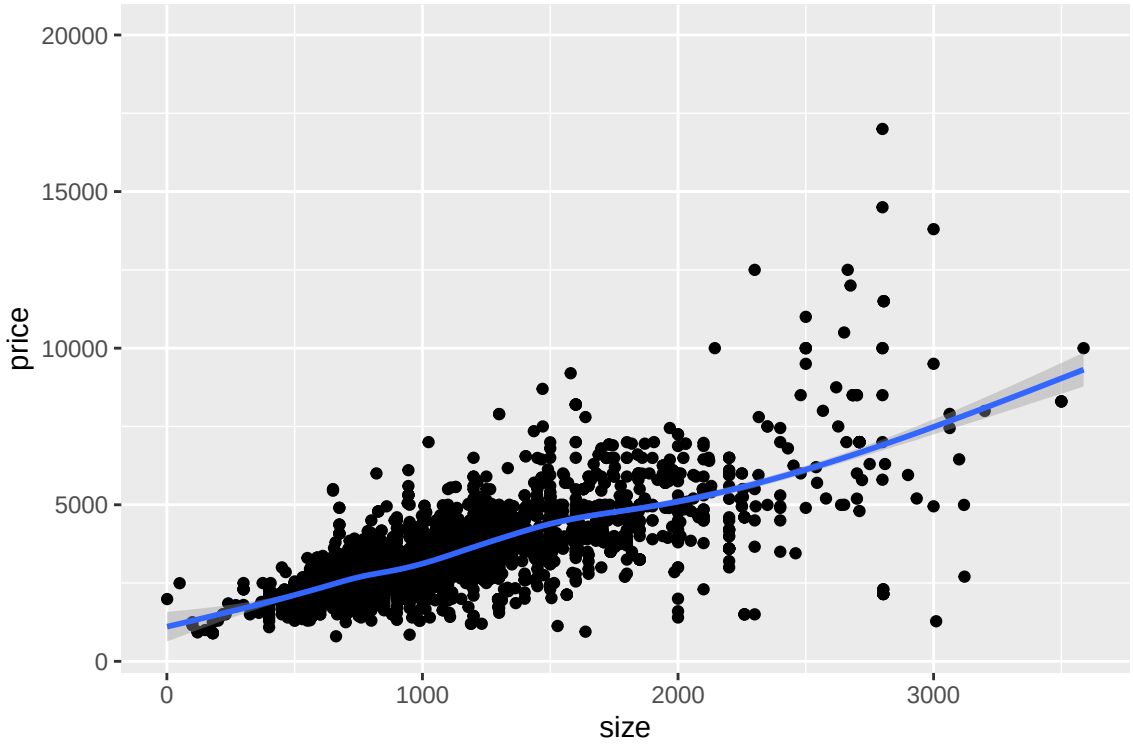
```
## Warning: Removed 1492 rows containing missing values (geom_point).
```



Add a smoothed fit layer.

```
p3 + geom_point() + geom_smooth()
```

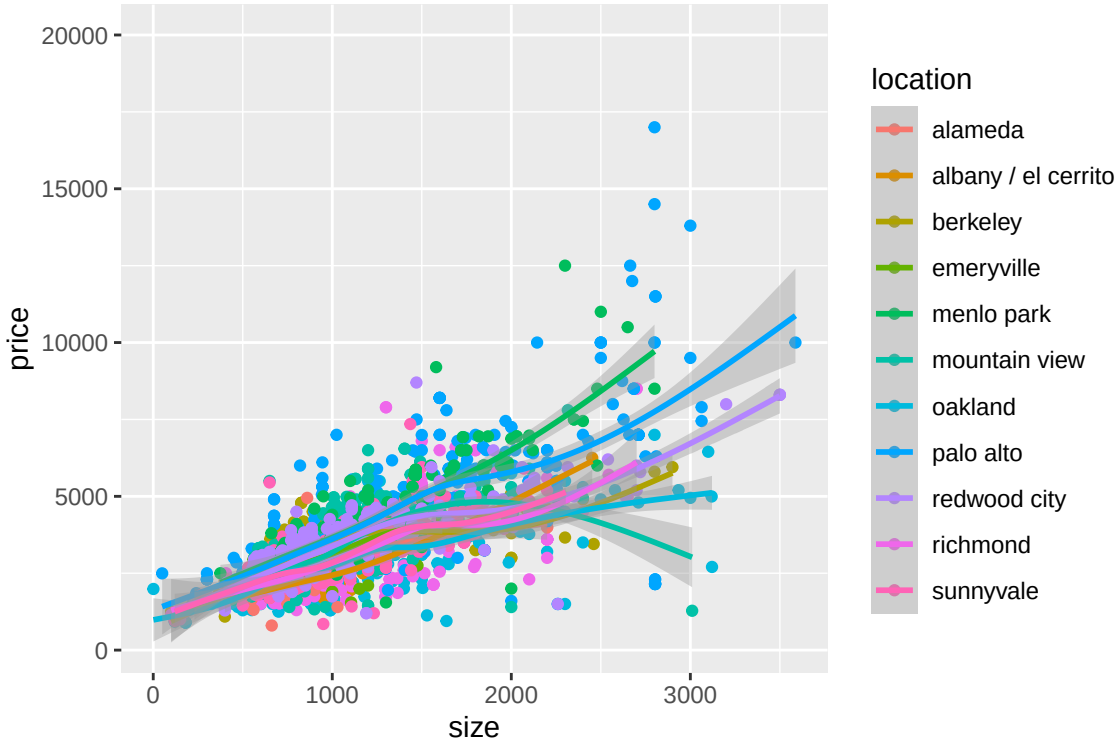
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 1492 rows containing non-finite values (stat_smooth).
## Warning: Removed 1492 rows containing missing values (geom_point).
```



Plot the curve by group.

```
p3 + geom_point(aes(colour=location)) + geom_smooth(aes(colour=location))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 1492 rows containing non-finite values (stat_smooth).
## Warning: Removed 1492 rows containing missing values (geom_point).
```



Exercise 3

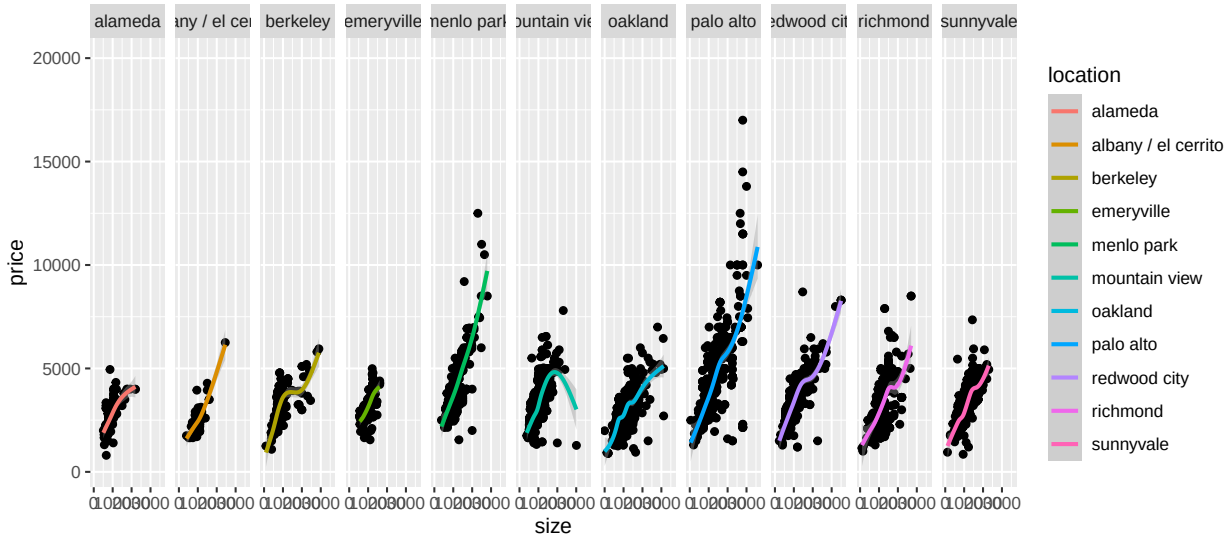
Draw the scatter plot and smooth curve. Use `facet_grid()` to group by location. Arrange these plots so that they are plotted side by side. Leave the points black and color the smooth curve by group using the `location` variable.

```
p3 + geom_point() + geom_smooth(aes(color = location)) +
  facet_grid(. ~ location)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 1492 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1492 rows containing missing values (geom_point).
```



Example 4: organize multiple plots in one single plot

One equivalence of `par(mfrow) = c(, )` in `ggplot` is `grid.arrange()` function in the package `gridExtra`.

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(p2 + geom_histogram() + ggtitle("Histogram"),
             p2 + geom_density() + ggtitle("Density plot"),
             p3 + geom_point() + ggtitle("Scatter plot"),
             p3 + geom_point() + geom_smooth() + ggtitle("Scatter plot with smooth curve"),
             nrow = 2, ncol = 2, # plot 2 by 2 grid
             top = "Examples for organizing  plots")
```

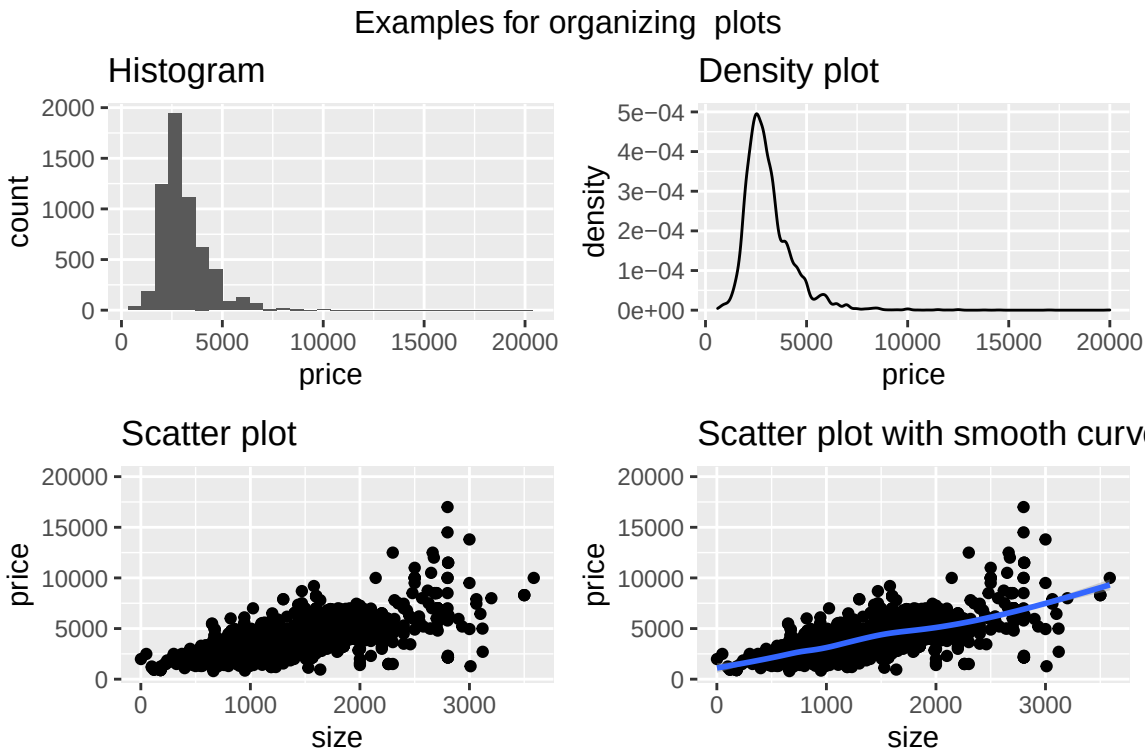
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1492 rows containing missing values (geom_point).
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 1492 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1492 rows containing missing values (geom_point).
```



- Reference: Build a plot layer by layer