STAT 408 - R Overview

January 11, 2018

R Intro

Why use R?

R is:

- a free public domain implementation of S,
- the standard among (academic) professional statisticians,
- available for Windows, Mac, and Linux,
- an object-oriented and functional programming structure, and
- designed to connect to high-performance programming languages like C and Fortran.

Why use R?

R has:

- an open-software environment with a large community that makes getting help easy,
- a massive set of packages for statistical modeling, data science, visualization, and importing and manipulating data,
- powerful tools for replicating and communicating your results,
- an interactive development environment (R Studio) tailored for interactive data analysis and statistical programmming,
- available for Windows, Mac, and Linux, and
- an object-oriented and functional programming structure.

Reading Data files

The ability to datasets into R is an essential skill. For this class, most of the files will be on the course webpage and can be directly downloaded using `read.csv`. Consider a dataset available at: `http://math.montana.edu/ahoegh/teaching/stat408/datasets/SeattleHousing.csv`

```
Seattle <- read.csv(
  'http://math.montana.edu/ahoegh/teaching/stat408/datasets/SeattleHousing.csv',
  stringsAsFactors = F)
```

Viewing Data files

A common function that we will use is `head`, which shows the first few rows of a data frame.

```
head(Seattle)
```

```
##     price bedrooms bathrooms sqft_living sqft_lot floors waterfront
## 1 1350000        3      2.50        2753    65005    1.0          1
## 2  228000        3      1.00        1190     9199    1.0          0
## 3  289000        3      1.75        1260     8400    1.0          0
## 4  720000        4      2.50        3450    39683    2.0          0
## 5  247500        3      1.75        1960    15681    1.0          0
## 6  850830        3      2.50        2070    13241    1.5          0
##   sqft_above sqft_basement zipcode     lat     long yr_sold mn_sold
## 1       2165           588   98070 47.4041 -122.451    2015       3
## 2       1190             0   98148 47.4258 -122.322    2014       9
## 3       1260             0   98148 47.4366 -122.335    2014       8
## 4       3450             0   98010 47.3420 -122.025    2015       3
## 5       1960             0   98032 47.3576 -122.277    2015       3
## 6       1270           800   98102 47.6415 -122.315    2014       6
```

R Data Structures

Data structure Overview

R has four common types of data structures:

- Vectors
- Matrices (and Arrays)
- Lists
- Data Frames

Data structure Overview

The base data structures in R can be organized by dimensionality and whether they are homogenous.

Dimension	Homogenous	Heterogenous
1d	Vector	List
2d	Matrix	Data Frame
no d	Array	

Vector Types

There are four common types of vectors: logical, integer, double (or numeric), and character. The `c()` function is used for combining elements into a vector

```
dbl <- c(1,2.5,pi)
int <- c(1L,4L,10L)
log <- c(TRUE,FALSE,F,T)
char <- c('this is','a character string')
```

Vector Types

They type of vector can be identified using the `typeof()` function. Note that only a single data type is allowed.

```
  typeof(dbl)
```

```
## [1] "double"
```

```
  comb <- c(char,dbl)
  typeof(comb)
```

```
## [1] "character"
```

```
  comb
```

```
## [1] "this is"                "a character string" "1"
## [4] "2.5"                    "3.14159265358979"
```

Exercise: Vectors

Create a vector with your first, middle, and last names.

Solution: Vectors

1. Create a vector with your first, middle, and last names.

```
andy.names <- c("Andrew","Blake","Hoegh")
andy.names
```

```
## [1] "Andrew" "Blake"  "Hoegh"
```

Data Frame Overview

A data frame:

- is the most common way of storing data in R
- is like a matrix with rows-and-column structure; however, unlike a matrix each column may have a different mode
- in a technical sense, a data frame is a list of equal-length vectors.

```
df <- data.frame(x = 1:3, y = c('a','b','c'))
kable(df)
```

x	y
1	a
2	b
3	c

Subsetting

Vector Subsetting: I

Subsetting allows you to extract certain elements from a data frame or vector (or matrix, array, list).

```
num.vec <- seq(from = 1, to = 9, by = 1); num.vec
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

```
num.vec[1:3]
```

```
## [1] 1 2 3
```

```
num.vec[c(1,5,8)]
```

```
## [1] 1 5 8
```

```
num.vec[-5]
```

```
## [1] 1 2 3 4 6 7 8 9
```

Vector Subsetting: II

Subsetting also works with logical values or expressions.

```
num.vec[num.vec > 5]
```

```
## [1] 6 7 8 9
```

```
num.vec[num.vec != 6]
```

```
## [1] 1 2 3 4 5 7 8 9
```

```
num.vec[rep(c(TRUE,FALSE,TRUE),each=3)]
```

```
## [1] 1 2 3 7 8 9
```

Data Frame Subsetting: I

The same ideas apply to data frames, but the indices now constitute rows and columns of the data frame.

```
df <- data.frame(x=1:3, y=3:1, z=c('a','b','c'))
df[,1]
```

```
## [1] 1 2 3
```

```
df[-1,c(2:3)]
```

```
##   y z
## 2 2 b
## 3 1 c
```

Data Frame Subsetting: II

There are also a couple built in functions in R for subsetting data frames.

```
df$x
```

```
## [1] 1 2 3
```

```
new.df <- subset(df, x >1); new.df
```

```
##   x y z
## 2 2 2 b
## 3 3 1 c
```

Exercise: Subsetting

1. Create a new data frame that only includes houses worth more than $1,000,000.
2. (bonus) From this new data frame what is the average living square footage of houses. Hint columns in a data.frame can be indexed by `Seattle$sqft_living`

Exercise: Subsetting - Solutions

1. Create a new data frame that only includes houses worth more than $1,000,000.

```
expensive.houses <- subset(Seattle, price > 1000000)
```

2. (bonus) From this new data frame what is the average living square footage of houses. Hint columns in a data.frame can be indexed by `Seattle$sqft_living`

```
mean(expensive.houses$sqft_living)
```

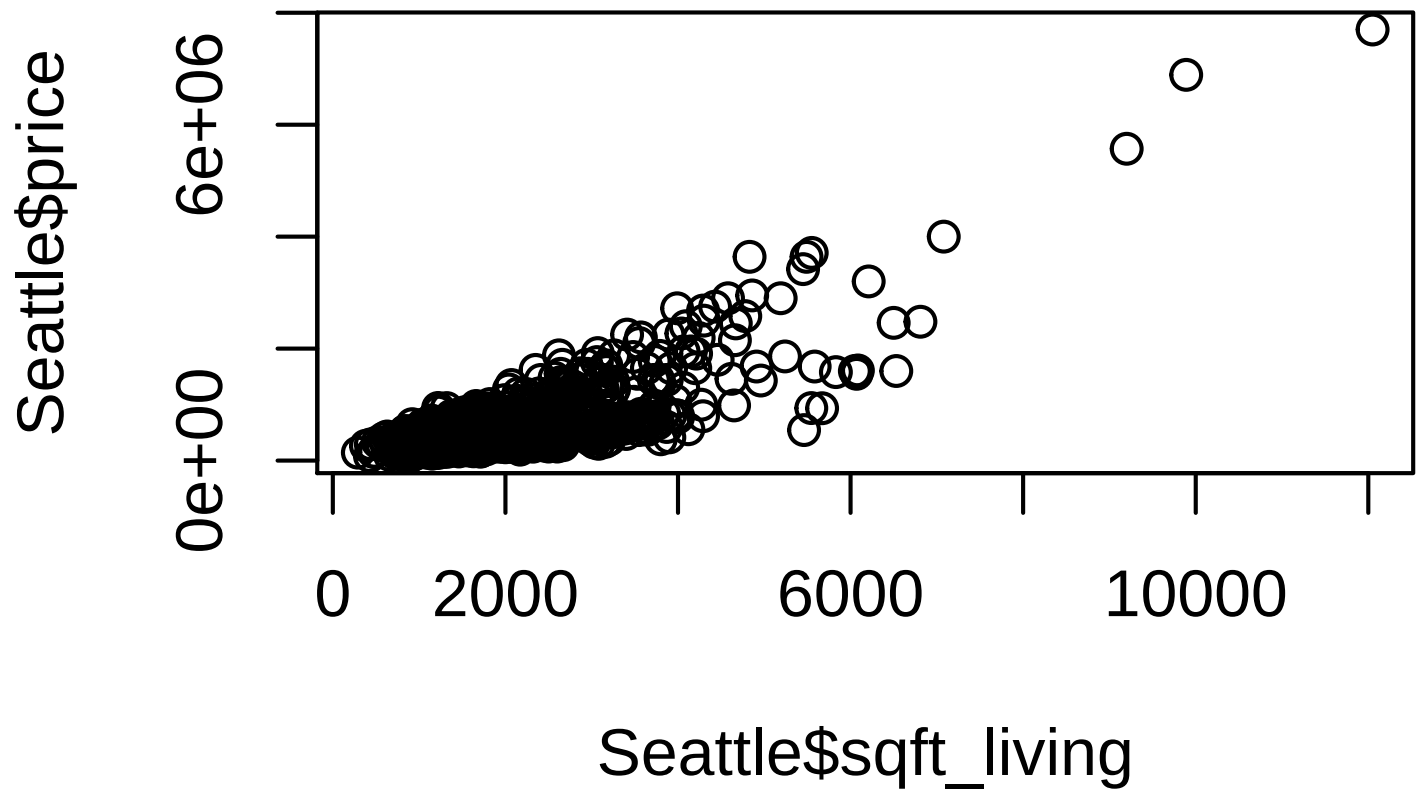
```
## [1] 3890.065
```

Graphics

Basic Plotting in R: Scatterplot

Later in the course, we will spend considerable time on graphics. For now, let's consider some of the basic functionality in R.

```
plot(Seattle$price~Seattle$sqft_living)
```

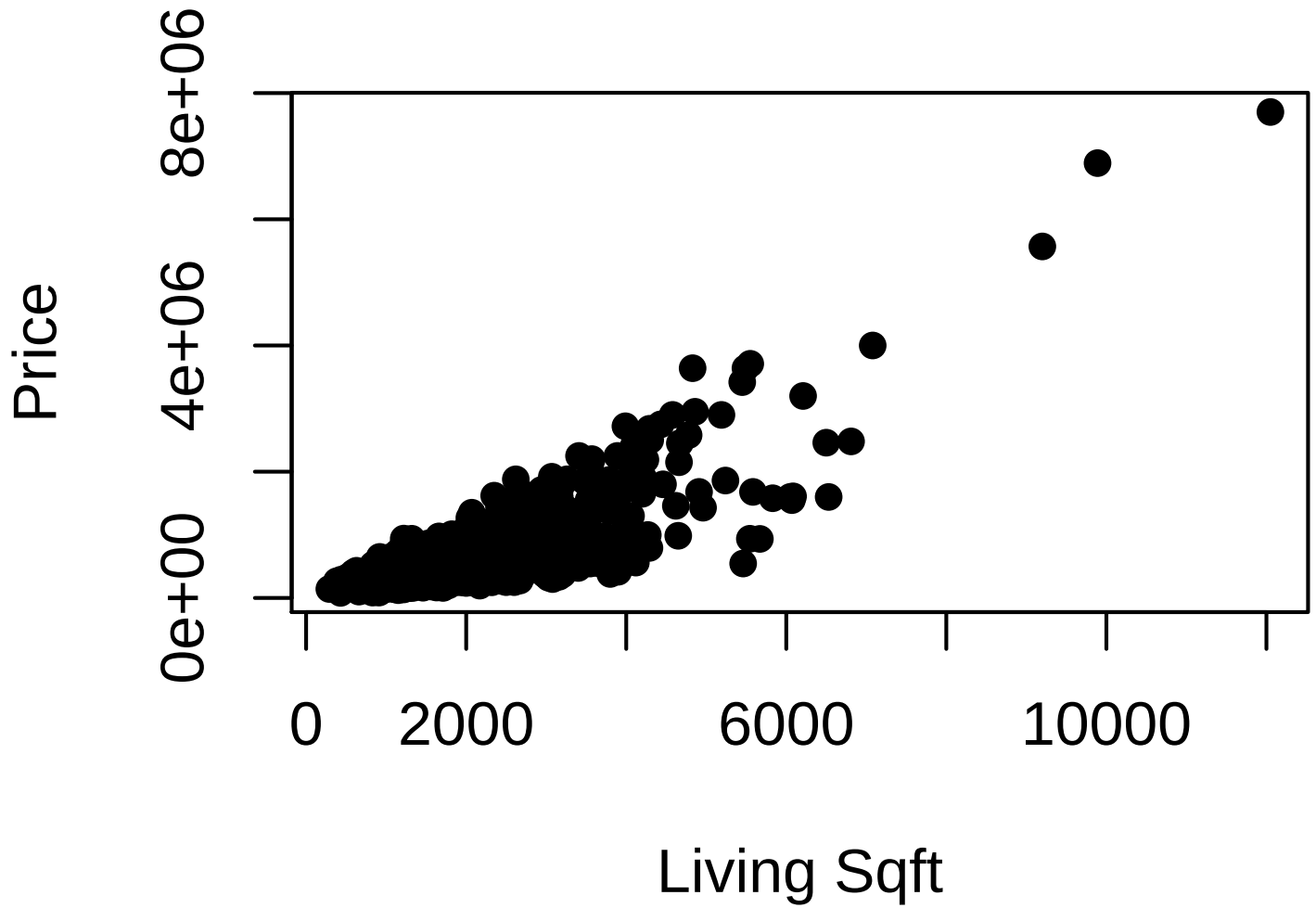


Basic Plotting in R: labels


```
plot(Seattle$price~Seattle$sqft_living,
     ylab='Price',xlab='Living Sqft')
```

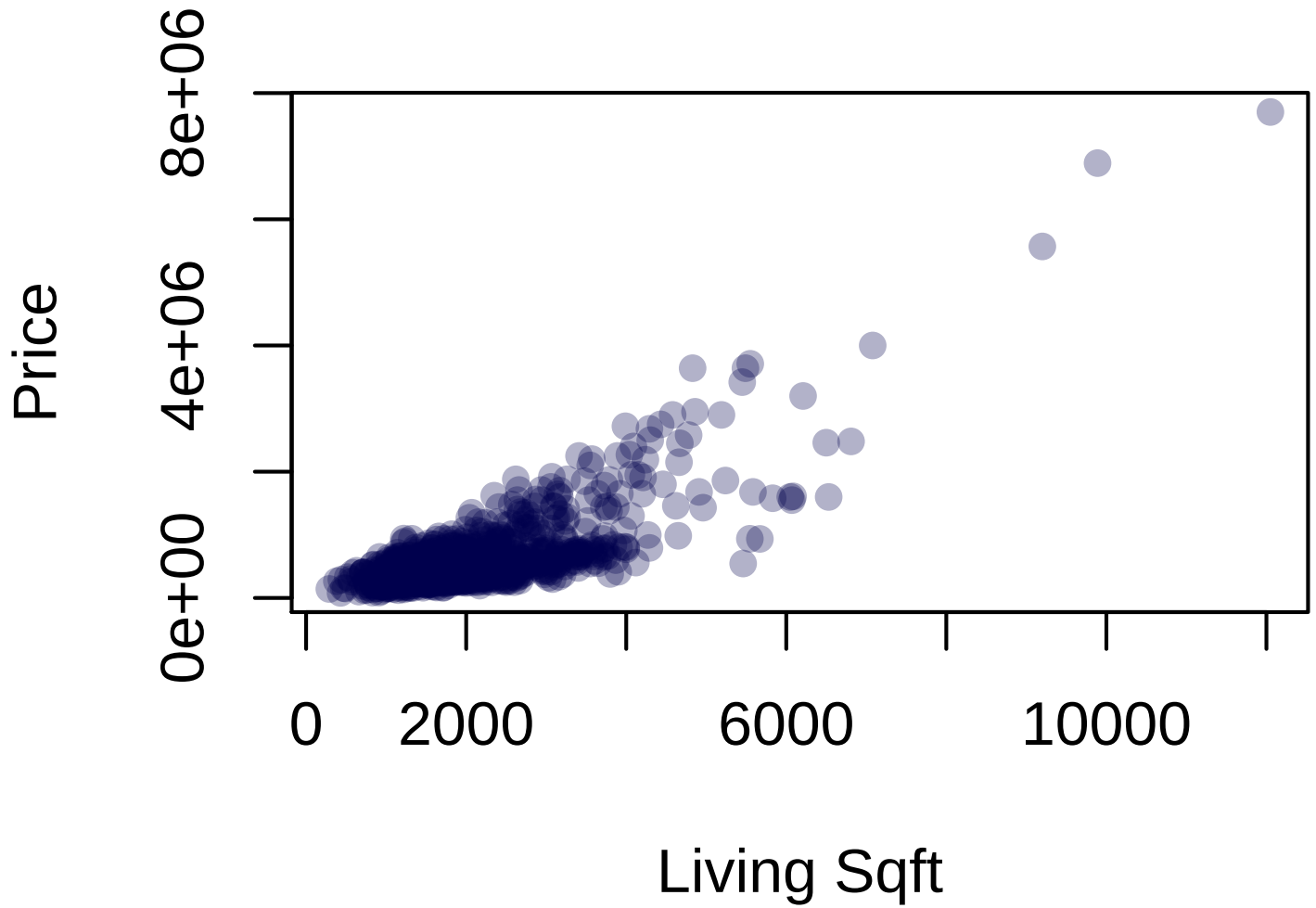
Basic Plotting in R: pch


```
plot(Seattle$price~Seattle$sqft_living,
     ylab='Price',xlab='Living Sqft', pch=16)
```



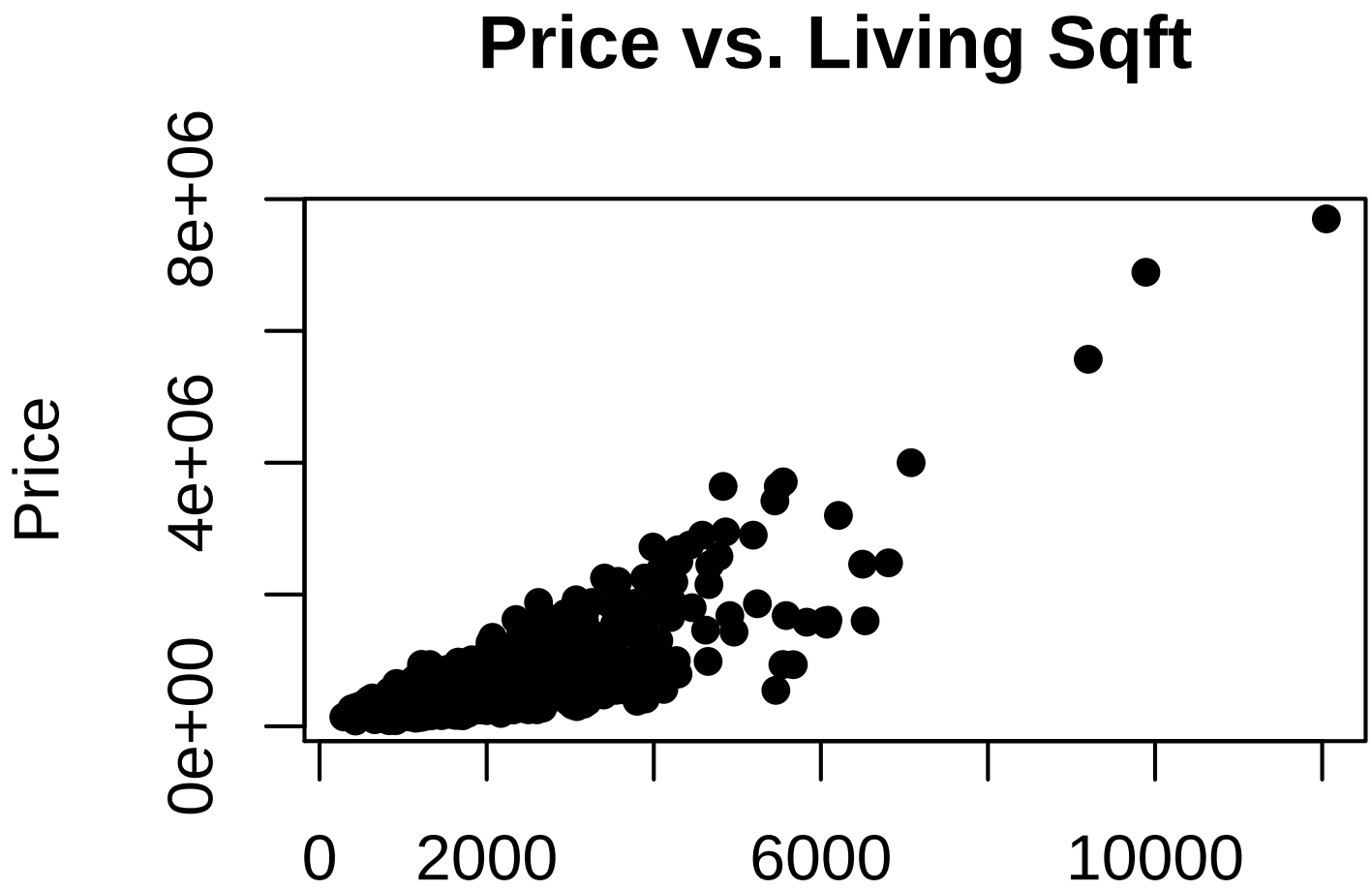
Basic Plotting in R: color


```
plot(Seattle$price~Seattle$sqft_living, pch=16,
     col=rgb(0,0,.3,.3),ylab='Price',xlab='Living Sqft')
```



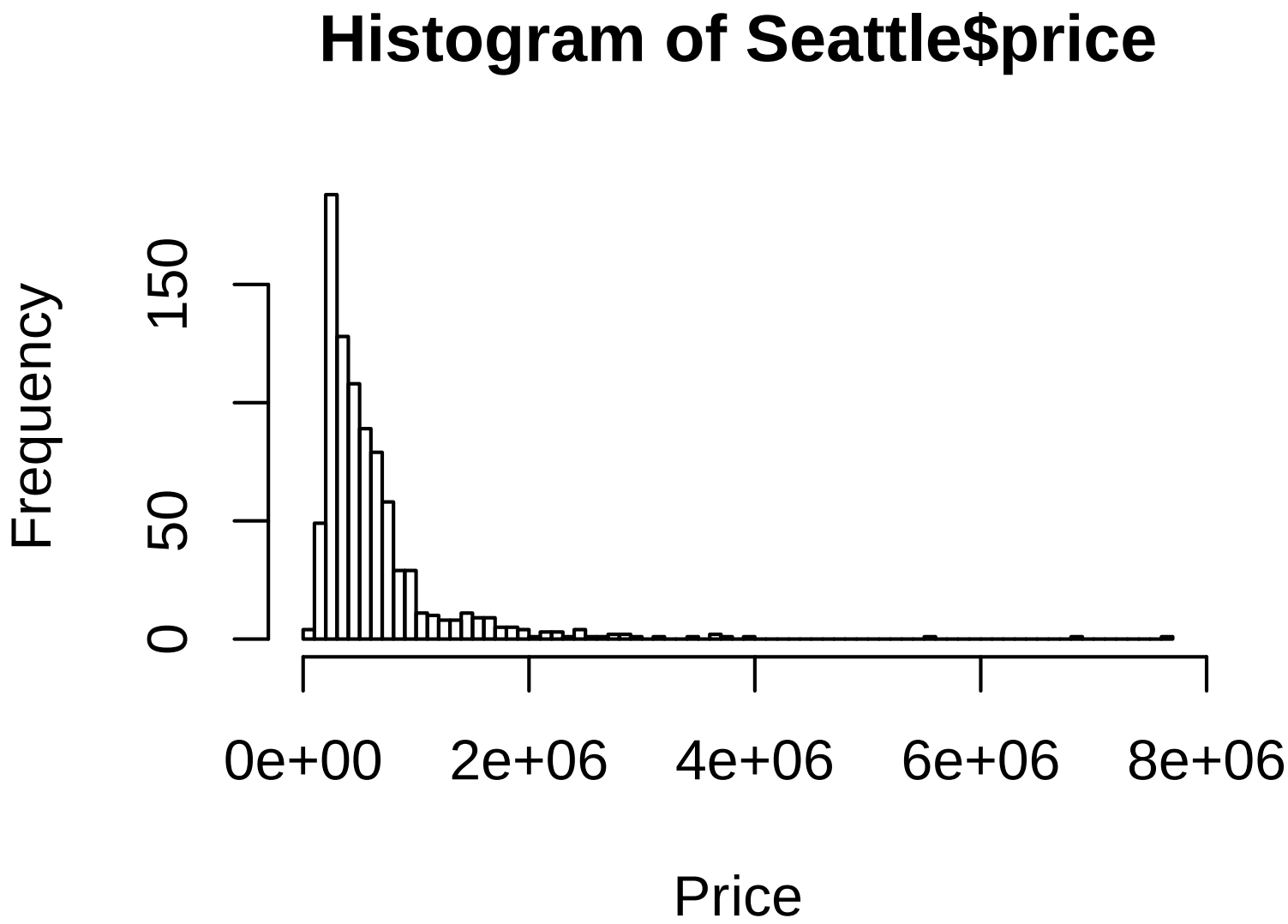
Basic Plotting in R: title


```
plot(Seattle$price~Seattle$sqft_living, pch=16, ylab='Price',
     xlab='Living Sqft',main='Price vs. Living Sqft')
```



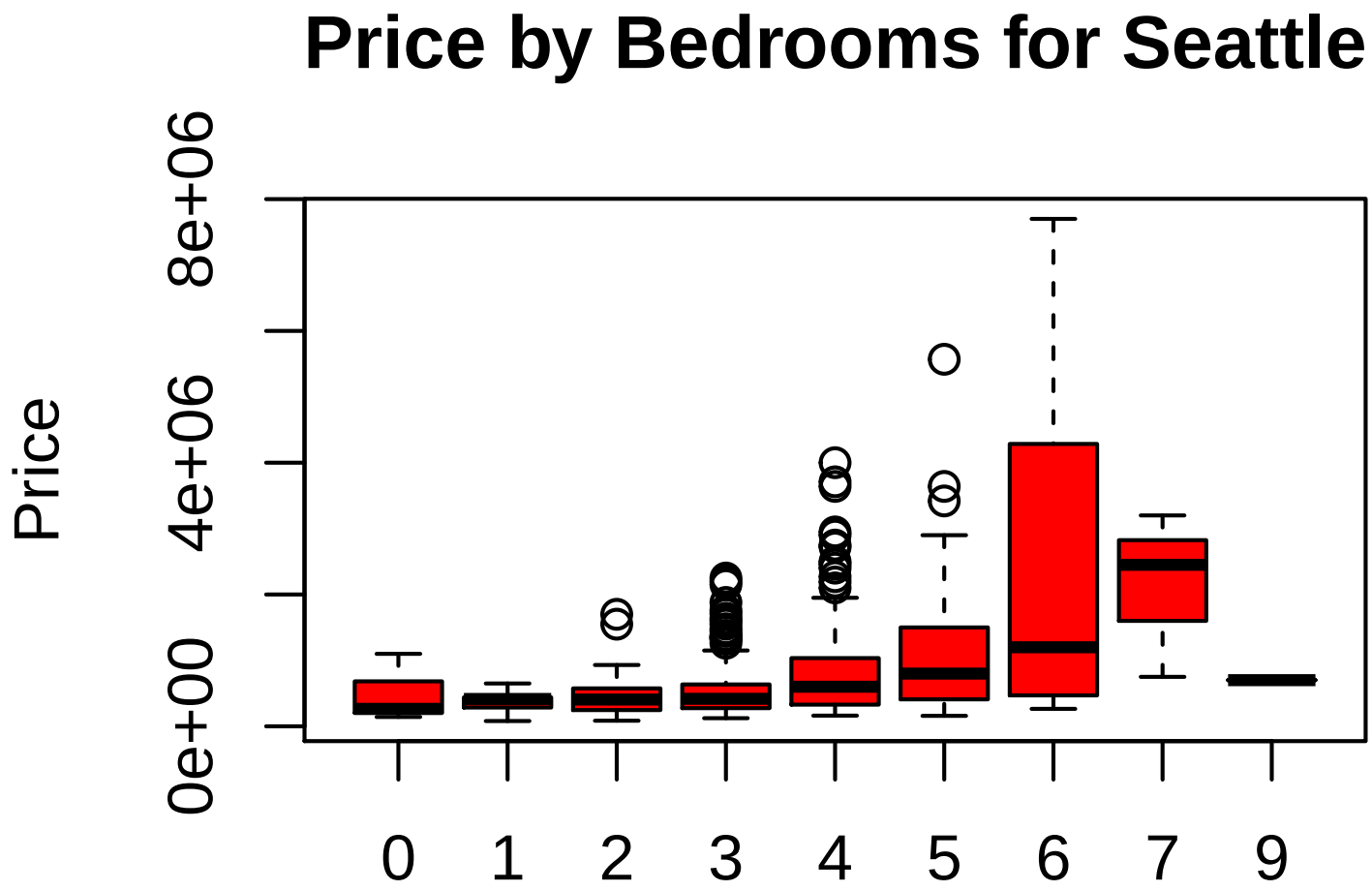
Basic Plotting in R: histogram


```
hist(Seattle$price,xlab='Price', breaks='FD')
```



Basic Plotting in R: histogram

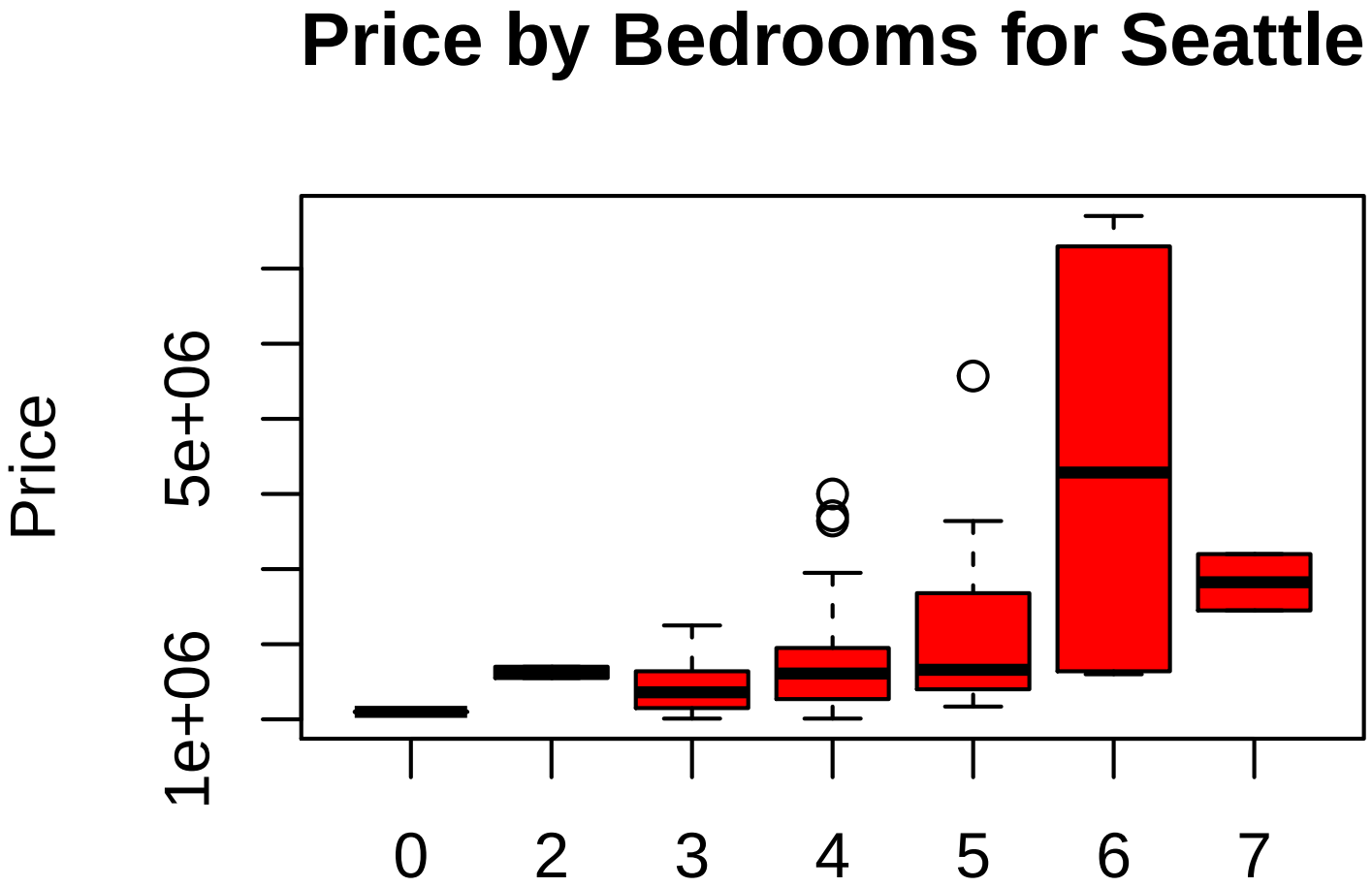

```
boxplot(Seattle$price~Seattle$bedrooms,ylab='Price', col='red',
        xlab='bedrooms',main='Price by Bedrooms for Seattle')
```



Exercise: Basic Plot

- Using only the subset of homes worth more than a million dollars, create a graphic.

Solution: Basic Plot



Solution: Basic Plot - with Code


```
boxplot(expensive.houses$price ~
          expensive.houses$bedrooms,
        ylab='Price', col='red', xlab='bedrooms',
        main='Price by Bedrooms for Seattle',
        sub='For homes worth more than $1,000,000')
```