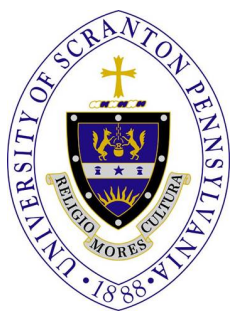
Data Visualization
Part 1: Basic Plots and Distribution Plots

Aram Balagyozyan

Department of Operations and Information Management
Kania School of Management
The
University of Scranton

February 27, 2022



Outline

1. Introduction
2. Basic Plots: Line Charts, Bar Charts, and Scatter Plots.
3. Distribution Plots

Outline

1. Introduction
2. Basic Plots: Line Charts, Bar Charts, and Scatter Plots.
3. Distribution Plots

Some Preliminaries

- In this module, we will describe a set of plots that can be used to explore the multidimensional nature of a data set.
- We will present basic plots (line graphs, bar charts and scatter plots), distribution plots (histograms, Q-Q plots, boxplots, violin plots), and different enhancements that expand the capabilities of these plots to visualize more information.
- We will focus on how the different visualizations and operations can support data mining tasks, from supervised tasks (prediction, classification, and time series forecasting) to unsupervised tasks, as well as provide a few guidelines on specific visualizations to use with each data mining task.

Some Preliminaries

- In particular, we will learn how to visualize correlations using correlation heatmaps and parallel coordinate plots and how to enhance different types of graphs using color, shape, and size.
- We will also describe the advantages of interactive visualization over static plots.
- The module will conclude with a presentation of specialized plots suitable for data with special structure (hierarchical, network, and geographical).

Two Data Examples, an Initial Look

- To illustrate various data visualization approaches, in this module we will be relying on two datasets.
  1. The Boston Housing Data
  2. Ridership on Amtrack Trains
- The Boston Housing Data contain information on census tracts in Boston for which several measurements are taken. It has 14 variables.
- Amtrack routinely collects data on ridership. The Amtrack Ridership data that we will use contain information on monthly ridership between January 1991 and March 2004.
- The R code below imports both datasets as well as the data dictionary for the Boston housing data into the R environment.

```
library(readr)
housing.df<-read_csv("BostonHousing.csv")
Amtrak.df<-read_csv("Amtrak.csv")
housing.dict<-read_csv("BostonHousingDataDict.csv")
```

Boston Housing Data

- The Boston Housing dataset contains information on 506 census tracts and 14 measurements on each tract (14 variables).

```
dim(housing.df)
```

```
## [1] 506  14
```

- Below is the variable definitions of the dataset.

Variable	Description
CRIM	Crime Rate
ZN	Percentage of residential land zoned for lots over 25,000 sq. ft.
INDUS	Percentage of land occupied by non-retail business
CHAS	Does tract bound Charles River (1 if yes and 0 if no)
NOX	Nitric oxide concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Percentage of owner-occupied homes built prior to 1940
DIS	Weighted distance to five Boston employment centers
RAD	Index of accessibility to radial highways
TAX	Full-value property tax rate per $10,000
PTRATIO	Pupil-to-teacher ratio by town
LSTAT	Percentage of lower status of the population
MEDV	Median Value of homes in $1000s
CAT.MEDV	Median value of homes in tract is above $30,000 (1 if yes and 0 if no)

Base R, ggplot2, or Else?

- Base R has its own plotting functions. However, the **ggplot2** visualization package by Hadley Wickham became one of the most popular graphics packages in R. It is due to its power and flexibility.
- The "gg" in **ggplot2** refers to the "Grammar of Graphics" that defines plotting theory and nomenclature. Thus if you want to become an effective data visualizer in R, you must become familiar with this philosophy and technical language of plotting.
- If you are likely to be using data visualization on a regular basis, it is worth to get up to speed on **ggplot2**.
- There are certain types of plots that require a specialized function within a certain package. While discussing these types of plots, I will be relying on functions external to base R or **ggplot2**.
- Alternatively, if a graph can be produced by the base R syntax, we will show how. Whenever possible, I will also provide and explain the **ggplot2** syntax.

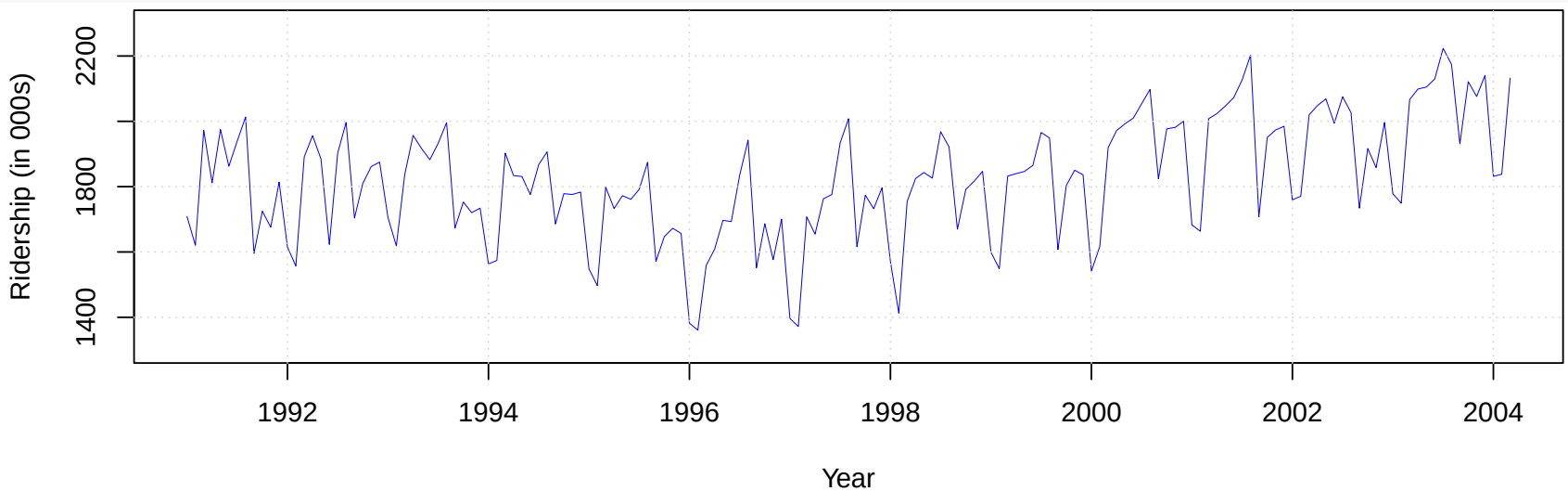
Outline

Line Chart

- Line charts are mostly used to display the evolution of variables over time (time series).
- The two plots on the next two slides demonstrate the same time series graph of monthly railway passengers on Amtrak, one produced using the base R `plot()` while the second is by function `ggplot()`.
- I also provide the code used to produced each plot

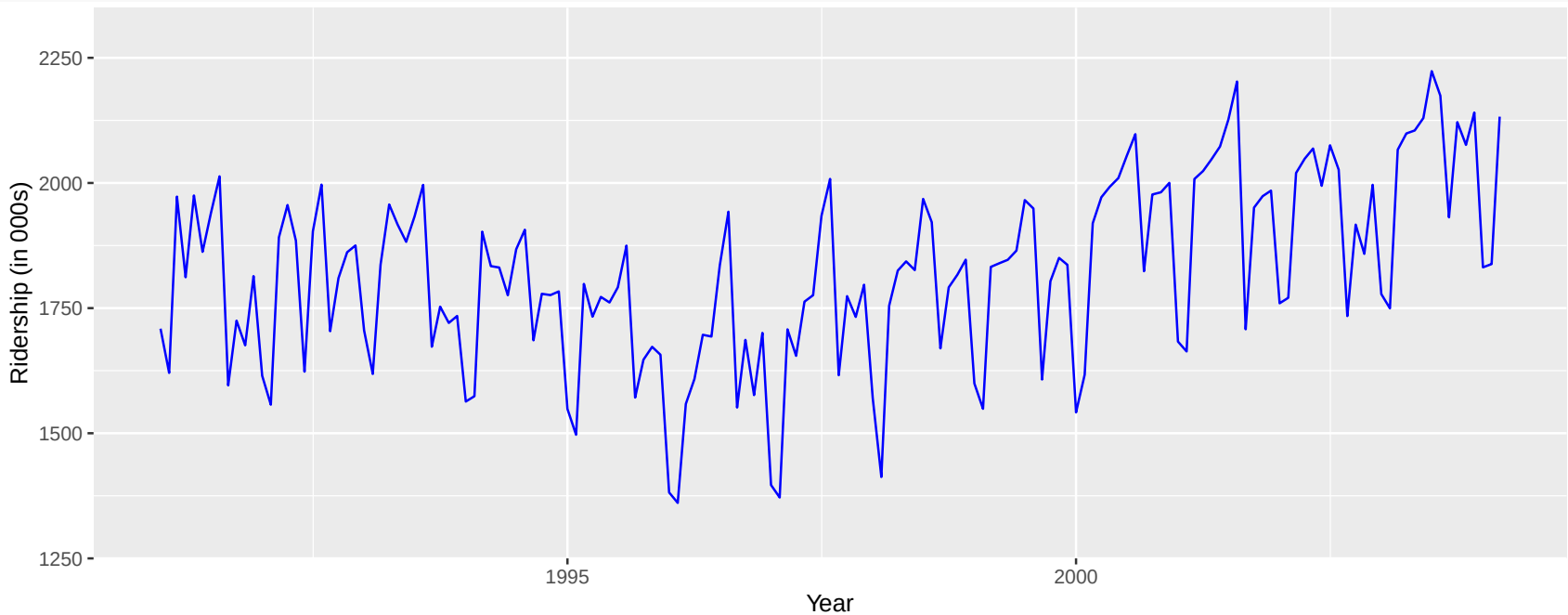
Line Chart Using Base R

```
# Convert the Amtrak.df into a time-series object
library(forecast) # using the forecast package
ridership.ts <- ts(Amtrak.df$Ridership,
  start = c(1991, 1), end = c(2004, 3), freq = 12)
plot(ridership.ts, col="blue",lwd=0.5,
  ylim = c(1300, 2300), ylab = "Ridership (in 000s)",
  xlab = "Year")
grid() # add grid
```



Line Chart Using ggplot()

```
library(ggplot2)
library(lubridate)
# Convert Month in Amtrak.df into the date format
Amtrak.df$Month<-dmy(Amtrak.df$Month)
ggplot(Amtrak.df)+
 geom_line(mapping = aes(x= Month,y = Ridership),
        color='blue', size=0.5)+ylim(1300, 2300)+
       ylab("Ridership (in 000s)") + xlab("Year")
```



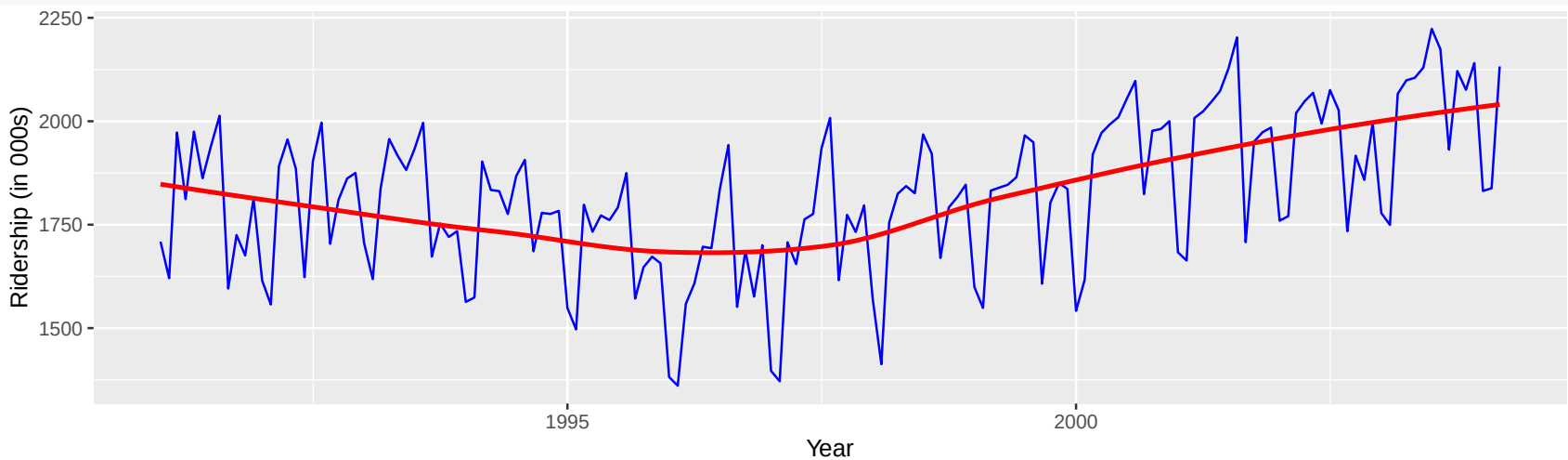
Basics of the **ggplot2** Syntax

- With **ggplot2** you begin a plot with the function `ggplot()`. Note the function name is **not** `ggplot2()`. `ggplot()` creates a coordinate system that you can add layers to.
- The first argument of `ggplot()` is the dataset to use in the graph. The `ggplot(Amtrak.df)` call creates an empty graph.
- The + sign in `ggplot()` calls signifies additional layers. The + sign should never be placed at the beginning of a new line of code.
- You complete a graph by adding one or more layers to `ggplot()`. The function `geom_line()` adds a line chart to your graph. **ggplot2** comes with many geom functions that each add a different type of layer to the plot. You will learn a bunch of them in this and subsequent modules.
- Each geom function in **ggplot2** takes a `mapping` argument. This defines how variables in the dataset are mapped to visual properties. The `mapping` argument is always pared with `aes()` and the x and y arguments of `aes()` that specify which variables to map to the x- and y-axes.

Line Chart Using ggplot() Revisited

- Just to begin demonstrate what can be done with ggplot(), suppose we wanted to add to the previous line chart a layer representing a smoothed trend line. This could be achieved with one additional line of code using geom_smooth():

```
ggplot(Amtrak.df)+
   geom_line(mapping = aes(x= Month, y = Ridership),
   color='blue', size=0.5)+
  ylab("Ridership (in 000s)") + xlab("Year")+
geom_smooth(mapping = aes(x= Month,y = Ridership),
  color="red", se=FALSE)
```



Bar Chart Using Base R

- Bar charts are useful for comparing single statistic (e.g. average, count, percentage) across groups. The height of the bar represents the value of the statistics and different bars correspond to different groups.
- Suppose we are interested in comparing `MEDV` for homes near the Charles River vs. those that are not. The code below accomplishes the task.
- First prepare the data for the plot.

```
# compute mean MEDV per CHAS = (0, 1)
data.for.plot <- aggregate(
                 housing.df$MEDV,
                 by = list(housing.df$CHAS),
                 FUN = mean)
names(data.for.plot) <- c("CHAS", "MeanMEDV")
```

aggregate()

- The code chunk above uses the function `aggregate()` (base R) that is in general very handy for summarizing data for certain subgroups.
- For example, command `aggregate(housing.df$MEDV,by = list(housing.df$CHAS),FUN = mean)` calculates the mean (`FUN=mean`) of `MEDV` broken down by `CHAS` (`by = list(housing.df$CHAS)`).

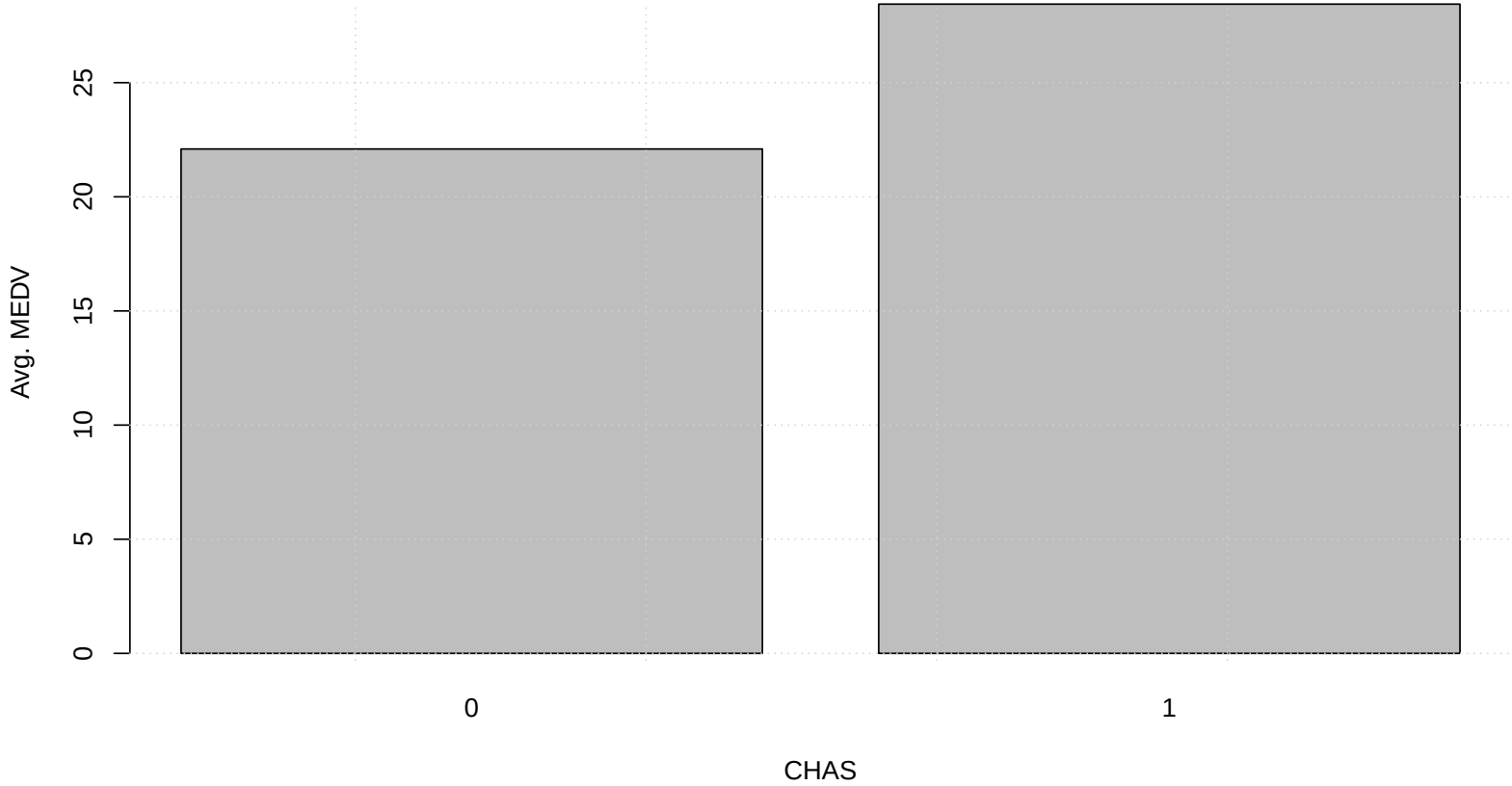
```
data.for.plot
```

```
##   CHAS MeanMEDV
## 1    0 22.09384
## 2    1 28.44000
```

Bar Chart Using Base R

- Below is the base R code that produces a bar chart:

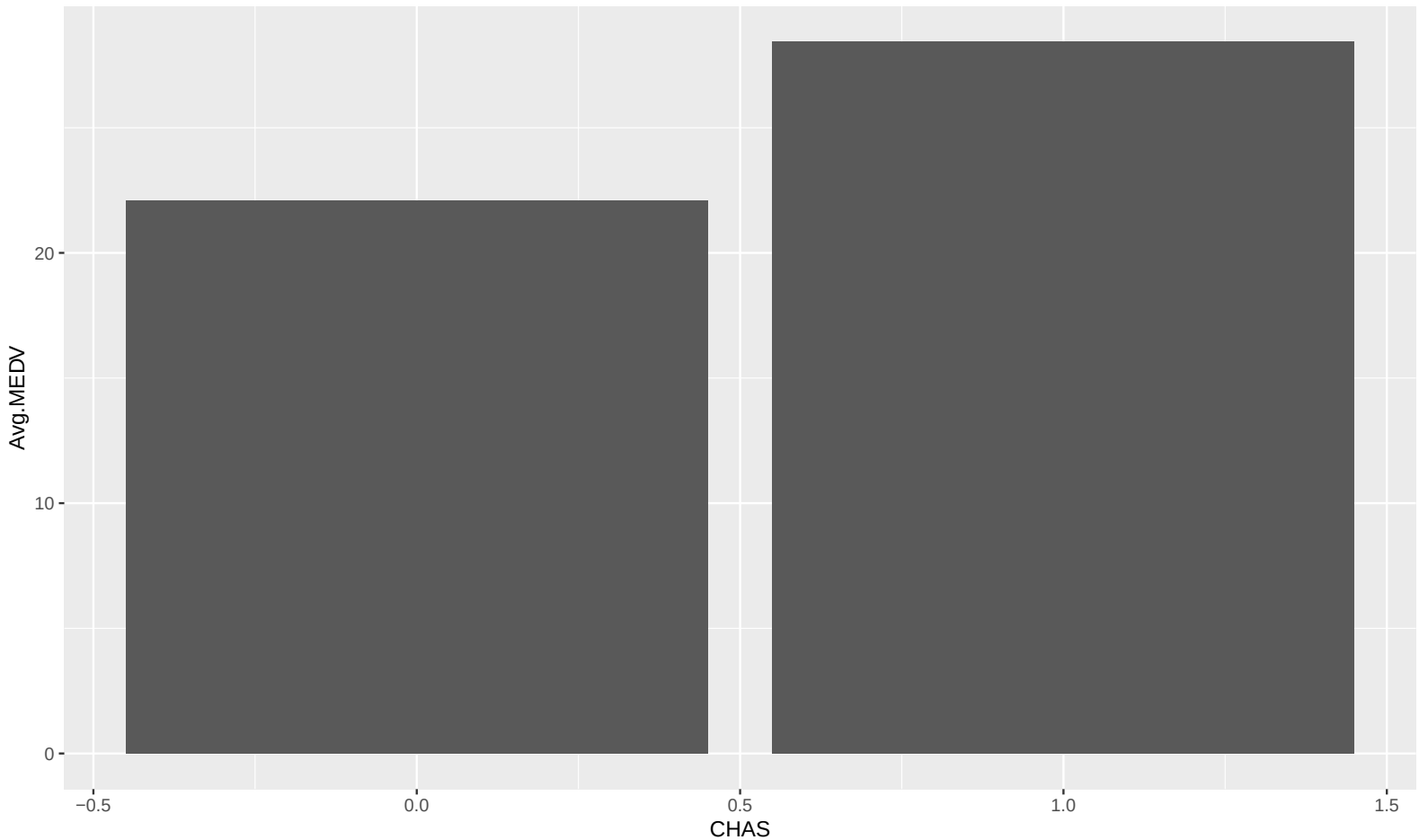
```
barplot(data.for.plot$MeanMEDV,
    names.arg = data.for.plot$CHAS,
    xlab = "CHAS", ylab = "Avg. MEDV")
grid() # add grid
```



Bar Chart Using ggplot()

- Below is the ggplot() code that produces a bar chart.

```
ggplot(data.for.plot) +
  geom_col(aes(x = CHAS, y = MeanMEDV))+
ylab("Avg.MEDV")
```



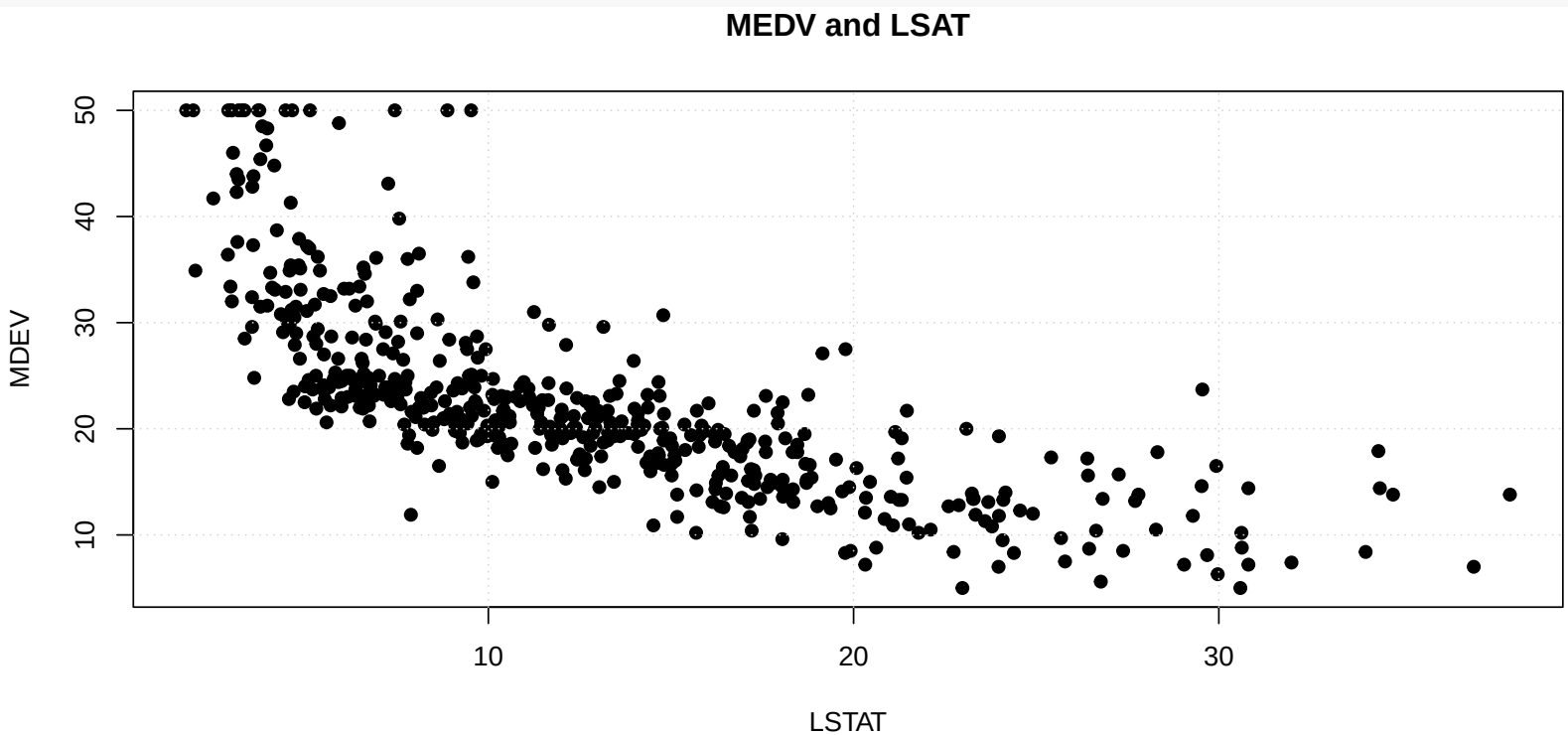
Scatter Plots Using Base R

- Scatter plots are widely used to uncover a relationship (correlation if you will) between two or more variables.
- Since in supervised learning, there is more focus on the outcome variable, in scatter plots that are used in conjunction with supervised learning exercises, the outcome variable is typically associated with the y axis.
- Suppose we wanted to investigate if there is an obvious relationship between tract median house value (MEDV) and percentage of lower status of the population in the tract (LSAT).

Scatter Plots Using Base R

- You could easily produce a scatter plot for `MEDV` and `LSTAT` using the code below.

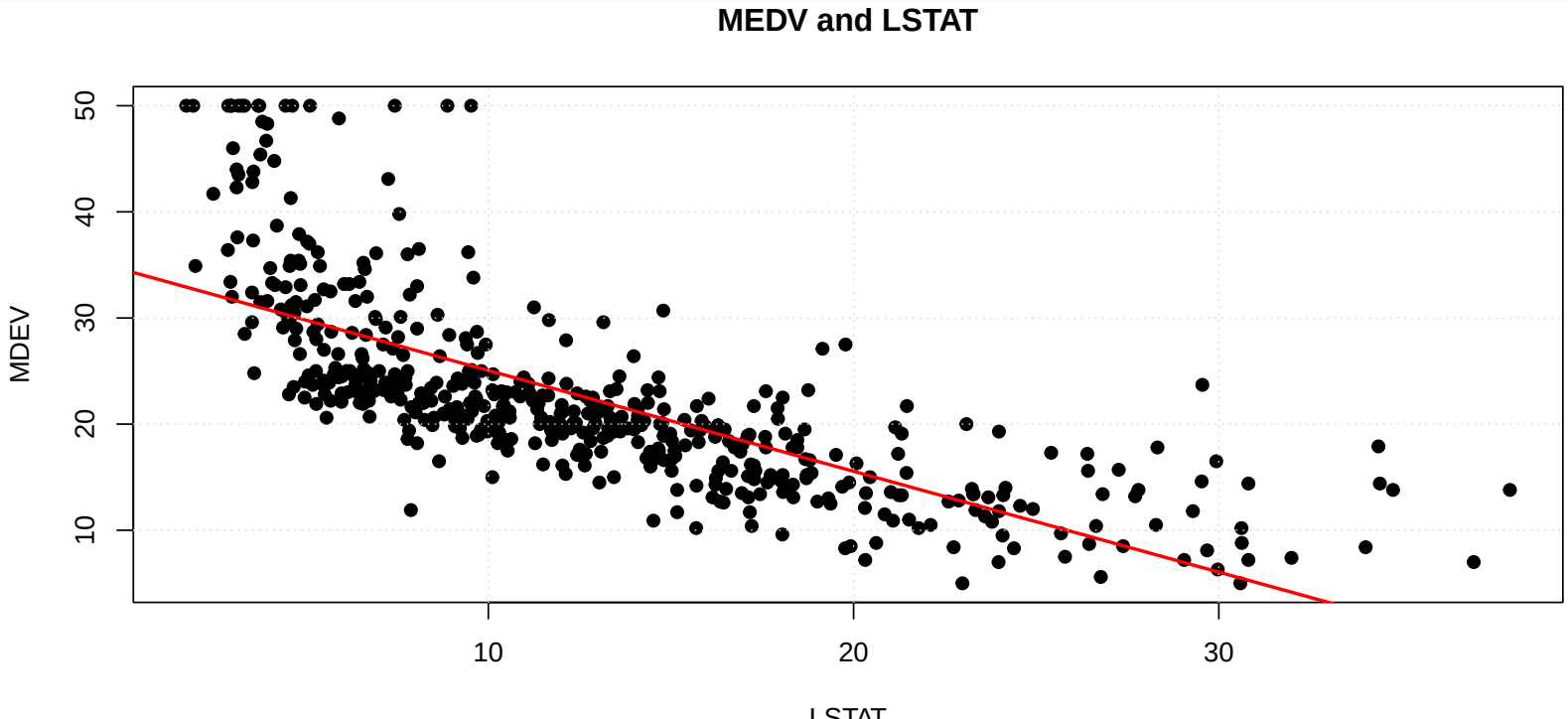
```
plot( housing.df$LSTAT,housing.df$MEDV,
  ylab = "MDEV", xlab = "LSTAT",
  pch=19, #pch = 19 -- solid circle markers
  main = "MEDV and LSAT") # plot title
grid() # add grid
```



Scatter Plots Using Base R

- The code below adds a straight regression line.

```
plot( housing.df$LSTAT,housing.df$MEDV,
  ylab = "MDEV", xlab = "LSTAT",
  pch=19, #pch = 19 -- solid circle markers
  main = "MEDV and LSTAT") # plot title
abline(lm(housing.df$MEDV ~ housing.df$LSTAT),
       col="red", lwd=2) #lwd = line width
grid() # add grid
```



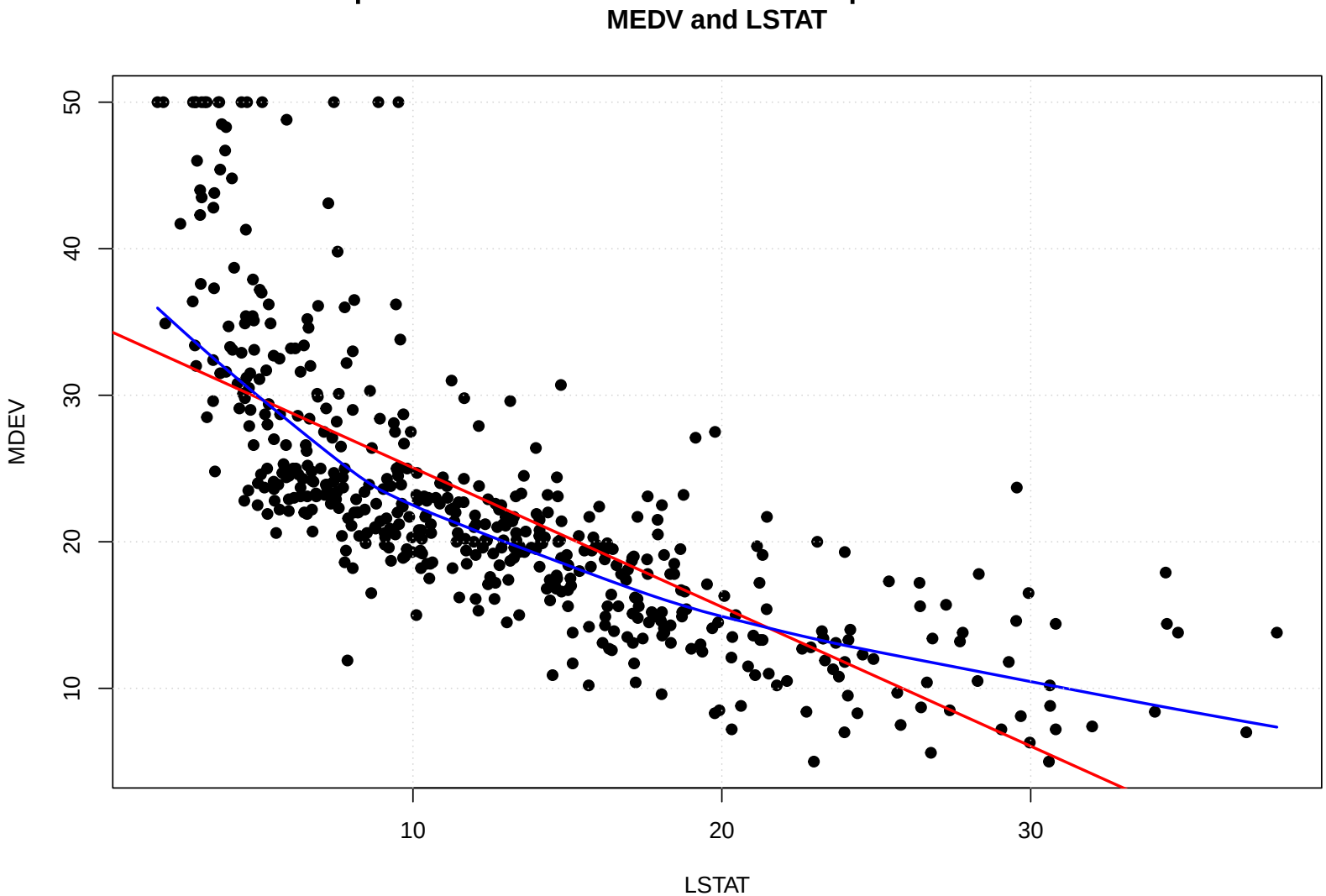
Scatter Plots Using Base R

- You could also add a non-linear regression line. The code below accomplishes it.

```
plot( housing.df$LSTAT,housing.df$MEDV,
  ylab = "MDEV", xlab = "LSTAT",
  pch=19, #pch = 19 -- solid circle markers
  main = "MEDV and LSTAT") # plot title
abline(lm(housing.df$MEDV ~ housing.df$LSTAT),
       col="red", lwd=2) # regression line
lines(lowess(housing.df$LSTAT,housing.df$MEDV),
       col="blue", lwd=2) #lowess line
grid() # add grid
```

Scatter Plots Using Base R

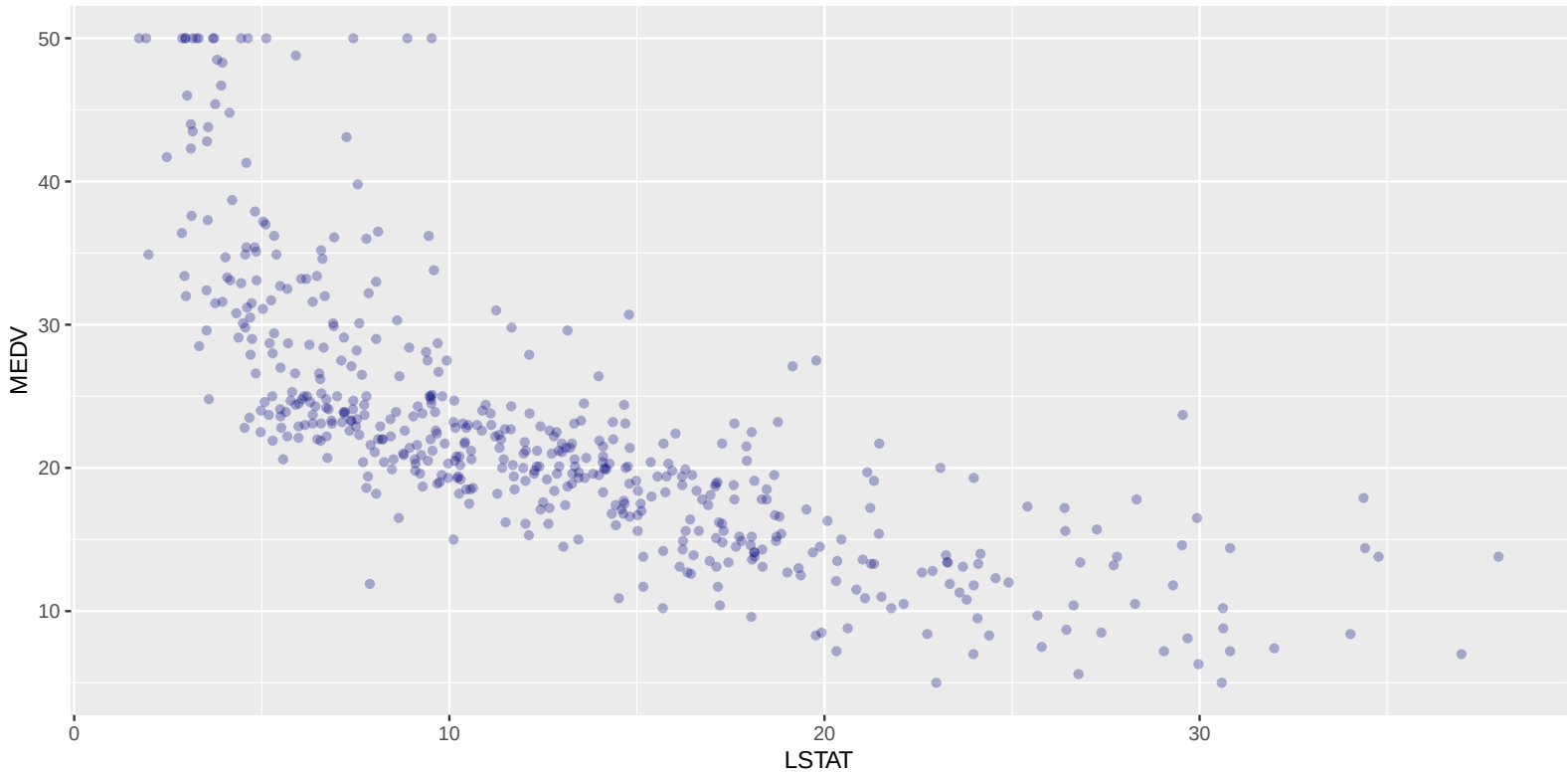
- Below is the output of the code on the previous slide:



Scatter Plot Using ggplot()

- In order to produce a scatter plot using ggplot() use the following code:

```
ggplot(housing.df) +
  geom_point(aes(x = LSTAT, y = MEDV),
     colour = "navy", alpha = 0.3)
```

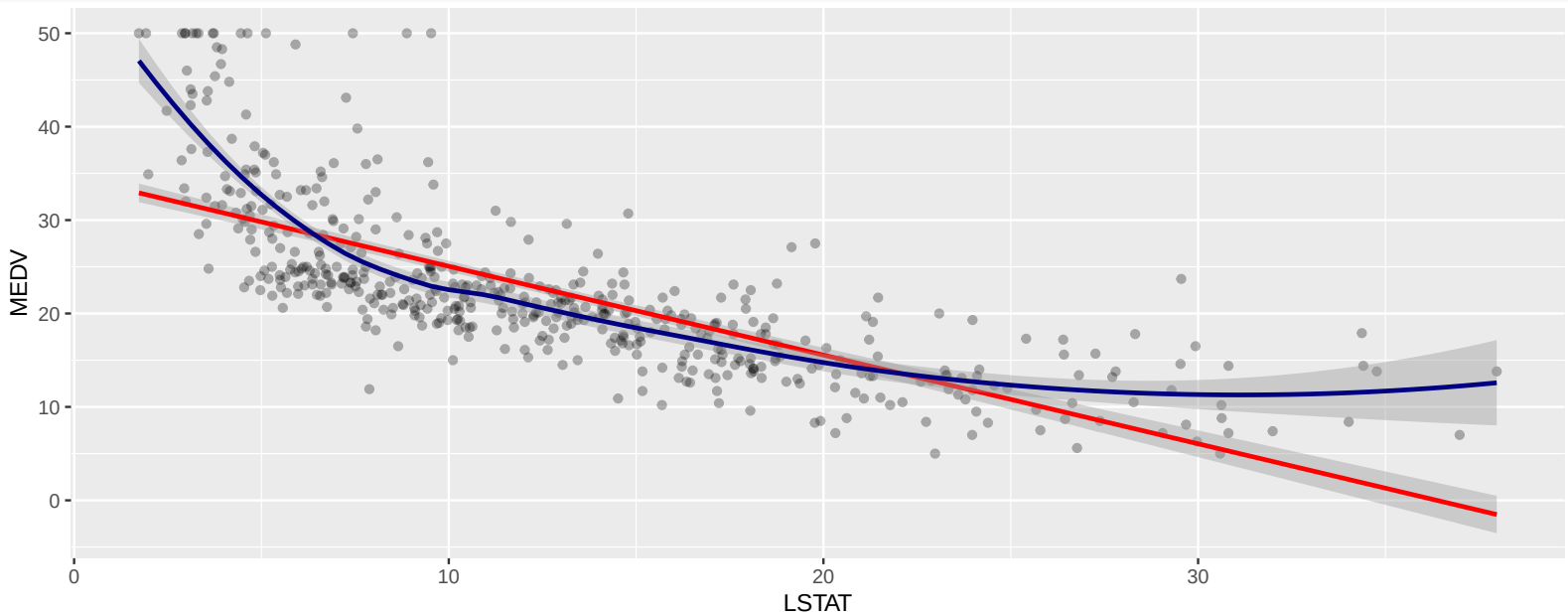



```
#alpha controls transparency
```

Scatter Plot Using ggplot()

- The following code adds straight and lowess regression lines

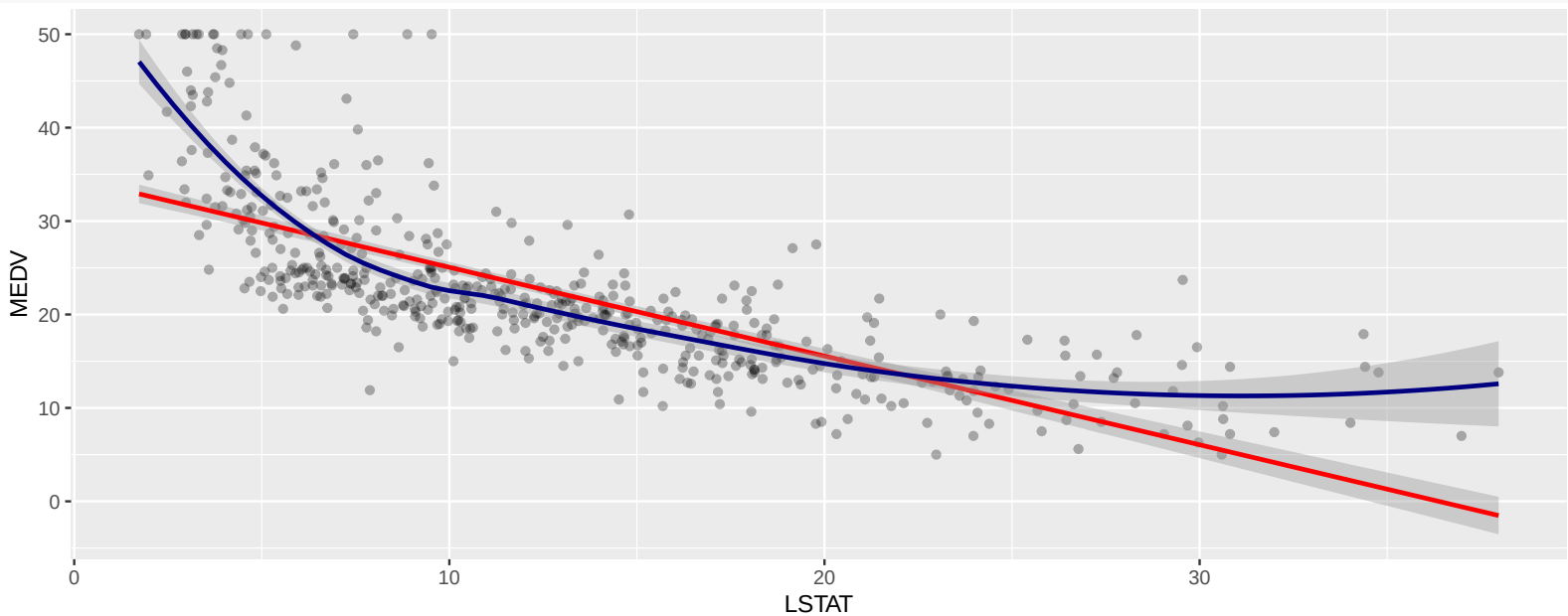
```
ggplot(housing.df) +
  geom_point(aes(x = LSTAT, y = MEDV),
     colour = "black", alpha = 0.3)+
  geom_smooth(aes(x = LSTAT, y = MEDV),
     method="lm",colour = "red")+ #regression line
  geom_smooth(aes(x = LSTAT, y = MEDV),
     method="loess",colour = "navy") #lowess line
```



Scatter Plot Using ggplot()

- Note that the code on the previous slide is a bit cumbersome, `(aes(x = LSTAT, y = MEDV)` is repeated in all `geom` layers.
- You could avoid that by specifying the aesthetics parameters in the `ggplot` statement instead:

```
ggplot(housing.df, aes(x = LSTAT, y = MEDV)) +
  geom_point(colour = "black", alpha = 0.3)+
  geom_smooth(method="lm",colour = "red")+
  geom_smooth(method="loess",colour = "navy")
```



Outline

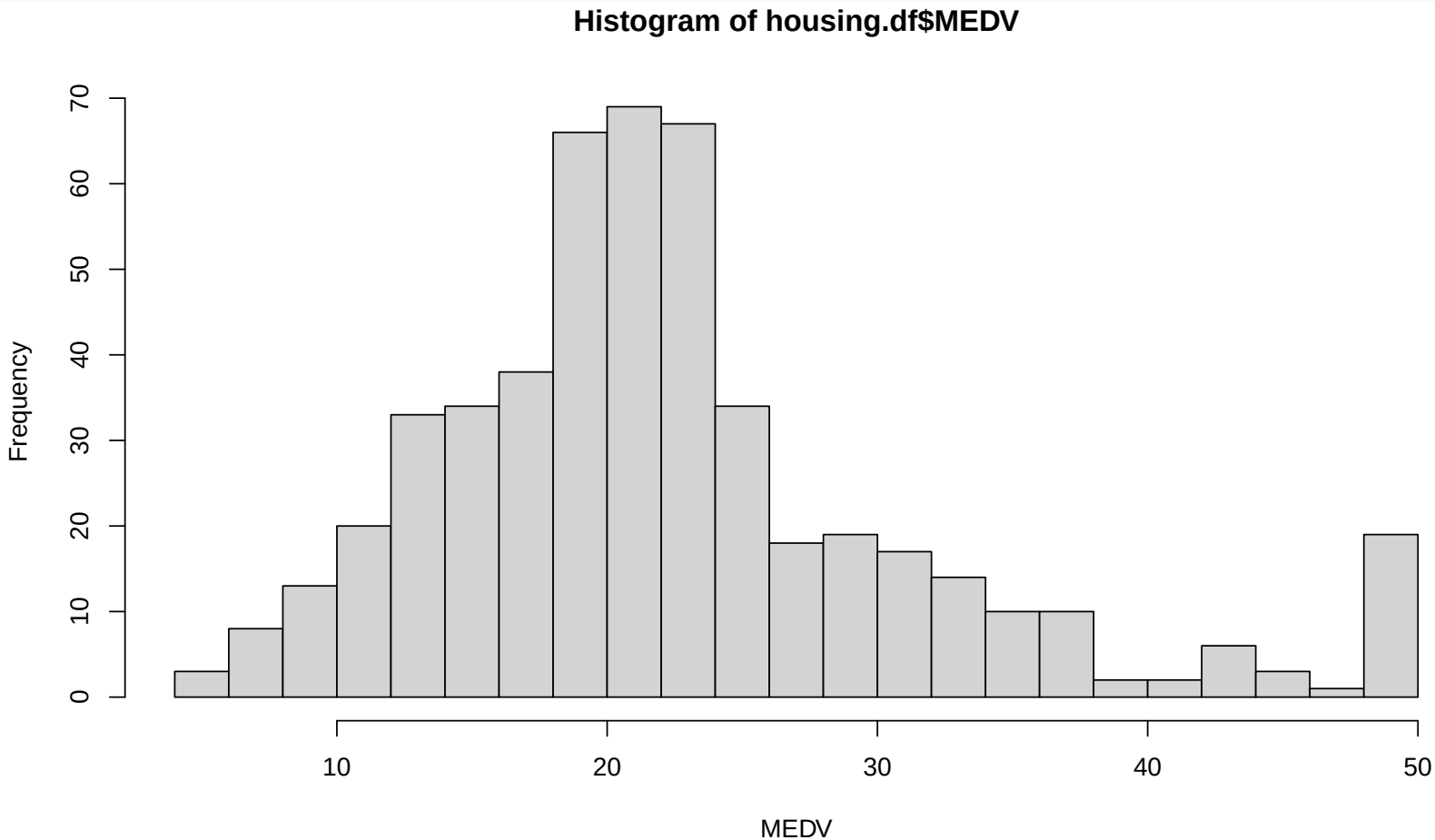
Distribution Plots: Overview

- As the name suggests, distribution plots are useful for examining the entire distribution of numerical variables.
- Distribution plots are useful in supervised learning for determining potential data mining methods and variable transformations
- Histograms and Q-Q plots are useful for visualizing the entire distribution of a variable.
- Box-plots and violin plots are effective for comparing sub-groups by generating side-by-side box plots, or by looking at the evolution of a single distribution over time.

Histogram Using Base R

- Histogram represents the frequencies of all x values with a series of vertical connected bars.

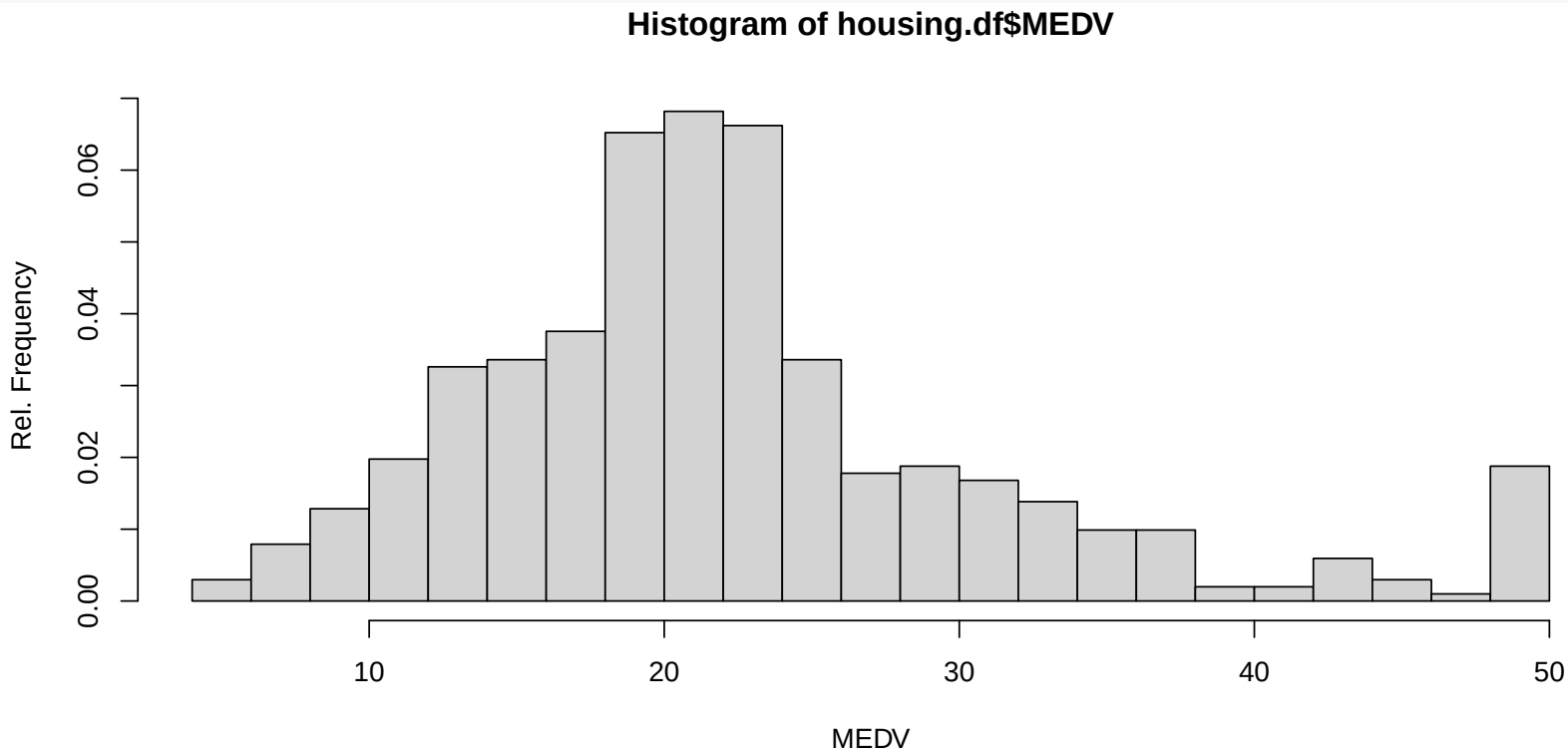
```
hist(housing.df$MEDV, xlab = "MEDV", freq = TRUE,
# freq = TRUE plots counts (not percentage)
   nclass=20) #number of bins
```



Histogram Using Base R: Frequency vs. Relative Frequency

- If you want the vertical axis to display the percentage of records (as opposed to simple counts) use `freq=FALSE` instead.

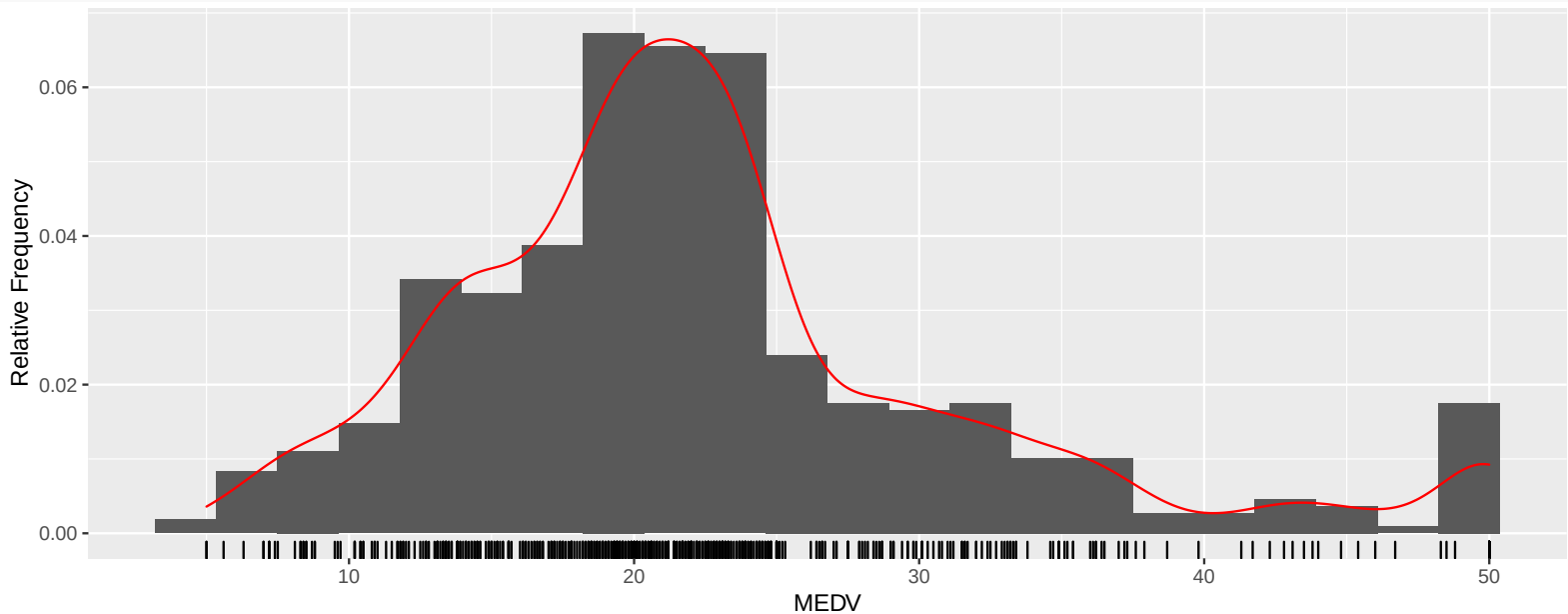
```
hist(housing.df$MEDV,
    xlab = "MEDV", ylab = "Rel. Frequency",
    freq = FALSE, # display density
    nclass=20) #number of bins
```



Histogram Using ggplot()

- The code below produces a histogram overlayed with a density line (smooth histogram) and rug plot (vertical dashes).

```
ggplot(housing.df,aes(x = MEDV))+
  geom_histogram(aes(y=..density..), bins = 22)+
  xlab("MEDV")+ylab("Relative Frequency")+
  geom_density(color="red")+geom_rug()
```



- If instead of plotting relative frequency you want to plot frequency, use `geom_histogram(aes(y=..count..))`

Histogram Using `ggplot()`

- The rug plot on the histogram above can be very useful allowing easy spotting of extremes or even outliers. For instance, we can observe that there is one value significantly greater than than all others.
- This kind of data inspection is very important as it may identify possible errors in the data sample, or even help to locate values that are so awkward that they may only be errors, or at least we would be better off by disregarding them in posterior analysis.
- You can also infer from the histogram and the density line above that the distribution of house values doesn't look normal, it is skewed to the left (positively skewed).

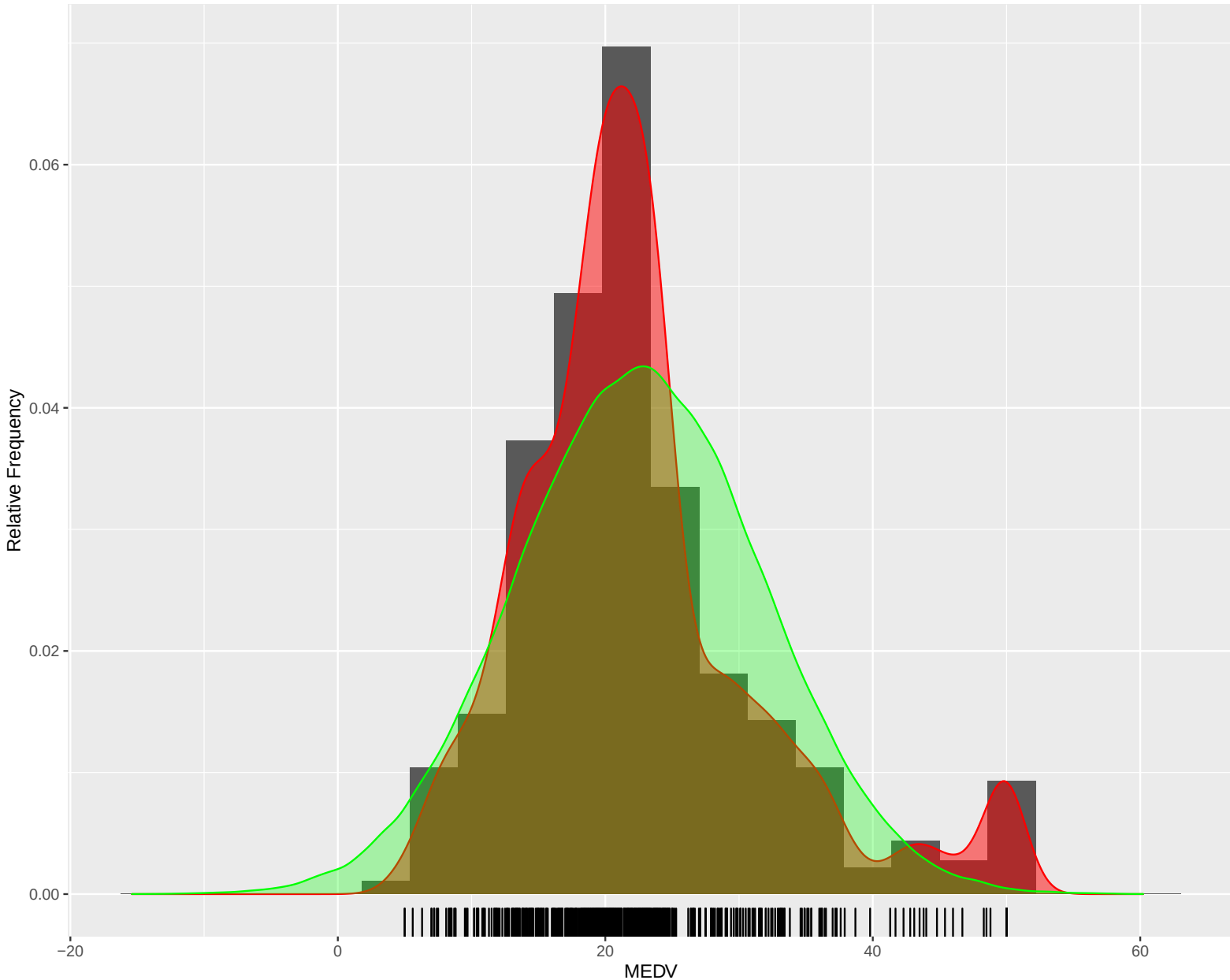
Histogram Using ggplot()

- Just to highlight the non-normality of MEDV, I am adding a normal density line to the plot above.

```
# first generat a normally
# distributed random variable
norm.df<-data.frame(normRV=rnorm(200000,
    mean = mean(housing.df$MEDV),
    sd=sd(housing.df$MEDV)))
# second generate the plot
ggplot(housing.df,aes(x = MEDV))+
  geom_histogram(aes(y=..density..), bins = 22)+
  xlab("MEDV")+ylab("Relative Frequency")+
  geom_density(color="red", fill="red", alpha=0.5)+
  geom_rug()+
  geom_density(data=norm.df,
  aes(x=normRV), color="green",
  fill="green", alpha=0.3)
```

Histogram Using `ggplot()`

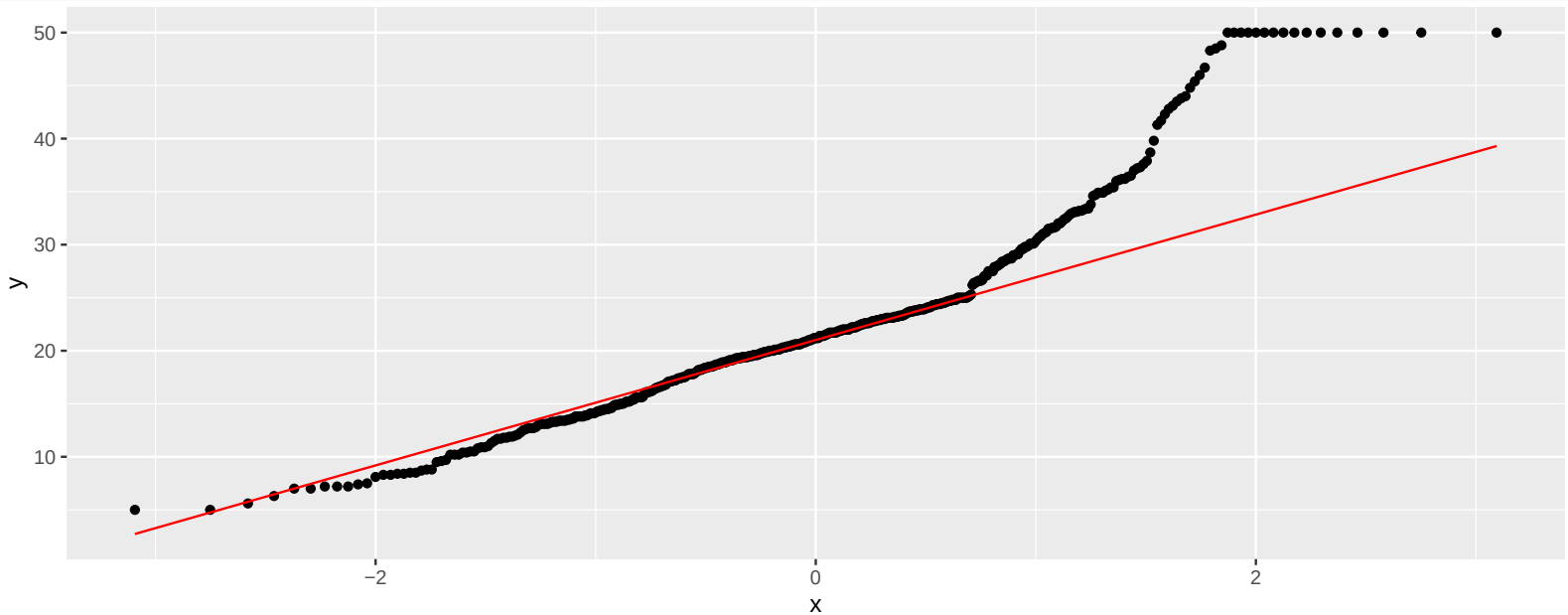
- Below is the output of the code on the previous slide:



Q-Q Plot Using `stat_qq` and `stat_qq`

- Another type of graph that allows for easy identification of outliers and non-normality is Q-Q (Quantile to Quantile) plot. Q-Q plots are generally used for comparing the actual empirical distribution with a hypothetical alternative (e.g normal distribution).

```
ggplot(housing.df,aes(sample = MEDV))+
  stat_qq()+stat_qq_line(color="red")
```

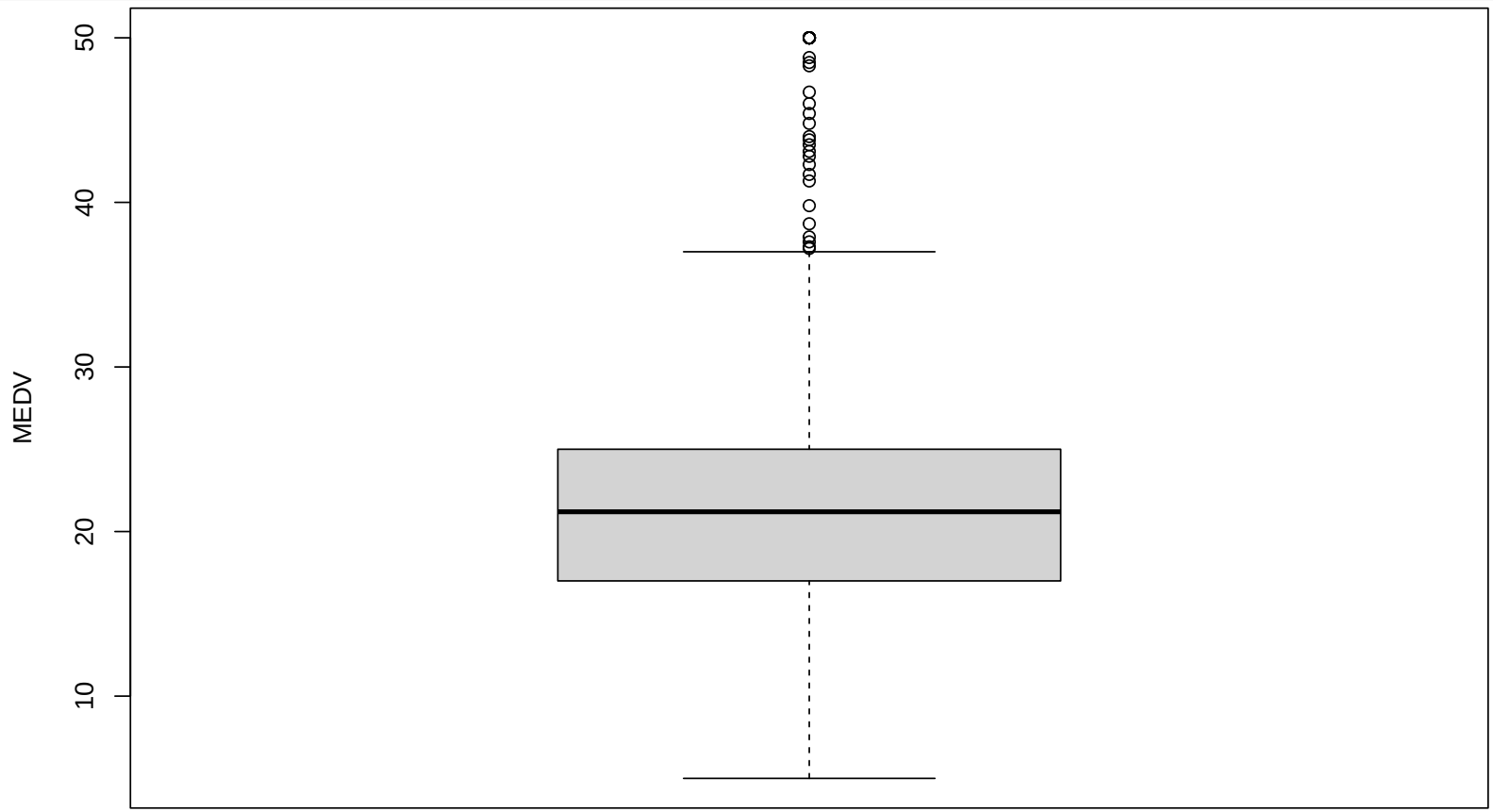


- `MEDV` is positively skewed with significant degree of heavy-tailedness. (https://xiongge.shinyapps.io/QQplots/)

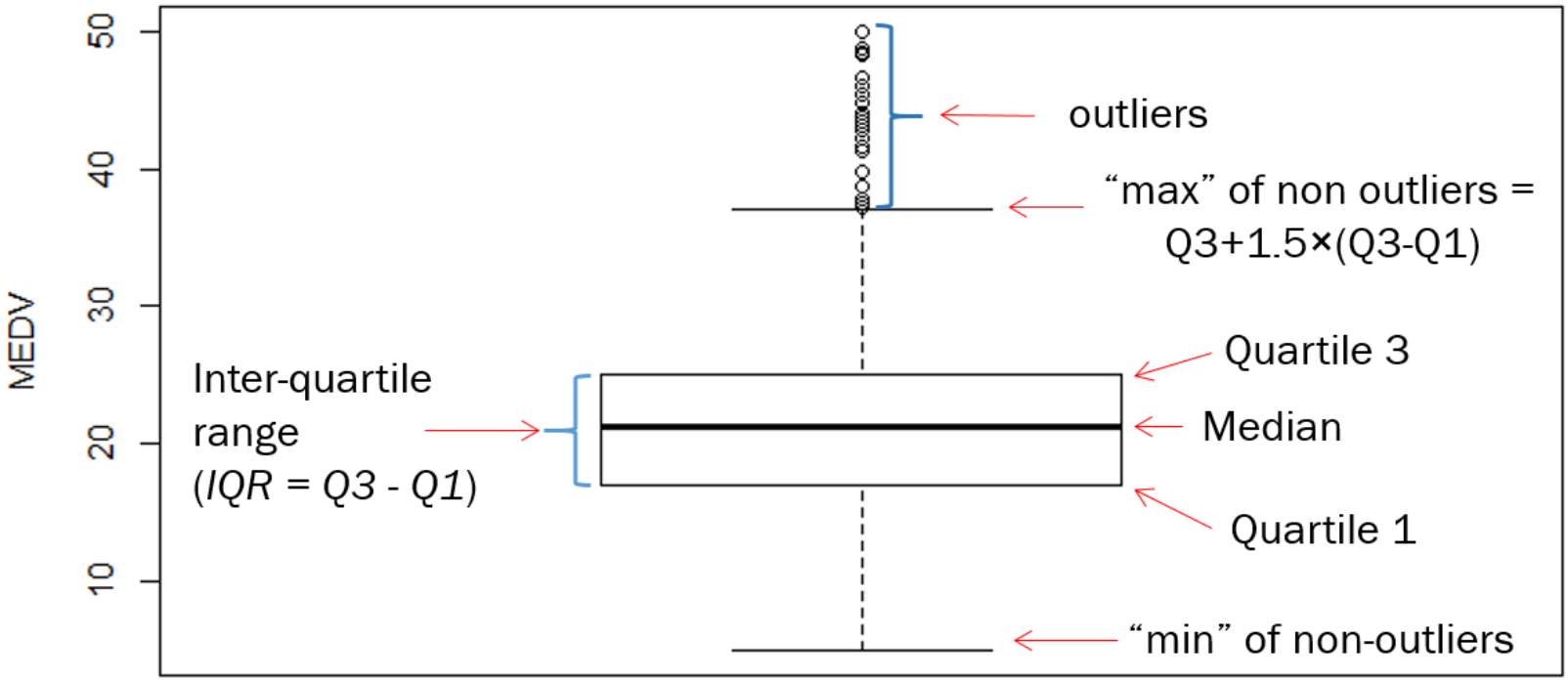
Box-Plot Using Base R

- We could also visualize the distribution of `MEDV` using a box-plot:

```
boxplot(housing.df$MEDV, ylab="MEDV")
```



Understanding Box-Plots



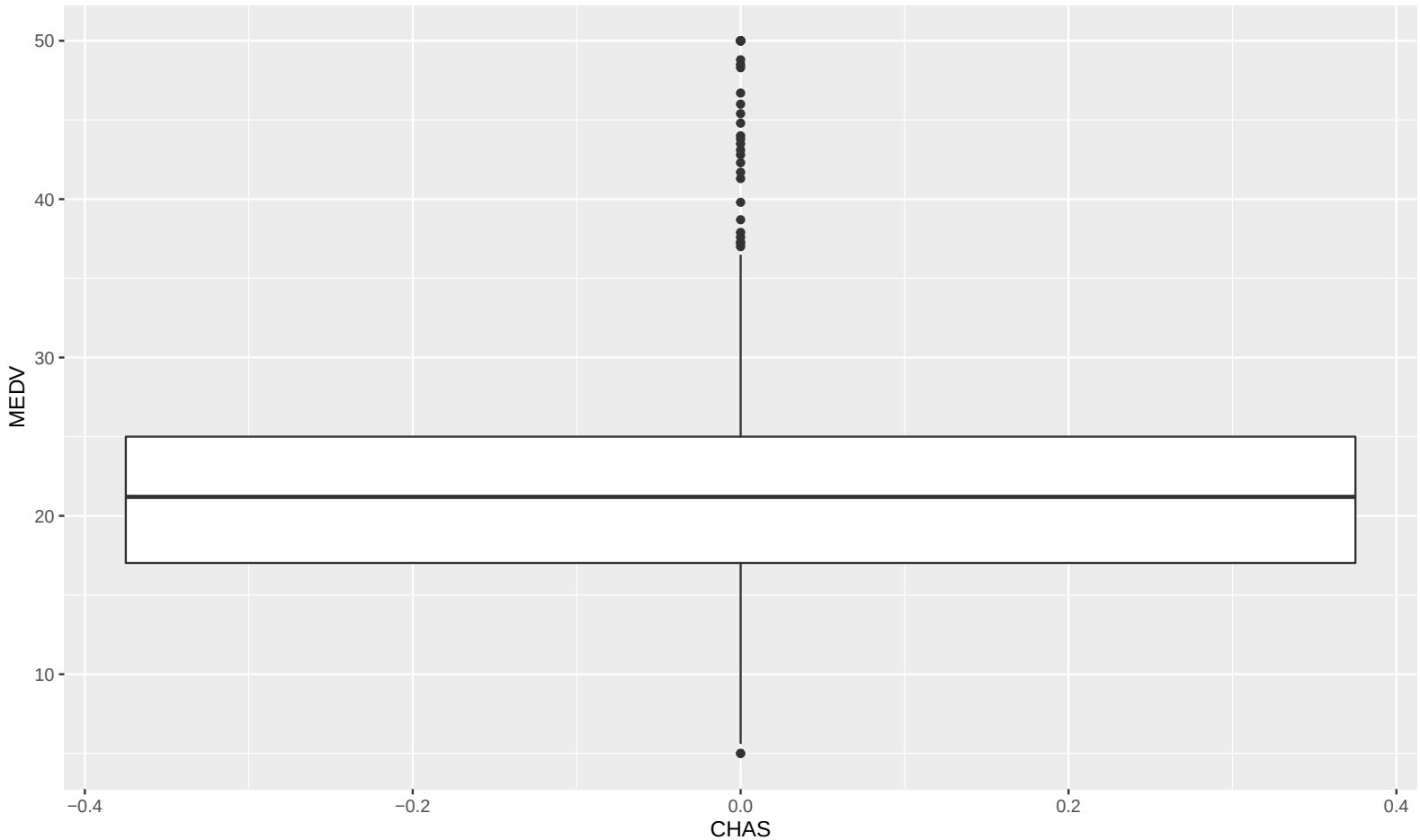
Understanding Box-Plots

- Define $Q3 - Q1$ as inter-quartile range (IQR).
- The top horizontal line on a box-plot is the smaller of $Q3 + 1.5 \times IQR$ and the maximum point of the data. If the largest observation in your data is greater than $Q3 + 1.5 \times IQR$ then any observation greater than $Q3 + 1.5 \times IQR$ is considered an outlier.
- Similarly, the bottom horizontal line is the greater of $Q1 - 1.5 \times IQR$ and the smallest observation in your data. If the smallest observation in your data is smaller than $Q1 - 1.5 \times IQR$ then any observation smaller than $Q1 - 1.5 \times IQR$ is an outlier.
- The details of the above convention of identifying outliers may be different across software packages.

Box-Plot Using ggplot()

- In order to produce a box-plot using ggplot() use the following code:

```
ggplot(housing.df) +
  geom_boxplot(aes(y = MEDV)) +
  xlab("CHAS")
```



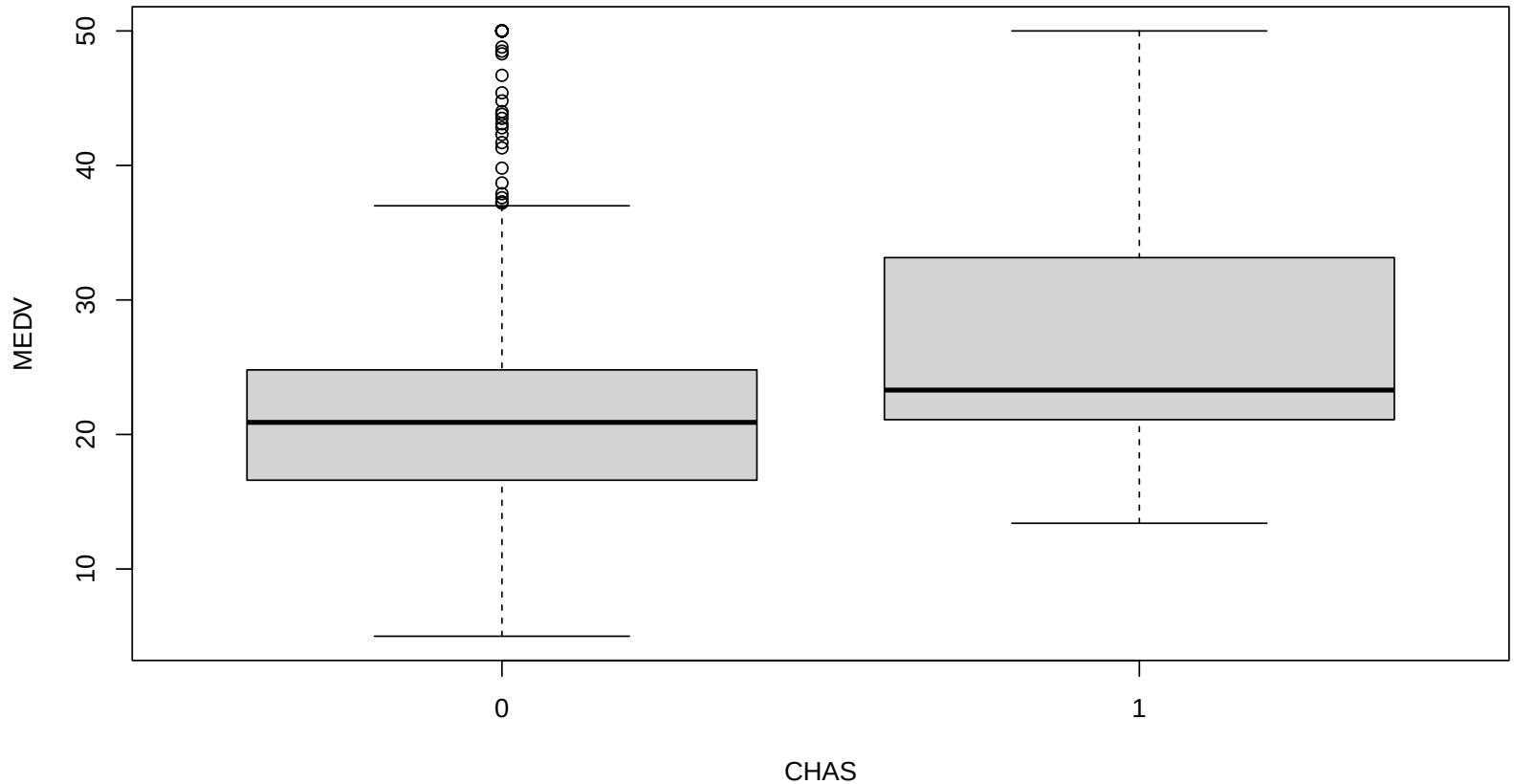
Using Box-Plots For Comparing the Distributions of Sub-Groups.

- Box-plots can be very useful in comparing the distributions of certain sub-groups in the data.
- As an example, suppose we want to compare the distribution of `MEDV` of the neighborhoods close to the Charles River (`CHAS = 1`) to those that are not `CHAS = 0`.

Side-by-side Box-Plot Using Base R

- The code below plots the box-plots of `MEDV` for neighborhoods that are next to the river and for the ones that are not.

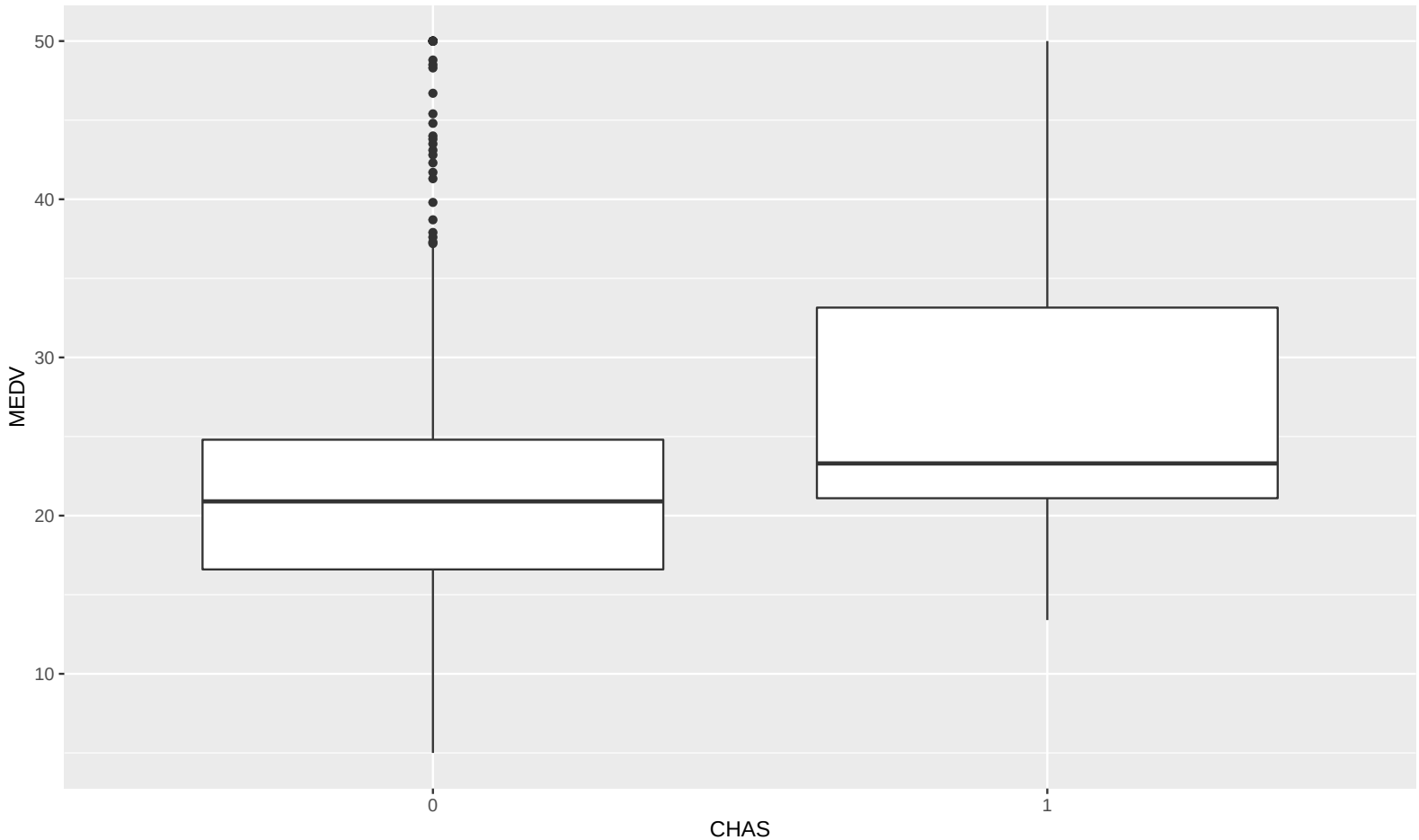
```
boxplot(housing.df$MEDV ~ housing.df$CHAS,
    xlab="CHAS", ylab="MEDV")
```



Side-by-side Box-Plot Using ggplot()

- To accomplish the same task using ggplot() use the following code:

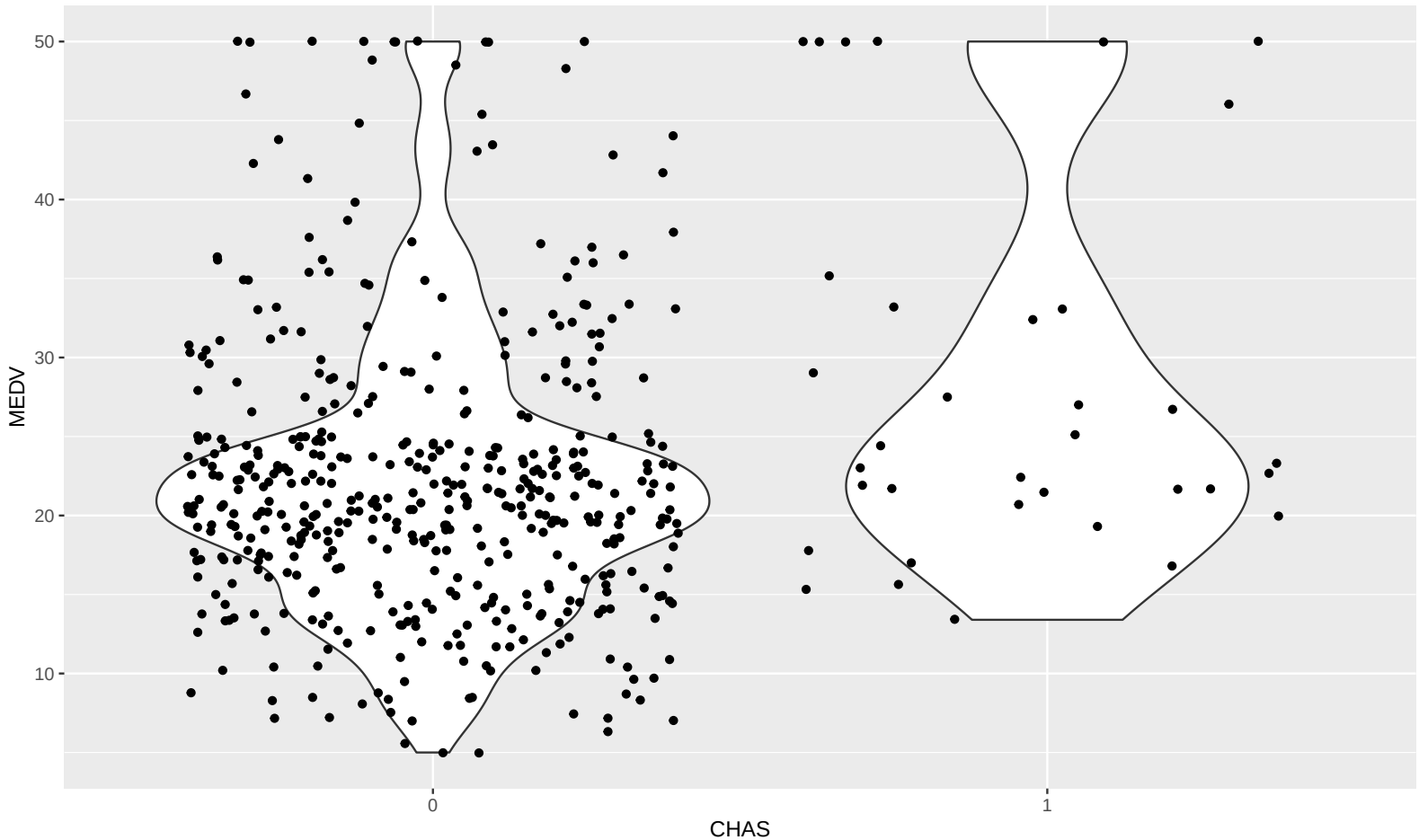
```
ggplot(housing.df) +
  geom_boxplot(aes(x = as.factor(CHAS),
                   y = MEDV)) + xlab("CHAS")
```



Side-by-side Violin Plot using ggplot()

- Another nice and useful plot that can be produced using ggplot() is the violin plot.

```
ggplot(housing.df,
aes(x = as.factor(CHAS),y = MEDV))+geom_violin() +
geom_jitter() + xlab("CHAS")+ ylab("MEDV")
```



Interpreting a Violin Plot

- Function `geom_jitter()` adds a small amount of random variation to the location of each point. This noise "unstacks" markers that hide markers underneath. Jittering is a useful way of handling overplotting caused by a concentration of many points in the same place. To see what `geom_jitter()` does, just try to run the code above without it.
- The white areas represent the distribution of `MEDV` for each type of neighborhood.
- Wider regions represent ranges that contain more data points.Thus, the neighborhoods with `MEDV` between around $12,000 and $30,000 are an obvious majority.
- You can also infer by examining the plot that there weren't any neighborhoods near the Charles River with median house value less than $10,000 or between $37,000 and $45,000.