MGS 3701 / MKT 3950: Data Mining

Chapter3: Data Visualization

Chungil Chae

温州肯恩大学
WENZHOU-KEAN UNIVERSITY

2022/01/01 (updated: 2022-02-14)

Chapter Objectives

- In this chapter, we describe a set of plots that can be used to explore the multidimensional nature of a data set. We present
  - basic plots (bar charts, line graphs, and scatter plots),
  - distribution plots (box-plots and histograms), and
  - different enhancements that expand the capabilities of these plots to visualize more information.
- We focus on how the different visualizations and operations can support data mining tasks, from supervised tasks (prediction, classification, and time series forecasting)
- We also describe the advantages of interactive visualization over static plots.
- The chapter concludes with a presentation of specialized plots suitable for data with special structure (hierarchical, network, and geographical).

Uses of Data Visualization

“A picture is worth a thousand words”

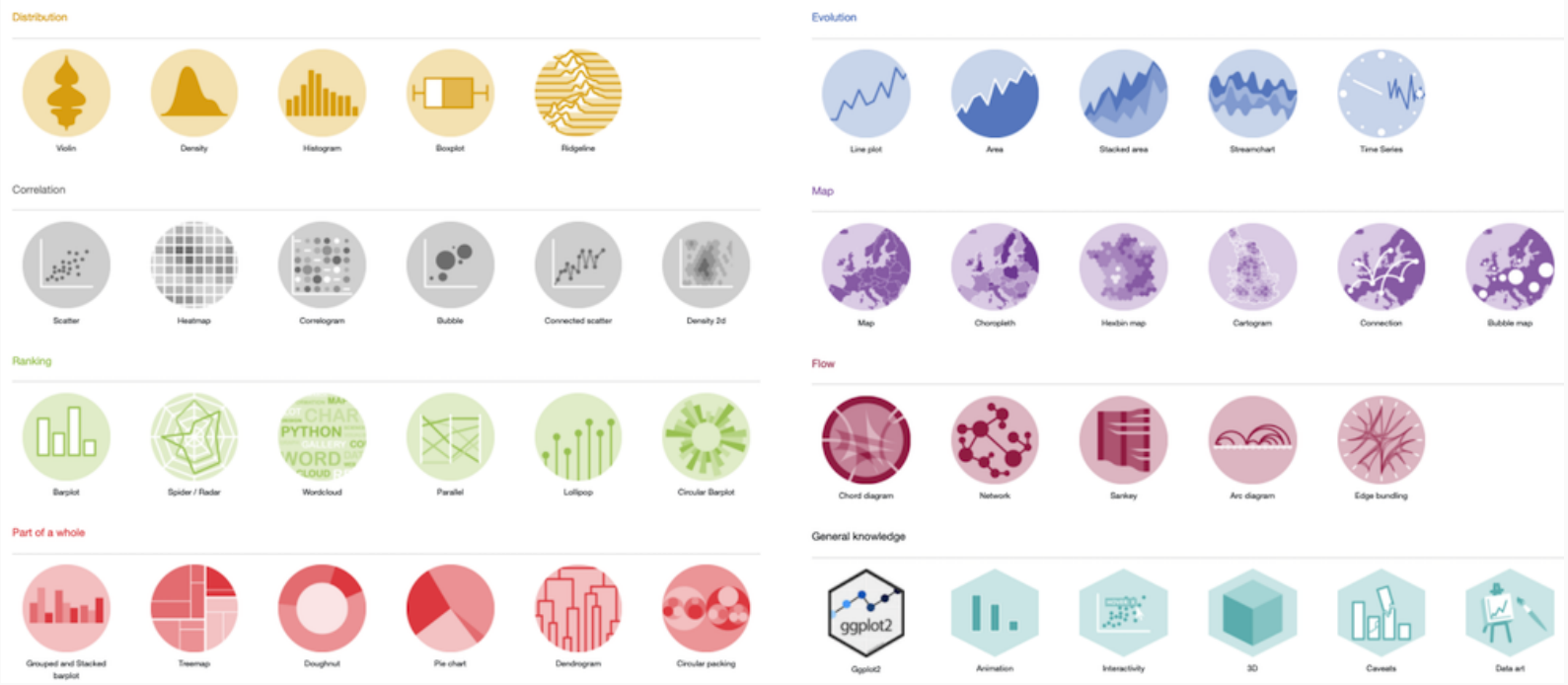
- The ability to condense diffused verbal information into a compact and quickly understood graphical image.
- Data visualization and numerical summarization provide us with both a powerful tool to explore data

Where do visualization techniques fit into the data mining process?

- In the pre-processing portion of the data mining process.
- Visualization supports data cleaning by
  - finding incorrect values
  - missing values, duplicate rows
  - columns with all the same value
- Variable derivation and selection
- Determining appropriate bin sizes
- Data reduction process.
- Determine which variables and metrics are useful.

Graphs for Data Exploration

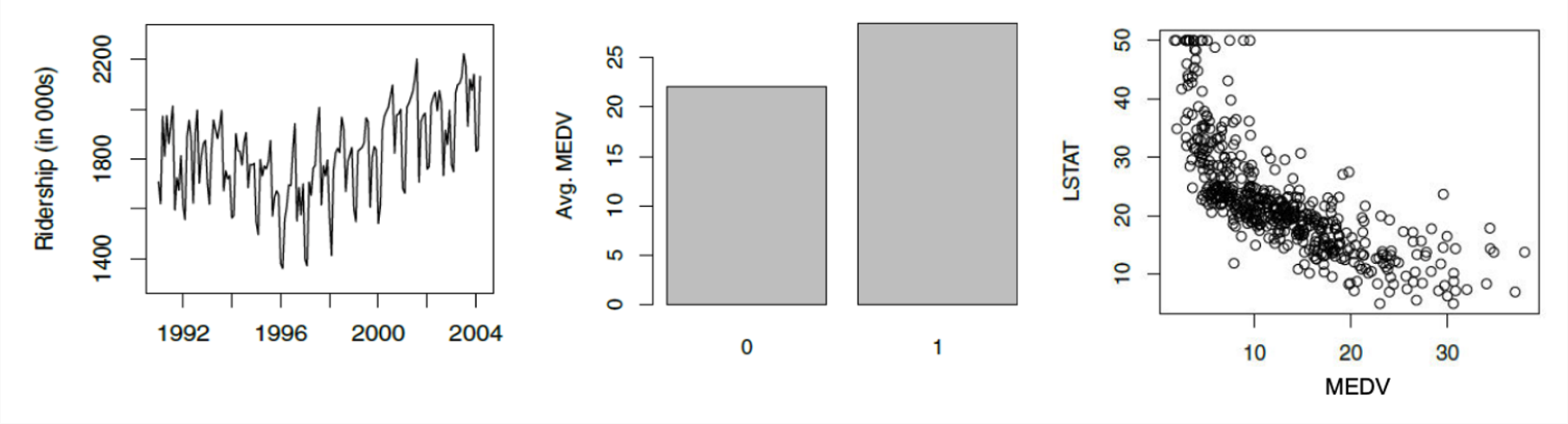
- Basic Plot
  - Line graphs
  - Bar chart
  - Scatterplots
- Distribution Plots
  - Boxplots
  - Histograms



plot chart

Basic Graphs

- **Line graphs** are used primarily for showing time series. The choice of time frame to plot, as well as the temporal scale, should depend on the horizon of the forecasting task and on the nature of the data.
- **Bar charts** are useful for comparing a single statistic (e.g., average, count, percentage) across groups.
- **A scatter plot** (aka scatter chart, scatter graph) uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

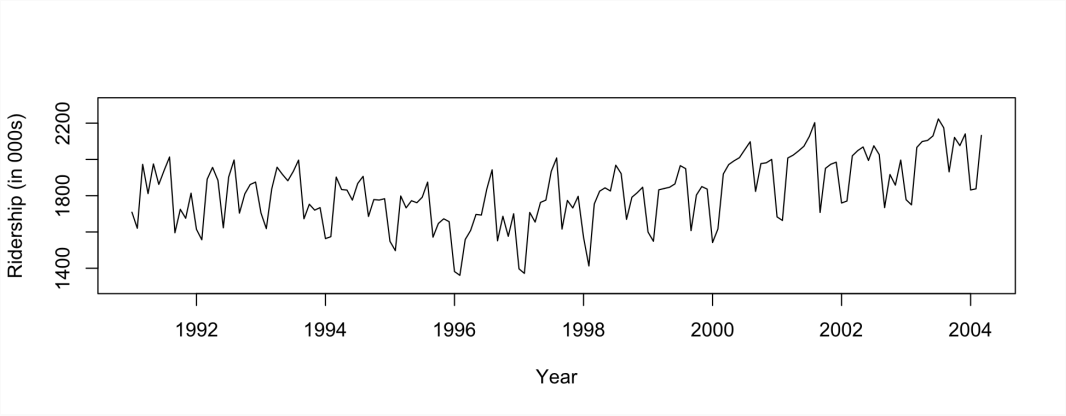


Line Graph for Time Series

```
library(forecast)
ridership.ts <- ts(Amtrak.df$Ridership,
                   start = c(1991, 1),
                   end = c(2004, 3),
                   freq = 12)
head(ridership.ts)
```

```
##           Jan      Feb      Mar      Apr      May      Jun
## 1991 1708.917 1620.586 1972.715 1811.665 1974.964 1862.356
```

```
plot(ridership.ts,
     xlab = "Year",
     ylab = "Ridership (in 000s)",
     ylim = c(1300, 2300))
```

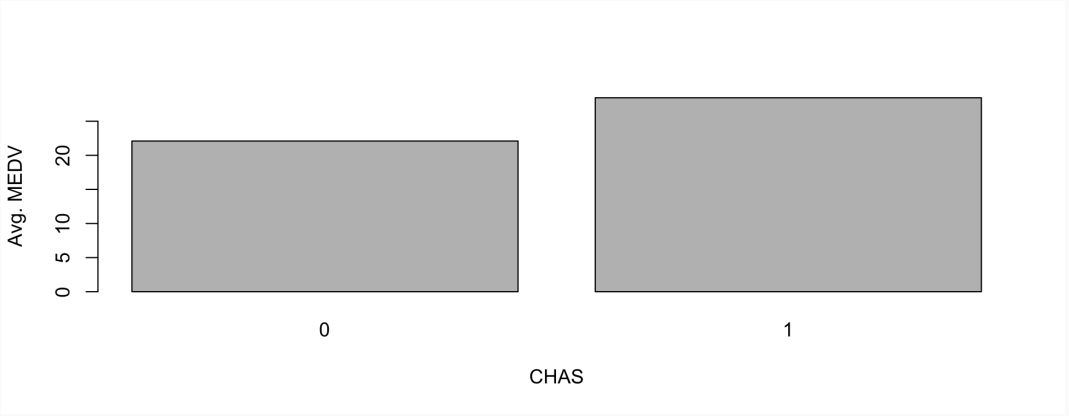


Bar Chart for Categorical Variable

```
# barchart of CHAS vs. mean MEDV
# 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")
head(data.for.plot)
```

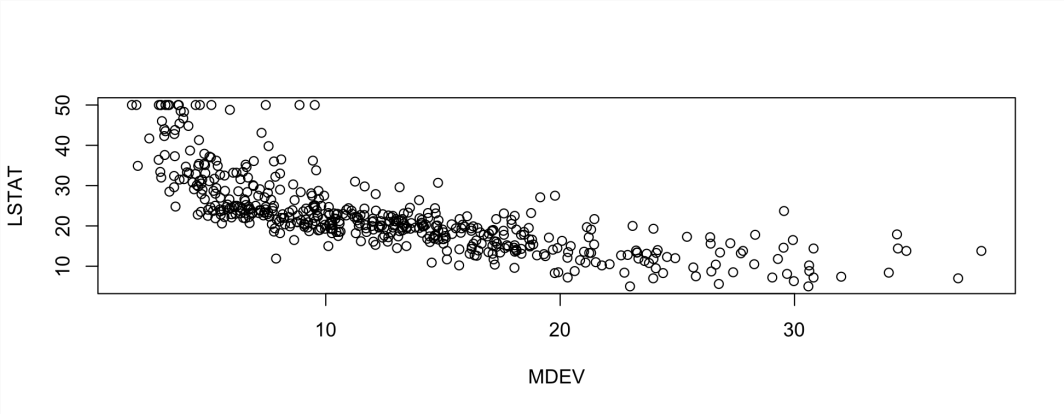
```
##   CHAS MeanMEDV
## 1    0 22.09384
## 2    1 28.44000
```

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



Scatterplot

```
## scatter plot with axes names
plot(housing.df$MEDV ~
       housing.df$LSTAT,
     xlab = "MDEV",
     ylab = "LSTAT")
```



Distribution plots

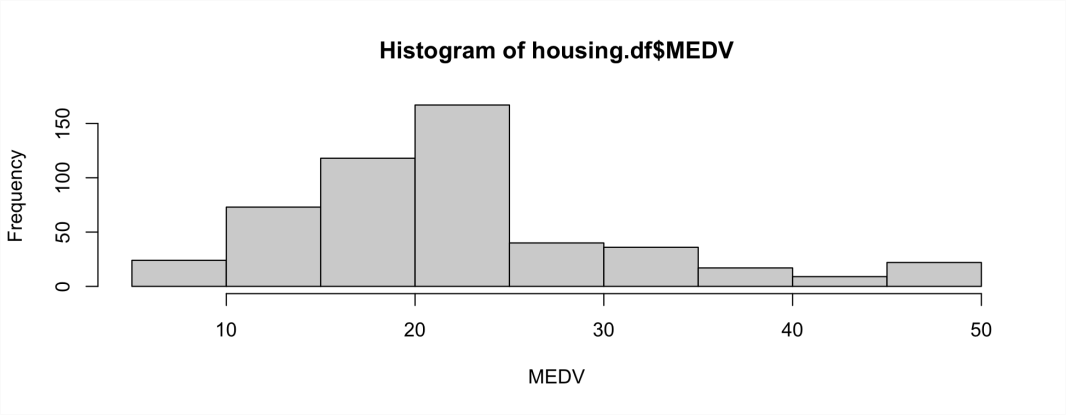
Display "how many" of each value occur in a data set

Or, for continuous data or data with many possible values, “how many” values are in each of a series of ranges or “bins”

Histogram

Boston Housing example: Histogram shows the distribution of the outcome variable (median house value)

```
hist(housing.df$MEDV, xlab = "MEDV")
```

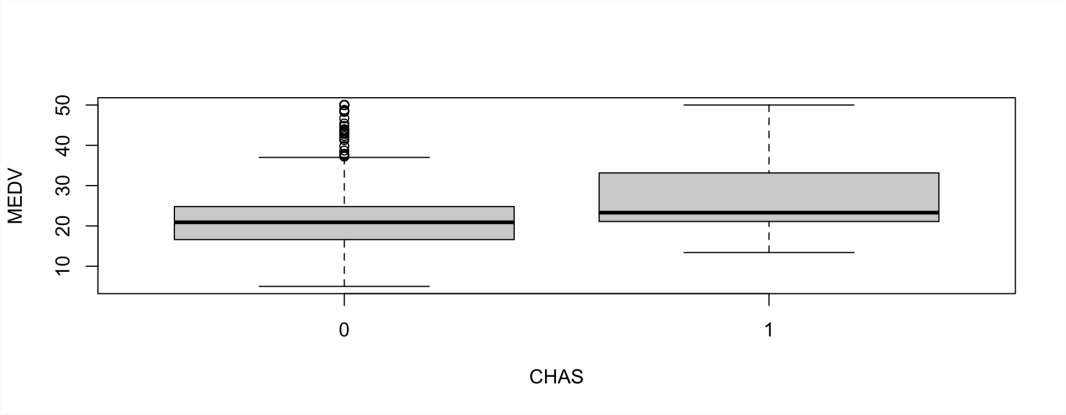


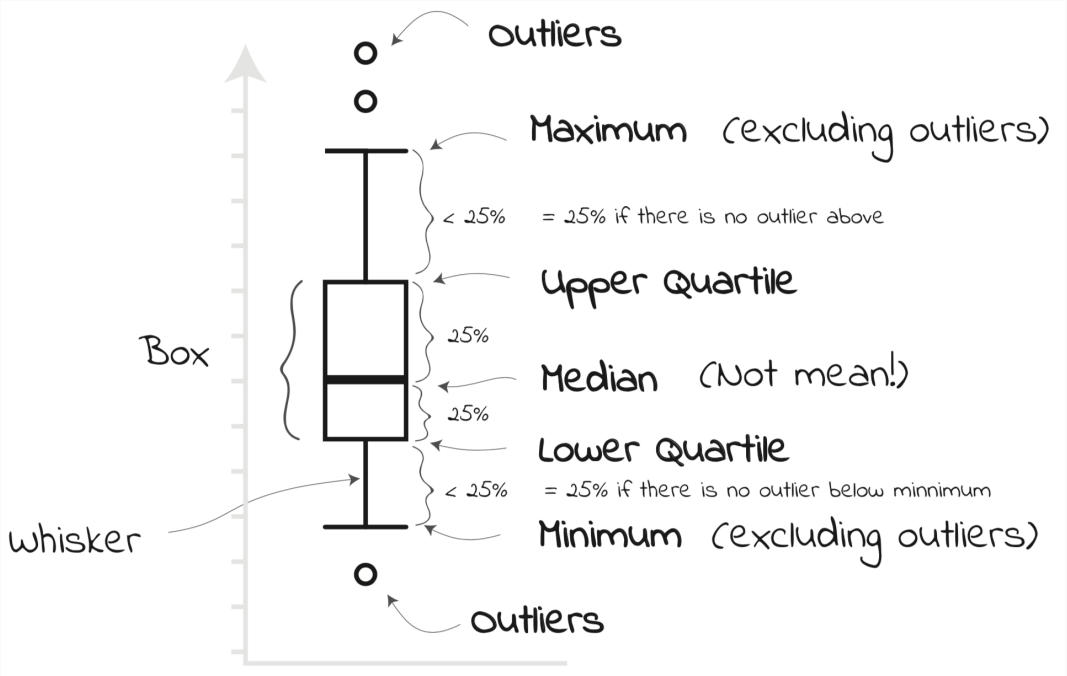
About 40 neighborhoods had a median house value < $10,000 (these data are from mid-20th century)

Boxplots

- Side-by-side boxplots are useful for comparing
- Houses in neighborhoods on Charles river (1) are more valuable than those not (0)

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





- Top outliers defined as those above Q3+1.5(Q3-Q1).
- “max” = maximum of non-outliers
- Analogous definitions for bottom outliers and for “min”
- Details may differ across software

Multidimensional Visualization

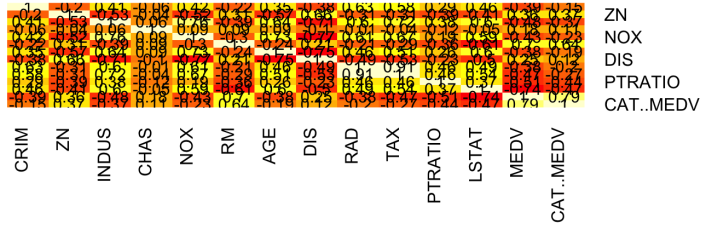
Heat Maps

- Color conveys information
- Visualize
  - Correlations
  - Missing Data

Heatmap to highlight correlations

- Darker & redder = more negative correlation
- Lighter and yellower = more positive correlation

```r
## heatmap with values
library(gplots)
heatmap.2(cor(housing.df),
          Rowv = FALSE,
          Colv = FALSE,
          dendrogram = "none",
          cellnote = round(cor(housing.df),2),
          notecol = "black",
          key = FALSE,
          trace = 'none',
          margins = c(10,10))
```

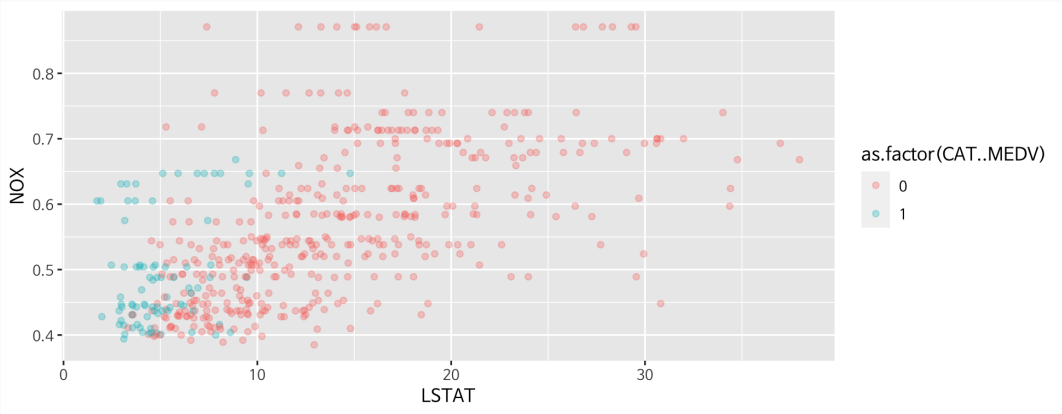


Scatterplot with color/shade added

Boston Housing NOX vs. LSTAT

light shade = low median value | dark shade = high median value

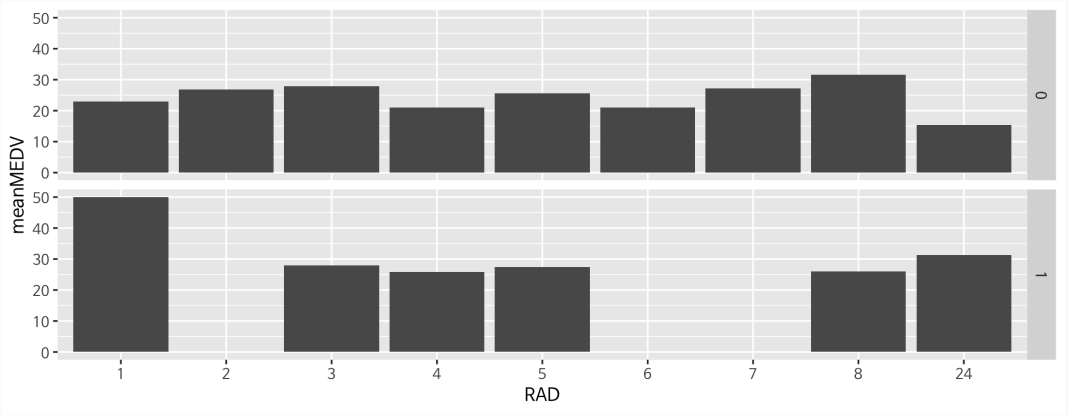
```
ggplot(housing.df,
       aes(y = NOX,
           x = LSTAT,
           colour= as.factor(CAT..MEDV))) +
  geom_point(alpha = 3/10)
```



Bar chart for MEDV vs. RAD

first for low-value neighborhoods, then for high value neighborhoods

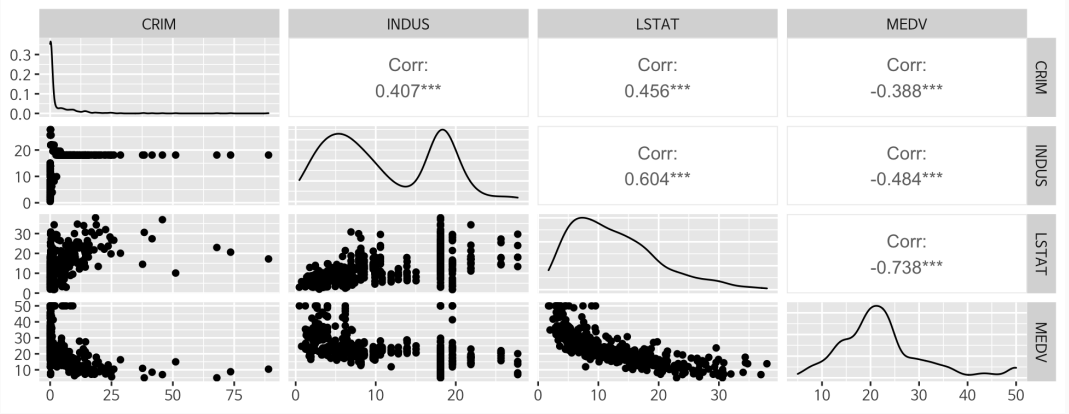
```
data.for.plot <- aggregate(housing.df$MEDV,
                           by = list(housing.df$RAD,
                                     housing.df$CHAS),
                           FUN = mean,
                           drop = FALSE)
colnames(data.for.plot) <- c("RAD","CHAS","meanMEDV")
ggplot(data.for.plot) +
  geom_bar(
    aes(x = as.factor(RAD),
        y = meanMEDV),
    stat = "identity") +
  xlab("RAD") +
  facet_grid(CHAS ~ .)
```



Matrix scatterplot

Diagonal plot is the frequency distribution for the variable

```
## simple plot
# use plot() to generate a matrix of 4X4 panels with va
# plot(housing.df[, c(1, 3, 12, 13)])
# alternative, nicer plot (displayed)
library(GGally)
ggpairs(housing.df[, c(1, 3, 12, 13)])
```



Manipulation

Rescaling

- Rescaling Changing the scale in a display can enhance the plot and illuminate relationships. The rescaling removes this crowding and allows a better view of the linear relationship between the two log-scaled variables (indicating a log–log relationship).

Aggregation

- Another useful manipulation of scaling is changing the level of aggregation. For a temporal scale, we can aggregate by different granularity (e.g., monthly, daily, hourly)
  - A popular aggregation for time series is a moving average, where the average of neighboring values within a given window size is plotted.
  - Non-temporal variables can be aggregated if some meaningful hierarchy exists:
    - geographical (tracts within a zip code in the Boston Housing example),
    - organizational (people within departments within units)

Zooming and Panning

- The ability to zoom in and out of certain areas of the data on a plot is important for revealing patterns and outliers. We are often interested in more detail on areas of dense information or of special interest.
- Panning refers to the operation of moving the zoom window to other areas (popular in mapping applications such as Google Maps).

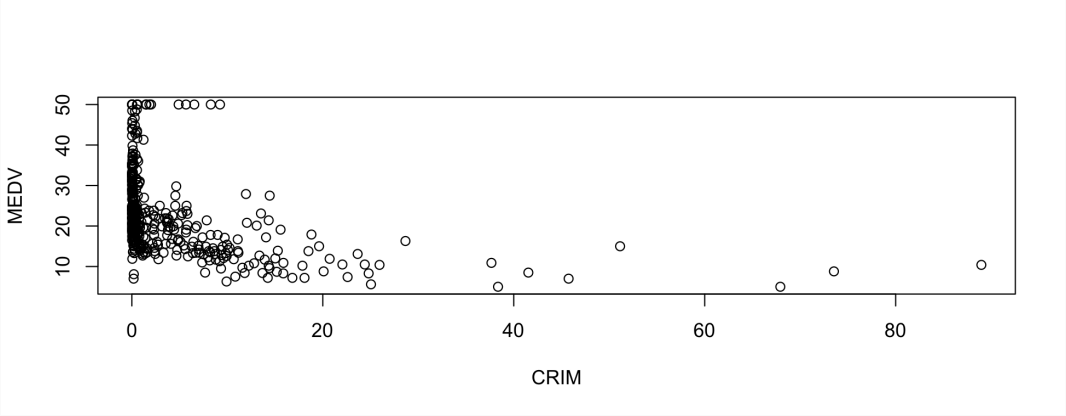
Filtering

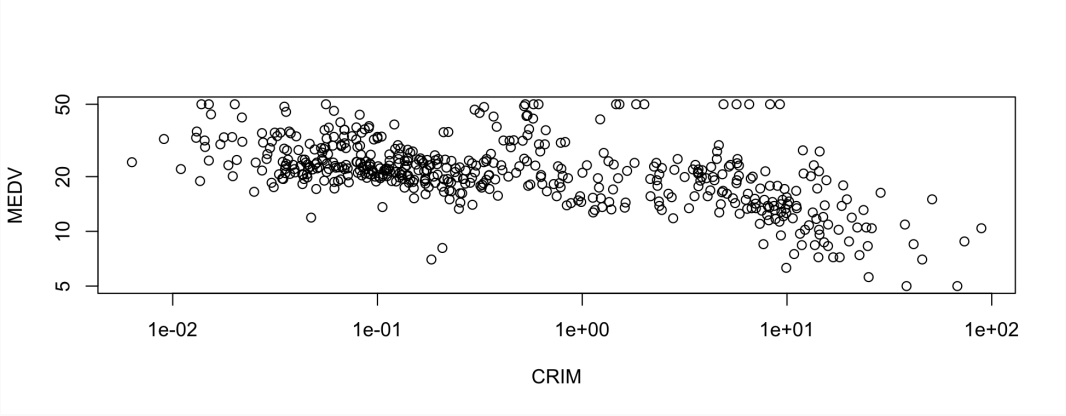
- Filtering means removing some of the observations from the plot.
- The purpose of filtering is to focus the attention on certain data while eliminating “noise” created by other data.

Rescaling to log scale (on right) “uncrowds” the data

```
## scatter plot: regular and log scale
plot(housing.df$MEDV ~ housing.df$CRIM,
     xlab = "CRIM",
     ylab = "MEDV")
```

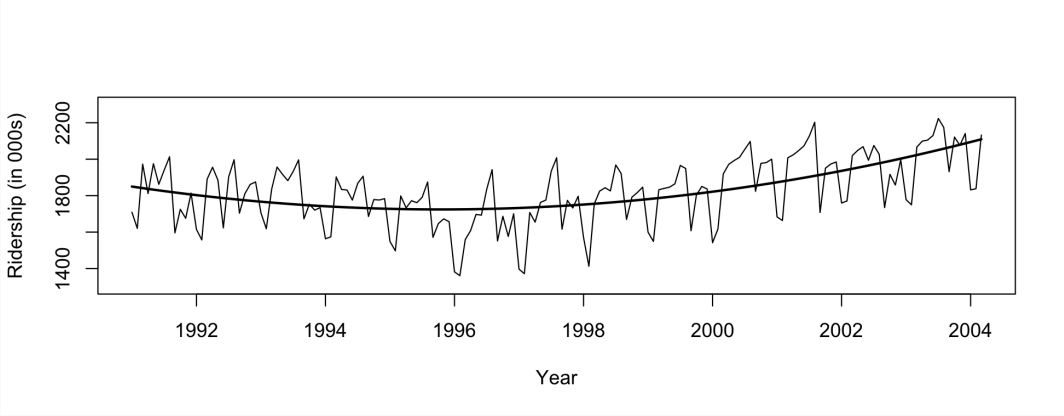
```
# to use logarithmic scale set argument log = to either
plot(housing.df$MEDV ~ housing.df$CRIM,
     xlab = "CRIM",
     ylab = "MEDV",
     log = 'xy')
```





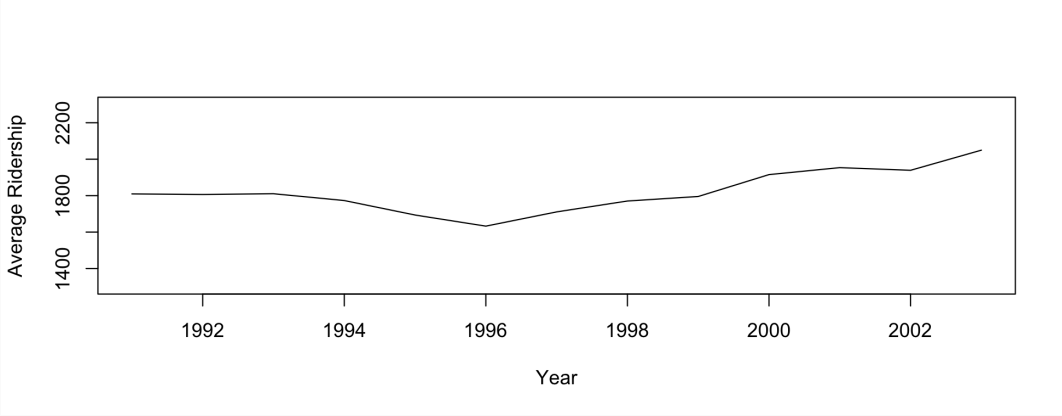
Amtrak Ridership – Monthly Data – Curve Added

```
ridership.lm <- tslm(ridership.ts ~ trend +
                       I(trend^2))
plot(ridership.ts,
     xlab = "Year",
     ylab = "Ridership (in 000s)",
     ylim = c(1300, 2300))
lines(ridership.lm$fitted, lwd = 2)
```



Annual

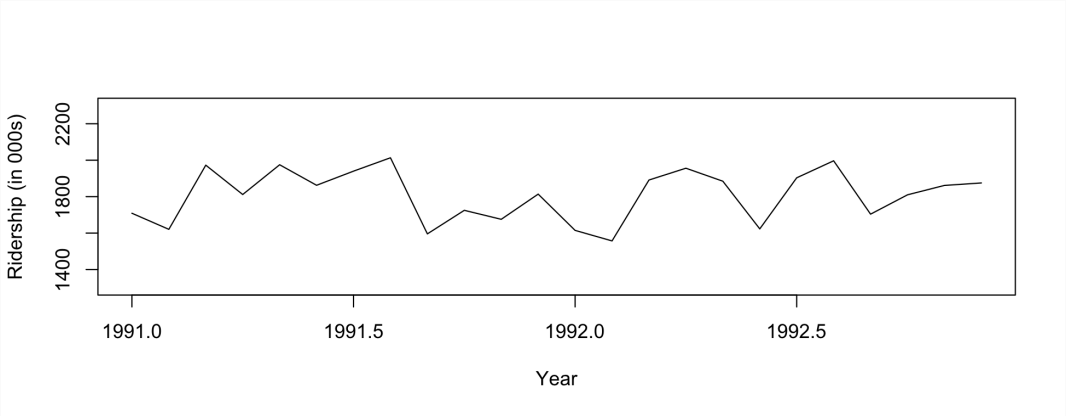
```
annual.ridership.ts <- aggregate(ridership.ts,
                                 FUN = mean)
plot(annual.ridership.ts,
     xlab = "Year",
     ylab = "Average Ridership",
     ylim = c(1300, 2300))
```



Amtrak Ridership

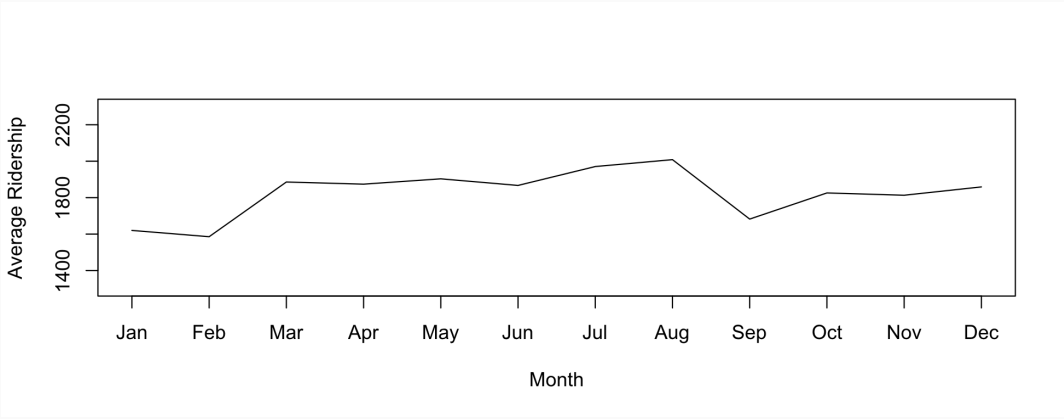
2 Year (1991-1992)

```
ridership.2yrs <- window(ridership.ts,
                         start = c(1991,1),
                         end = c(1992,12))
plot(ridership.2yrs,
     xlab = "Year",
     ylab = "Ridership (in 000s)",
     ylim = c(1300, 2300))
```



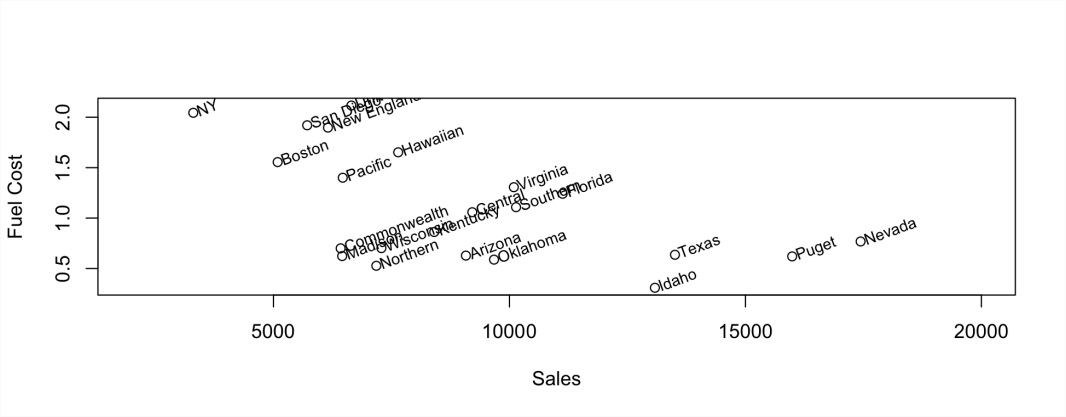
Month

```
monthly.ridership.ts <- tapply(ridership.ts,
                               cycle(ridership.ts),
                               mean)

plot(monthly.ridership.ts,
     xlab = "Month",
     ylab = "Average Ridership",
     ylim = c(1300, 2300),
     type = "l",
     xaxt = 'n')

## set x labels
axis(1, at = c(1:12),
     labels = c("Jan","Feb","Mar",
                "Apr","May","Jun",
                "Jul","Aug","Sep",
                "Oct","Nov","Dec"))
```



Scatter Plot with Labels (Utilities)

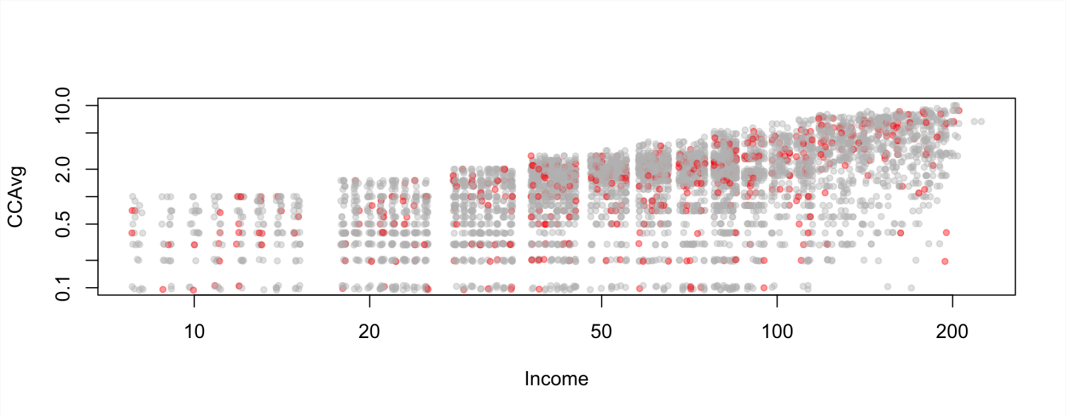
```
plot(utilities.df$Fuel_Cost ~
       utilities.df$Sales,
     xlab = "Sales",
     ylab = "Fuel Cost",
     xlim = c(2000, 20000))
text(x = utilities.df$Sales,
     y = utilities.df$Fuel_Cost,
     labels = utilities.df$Company,
     pos = 4,
     cex = 0.8,
     srt = 20,
     offset = 0.2)
```



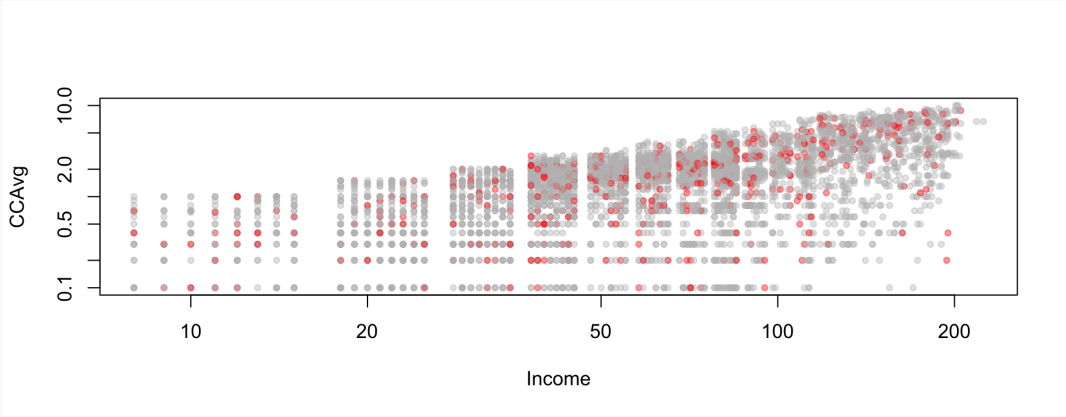
Scaling: Smaller markers, jittering, color contrast

(Universal Bank; red = accept loan)

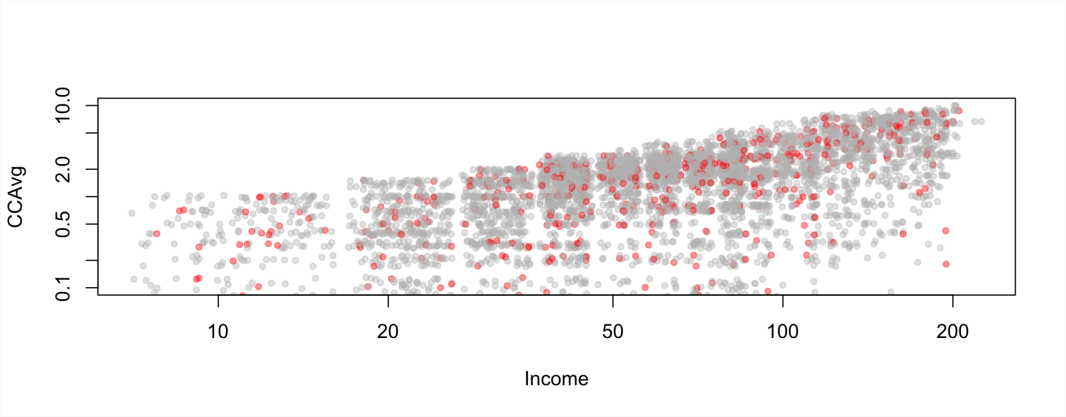
```
# use function alpha() in library scales to add transpa
library(scales)
plot(
    jitter(universal.df$CCAvg, 1) ~
        jitter(universal.df$Income, 1),
    col = alpha(
        ifelse(universal.df$Securities.Account == 0,
        "gray",
        "red"),
        0.4),
    pch = 20,
    log = 'xy',
    ylim = c(0.1, 10),
    xlab = "Income",
    ylab = "CCAvg")
```



Graph without jitter



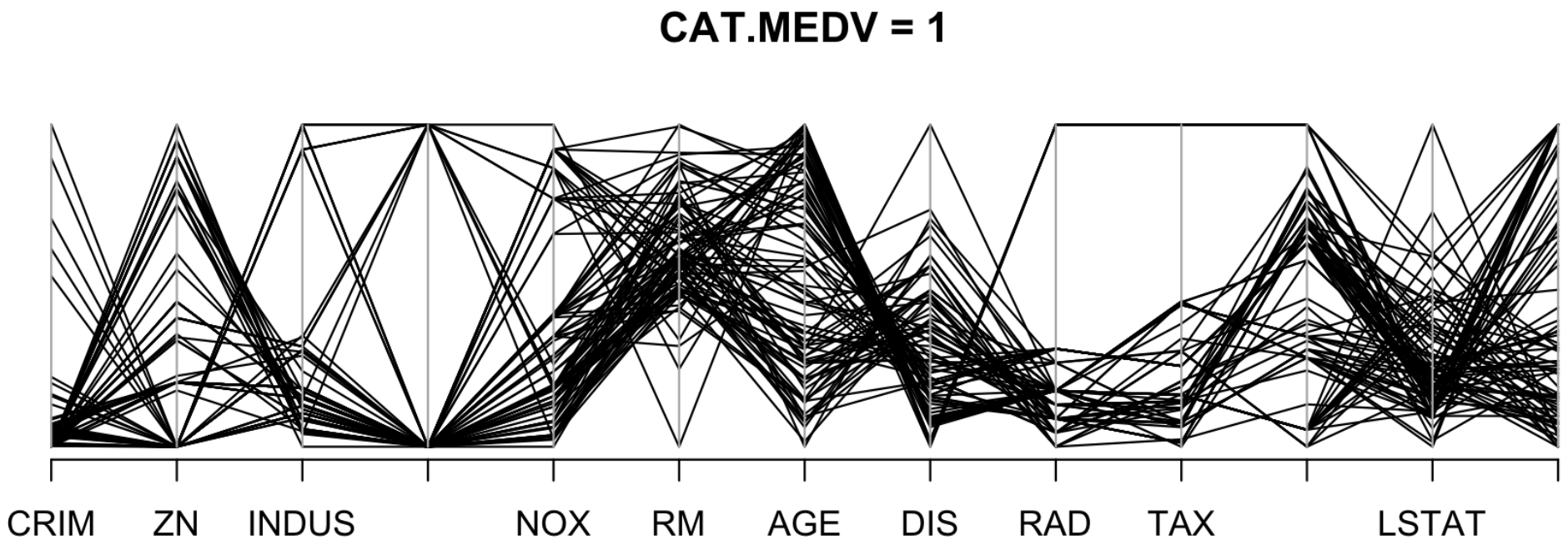
Graph with jitter



Parallel Coordinate Plot (Boston Housing)

```
library(MASS)
parcoord(housing.df[housing.df$CAT..MEDV == 0, -14], main = "CAT.MEDV = 0")
```

```
parcoord(housing.df[housing.df$CAT..MEDV == 1, -14], main = "CAT.MEDV = 1")
```



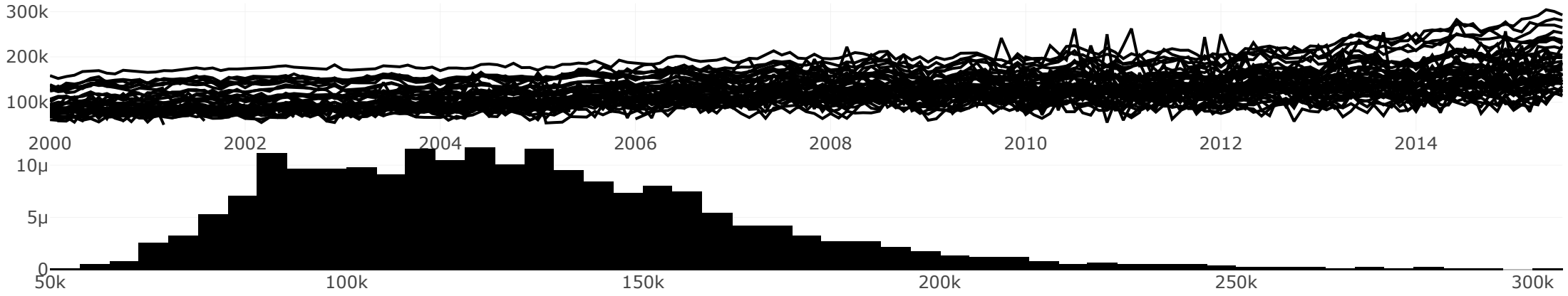
Linked Plot

Brush color SharedData6656de4d

rgba(228

Brush color asdasd

SharedData6656de4d

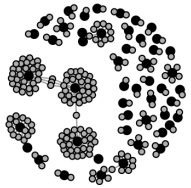


Network Graph

- Network analysis techniques were spawned by the explosion of social and product network data.
    - Networks of sellers and buyers on eBay and networks of users on Facebook.
    - Network of products on Amazon (linked through the recommendation system).
- Networks can also have nodes of multiple types.
    - A common structure is networks with two types of nodes.
    - A two-type node network
        - a network of sellers and buyers on the online auction site

eBay Auctions

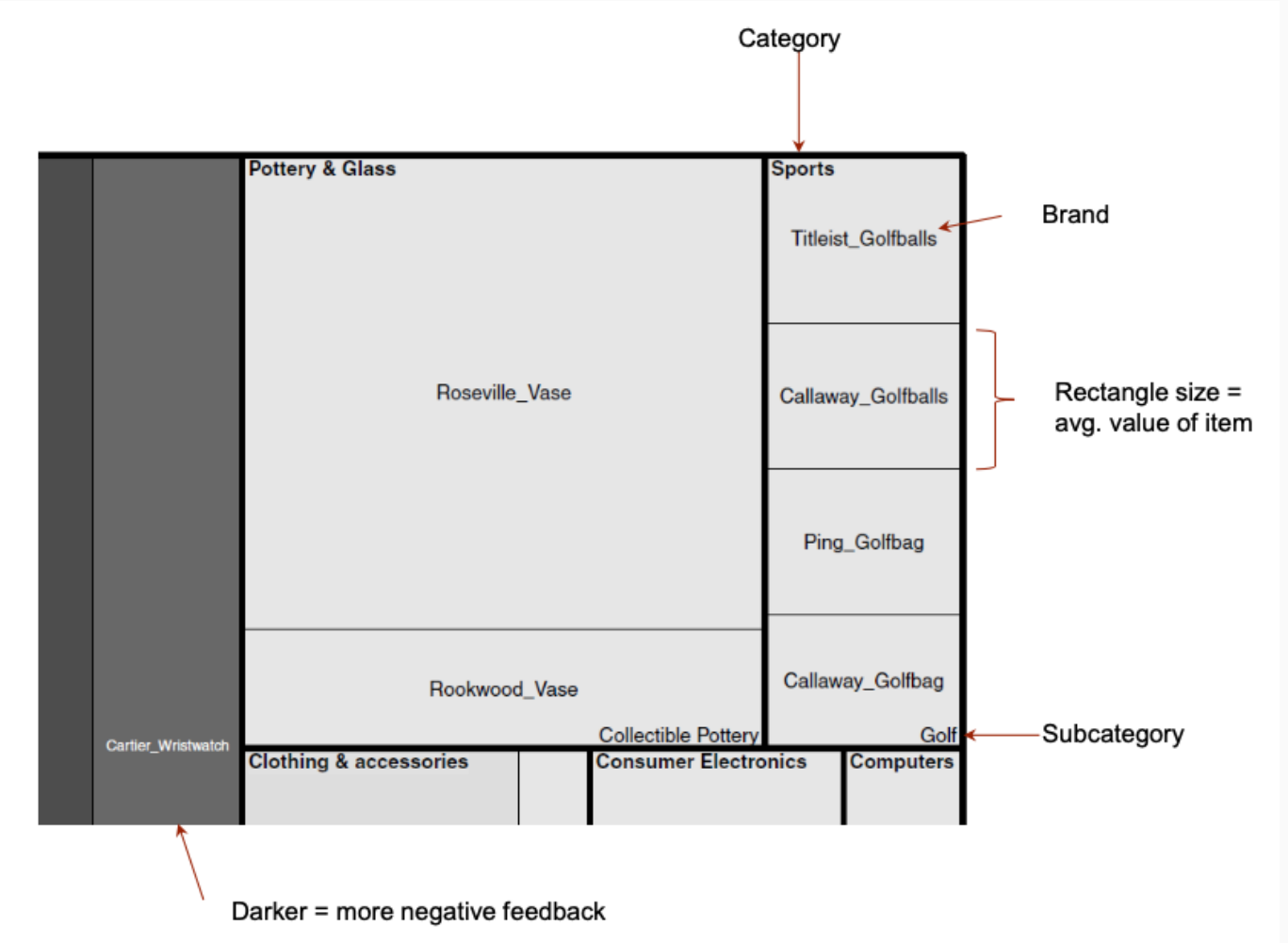
```
library(igraph)
ebay.df <- read.csv("data/ch3/eBayNetwork.csv")

# transform node ids to factors
ebay.df[,1] <- as.factor(ebay.df[,1])
ebay.df[,2] <- as.factor(ebay.df[,2])
graph.edges <- as.matrix(ebay.df[,1:2])

g <- graph.edgelist(graph.edges, directed = FALSE)
isBuyer <- V(g)$name %in% graph.edges[,2]

plot(g, vertex.label = NA,
     vertex.color = ifelse(isBuyer, "gray", "black"),
     vertex.size = ifelse(isBuyer, 7, 10))
```



Visualizing Hierarchical Data: Treemaps

Hierarchical data and the exploration of data at different hierarchy levels

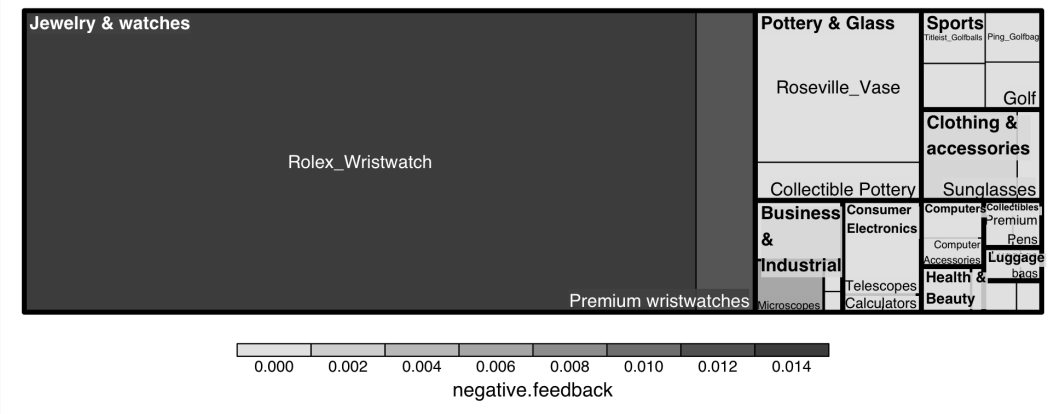
- Treemaps are useful visualizations specialized for exploring large data sets that are hierarchically structured (tree-structured).
- They allow exploration of various dimensions of the data while maintaining the hierarchical nature of the data.
- Auctions from eBay.com
  - Hierarchically ordered by item category, sub-category, and brand.
  - The levels in the hierarchy of the treemap are visualized as rectangles containing sub-rectangles.
- Categorical variables can be included in the display by using hue. - Numerical variables can be included via rectangle size and color intensity


Category
Pottery & Glass
Sports
Titleist_Golfballs
Brand
Roseville_Vase
Callaway_Golfballs
Rectangle size =
avg. value of item
Ping_Golfbag
Rookwood_Vase
Callaway_Golfbag
Collectible Pottery
Golf
Subcategory
Cartier_Wristwatch
Clothing & accessories
Consumer Electronics
Computers
Darker = more negative feedback

Treemap – eBay Auctions)

Hierarchical eBay data: Category> sub-category> Brand

```
library(treemap)
tree.df <- read.csv("data/ch3/EbayTreemap.csv")

# add column for negative feedback
tree.df$negative.feedback <- 1* (tree.df$Seller.Feedbac

# draw treemap
treemap(tree.df,
        index = c("Category","Sub.Category", "Brand"),
        vSize = "High.Bid",
        vColor = "negative.feedback",
        fun.aggregate = "mean",
        align.labels = list(
          c("left", "top"),
          c("right", "bottom"),
          c("center", "center")),
        palette = rev(gray.colors(3)),
        type = "manual",
```



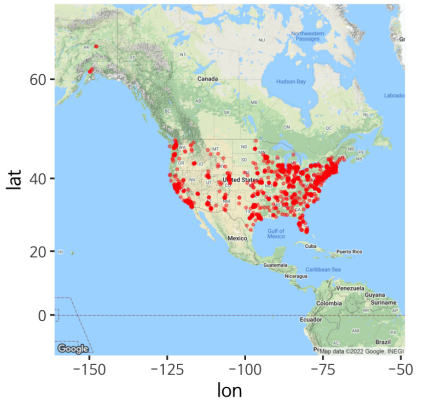
Visualizing Geographical Data: Map Charts

- Geographical information.
  - Zip codes are one example of a categorical variable
  - Plotting the data on a geographic map can often reveal patterns that are harder to identify
- A map chart uses a geographical map as background,
  - Color, hue, and other features are used to include categorical or numerical variables.
  - Besides specialized mapping software, maps are now becoming part of general-purpose software

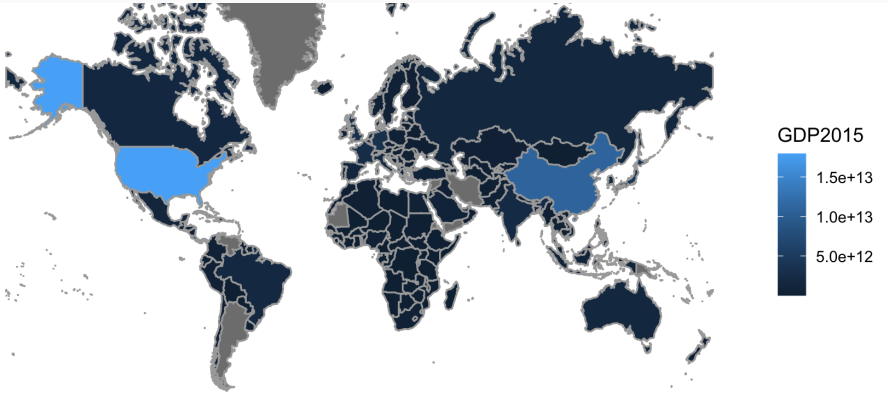
Using Google Maps

Location of Statistics.com students and instructors

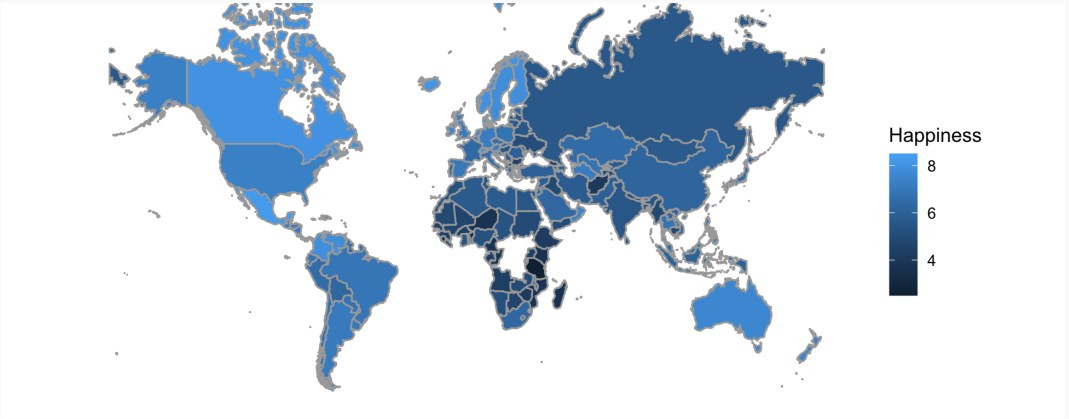
```
library(ggmap)
#register_google(key = "YOUR API KEY")
SCstudents <- read.csv("data/ch3/SC-US-students-GPS-dat
Map <- get_map("Denver, CO", zoom = 3)
ggmap(Map) +
  geom_point(
    aes(x = longitude,
        y = latitude),
    data = SCstudents,
    alpha = 0.4,
    colour = 'red',
    size = 0.5)
```



```
library(mosaic)
gdp.df <- read.csv("data/ch3/gdp.csv",
                   skip = 4,
                   stringsAsFactors = FALSE)
names(gdp.df)[5] <- "GDP2015"
happiness.df <- read.csv("data/ch3/Veerhoven.csv")

# gdp map
mWorldMap(gdp.df,
          key = "Country.Name",
          fill = "GDP2015") +
  coord_map()
```



```
# well-being map
mWorldMap(happiness.df,
          key = "Nation",
          fill = "Score") +
  coord_map() +
  scale_fill_continuous(name="Happiness")
```



Summary: Major Visualizations and Operations, by Da

1. Prediction

- Plot outcome on the y-axis of boxplots, bar charts, and scatter plots.
- Study relation of outcome to categorical predictors via side-by-side box plots, bar charts, and multiple panels.
- Study relation of outcome to numerical predictors via scatter plots.
- Use distribution plots (boxplot, histogram) for determining needed transformations of the outcome variable (and/or numerical predictors).
- Examine scatter plots with added color/panels/size to determine the need for interaction terms.
- Use various aggregation levels and zooming to determine areas of the data with different behavior, and to evaluate the level of global vs. local patterns.

2. Classification

- Study relation of outcome to categorical predictors using bar charts with the outcome on the y-axis.
- Studyrelationofoutcometopairsofnumericalpredictorsviacolor-coded scatter plots (color denotes the outcome).
- Study relation of outcome to numerical predictors via side-by-side box-plots: Plot boxplots of a numerical variable by outcome. Create similar displays for each numerical predictor. The most separable boxes indicate potentially useful predictors.
- Use color to represent the outcome variable on a parallel coordinate plot.
- Use distribution plots (boxplot, histogram) for determining needed trans- formations of numerical predictor variables.
- Examine scatter plots with added color/panels/size to determine the need for interaction terms.
- Use various aggregation levels and zooming to determine areas of the data with different behavior, and to evaluate the level of global vs. local patterns.

3. Time Series Forecasting

- Create line graphs at different temporal aggregations to determine types of patterns.
- Use zooming and panning to examine various shorter periods of the series to determine areas of the data with different behavior.
- Use various aggregation levels to identify global and local patterns.
- Identify missing values in the series (that will require handling).
- Overlay trend lines of different types to determine adequate modeling choices.

4. Unsupervised Learning

- Create scatter plot matrices to identify pairwise relationships and clustering of observations.
- Use heatmaps to examine the correlation table.
- Use various aggregation levels and zooming to determine areas of the data with different behavior.
- Generate a parallel coordinates plot to identify clusters of observations.

Chapter Video

References

Shmueli, G., P. C. Bruce, P. Gedeck, and N. R. Patel (2019). Data mining for business analytics: concepts, techniques and applications in Python. John Wiley & Sons.

Shmueli, G., P. C. Bruce, I. Yahav, N. R. Patel, and K. C. Lichtendahl Jr (2017). Data mining for business analytics: concepts, techniques, and applications in R. John Wiley & Sons.

Assignment