Data Visualization
Part 2: Heatmaps and Multidimensional Visualization

Aram Balagyozyan

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

February 23, 2022

Outline

Outline

Heatmaps: Visualizing Correlations and Missing Values

- A heatmap is a graphical display of numerical data where color is used to denote values.
- Heatmaps are especially useful for
  1. visualizing correlation tables and
  2. visualizing missing values in the data.
- Correlation tables can be very useful in quickly assessing the correlation between the variables in the data.
- Suppose we wanted to visualize the correlation between the variables in `housing.df`. As a first step, we could use the `cor()` function in the base R.

Visualizing Correlations

- Try running the following code:

```
cor(housing.df)
```

- As you can see, the output is a 14×14 matrix whereby the element located on the intersection of i-th row and j-th column is the correlation between the i-th-row variable and j-th-column variable.
- For example, you can observe from the output of the `cor()` command above that the correlation between `INDUS` and `NOX` is 0.764.
- Since the correlation between any pair of variables `x` and `y` is the same as between `y` and `x`, the correlation matrix is symmetric, with the triangle below the diagonal being the exact copy of the triangle above the diagonal.
- Since the correlation between and variable `x` and itself is 1, the diagonal of a correlation matrix contains only unit elements.
- Heatmaps, can significantly improve the readability of a correlation matrix

Correlation Heatmap Using Base R and the **corrplot** Package

- To produce a correlation heatmap without correlation values, try running the following code:

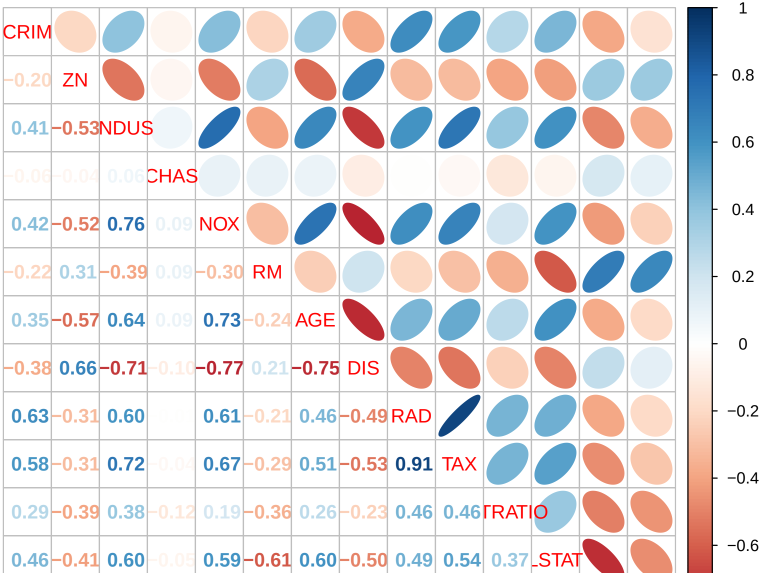
```
heatmap(cor(housing.df), Rowv = NA, Colv = NA)
```

- Although the output of the code above is a pretty picture with the color demonstrating the degree of correlation, it would be much more useful if we could also observe correlation values. To achieve that, I am going to rely on the **corrplot** package.

```
library(corrplot)
cm<-cor(housing.df)
corrplot.mixed(cm,lower="number",
               upper = "ellipse",tl.pos="d")
```

Correlation Heatmap Using Base R and the **corrplot** Package

- Below is the correlation heatmap produced by the above code.



Understanding and Interpreting the Output of `corrplot()`

- `corplot.mixed()` creates an heatmap correlation matrix where the degrees of correlation are presented in color plus in two different formats, graphical (ellipses) and numerical.
- The color spectrum varies from dark red (correlation = -1) to dark blue (correlation = + 1).
- The `lower="number"` option populates the lower triangle with correlation coefficients. Similarly, the `upper="ellipse"` option populates the upper triangle with ellipses. The thinner an ellipse the stronger the correlation (whether positive or negative).
- `tl.pos="d"` places the labels on the diagonal.
- The two most closely correlated variables in `housing.df` are `RAD` and `TAX` with the correlation coefficient being 0.91. This indicates that the more accessible a neighborhood is from a highway, the higher on average the property tax rate.
- If predicting `MEDV` is your goal, then it is very helpful to know that `LSTAT`, `RM`, and `PTRATIO` are the three variables that correlate most closely with `MEDV`.

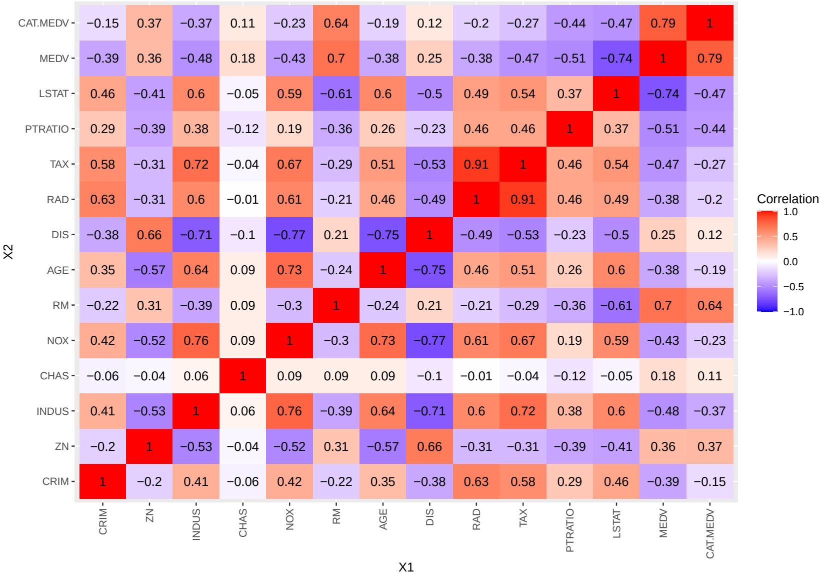
Correlation Heatmap Using ggplot()

- Just for the sake of completeness, below is the ggplot code that will produce a correlation heatmap matrix:

```
library(ggplot2)
library(reshape) # need this to melt
cor.mat <- round(cor(housing.df),2)
melted.cor.mat <- melt(cor.mat)
ggplot(melted.cor.mat,
   aes(x = X1, y = X2, fill = value)) +
  scale_fill_gradient2(low = "blue",
        high = "red", mid = "white",
  midpoint = 0, limit = c(-1,1),
  name="Correlation")+
  geom_tile() +
  geom_text(aes(label = value))+
  # + x-axis label vertical alignment
  theme(axis.text.x = element_text(
angle = 90, vjust = 0.5, hjust=1))
```

Correlation Heatmap Using ggplot()

Below is the output of the code on the previous slide.



X2 \ X1	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV	CAT.MEDV
CAT.MEDV	−0.15	0.37	−0.37	0.11	−0.23	0.64	−0.19	0.12	−0.2	−0.27	−0.44	−0.47	0.79	1
MEDV	−0.39	0.36	−0.48	0.18	−0.43	0.7	−0.38	0.25	−0.38	−0.47	−0.51	−0.74	1	0.79
LSTAT	0.46	−0.41	0.6	−0.05	0.59	−0.61	0.6	−0.5	0.49	0.54	0.37	1	−0.74	−0.47
PTRATIO	0.29	−0.39	0.38	−0.12	0.19	−0.36	0.26	−0.23	0.46	0.46	1	0.37	−0.51	−0.44
TAX	0.58	−0.31	0.72	−0.04	0.67	−0.29	0.51	−0.53	0.91	1	0.46	0.54	−0.47	−0.27
RAD	0.63	−0.31	0.6	−0.01	0.61	−0.21	0.46	−0.49	1	0.91	0.46	0.49	−0.38	−0.2
DIS	−0.38	0.66	−0.71	−0.1	−0.77	0.21	−0.75	1	−0.49	−0.53	−0.23	−0.5	0.25	0.12
AGE	0.35	−0.57	0.64	0.09	0.73	−0.24	1	−0.75	0.46	0.51	0.26	0.6	−0.38	−0.19
RM	−0.22	0.31	−0.39	0.09	−0.3	1	−0.24	0.21	−0.21	−0.29	−0.36	−0.61	0.7	0.64
NOX	0.42	−0.52	0.76	0.09	1	−0.3	0.73	−0.77	0.61	0.67	0.19	0.59	−0.43	−0.23
CHAS	−0.06	−0.04	0.06	1	0.09	0.09	0.09	−0.1	−0.01	−0.04	−0.12	−0.05	0.18	0.11
INDUS	0.41	−0.53	1	0.06	0.76	−0.39	0.64	−0.71	0.6	0.72	0.38	0.6	−0.48	−0.37
ZN	−0.2	1	−0.53	−0.04	−0.52	0.31	−0.57	0.66	−0.31	−0.31	−0.39	−0.41	0.36	0.37
CRIM	1	−0.2	0.41	−0.06	0.42	−0.22	0.35	−0.38	0.63	0.58	0.29	0.46	−0.39	−0.15

Correlation Heatmap Using `ggplot()`

- To understand what `melt()` does, consider the following 2×2 matrix.

```
champs
```

```
##      pullups pushups
## John       1       3
## Paul       2       4
```

```
melt(champs)
```

```
##     X1      X2 value
## 1 John pullups     1
## 2 Paul pullups     2
## 3 John pushups     3
## 4 Paul pushups     4
```

- `melt()` reshapes a matrix into a form suitable for (gg)plotting.
- To see the difference between `cor.mat` and `melt(cor.mat)`, type `cor.mat` in the Console and hit Enter then type `melt(cor.mat)` again in the Console and hit Enter.

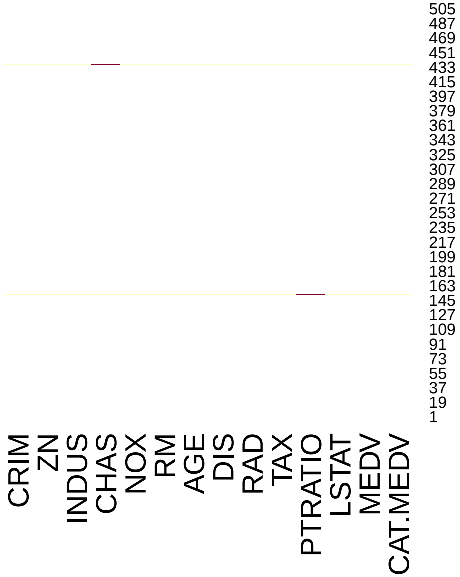
Using `heatmap()` to Detect `NA`s in the Data.

- In order to demonstrate the use of `heatmap()` for detecting missing values, let's first randomly replace 2 values in `housing.df` with `NA`s and then detect them using the proposed technique.
- To replace some values in `housing.df` with NAs, I am going to use the `random_NAs()` function the script of which is located in the working directory.
- The `random_NAs()` function takes in a data frame and a given number *k* and replaces *k* random elements in the data frame with NAs.

```
# Import function random_NAs.R that randomly
# replaces some elements of a data frame with NAs
source("random_NAs.R")
# Replace 2 elements of housing.df with NAs
housing.dfwNAs<-random_NAs(housing.df,2)
heatmap(1*is.na(housing.dfwNAs),
        Rowv = NA, Colv = NA)
```

Using `heatmap()` to Detect `NA`s in the Data.

- Below is the output of the above code.



Using `heatmap()` to Detect `NA`s in the Data.

- Not only the plot helps you visualize the level and amount of "missingness" but also, in the case of few missing values, it allows you to spot exactly what variables and what row ranges contain the NAs.

Outline

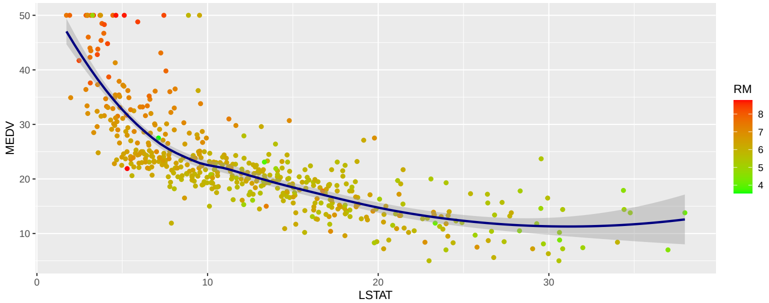
Multidimensional Visualization

- You've already seen in a couple of examples above (density and heatmaps plots) how color can enhance visual properties of a graph.
- Basic plots can be enhanced not only by color but also by size, shape, size, and by simultaneously exhibiting more than one plot on multiple panels.
- Temporal information can be added via animation.
- These enhancements allow to convey information on more than two variables at a time. They also allow to simultaneously explore the relationship between more than two variables.

Color-Enhanced Scatter Plot Using `ggplot()`

- Just to begin showing how color can enhance the informational quality of a plot reconsider a scatter plot introduced above.

```
p3<-ggplot(housing.df, aes(x = LSTAT, y = MEDV)) +
  geom_point(aes(colour = RM))+ # add color
  geom_smooth(method="loess",colour = "navy")
p3 + scale_color_gradient(low="green", high="red")
```

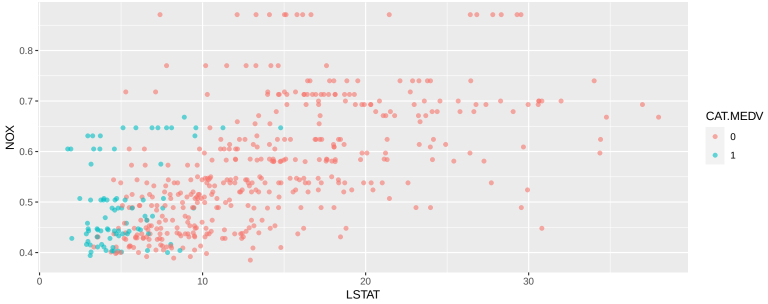


- Tracts with small percentage of low-status population AND large average number of rooms per dwelling were more likely to have higher real-estate valuation.

Color-Enhanced Scatter Plot Using ggplot()

- You could also “sprinkle” a scatter plot with color representing a categorical variable.

```
ggplot(data=housing.df, aes(y = NOX,
  x = LSTAT,color= factor(CAT.MEDV)))+
  geom_point(alpha = 0.6)+
  #change the color legend title
  labs(color='CAT.MEDV')
```



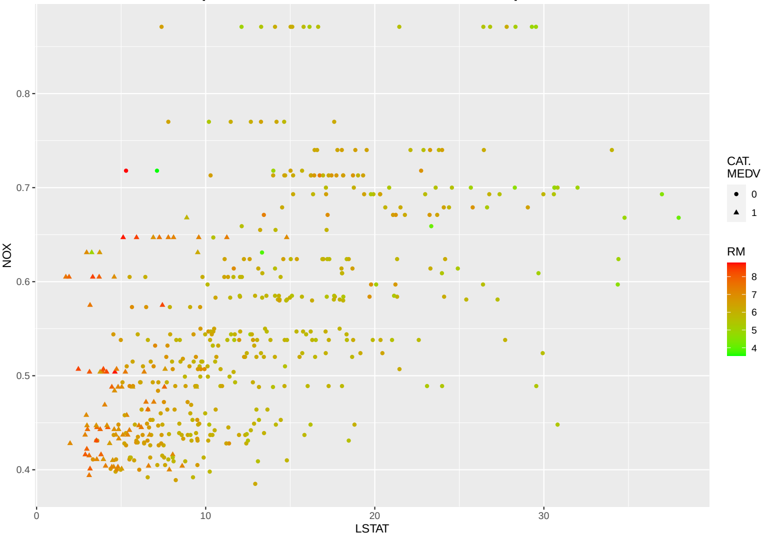
Color-and Shape-Enhanced Scatter Plot Using ggplot()

- Or you could notch it up and use marker shapes to denote CAT.MEDV and color to represent RM.

```
ggplot(data=housing.df, aes(y = NOX,
       x = LSTAT,
       shape= factor(CAT.MEDV),# specify shape
       color=RM))+ # specify color
       geom_point()+ # produce scatterplot
  #change the color and shape legend titles
labs(color='RM', shape='CAT.\nMEDV')+
  # change the spectrum of colors
scale_color_gradient(low="green", high="red")
```

Color-and Shape-Enhanced Scatter Plot Using `ggplot()`

- Below is the output of the code from the previous slide.



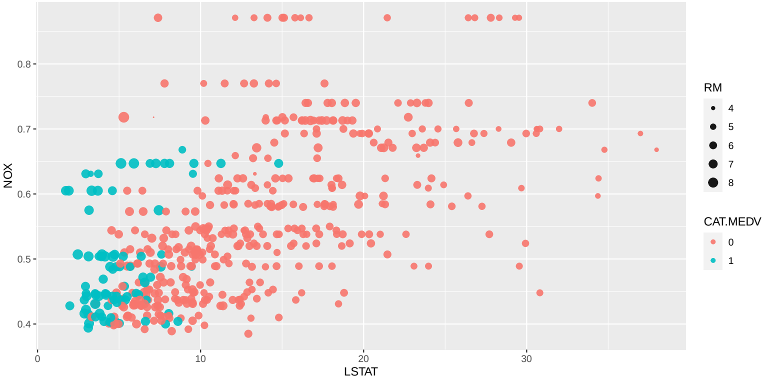
Color-and Size-Enhanced Scatter Plot Using ggplot()

- As you might have noticed, in these settings, the color of markers is not an excellent representation of average rooms, RM. It would be great if instead we could use the size of markers to represent RM.

```r
ggplot(data=housing.df, aes(y = NOX,
  x = LSTAT,color= factor(CAT.MEDV),
  size=RM))+ #specify the size
  geom_point(alpha = 0.9)+
  #scale the marker size
  scale_size(range = c(0, 3))+
  #change the color legend title
  labs(color='CAT.MEDV')
```

Color-and Size-Enhanced Scatter Plot Using `ggplot()`

- The resulting graph shown below is often referred to as "bubble plot."



- High-value neighborhoods (`CAT.MEDV=1`) are characterized by relatively low levels of nitric oxide concentration (`NOX < 0.7`) AND low percentage of underprivileged population (`LSTAT < 15`) AND greater average number of rooms.

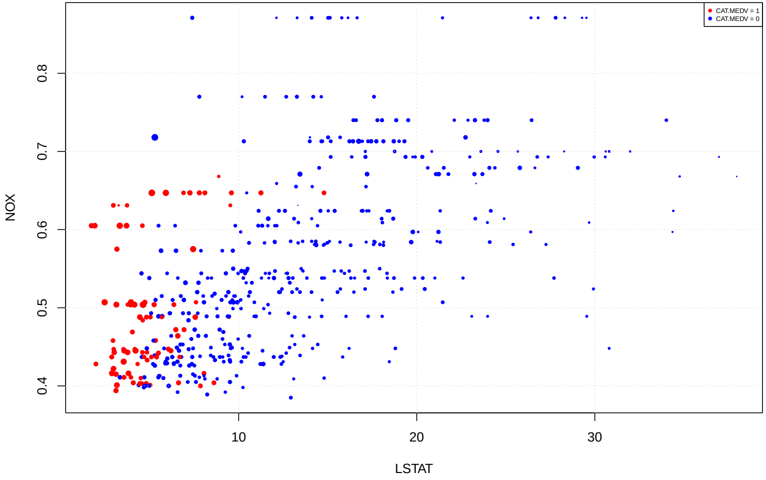
Color- and Size-Enhanced Scatter Plot Using Base R

- The same sorts of plots can be produced using base R, although, as you will probably find, as the plot becomes more and more sophisticated, coding it in base R becomes more and more cumbersome.

```r
plot(housing.df$NOX ~ housing.df$LSTAT,
 ylab = "NOX", xlab = "LSTAT", pch=19,
 col = ifelse(
       housing.df$CAT.MEDV == 1,"red","blue"),
#set marker size = RM scaled to be between 0 and 1.
 cex=(housing.df$RM-min(housing.df$RM))
     /(max(housing.df$RM)-min(housing.df$RM)))
legend("topright", # legend's location
   legend = c("CAT.MEDV = 1", "CAT.MEDV = 0"),
   col = c("red", "blue"),
   pch = 19, cex = 0.5)
grid()
```

Color- and Size-Enhanced Scatter Plot Using Base R

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



`ifelse()`

- Function `ifelse()` in the code above can come very handy when you need to convert a variable with certain values into another variable with values derived from it.

```r
grades<-c(19, 72, 68, 60, 99, 50, 37, 57,  6, 26)
ifelse(grades>60,"Pass", "Fail")
```

```
## [1] "Fail" "Pass" "Pass" "Fail" "Pass" "Fail" "Fail"
"Fail" "Fail" "Fail"
```

- `ifelse()` can be nested.

```r
passfail<-ifelse(grades>70,"Strong Pass",
        ifelse(grades>60,"Pass","Fail"))
passfail
```

```
## [1] "Fail" "Strong Pass" "Pass" "Fail" "Strong Pass"
## [6] "Fail" "Fail" "Fail" "Fail" "Fail"
```

`ifelse()`

- `ifelse()` can be used for condensing categories of categorical variables.

```r
ifelse(passfail=="Pass"|passfail=="Strong Pass",
   "Pass", "Fail")
```

```
## [1] "Fail" "Pass" "Pass" "Fail" "Pass" "Fail" "Fail"
"Fail" "Fail" "Fail"
```

- The $==$ operator is a logical operator in R that checks if each element of `passfail` is exactly equal to "Pass" and returns TRUE if yes and FALSE if not.
- The $|$ operator is another logical operator that denotes logical OR. `X|Y` means X **or** Y.
- Thus, `passfail=="Pass"|passfail=="Strong Pass"` checks if each element of `passfail` is exactly equal to "Pass" **or** to "Strong Pass" and returns TRUE if yes or FALSE if not.
- For other logical operators in R, see for example `https://www.statmethods.net/management/operators.html`

Color-Enhanced Barplot using ggplot()

- Color-enhancement may even be applied to bar charts. Suppose you wanted to observe the difference in MEDV between the neighborhoods that bound to the Charles river and the ones the are not, broken down by the index of highway accessibility. Let's prepare the data first.

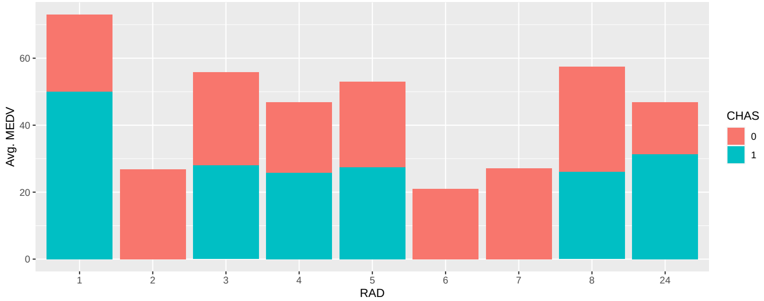
```
data.for.plot <- aggregate(
             housing.df$MEDV,
             by = list(housing.df$CHAS,
                       housing.df$RAD),
             FUN = mean, drop=FALSE)
names(data.for.plot) <-
  c("CHAS","RAD", "MeanMEDV")
```

- Statement `aggregate(housing.df$MEDV,by ...)`, calculates the mean of MEDV broken down by CHAS and RAD
- Option `drop = FALSE` forces R to include all combinations.

Color-Enhanced Barplot using ggplot()

- Now we can produce the plot using `geom_col()`

```
ggplot(data.for.plot) +
  geom_col(aes(x = factor(RAD),
               y = MeanMEDV,
               fill=factor(CHAS)))+
labs(x="RAD",y="Avg. MEDV", fill='CHAS')
```



- We can see from the plot above that there are no near-river homes in `RAD` levels 2, 6, and 7.

More than One Plot on Multiple Panels: `ggplot()`

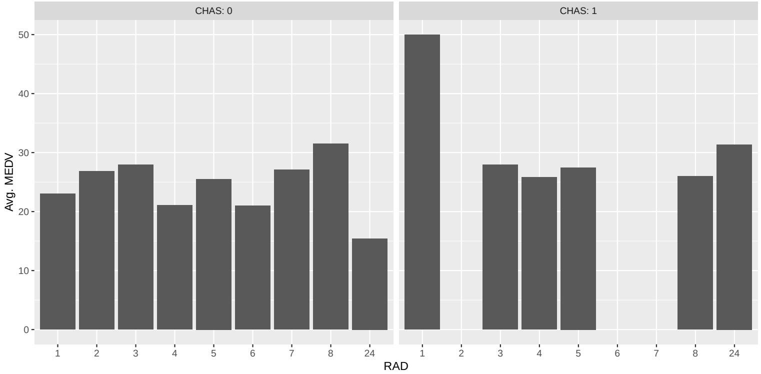
- The information on the previous plot would probably be more visually accessible if we could divide stacked up on the bars categories into two separate, side-by-side barcharts, each for a different category of `CHAS`.

```
ggplot(data.for.plot) +
geom_col(aes(x = factor(RAD),
               y = MeanMEDV))+
facet_wrap(vars(CHAS),labeller = label_both)+
labs(x="RAD",y="Avg. MEDV")
```

- The `facet_wrap()` command does the trick.
- You could also use the `facet_grid()` command of the **ggplot** package. In that case, the layout of the grid would be determined by the `cols=vars(CHAS)` `rows=vars(CHAS)` (you could specify both `cols=` and `rows=`).
- The `labeller = label_both` options forces R to print both, the variable name (CHAS) and value (0 or 1).

More than One Plot on Multiple Panels: `ggplot()`

- Below is the plot produced by the code on the previous slide.



- The average `MEDV` for different highway accessibility levels (`RAD`) behaves differently for homes near the river (right panel) compared to homes away from river (left panel). Specifically, neighborhoods near the river with `RAD = 1` had particularly high real-estate valuation.

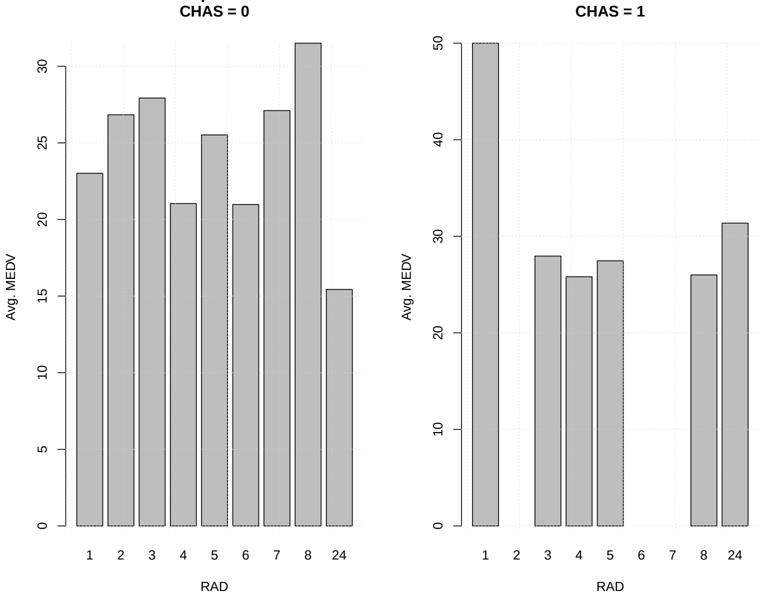
More than One Plot on Multiple Panels: Base R.

- Just for completeness, below is the code that will produce the same plot in base R.

```
par(mfcol = c(1,2))#creates a 1 by 2 graphical grid
barplot(height = data.for.plot$MeanMEDV
    [data.for.plot$CHAS == 0],
    names.arg = data.for.plot$RAD
    [data.for.plot$CHAS == 0],
    xlab = "RAD", ylab = "Avg. MEDV",
    main = "CHAS = 0")
grid()
barplot(height = data.for.plot$MeanMEDV
    [data.for.plot$CHAS == 1],
    names.arg = data.for.plot$RAD
    [data.for.plot$CHAS == 1],
    xlab = "RAD", ylab = "Avg. MEDV",
    main = "CHAS = 1")
grid()
```

More than One Plot on Multiple Panels: Base R.

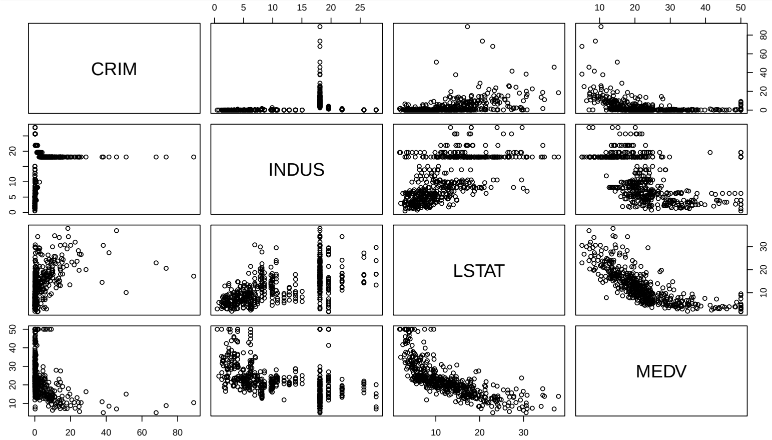
- Below is the output



Scatter-Plot Matrix Using Base R

- A scatter-plot matrix is a special plot with many scatterplots on multiple panels.
- As an example:

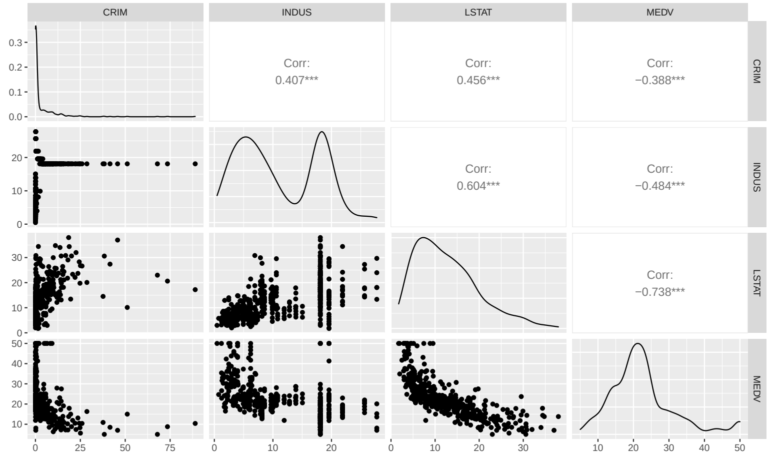
```
plot(housing.df[, c(1, 3, 12, 13)])
```



Scatter-Plot Matrix Using ggpairs()

- A nicer and more informative scatter-plot matrix can be produced using function ggpairs() in the GGally package that extends the ggplot2 package.

```
library(GGally)
ggpairs(housing.df[, c(1, 3, 12, 13)])
```



Interpreting a Scatter-Plot Matrix

- Each chart on the matrix is a scatter plot with the variable on the y-axis being the row variable and the x-axis being the column variable.
- For example, the plots in the bottom row all have `MEDV` on the y-axis and the x-axis determined by the respective column variable .
- Note that the interpretation of the information on a scatter-plot matrix is very similar to the interpretation of the correlation heatmap plot.

Interpreting a Scatter-Plot Matrix

- Also note that a scatter-plot matrix is (almost) symmetrical, with the plots on each corresponding pair across the diagonal being the rotated versions of each other.
- Along the diagonal, where just a single variable is involved, the frequency distribution for that variable is displayed. Above the diagonal are the correlation coefficients corresponding to the two variables.
- We can see different types of relationships from the different shapes (e.g., an exponential relationship between `MEDV` and `LSTAT` and a highly skewed relationship between `CRIM` and `INDUS`), which can indicate needed transformation.

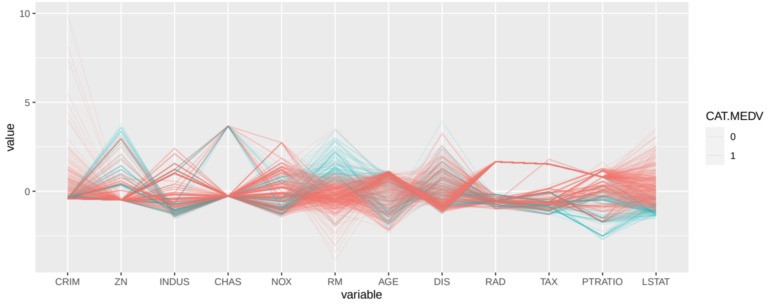
Parallel Coordinate Plot

- Another useful approach toward presenting multidimensional information on a two-dimensional plot is via specialized plots such as the *parallel coordinates* plot.
- In this plot a vertical axis is drawn for each variable. Then each observation is represented by drawing a line that connects its values on the different axes, thereby creating a "multivariate profile."

Parallel Coordinate Plot Using ggparcoord()

- As an example consider a parallel coordinates plot for the Boston housing dataset created using the `ggparcoord()` function of the **GGally** package:

```r
library(GGally)
# first convert CAT.MEDV into a factor
housing.df$CAT.MEDV<-factor(housing.df$CAT.MEDV)
ggparcoord(housing.df, groupColumn = "CAT.MEDV"
       ,columns = 1:12,
       alphaLines = 0.1) # controls opacity
```

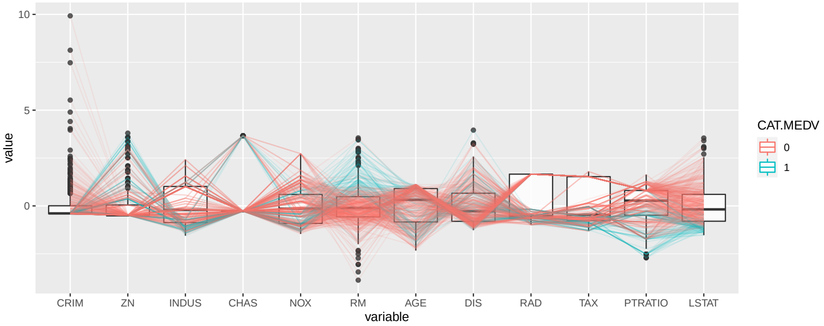


Interpreting a Parallel Coordinate Plot

- In the chart above, lines of different colors represent two values of `CAT.MEDV`.
- Usually, when observing a parallel coordinates plot one needs to pay close attention to the variables for which there is a distinct differential clustering of one specific color.
- For example, by looking at the chart above, one can learn that more expensive neighborhoods (blue lines) consistently have low `CRIM`, `PTRATIO`, `LSTAT`, and high `RM` compared to cheaper neighborhoods.
- All those low-valued neighborhoods with significantly high `CRIM` are probably outliers.
- In order to check the hypothesis, we could have added a box-plot for each variable to the parallel coordinates plot above.
- Adding box-plots can be easily done by adding the (boxplot = TRUE) option to the `ggparcoord` command above.

Box-plot Enhanced Parallel Coordinate Plot

```
# first convert CAT.MEDV into a factor
housing.df$CAT.MEDV<-factor(housing.df$CAT.MEDV)
ggparcoord(housing.df, groupColumn = "CAT.MEDV"
           ,columns = 1:12,
           alphaLines = 0.1, # controls opacity
           boxplot = TRUE) # enables boxplots
```



- Our hypothesis seems to be true.