Week 2: Data Pre-Processing and Exploration

Theory and Practice

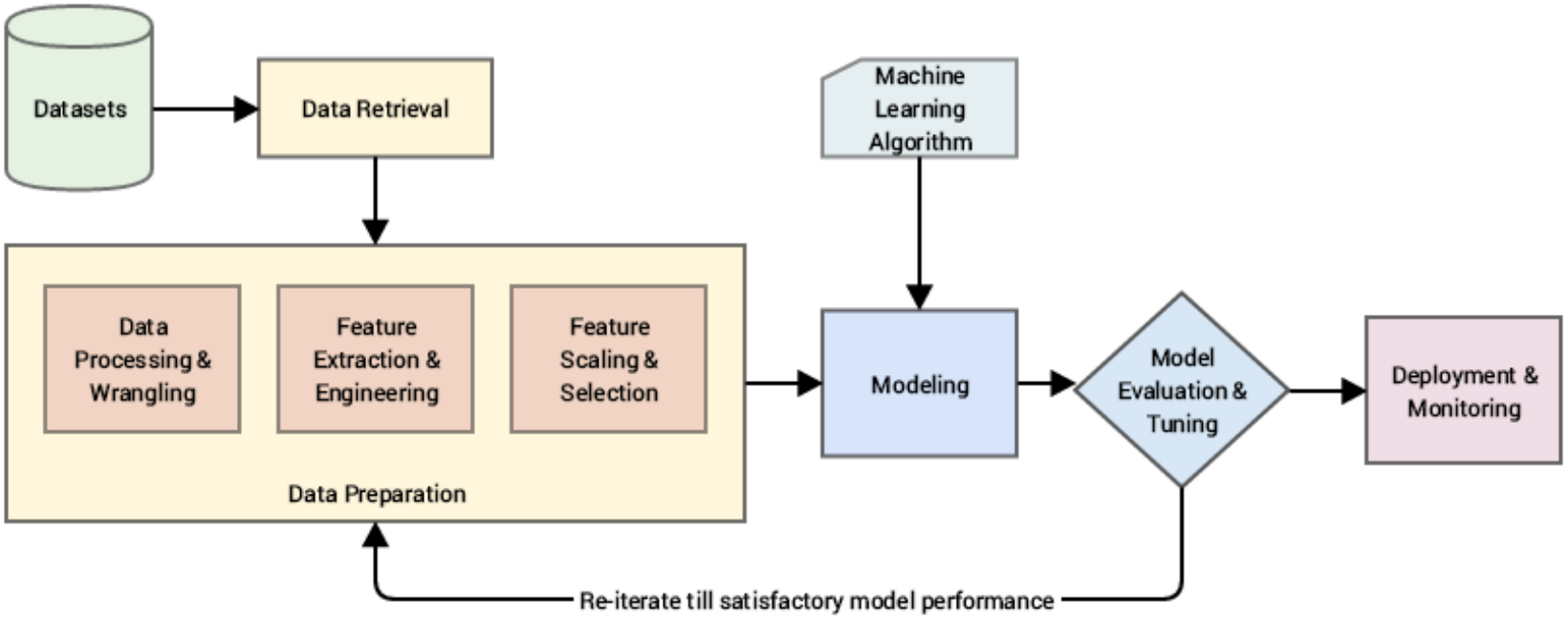
Ying Lin, Ph.D | ylin65@syr.edu

24 January, 2020

Outline of the Topics

- A standard machine learning pipeline
- Dataset types, attribute types and conversion
- Data quality issues and cleaning/transformation/preprocessing
- Exploratory data analysis and visualization
- R/Python demo

A Standard Machine Learning Pipeline



3/53

Dataset Types

- Structured data: Record dataset
  - Row: object, record, example
  - Column: feature, attribute, variable
- Unstructured data
  - Market basket data
  - Text data
  - Image data
  - Sequence data

Attribute Types

- Categorical (Qualitative)
  - Nominal: e.g. name, zip code
  - Ordinal: e.g. rating, rank
- Numerical (Quantitative)
  - Interval: e.g. IQ, temperature in fahrenheit, date
  - Ratio: e.g. height, weight
- Properties of Attributes
  - Distinctness
  - Order
  - Substraction
  - Division

Discretization

- Discrete vs. continuous
- Discretization: one type of data transformation from a continuous attribute to a discrete one
  - Equal interval
  - Equal frequency
  - K-means
  - Equal entropy
- Some data mining and machine learning algorithms don't work with continuous attributes: decision tree induction, association rule mining, naive Bayes classifier

Numerization

- Label/integer encoding: assume ordinal property of the categorical attributes
- One-hot encoding: create the same number of dummy variables/binary attributes for the categorical attributes with the same number of possible values (minus one), i.e. dummify
- Target encoding: use the target attribute (categorical and numerical) to encode. Careful with overfitting.
- Frequency encoding
- Embedding
- When needed: regression, PCA, distance-based methods (KNN, clustering analysis), nueral network, SVM

Data Quality Issues

- Data noise vs. outliers
  - Risk of overfitting
- Missing values
  - Can’t be tolerated by neural network, linear/logistic regression, etc.
- Duplicate data
- Multicollinearity
- Attributes with low to no variance
- Imbalanced dataset

Data Transformation

- Transformation based on rows (records, objects)
- Transformation based on columns (attributes, features)
- Aggregation and data roll-up
- Attribute transformation/feature engineering
- Sampling: e.g. bootstrap resampling
- Dimension reduction:
  - Principal Component Analysis (PCA): represent data in lower dimension and address multicollinearity
  - Remove attributes with low to no variance

Aggregation

- Combine multiple attributes (or objects) into a single attribute (or object)
- Benefits
  - Data reduction
  - Reduce data variability
  - Change of analysis scale
  - Generate summary/descriptive analysis

Feature Engineering and Attribute Transformation

- Discretization and numerization
  - Many ML algorithms can only work with categorical or numerical attributes
- Apply transformation function, such as $\log(x)$, e^x, x^k, $\frac{1}{x}$
  - Reduce effects of outliers
- Normalization and standardization (scale and center)
- Algorithms that require both numerization and normalization
  - Distance-based algorithms: KNN, Cluster Analysis, kernel based methods such as SVM
  - Neural network and deep learning
  - PCA

Normalization and Standardization

- Z-score transformation: convert numerical values into the distance in standard deviation units form the mean.

$$z_i = \frac{x_i - \overline{x}}{\sigma}$$

- Min-Max standardization: Standardize the range of all the numerical attributes to [0, 1]

$$f(x) = \frac{x - min}{max - min}$$

Sampling

- Create data samples with the similar properties as the original distribution
  - Sampling with vs. without replacement
  - Sample size
- Sampling Methods
  - Convenience sampling
  - Random sampling
  - Stratified sampling
  - Systematic sampling

Exploratory Data Analysis (EDA)

- EDA is an iterative cycle
  - Generate questions/hypotheses about your data
  - Search for answers by cleaning, visualising, transforming and modeling your data
  - Use what you learn to refine your questions and generate new questions
- EDA is a state of minds: curiosity and inquisitiveness
  - EDA is fundamentally a creative process that is shaped by a series of Q & A
  - Key to asking questions is to generate a large of questions followed up with data checking and exploration

Visualization

- Multivariate analysis to intuitively recognize higher-dimensional data patterns and derive insights to guide data modeling
- Increase expressiveness through integrating more and various type of attributes
- Common visualization techniques and their variants
  - Basic types: histogram, box plot, bar chart, scatter plot, bubble chart, pie/donut pie chart, area graph, density plot, heatmap, etc.
  - Special types: word cloud, tree diagram, network diagram, radar chart, sankey diagram, etc.
  - Combination: faceted plot, dashboarding

Demo Dataset: Car MPG Dataset

```
library(ggplot2)
data(mpg, package = "ggplot2")
dim(mpg)
```

```
## [1] 234  11
```

```
str(mpg)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    234 obs. of  11 variables:
##  $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
##  $ model       : chr  "a4" "a4" "a4" "a4" ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr  "f" "f" "f" "f" ...
##  $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr  "p" "p" "p" "p" ...
##  $ class       : chr  "compact" "compact" "compact" "compact" ...
```

A Preview of Data

```
knitr::kable(mpg)
```

manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

Meta Data of "mpg" Data frame (1)

Variable	Description
manufacturer	carmarker
model	model name
displ	engine displacement (in litres)
year	year of manufacture
cyl	number of cylinders
trans	type of transmission
drv	f = front-wheel drive, r = rear wheel drive, 4 = 4wd
cty	city miles per gallon
hwy	highway miles per gallon
fl	fuel type
class	"type" of car

Change Attribute Type

```
char_var <- sapply(mpg, is.character)
mpg[, char_var] <- lapply(mpg[, char_var], as.factor)
str(mpg)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    234 obs. of  11 variables:
##  $ manufacturer: Factor w/ 15 levels "audi","chevrolet",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ model       : Factor w/ 38 levels "4runner 4wd",..: 2 2 2 2 2 2 2 3 3 3 ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : Factor w/ 10 levels "auto(av)","auto(l3)",..: 4 9 10 1 4 9 1 9 4 10 ...
##  $ drv         : Factor w/ 3 levels "4","f","r": 2 2 2 2 2 2 2 1 1 1 ...
##  $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : Factor w/ 5 levels "c","d","e","p",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ class       : Factor w/ 7 levels "2seater","compact",..: 2 2 2 2 2 2 2 2 2 2 ...
```

Illustration of EDA

Descriptive data analysis to identify interesting data patterns; e.g. compare fuel economy of different car classes

```
library(dplyr)
mpg %>% group_by(class) %>%
  summarise(ct=n(), cty=round(mean(cty, na.rm=T), 1),
            hwy=round(mean(hwy, na.rm=T), 1),
            displ=round(mean(displ, na.rm=T), 1)) %>%
  arrange(desc(cty))
```

```
## # A tibble: 7 x 5
##   class         ct   cty   hwy displ
##   <fct>      <int> <dbl> <dbl> <dbl>
## 1 subcompact    35  20.4  28.1   2.7
## 2 compact       47  20.1  28.3   2.3
## 3 midsize       41  18.8  27.3   2.9
## 4 minivan       11  15.8  22.4   3.4
## 5 2seater        5  15.4  24.8   6.2
## 6 suv           62  13.5  18.1   4.5
## 7 pickup        33  13    16.9   4.4
```

Discretization: Conversion from Numerical to Categorical Attributes

```
library(dplyr)
library(arules)
mpg$displ_grp <- arules::discretize(mpg$displ, method = "frequency",
                           breaks = 3, labels = c("low",
                                                  "medium", "high"))
mpg %>%
  group_by(displ_grp) %>%
  summarise(avg_displ = mean(displ), count = n(), min = min(displ), max = max(displ))
```

```
## # A tibble: 3 x 5
##   displ_grp avg_displ count   min   max
##   <fct>         <dbl> <int> <dbl> <dbl>
## 1 low            2.02    62   1.6   2.4
## 2 medium         3.06    86   2.5   3.9
## 3 high           4.93    86   4     7
```

Convert from Categorical to Numerical Attributes

- function from package creates a full set of dummy variables, i.e. less than full rank parameterization

```
library()
unique(mpg$drv)
```

```
## [1] f 4 r
## Levels: 4 f r
```

```
dmy <- caret::dummyVars(" ~ drv", data = mpg, fullRank = T)
head(data.frame(predict(dmy, newdata = mpg)), 5)
```

```
##   drv.f drv.r
## 1     1     0
## 2     1     0
## 3     1     0
## 4     1     0
## 5     1     0
```

Dimension Reduction and Feature Engineering: PCA

```
trans = caret::preProcess(mpg[, sapply(mpg, is.numeric)], method=c("center", "scale", "pca"),
                  pcaComp = 5)
PC = predict(trans, mpg[, sapply(mpg, is.numeric)])
head(PC, 5)
```

```
##          PC1        PC2        PC3         PC4         PC5
## 1 -1.8858984  0.9723721 -0.4244892 -0.18149906 -0.52192770
## 2 -2.2424056  0.9025047 -0.1179838 -0.07438406 -0.01253529
## 3 -2.0917184 -1.0910392 -0.3338737 -0.07819127 -0.37494694
## 4 -2.1275729 -1.0902013 -0.3212256 -0.01996586 -0.09463483
## 5 -0.3953825  0.9725482  0.1057113 -0.48523023 -0.38418629
```

```
str(PC)
```

```
## 'data.frame':    234 obs. of  5 variables:
##  $ PC1: num  -1.886 -2.242 -2.092 -2.128 -0.395 ...
##  $ PC2: num  0.972 0.903 -1.091 -1.09 0.973 ...
##  $ PC3: num  -0.424 -0.118 -0.334 -0.321 0.106 ...
##  $ PC4: num  -0.1815 -0.0744 -0.0782 -0.02 -0.4852 ...
##  $ PC5: num  -0.5219 -0.0125 -0.3749 -0.0946 -0.3842 ...
```

Remove Features with No or Low Variances

```
nzv <- caret::nearZeroVar(mpg[mpg$year == 1999, ], saveMetrics = T)
colnames(mpg)[nzv$nzv]
```

```
## [1] "year"
```

```
nzv[nzv$nzv, ]
```

```
##      freqRatio percentUnique zeroVar  nzv
## year         0     0.8547009    TRUE TRUE
```

Data Normalization and Standardization

```
mpg_cp <- mpg
mpg_cp$displ_scale <- scale(mpg_cp$displ, center = TRUE, scale = TRUE)
summary(mpg_cp$displ)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.600   2.400   3.300   3.472   4.600   7.000
```

```
sd(mpg_cp$displ)
```

```
## [1] 1.291959
```

```
summary(mpg_cp$displ_scale)
```

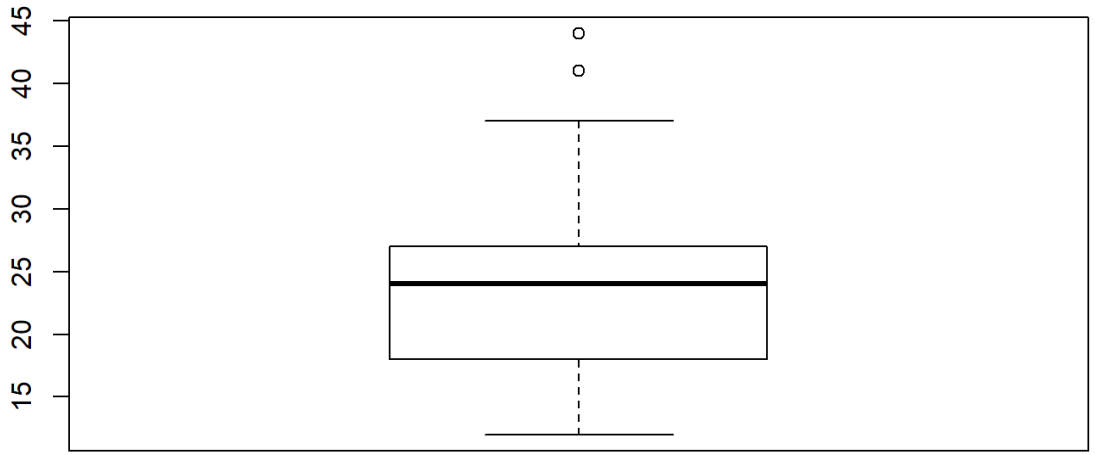
```
##        V1
##  Min.   :-1.4488
##  1st Qu.:-0.8296
##  Median :-0.1330
##  Mean   : 0.0000
##  3rd Qu.: 0.8733
##  Max.   : 2.7309
```

```
sd(mpg_cp$displ_scale)
```

Deal with Data Outliers and Noises (1)

Before removing outliers

```
boxplot(mpg$hwy)
```

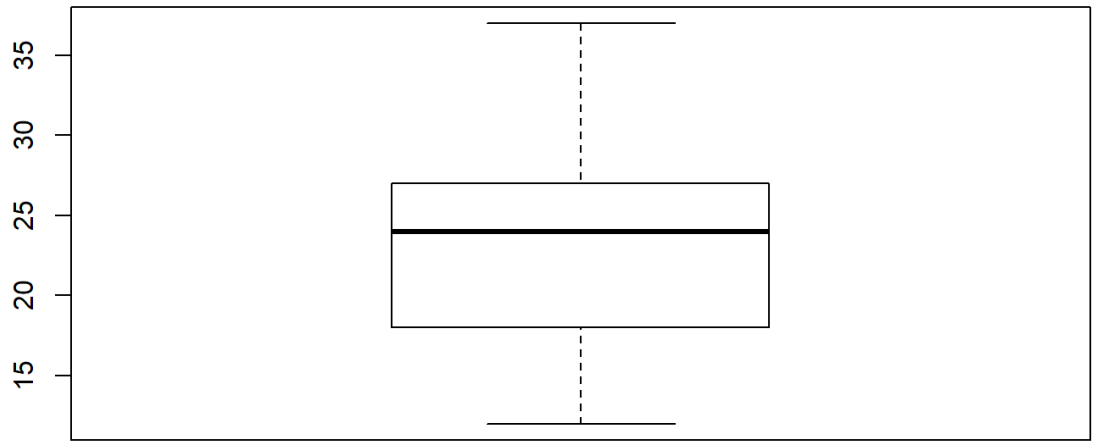



```
mpg$hwy[mpg$hwy %in% boxplot.stats(mpg$hwy)$out] <- median(mpg$hwy, na.rm = T)
boxplot(mpg$hwy)
```

Deal with Data Outliers and Noises (2)

After removing outliers

```
mpg$hwy[mpg$hwy %in% boxplot.stats(mpg$hwy)$out] <- median(mpg$hwy, na.rm = T)
boxplot(mpg$hwy)
```



Remove Duplicate Data Record

```
nrow(mpg)
```

```
## [1] 234
```

```
nrow(mpg[!duplicated(mpg), ])
```

```
## [1] 225
```

```
mpg %>%
  distinct(.keep_all = T) %>%
  nrow()
```

```
## [1] 225
```

Output those Duplicated Records

```
mpg[duplicated(mpg)|duplicated(mpg, fromLast = T), ]
```

```
## # A tibble: 18 x 12
##    manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
##    <fct>        <fct> <dbl> <int> <int> <fct> <fct> <int> <dbl> <fct> <fct>
##  1 chevrolet    c150~   5.3  2008     8 auto~ r        14    20 r     suv
##  2 chevrolet    c150~   5.3  2008     8 auto~ r        14    20 r     suv
##  3 dodge        cara~   3.3  1999     6 auto~ f        16    22 r     mini~
##  4 dodge        cara~   3.3  1999     6 auto~ f        16    22 r     mini~
##  5 dodge        cara~   3.3  2008     6 auto~ f        17    24 r     mini~
##  6 dodge        cara~   3.3  2008     6 auto~ f        17    24 r     mini~
##  7 dodge        dako~   4.7  2008     8 auto~ 4        14    19 r     pick~
##  8 dodge        dako~   4.7  2008     8 auto~ 4        14    19 r     pick~
##  9 dodge        dura~   4.7  2008     8 auto~ 4        13    17 r     suv
## 10 dodge        dura~   4.7  2008     8 auto~ 4        13    17 r     suv
## 11 dodge        ram ~   4.7  2008     8 manu~ 4        12    16 r     pick~
## 12 dodge        ram ~   4.7  2008     8 auto~ 4        13    17 r     pick~
## 13 dodge        ram ~   4.7  2008     8 auto~ 4        13    17 r     pick~
## 14 dodge        ram ~   4.7  2008     8 manu~ 4        12    16 r     pick~
## 15 ford         expl~   4    1999     6 auto~ 4        14    17 r     suv
## 16 ford         expl~   4    1999     6 auto~ 4        14    17 r     suv
## 17 honda        civic   1.6  1999     4 auto~ f        24    32 r     subc~
## 18 honda        civic   1.6  1999     4 auto~ f        24    32 r     subc~
## # ... with 1 more variable: displ_grp <fct>
```

Aggregation

Goal: output the five most fuel economy models

```
mpg %>%
  group_by(manufacturer) %>%
  summarise(avg_cty = mean(cty)) %>%
  arrange(desc(avg_cty)) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   manufacturer avg_cty
##   <fct>          <dbl>
## 1 honda           24.4
## 2 volkswagen      20.9
## 3 subaru          19.3
## 4 hyundai         18.6
## 5 toyota          18.5
```

Random Data Sampling without Replacement

```
mpg_sample_index <- sample(1:nrow(mpg),
                           size = nrow(mpg) * 0.3,
                           replace = F)
prop.table(table(mpg[mpg_sample_index, "drv"]))
```

```
##
##         4         f         r
## 0.4285714 0.4571429 0.1142857
```

```
mpg[mpg_sample_index, ]
```

```
## # A tibble: 70 x 12
##    manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
##    <fct>        <fct> <dbl> <int> <int> <fct> <fct> <int> <dbl> <fct> <fct>
##  1 dodge        dako~   4.7  2008     8 auto~ 4        14    19 r     pick~
##  2 nissan       maxi~   3    1999     6 auto~ f        18    26 r     mids~
##  3 volkswagen   gti     2    1999     4 auto~ f        19    26 r     comp~
##  4 volkswagen   pass~   2    2008     4 manu~ f        21    29 p     mids~
##  5 chevrolet    corv~   7    2008     8 manu~ r        15    24 p     2sea~
##  6 chevrolet    corv~   6.2  2008     8 manu~ r        16    26 p     2sea~
##  7 toyota       camry   2.2  1999     4 manu~ f        21    29 r     mids~
##  8 audi         a4 q~   3.1  2008     6 manu~ 4        15    25 p     comp~
##  9 dodge        dako~   5.2  1999     8 auto~ 4        11    15 r     pick~
```

Stratified Data Sampling

```
prop.table(table(mpg$drv))
```

```
##
##         4         f         r
## 0.4401709 0.4529915 0.1068376
```

```
sample_index <- caret::createDataPartition(mpg$drv, p = 0.3,
                                           list = F)
prop.table(table(mpg[sample_index, "drv"]))
```

```
##
##         4         f         r
## 0.4366197 0.4507042 0.1126761
```

Missing Value Manipulations (1)

Check if missing values exist in the dataset. If not, then randomly assign some missing values and check how many missing values exist for which variables.

```
sum(!complete.cases(mpg))
```

```
## [1] 0
```

```
mpg$hwy[sample(1:length(mpg$hwy), size = 5, replace = F)] <- NA
sum(!complete.cases(mpg))
```

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

```
sapply(mpg, function(x) sum(is.na(x)))
```

```
## manufacturer        model        displ         year          cyl
##            0            0            0            0            0
##        trans          drv          cty          hwy           fl
##            0            0            0            5            0
##        class    displ_grp
##            0            0
```

Missing Value Manipulations (2)

Check number of missing values, records with missing values, and attributes with missing values

```
sum(is.na(mpg))
```

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

```
which(apply(mpg, 1, function(x) sum(is.na(x))) > 0)
```

```
## [1]  37 116 135 150 155
```

```
which(apply(mpg, 2, function(x) sum(is.na(x))) > 0)
```

```
## hwy
##   9
```

```
sum(!complete.cases(mpg))
```

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

Missing Value Manipulations (3)

Simple imputation method for missing value: replace with mean of the attribute

```
mpg2 <- mpg
mpg2$hwy[is.na(mpg2$hwy)] <- mean(mpg2$hwy, na.rm = T)
sum(!complete.cases(mpg2))
```

```
## [1] 0
```

More advanced missing value imputation algorithms: kNN imputation

```
preprocess <- caret::preProcess(mpg, method = c("knnImpute", "center", "scale"))
mpg3 <- predict(preprocess, mpg)
sum(!complete.cases(mpg3))
```

```
## [1] 0
```

Examine the Missing Value Imputation Values

```
row <- which(apply(mpg, MARGIN = 1, function(x) sum(is.na(x)) > 0))
col <- which(apply(mpg, MARGIN = 2, function(x) sum(is.na(x)) > 0))
print(as.data.frame(unique(mpg2[row, col])))
```

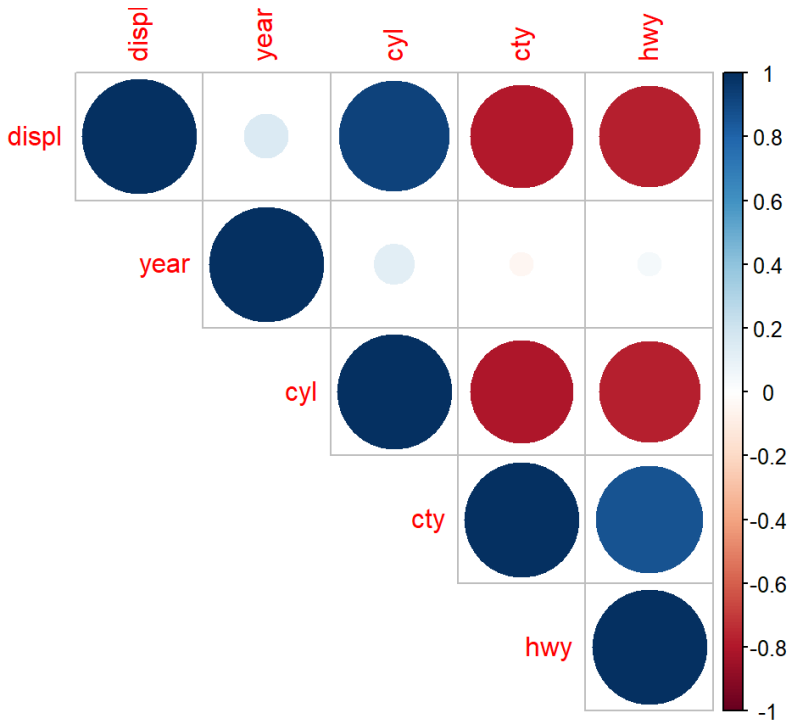
```
##        hwy
## 1 23.17904
```

```
mpg3[row, c("manufacturer", "model", "class", "displ", "displ_grp", "cyl", "cty", "hwy")]
```

```
## # A tibble: 5 x 8
##   manufacturer model      class      displ displ_grp     cyl     cty    hwy
##   <fct>        <fct>      <fct>      <dbl> <fct>       <dbl>   <dbl>  <dbl>
## 1 chevrolet    malibu     midsize   0.0992 medium     0.0689  0.0331  0.364
## 2 hyundai      tiburon    subcomp~ -1.14   low       -1.17    0.503   0.616
## 3 lincoln      navigator~ suv       1.49   high       1.31   -1.38   -1.26
## 4 nissan       maxima     midsize   0.0218 medium     0.0689  0.503   0.796
## 5 pontiac      grand prix midsize  -0.288  medium     0.0689  0.268   0.508
```

Detect Attributes' Correlation: Correlogram

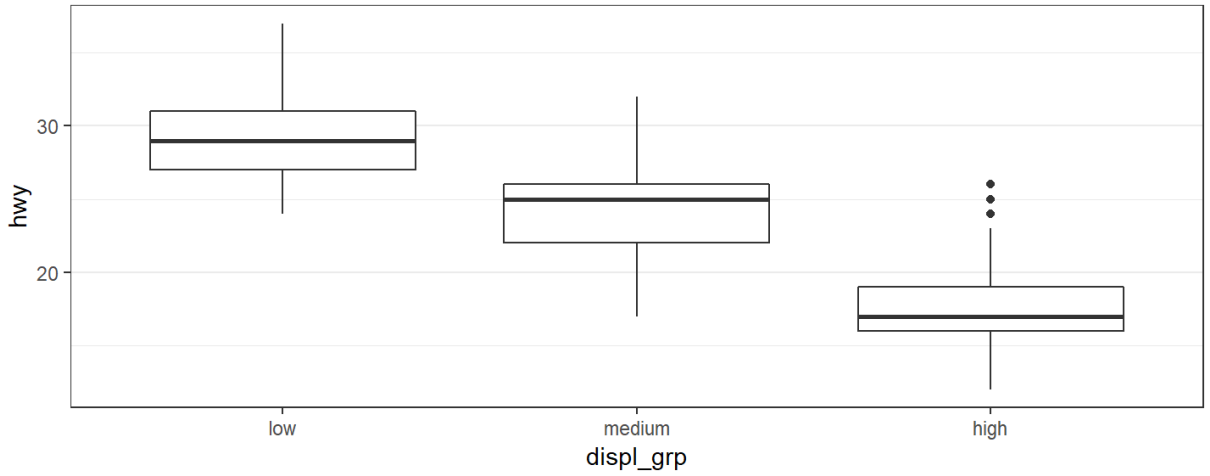
```
library(corrplot)
cor_matrix <- cor(mpg[complete.cases(mpg), sapply(mpg, is.numeric)], method = "pearson")
corrplot(cor_matrix, type = "upper")
```



Visualization: Box Plot

Apply to two attributes: one categorical and one numeric attribute.

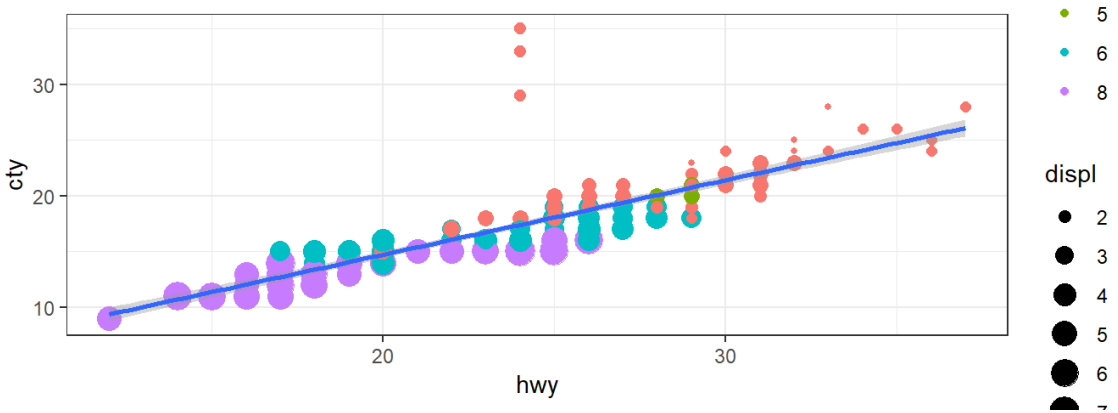
```
theme_set(theme_bw())

ggplot(mpg, aes(x = displ_grp, y = hwy)) +
  geom_boxplot() +
  theme(panel.grid.major.x = element_blank())
```



Visualization: Scatter Plot (and its Variants)

Apply to two numeric attributes.

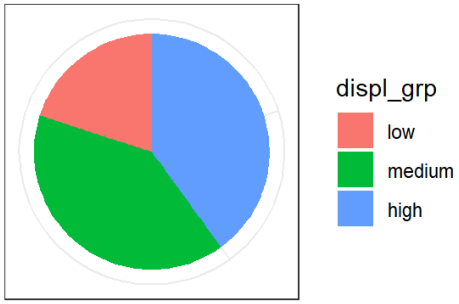
```
ggplot(mpg, aes(x = hwy, y = cty)) +
  geom_point(aes(color = as.factor(cyl), size = displ)) +
  geom_smooth(method = "lm")
```



Visualization: Pie Chart

Apply to one categorical attribute (optional: numeric count).

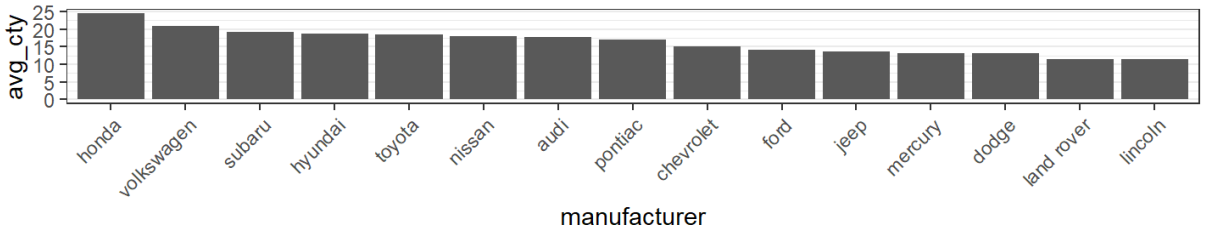
```
library(scales)
mpg %>%  group_by(displ_grp) %>%
  summarise(pct = percent(n()/nrow(mpg))) %>%
  ggplot(aes(x = factor(1), y = pct, fill = displ_grp)) +
  geom_bar(stat = "identity") +
  coord_polar(theta = "y") +
  theme(axis.ticks = element_blank(),
        axis.text = element_blank(),
        axis.title = element_blank())
```



Visualization: Ordered Bar Plot

Apply to two attributes: one categorical and one numeric attribute.

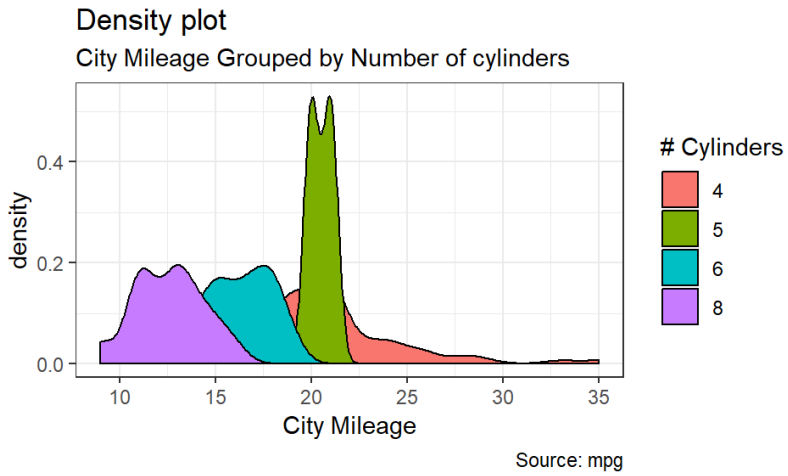
```
mpg_by_maker <- mpg %>%
  group_by(manufacturer) %>%
  summarise(avg_cty = mean(cty, na.rm = T)) %>%
  arrange(desc(avg_cty))

mpg_by_maker$manufacturer <- factor(mpg_by_maker$manufacturer,
                                    levels = mpg_by_maker$manufacturer)

ggplot(mpg_by_maker, aes(x = manufacturer, y = avg_cty)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid.major.x = element_blank())
```



Visualization: Density Plot

Apply to two attributes: one categorical and one numeric attribute.

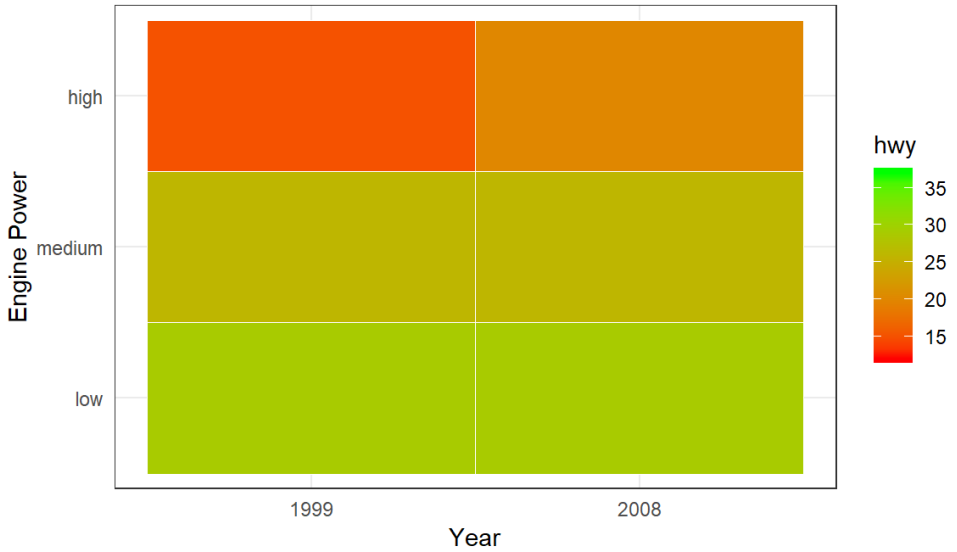
```
ggplot(mpg, aes(cty)) +
  geom_density(aes(fill=factor(cyl))) +
    labs(title="Density plot",
         subtitle="City Mileage Grouped by Number of cylinders",
         caption="Source: mpg",
         x="City Mileage",
         fill="# Cylinders")
```



Visualization: Heatmap

Apply to two attributes: one categorical and one numeric attribute.

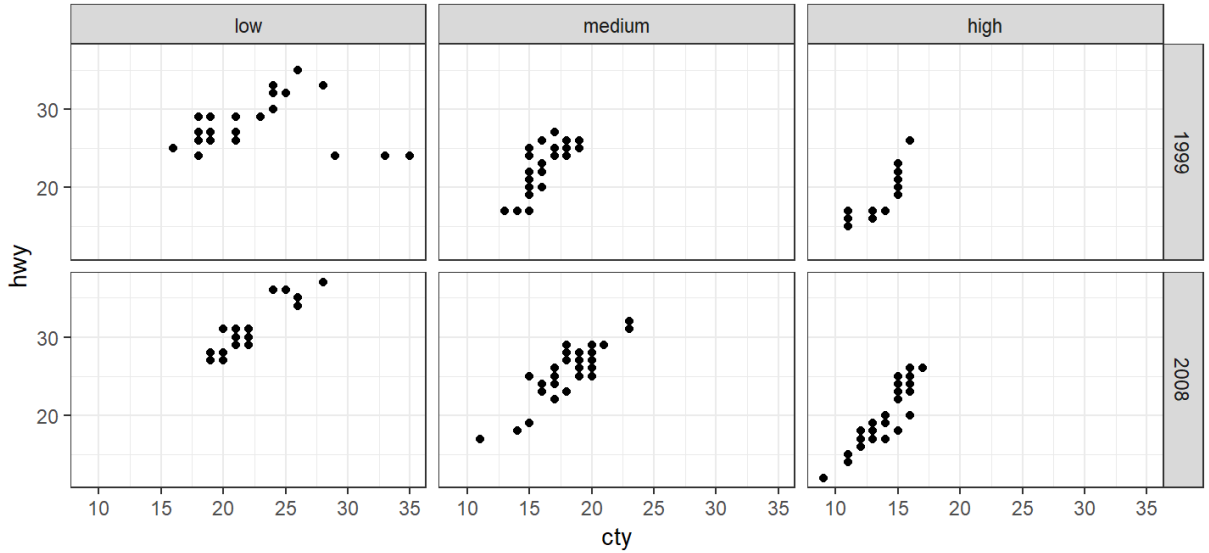
```
ggplot(mpg, aes(x = as.factor(year), y = displ_grp)) +
  geom_tile(aes(fill = hwy), color = "white") +
  scale_fill_gradient(low = "red", high = "green") +
  theme(axis.ticks = element_blank()) +
  labs(x = "Year", y = "Engine Power")
```



Visualization: Faceting

Combine multiple plots by adding two additional categorical attributes.

```
ggplot(mpg, aes(cty, hwy)) +
  geom_point() +
  facet_grid(year ~ displ_grp)
```



Setup Python in Rmarkdown

```
library(reticulate)
use_python("C:/Users/ylin65/AppData/Local/Programs/Python/Python37/python.exe", required = T)
py_discover_config()
```

```
## python:         C:/Users/ylin65/AppData/Local/Programs/Python/Python37/python.exe
## libpython:      C:/Users/ylin65/AppData/Local/Programs/Python/Python37/python37.dll
## pythonhome:     C:\Users\ylin65\AppData\Local\Programs\Python\Python37
## version:        3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)]
## Architecture:   64bit
## numpy:          C:\Users\ylin65\AppData\Local\Programs\Python\Python37\lib\site-packages\numpy
## numpy_version:  1.17.1
##
## NOTE: Python version was forced by use_python function
```

Data exploration

```
import pandas as pd
import numpy as np
mpg = pd.DataFrame(r.mpg)
mpg.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 234 entries, 0 to 233
## Data columns (total 12 columns):
## manufacturer    234 non-null category
## model           234 non-null category
## displ           234 non-null float64
## year            234 non-null int32
## cyl             234 non-null int32
## trans           234 non-null category
## drv             234 non-null category
## cty             234 non-null int32
## hwy             229 non-null float64
## fl              234 non-null category
## class           234 non-null category
## displ_grp       234 non-null category
## dtypes: category(7), float64(2), int32(3)
## memory usage: 11.6 KB
```

```
mpg.describe()
```

Data Discretization

```
mpg['displ_grp_py'] = pd.qcut(mpg.displ, 3,
                    labels = ["low", "medium", "high"])
mpg.groupby(['displ_grp_py']).agg({'displ': ['count', 'mean']})
```

```
##              displ
##              count      mean
## displ_grp_py
## low             82  2.137805
## medium          81  3.375309
## high            71  5.122535
```

Data Normalization and Standardization

```
from sklearn import preprocessing
mpg['displ_scale'] = preprocessing.scale(mpg['displ'])
mpg['displ_scale'].describe().round(2)
```

```
## count    234.00
## mean      -0.00
## std        1.00
## min       -1.45
## 25%       -0.83
## 50%       -0.13
## 75%        0.88
## max        2.74
## Name: displ_scale, dtype: float64
```

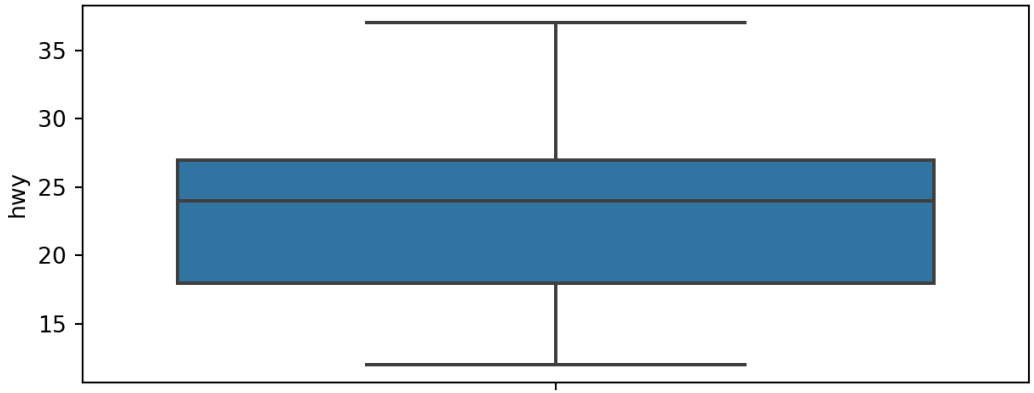
One Hot Encoding and Label Encoding

```
mpg_drv = pd.concat([mpg, pd.get_dummies(mpg["drv"], prefix="drv")], axis=1)
mpg_drv.sample(n=mpg_drv.shape[0]).head(10)
```

```
##     manufacturer                model  displ  ...  drv_4  drv_f drv_r
## 208   volkswagen                  gti    2.0  ...      0      1     0
## 176       toyota          4runner 4wd    3.4  ...      1      0     0
## 6           audi                   a4    3.1  ...      0      1     0
## 183       toyota                camry    3.0  ...      0      1     0
## 165       subaru          impreza awd    2.2  ...      1      0     0
## 81          ford         explorer 4wd    4.6  ...      1      0     0
## 120      hyundai              tiburon    2.7  ...      0      1     0
## 122         jeep  grand cherokee 4wd    3.0  ...      1      0     0
## 188       toyota         camry solara    2.4  ...      0      1     0
## 196       toyota              corolla    1.8  ...      0      1     0
##
## [10 rows x 17 columns]
```

Data Outliers and Noises

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(y="hwy", data=mpg)
plt.show()
```



Remove Duplicate Data Records

```
mpg2 = mpg.drop_duplicates(keep="first")
diff = mpg.shape[0] - mpg2.shape[0]
print(f"There are {diff} duplicate records")
```

```
## There are 9 duplicate records
```

```
mpg[mpg.duplicated()]
```

```
##     manufacturer                model  ...  displ_grp_py  displ_scale
## 20     chevrolet   c1500 suburban 2wd  ...          high     1.418098
## 40         dodge          caravan 2wd  ...        medium    -0.133257
## 42         dodge          caravan 2wd  ...        medium    -0.133257
## 53         dodge    dakota pickup 4wd  ...          high     0.952691
## 60         dodge          durango 4wd  ...          high     0.952691
## 67         dodge  ram 1500 pickup 4wd  ...          high     0.952691
## 68         dodge  ram 1500 pickup 4wd  ...          high     0.952691
## 79          ford         explorer 4wd  ...        medium     0.409717
## 103        honda                civic  ...           low    -1.451909
##
## [9 rows x 14 columns]
```

Aggregation

```
mpg2.groupby("manufacturer")['cty'].mean().sort_values(ascending=False)
```

```
## manufacturer
## honda         24.500000
## volkswagen    20.925926
## subaru        19.285714
## hyundai       18.642857
## toyota        18.529412
## nissan        18.076923
## audi          17.611111
## pontiac       17.000000
## chevrolet     15.055556
## ford          14.000000
## jeep          13.500000
## mercury       13.250000
## dodge         12.935484
## land rover    11.500000
## lincoln       11.333333
## Name: cty, dtype: float64
```

Scatter Plot

```
sns.scatterplot(x="hwy", y="cty", hue="cyl", size="displ", data=mpg)
```