Week 5

R Packages

The packages you will need to install for the week are **splitstackshape**, **lubridate**, **dplyr**, **rgl**.

Learning Goal for the Week

In the past four weeks, we have examined unsupervised (cluster analysis; frequent pattern analysis) and supervised (linear regressions; decision trees) learning algorithms. In Week 5, we will take a "step back" and discuss the **bane of our existence as data scientists/analysts**: data preprocessing.

Let's Talk Data Quality

According to Han, Kamber, and Pei (2012), data quality consists of six elements:

- Accuracy
- Completeness
- Consistency
- Timeliness
- Believability
- Interpretability

Achieving a high level of data quality is the reason for data preprocessing.

Link to a section from Han, Kamber, and Pei

Three Tasks of Data Preprocessing

- Data Cleaning: fill in missing values; smoothing noisy data; identifying and removing outliers; and resolving inconsistencies.
- Data Transformation: normalization; discretization; and concept hierarchy generation.
- Data Reduction: dimensionality and numerosity reduction.

Dataset

We are going to play with a dataset containing 5,000+ movies from IMDB. The dataset was scrapped, cleaned, and posted by Chuan Sun on Kaggle.

```
setwd("C:/Users/corylowe/OneDrive/Code/R Practice Code/Applied Data Mining_Portfolio/Week 5")
movie<-read.csv("movie_metadata.csv")
```

A Quick Data Cleaning Exercise

```
str(movie)
```

```
## 'data.frame':    5043 obs. of  28 variables:
##  $ color                    : Factor w/ 3 levels ""," Black and White",..: 3 3 3 3 1 3 3 3 3 3 ...
##  $ director_name            : Factor w/ 2399 levels "","A. Raven Cruz",..: 929 801 2027 380 606 109 2
##  $ num_critic_for_reviews   : int  723 302 602 813 NA 462 392 324 635 375 ...
##  $ duration                 : int  178 169 148 164 NA 132 156 100 141 153 ...
##  $ director_facebook_likes  : int  0 563 0 22000 131 475 0 15 0 282 ...
##  $ actor_3_facebook_likes   : int  855 1000 161 23000 NA 530 4000 284 19000 10000 ...
##  $ actor_2_name             : Factor w/ 3033 levels "","50 Cent","A. Michael Baldwin",..: 1408 2218 2
##  $ actor_1_facebook_likes   : int  1000 40000 11000 27000 131 640 24000 799 26000 25000 ...
##  $ gross                    : int  760505847 309404152 200074175 448130642 NA 73058679 336530303 2008
##  $ genres                   : Factor w/ 914 levels "Action","Action|Adventure",..: 107 101 128 288 75
##  $ actor_1_name             : Factor w/ 2098 levels "","50 Cent","A.J. Buckley",..: 305 983 355 1968
##  $ movie_title              : Factor w/ 4917 levels "#HorrorÂ ","[Rec] 2Â ",..: 398 2731 3279 3707 33
##  $ num_voted_users          : int  886204 471220 275868 1144337 8 212204 383056 294810 462669 321795
##  $ cast_total_facebook_likes: int  4834 48350 11700 106759 143 1873 46055 2036 92000 58753 ...
##  $ actor_3_name             : Factor w/ 3522 levels "","50 Cent","A.J. Buckley",..: 3442 1395 3134 17
##  $ facenumber_in_poster     : int  0 0 1 0 0 1 0 1 4 3 ...
##  $ plot_keywords            : Factor w/ 4761 levels "","10 year old|dog|florida|girl|supermarket",..
##  $ movie_imdb_link          : Factor w/ 4919 levels "http://www.imdb.com/title/tt0006864/?ref_=fn_tt_
##  $ num_user_for_reviews     : int  3054 1238 994 2701 NA 738 1902 387 1117 973 ...
##  $ language                 : Factor w/ 48 levels "","Aboriginal",..: 13 13 13 13 1 13 13 13 13 13 .
##  $ country                  : Factor w/ 66 levels "","Afghanistan",..: 65 65 63 65 1 65 65 65 65 63
##  $ content_rating           : Factor w/ 19 levels "","Approved",..: 10 10 10 10 1 10 10 9 10 9 ...
##  $ budget                   : num  2.37e+08 3.00e+08 2.45e+08 2.50e+08 NA ...
##  $ title_year               : int  2009 2007 2015 2012 NA 2012 2007 2010 2015 2009 ...
##  $ actor_2_facebook_likes   : int  936 5000 393 23000 12 632 11000 553 21000 11000 ...
##  $ imdb_score               : num  7.9 7.1 6.8 8.5 7.1 6.6 6.2 7.8 7.5 7.5 ...
##  $ aspect_ratio             : num  1.78 2.35 2.35 2.35 NA 2.35 2.35 1.85 2.35 2.35 ...
##  $ movie_facebook_likes     : int  33000 0 85000 164000 0 24000 0 29000 118000 10000 ...
```

```
summary(movie)
```

```
##              color                director_name  num_critic_for_reviews
##                 :  19                    : 104   Min.   :  1.0
##   Black and White: 209   Steven Spielberg:  26   1st Qu.: 50.0
##  Color           :4815   Woody Allen     :  22   Median :110.0
##                          Clint Eastwood  :  20   Mean   :140.2
##                          Martin Scorsese :  20   3rd Qu.:195.0
##                          Ridley Scott    :  17   Max.   :813.0
##                          (Other)         :4834   NA's   :50
##     duration     director_facebook_likes actor_3_facebook_likes
##  Min.   :  7.0   Min.   :    0.0         Min.   :    0.0
##  1st Qu.: 93.0   1st Qu.:    7.0         1st Qu.:  133.0
##  Median :103.0   Median :   49.0         Median :  371.5
##  Mean   :107.2   Mean   :  686.5         Mean   :  645.0
##  3rd Qu.:118.0   3rd Qu.:  194.5         3rd Qu.:  636.0
##  Max.   :511.0   Max.   :23000.0         Max.   :23000.0
##  NA's   :15      NA's   :104             NA's   :23
```

```
##          actor_2_name  actor_1_facebook_likes     gross
##  Morgan Freeman :  20   Min.   :     0         Min.   :      162
##  Charlize Theron:  15   1st Qu.:   614         1st Qu.:  5340988
##  Brad Pitt      :  14   Median :   988         Median : 25517500
##                 :  13   Mean   :  6560         Mean   : 48468408
##  James Franco   :  11   3rd Qu.: 11000         3rd Qu.: 62309438
##  Meryl Streep   :  11   Max.   :640000         Max.   :760505847
##  (Other)        :4959   NA's   :7              NA's   :884
##                 genres                   actor_1_name
##  Drama                : 236   Robert De Niro   :  49
##  Comedy               : 209   Johnny Depp      :  41
##  Comedy|Drama         : 191   Nicolas Cage     :  33
##  Comedy|Drama|Romance: 187   J.K. Simmons     :  31
##  Comedy|Romance       : 158   Bruce Willis     :  30
##  Drama|Romance        : 152   Denzel Washington:  30
##  (Other)              :3910   (Other)          :4829
##                    movie_title   num_voted_users
##  Ben-HurÂ                  :   3   Min.   :      5
##  HalloweenÂ                :   3   1st Qu.:   8594
##  HomeÂ                     :   3   Median :  34359
##  King KongÂ                :   3   Mean   :  83668
##  PanÂ                      :   3   3rd Qu.:  96309
##  The Fast and the FuriousÂ :   3   Max.   :1689764
##  (Other)                   :5025
##  cast_total_facebook_likes         actor_3_name  facenumber_in_poster
##  Min.   :     0                          :  23   Min.   : 0.000
##  1st Qu.:  1411           Ben Mendelsohn:   8   1st Qu.: 0.000
##  Median :  3090           John Heard    :   8   Median : 1.000
##  Mean   :  9699           Steve Coogan  :   8   Mean   : 1.371
##  3rd Qu.: 13756           Anne Hathaway :   7   3rd Qu.: 2.000
##  Max.   :656730           Jon Gries     :   7   Max.   :43.000
##                           (Other)       :4982   NA's   :13
##                                                                                  plot_keywords
##                                                                                         : 153
##  based on novel                                                                         :   4
##  1940s|child hero|fantasy world|orphan|reference to peter pan                          :   3
##  alien friendship|alien invasion|australia|flying car|mother daughter relationship:   3
##  animal name in title|ape abducts a woman|gorilla|island|king kong                     :   3
##  assistant|experiment|frankenstein|medical student|scientist                           :   3
##  (Other)                                                                                :4874
##                                             movie_imdb_link
##  http://www.imdb.com/title/tt0077651/?ref_=fn_tt_tt_1:   3
##  http://www.imdb.com/title/tt0232500/?ref_=fn_tt_tt_1:   3
##  http://www.imdb.com/title/tt0360717/?ref_=fn_tt_tt_1:   3
##  http://www.imdb.com/title/tt1976009/?ref_=fn_tt_tt_1:   3
##  http://www.imdb.com/title/tt2224026/?ref_=fn_tt_tt_1:   3
##  http://www.imdb.com/title/tt2638144/?ref_=fn_tt_tt_1:   3
##  (Other)                                             :5025
##  num_user_for_reviews     language         country       content_rating
##  Min.   :   1.0       English :4704   USA      :3807   R        :2118
##  1st Qu.:  65.0       French  :  73   UK       : 448   PG-13    :1461
##  Median : 156.0       Spanish :  40   France   : 154   PG       : 701
##  Mean   : 272.8       Hindi   :  28   Canada   : 126            : 303
##  3rd Qu.: 326.0       Mandarin:  26   Germany  :  97   Not Rated: 116
```

```
##  Max.   :5060.0       German  :  19   Australia:  55   G         : 112
##  NA's   :21           (Other) : 153   (Other)  : 356   (Other)   : 232
##      budget            title_year   actor_2_facebook_likes   imdb_score
##  Min.   :2.180e+02   Min.   :1916   Min.   :     0         Min.   :1.600
##  1st Qu.:6.000e+06   1st Qu.:1999   1st Qu.:   281         1st Qu.:5.800
##  Median :2.000e+07   Median :2005   Median :   595         Median :6.600
##  Mean   :3.975e+07   Mean   :2002   Mean   :  1652         Mean   :6.442
##  3rd Qu.:4.500e+07   3rd Qu.:2011   3rd Qu.:   918         3rd Qu.:7.200
##  Max.   :1.222e+10   Max.   :2016   Max.   :137000         Max.   :9.500
##  NA's   :492         NA's   :108    NA's   :13
##   aspect_ratio   movie_facebook_likes
##  Min.   : 1.18   Min.   :     0
##  1st Qu.: 1.85   1st Qu.:     0
##  Median : 2.35   Median :   166
##  Mean   : 2.22   Mean   :  7526
##  3rd Qu.: 2.35   3rd Qu.:  3000
##  Max.   :16.00   Max.   :349000
##  NA's   :329
```

Plot Keywords

Only the first five plot keywords are captured from the web scrapping exercise. Here's the full IMDB page for the movie Avatar.

Let's find out what are the most commonly used words/phrases people use to describe movies.

```r
library(splitstackshape)
library(Matrix)
library(arules)
```

```
##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:arules':
##
##     intersect, recode, setdiff, setequal, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
movie$plot_keywords<-as.character(movie$plot_keywords)

movie_2<-cSplit(movie,"plot_keywords",sep="|")
pk<-select(movie_2,c(28:32))

keywords_basket<-paste(pk$plot_keywords_1,pk$plot_keywords_2,pk$plot_keywords_3,pk$plot_keywords_4,pk$pl

write(keywords_basket,file="keywords_basket") #write out as a text file

kb<-read.transactions("keywords_basket", format="basket", sep=",") #cannot use the read.transactions fu
```

```
## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string
```

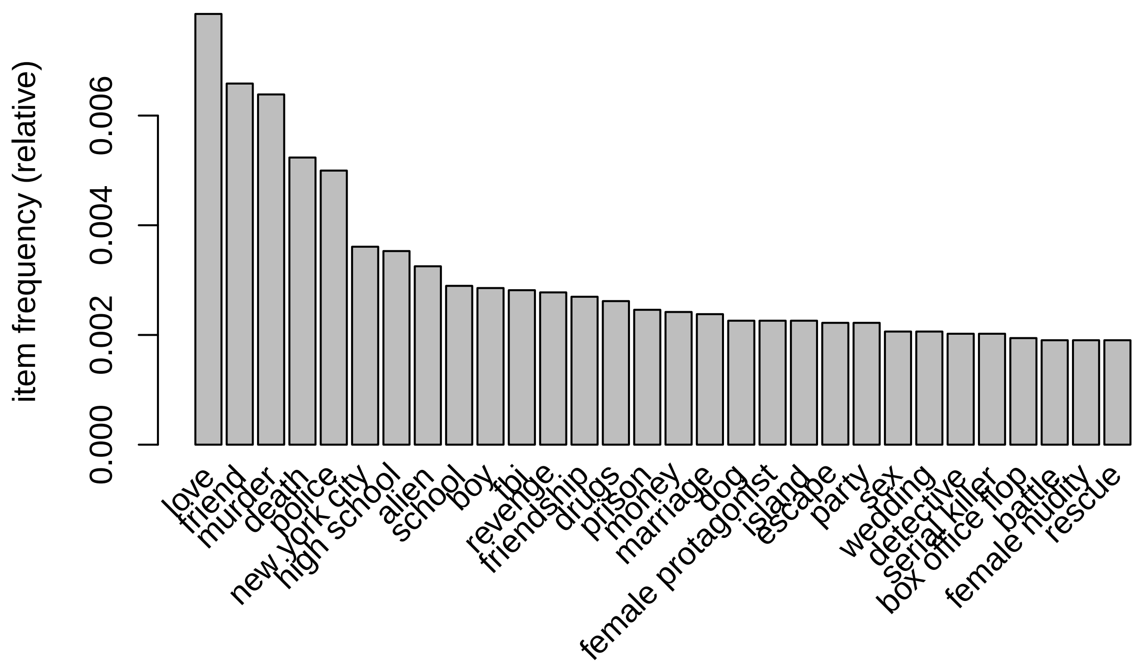
```
## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string
```

```
## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string
```

```
## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string
```

```
## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string

## Warning in scan(text = l, what = "character", sep = sep, quote = quote, :
## EOF within quoted string
```

```r
freq_kwb_df <-as.data.frame(itemFrequency(kb))

itemFrequencyPlot(kb,topN=30,names=TRUE)
```



Data Transformation

Discretization & Concept Hierarchies

Discretization is the process of turning a numeric attribute into interval labels. The purpose of discretization is to reduce the number of unique values in the data mining process. This is particularly useful for large datasets.

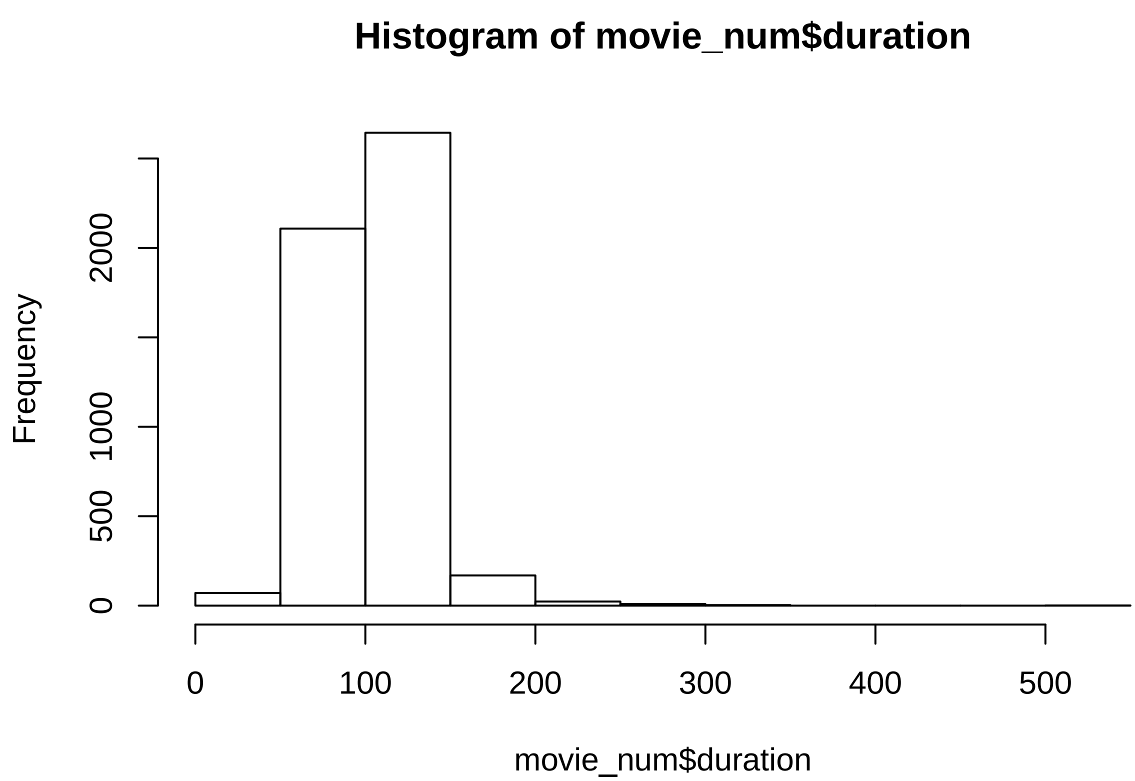
Concept hierarchies replace "lower level" raw data with "higher level" categories.

```
movie_3<-movie
names(movie_3)
```

```
##  [1] "color"                     "director_name"
##  [3] "num_critic_for_reviews"    "duration"
##  [5] "director_facebook_likes"   "actor_3_facebook_likes"
##  [7] "actor_2_name"              "actor_1_facebook_likes"
##  [9] "gross"                     "genres"
## [11] "actor_1_name"              "movie_title"
## [13] "num_voted_users"           "cast_total_facebook_likes"
## [15] "actor_3_name"              "facenumber_in_poster"
## [17] "plot_keywords"             "movie_imdb_link"
## [19] "num_user_for_reviews"      "language"
## [21] "country"                   "content_rating"
```

```
## [23] "budget"                     "title_year"
## [25] "actor_2_facebook_likes"     "imdb_score"
## [27] "aspect_ratio"               "movie_facebook_likes"
```

```
movie_num<-movie_3[,c(3,4,5,6,8,9,13,14,16,19,23,25,26,27,28)]
hist(movie_num$duration)
```




```
summary(movie_num$duration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     7.0    93.0   103.0   107.2   118.0   511.0      15
```

```
quantile(movie_num$duration,prob = seq(0, 1, length = 6),na.rm=TRUE)
```

```
##   0%  20%  40%  60%  80% 100%
##    7   91   99  108  122  511
```

Using Percentile Rank

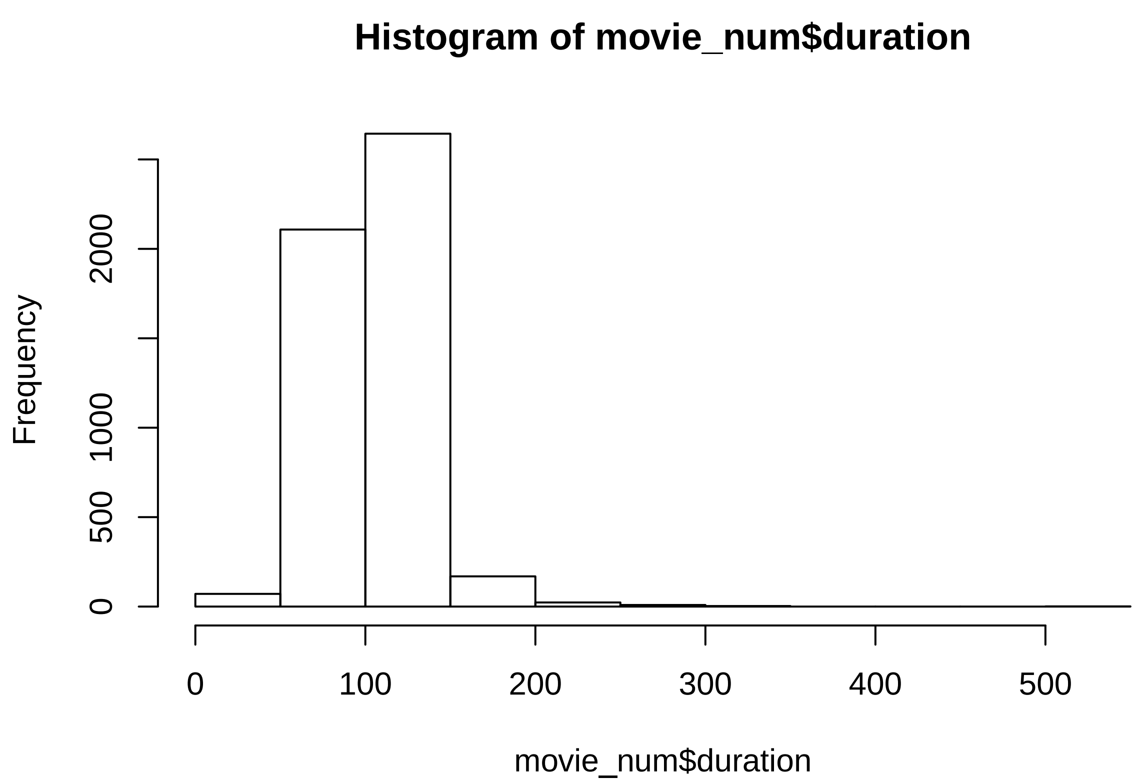
Let's create a factor variable for movie length (duration). 1 = shortest; 5 = longest.

```r
movie_num<-within(movie_num,quantile<-as.integer(cut(duration,quantile(movie_num$duration,prob = seq(0,

movie_num$quantile<-as.character(movie_num$quantile)

movie_num$movie_length_per <-factor(movie_num$quantile,levels=c(1,2,3,4,5), labels=c("Bottom Quantile",
```

Using Histogram

Most movies are between 100 to 150 minutes.

```r
hist(movie_num$duration)
```



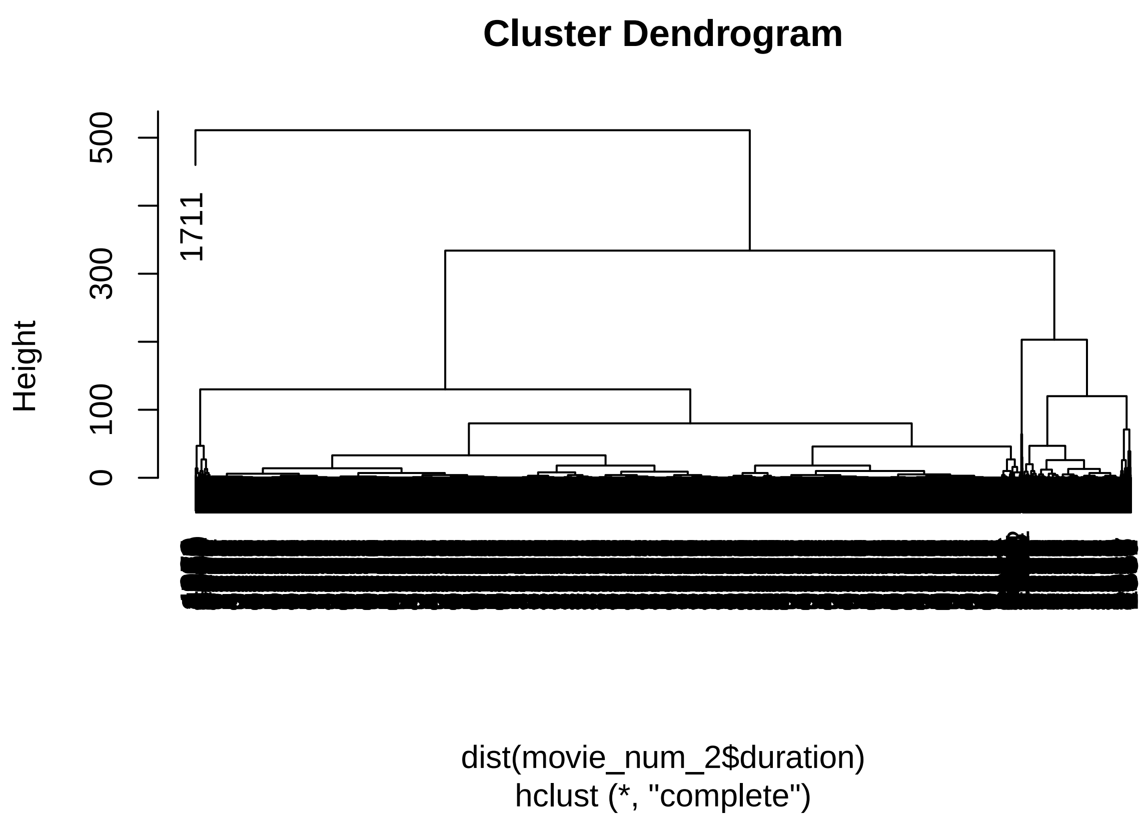

```r
movie_num$movie_length_hist<-
  ifelse(movie_num$duration<=100,"Short",
    ifelse (movie_num$duration<=150, "Average",
      ifelse(movie_num$duration<=511, "Long",
        ifelse(is.na(movie_num$duration), "NA"))))
```

Using Cluster Analysis

How many clusters? 4? 6?

Remember that cluster analysis does not like missing values. You can either recode missing values to "0" or remove them. We will recode in this exercise.

```r
movie_num_2<-movie_num

movie_num_2$duration<-ifelse(is.na(movie_num$duration),0,movie_num$duration) #recode

hc_duration<-hclust(dist(movie_num_2$duration), method="complete")

plot(hc_duration)
```




```r
movie_length_clusters <- cutree(hc_duration, k=4)

movie_num_2$movie_length_group<-movie_length_clusters
```

Cleaning up the environment.

```r
movie_num_2<-NULL
```

Cleaning up the data frame.

```r
movie_num$quantile<-NULL
movie_num$movie_length_per<-NULL
movie_num$movie_length_hist<-NULL
```

Normalization

Normalization is when numeric attribute is transformed to be on a smaller scale. Normalization is useful for data mining techniques that uses a distance measure (knn; cluster analysis).

Min-Max Normalization

```
normalize<- function(x,na.rm=TRUE){(x-min(x,na.rm=TRUE))/(max(x,na.rm=TRUE)-min(x.na.rm=TRUE))}
movie_num$budget_norm<-normalize(movie_num$budget)

summary(movie_num$budget_norm) #Checking the range
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.0000  0.0005  0.0016  0.0033  0.0037  1.0000     492
```

Z-Score Normalization (Or Mean Zero Normalization)

```
movie_num$budget_z<-(movie_num$budget - mean(movie_num$budget,na.rm=TRUE))/sd(movie_num$budget,na.rm=TRU
```

Z Normalization with Mean Absolute Deviation (MAD)

More robust to outliers.

```
movie_num$budget_z_mad<-(movie_num$budget - mean(movie_num$budget,na.rm=TRUE))/mad(movie_num$budget,na.
```

Decimal Scaling

```
max_budget<-max(movie_num$budget, na.rm=TRUE)

digits <- floor(log10( max_budget))+1

print(digits)
```

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

```
movie_num$budget_decimal<-(movie_num$budget)/(10^(digits))
```

Note: Digits code chunk above is from here

Let's clean up what we have done.

```
movie_num$budget_norm<-NULL
movie_num$budget_z_mad<-NULL
movie_num$budget_z<-NULL
movie_num$budget_decimal<-NULL
```

Data Reduction

The purpose of data reduction is to "obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should produce more efficient yet produce the same (or almost the same) analytical results" (Han & Kamber 2006, p. 73).

We will discuss one way to achieve data reduction: Principal Component Analysis.

The Curse of Dimensionality

The curse of dimensionality refers to the fact that algorithms that work in low dimensional space do not have the same performance in high dimensional space.

In a paper titled *A Few Useful Things to Know about Machine Learning*, Pedro Domingos wrote:

After overfitting, the biggest problem in machine learning is the curse of dimensionality....Generalizing correctly becomes exponentially harder as the dimensionality (number of features) of the xamples grows, because a fixed-size training set covers a dwindling fraction of the input space.

...our intuitions, which come from a three-dimensional world, often do not apply in high-dimensional ones. In high dimensions, most of the mass of a multivariate Gaussian distribution is not near the mean, but in an increasingly distant "shell" around it; and most of the volume of a high-dimensional orange is in the skin, not in the pulp. If a constant number of examples is distributed uniformly in a a high-dimensional hypercube, beyond some dimensionality samples are closer to the face of the hypercube than to their nearest neighbor (p.4).

An example may also make things a bit clearer. Keogh and Mueen (2010) said:

For example, 100 evenly-spaced sample points suffice to sample a unit interval with no more than 0.01 distance between points; an equivalent sampling of a 10-dimensional unit hypercube with a grid with a spacing of 0.01 between adjacent points would require 10ˆ20 sample poits: thus, in some sense, the 10D hypercube can be said to be a factor of 10ˆ18 "larger" than the unit interval (p.257).

What Can Go Wrong in Higher Dimensional Space?

Keogh & Mueen (2010) and Domingos (2012) noted several problems:

- Increase in dimensionality requires a large increase in observations to keep the same performance for machine learning algorithms.
- Notion of "distance" becomes meaningless. All neighbors become equidistant (=1) in high dimensional space.

Principal Component Analysis (PCA)

PCA has many functions:

- Allows us to tackle the curse of dimensionality problem.
- Allows us to handle multicollinearity in regressions because we can use the principal components as "predictors."
- Visualize data in a lower dimensional space.

PCA: Intuitive Explanation

An intuitive explanation for principal components is given in Gareth et al 2013:

"Principal components are the dimensions that are closest to the n observations" (p.379)

"The first principal component loading vector has a very special property: it is the line in p-dimensional space that is closest to the n observations (using average squared Euclidean distance as a measure of closeness" (p. 379)

PCA: Formal Definition

The formal definition of principal components are as follows:

The first principal component of a set of features X1, X2,. . . ,Xp is the normalized linear combination of the features Z1 = a11*X1 + a2*X2 + . . . + ap1*Xp that has the largest variance.

a11, a21,. . . ,ap1 are called the loadings of the principal component. These values equal to 1.

The second principal component is the linear combination of X1,. . . ,Xp that has maximal variance out of all linear combinations that are uncorrelated with Z1 (p.376).

In this exercise, we are going to take a 15-dimensional dataset and reducing it down to something smaller.

The function prcomp does not like missing values. We will recode to 0.

```
movie_num[is.na(movie_num)]<-0 #3801 observations

p <- prcomp(movie_num, scale=TRUE, center=TRUE)

summary(p)
```

```
## Importance of components:
##                           PC1    PC2     PC3    PC4     PC5     PC6
## Standard deviation     2.1153 1.4365 1.03244 1.0151 0.99674 0.97764
## Proportion of Variance 0.2983 0.1376 0.07106 0.0687 0.06623 0.06372
## Cumulative Proportion  0.2983 0.4359 0.50694 0.5756 0.64187 0.70559
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.94455 0.90283 0.87327 0.82182 0.65252 0.63583
## Proportion of Variance 0.05948 0.05434 0.05084 0.04503 0.02839 0.02695
## Cumulative Proportion  0.76507 0.81941 0.87025 0.91527 0.94366 0.97061
##                           PC13   PC14    PC15
## Standard deviation     0.53776 0.3873 0.04058
## Proportion of Variance 0.01928 0.0100 0.00011
## Cumulative Proportion  0.98989 0.9999 1.00000
```

Looking at the loadings of the principal components.

```
loadings<-p$rotation[,]
#View(loadings)
```

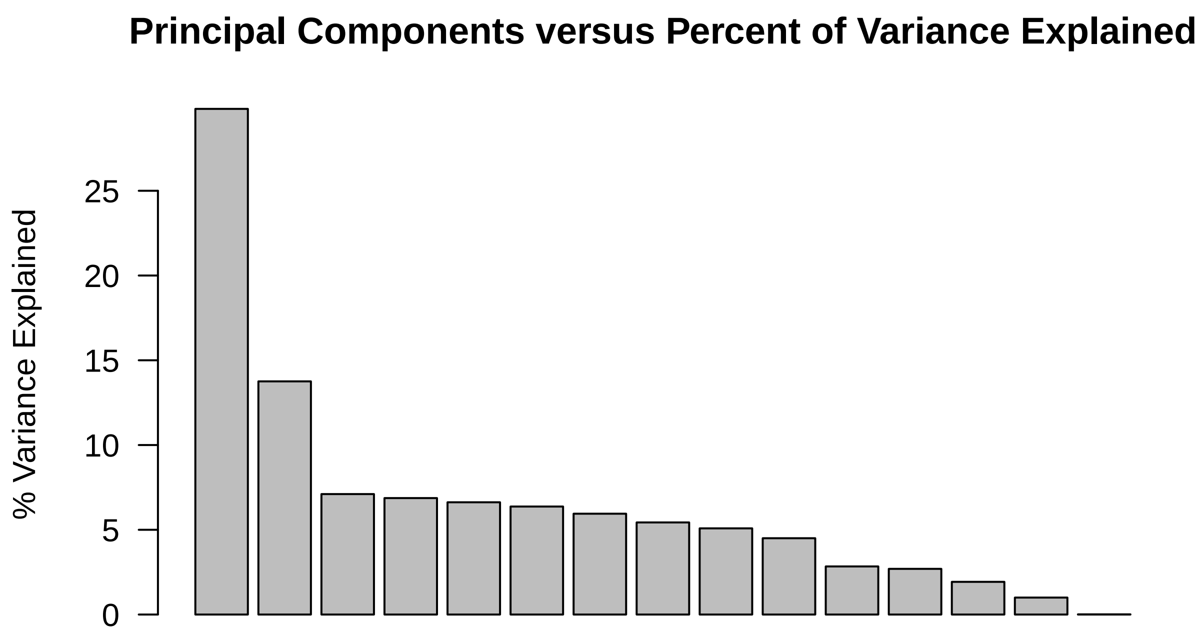
Let's see if we can visualize our principal components.

Bar Chart of Variance Explained by PCA

```
p.variance.explained <-(p$sdev^2 / sum(p$sdev^2))*100
print(p.variance.explained)
```

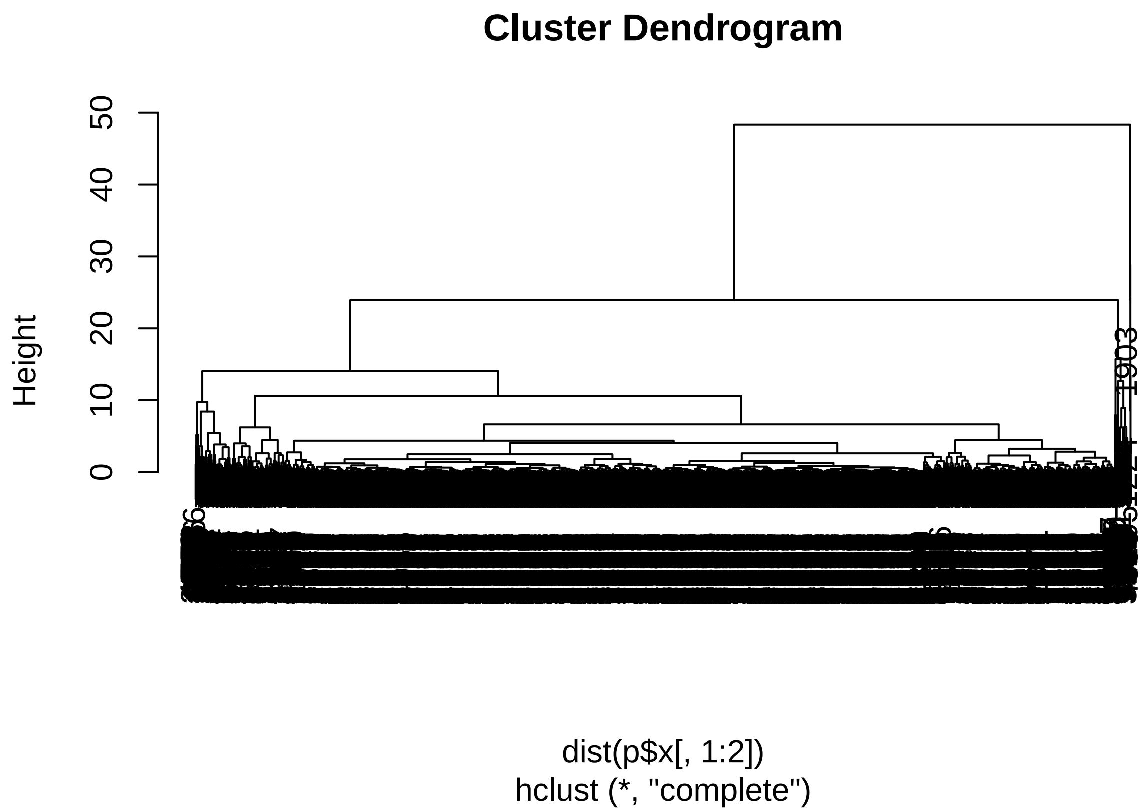
```
##  [1] 29.83093092 13.75642758  7.10620829  6.87010198  6.62322098
##  [6]  6.37191038  5.94777866  5.43406130  5.08396454  4.50255429
## [11]  2.83854886  2.69522696  1.92792681  1.00015911  0.01097933
```

```
# plot percentage of variance explained for each principal component
barplot(p.variance.explained, las=2, xlab="", ylab="% Variance Explained", main="Principal Components ve
```



Shall We Cluster the Principal Components?

```
hc_tree<-hclust(dist(p$x[,1:2]), method="complete") # 1:2 = based on 2 components
plot(hc_tree)
```




```
groups <- cutree(hc_tree, k=4)

movie$cluster<-groups #writing out the cluster assignment for each movie


# k = number of groups
```

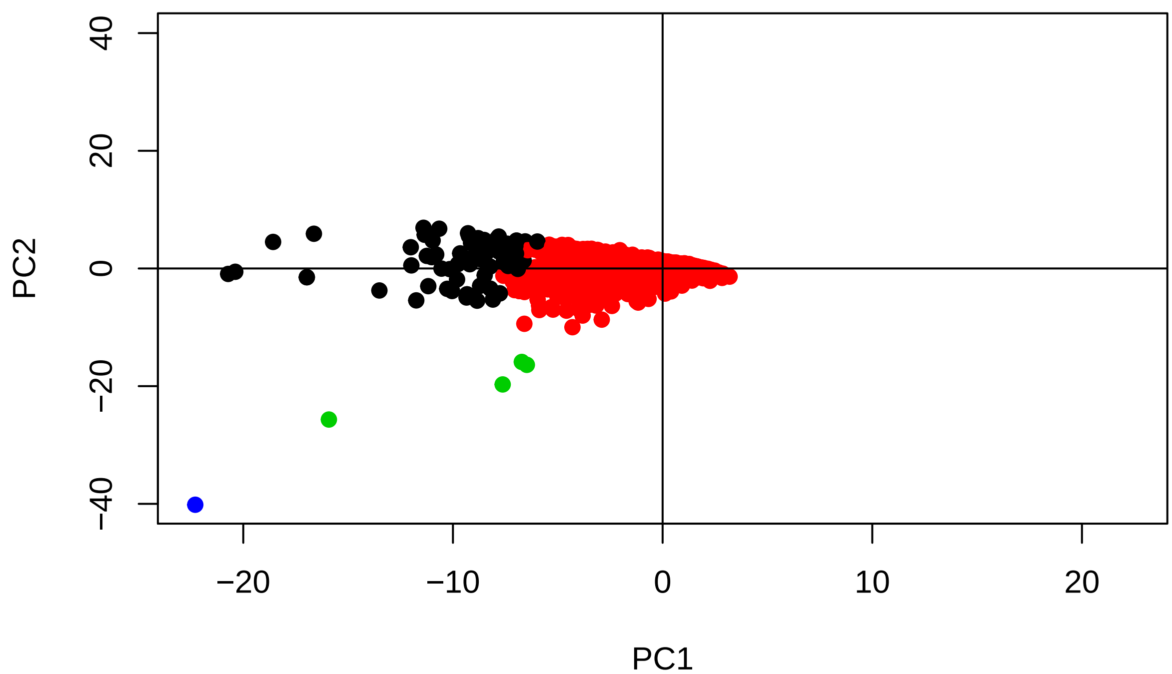
This function creates 2-dimensional scatter plot for 2 principal components & 3-dimensional scatter plot for 3 principal components.

The function is written by Karolis Koncevicius.

```
library(rgl)
plotPCA <- function(x, nGroup) {
    n <- ncol(x)
    if(!(n %in% c(2,3))) { # check if 2d or 3d
        stop("x must have either 2 or 3 columns")
    }

    fit <- hclust(dist(x), method="complete") # cluster
    groups <- cutree(fit, k=nGroup)

    if(n == 3) { # 3d plot
        plot3d(x, col=groups, type="s", size=1, axes=F)
        axes3d(edges=c("x--", "y--", "z"), lwd=3, axes.len=2, labels=TRUE)
        grid3d("x")
```

```r
        grid3d("y")
        grid3d("z")
    } else { # 2d plot
        maxes <- apply(abs(x), 2, max)
        rangeX <- c(-maxes[1], maxes[1])
        rangeY <- c(-maxes[2], maxes[2])
        plot(x, col=groups, pch=19, xlab=colnames(x)[1], ylab=colnames(x)[2], xlim=rangeX, ylim=rangeY)
        lines(c(0,0), rangeX*2)
        lines(rangeY*2, c(0,0))
    }
}
```

```r
plotPCA(p$x[,1:2],4) #2D
```




```r
#plotPCA(p$x[,1:3],4) #3D
```

Some Interesting Characteristics about Movies

```r
table(movie$cluster)
```

```
##
```

```
##    1    2    3    4
##   80 4958    4    1
```

```r
aggregate(data = movie, duration ~ cluster, mean)
```

```
##   cluster duration
## 1       1 144.9875
## 2       2 106.6071
## 3       3  87.7500
## 4       4  98.0000
```

```r
aggregate(data = movie, budget ~ cluster, mean)
```

```
##   cluster    budget
## 1       1 133560000
## 2       2  38097795
## 3       3  14656250
## 4       4  26000000
```

```r
aggregate(data = movie, gross ~ cluster, mean)
```

```
##   cluster     gross
## 1       1 280763209
## 2       2  43907839
## 3       3  33281553
## 4       4  84136909
```

```r
aggregate(data = movie, imdb_score ~ cluster, mean)
```

```
##   cluster imdb_score
## 1       1   7.793750
## 2       2   6.420573
## 3       3   5.950000
## 4       4   7.200000
```

Cluster 1: **The Blockbusters**

Long movies (average 144 minutes); big budget (average $133 million); box office hits (average $280 million); highest IMDB score (average 7.8).

Cluster 2: **Weekend Standard Fare**

Average length movies (average 106 minutes); mid-sized budget (average $38 million); weekend opening leaders (average $43 million); average IMDB score (average 6.4)

Cluster 3: **I Wish I Didn't Waste My Time on That!**

Short movies (average 87 minutes); small budget (average $14 million); box office flops (average $33 million); terrible IMDB score (average 6.0)

Cluster 4: **The Unicorn**

Average length movie (average 98 minutes); small budget (averge $26 million); box office hit ($84 million); high IMDB score (7.2)

Let's see what these movies are.

```
Cluster_1<-subset(movie,cluster==1)
Cluster_2<-subset(movie,cluster==2)
Cluster_3<-subset(movie,cluster==3)
Cluster_4<-subset(movie,cluster==4)
```

Another PCA Application

We are going to replicate an exercise conducted by Hakkio and Willis (2013) of the Federal Reserve Bank of Kansas City. Hakkio and Willis created two measures of labor market conditions from 23 labor market variables. One measure tracks the level of activity of labor market conditions. The second measure tracks the rate of change.

Here are some useful links:

LMCI

Macro Bulletin

In our exercise, we can only get access to 15 variables since the remaining ones are proprietary information. In any case, our goal is to take a dataset in 15 dimensions and reducing it down to 2 dimensions.

First, we need to do some data wrangling.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```
LMCI_large<-read.csv("LMCI_month.csv")
LMCI_small<-read.csv("LMCI_week.csv")

#let's handle the LMCI_small dataset first.
LMCI_small$date<-ymd(as.character(LMCI_small$DATE)) #we have to convert the DATE into "Date" format.
LMCI_small$year<-year(LMCI_small$date)
LMCI_small$month<-month(LMCI_small$date)
LMCI_small.2<-as.data.frame(aggregate(ICSA~year+month, LMCI_small, mean))
LMCI_small.3<-LMCI_small.2[order(LMCI_small.2$year,LMCI_small.2$month),]#sort by year and month

#and now we deal with the LMCI_month dataset.
LMCI_large$date<-ymd(as.character(LMCI_large$DATE))
LMCI_large$year<-year(LMCI_large$date)
LMCI_large$month<-month(LMCI_large$date)
LMCI_large.2<-LMCI_large[order(LMCI_large$year,LMCI_large$month),]
LMCI_large.3<-LMCI_large.2[,c(-1,-16)]

#merging will cause us to lose one observation for ISCA since there is a value for the month of April 2
LMCI_combined<-merge(LMCI_small.3,LMCI_large.3, by=c("year", "month"))
LMCI_combined<-LMCI_combined[,c(-1,-2)]
```

And now onto the PCA.

```r
LMCI_combined_zscore<-data.frame(scale(LMCI_combined))

#Checking to see if z-score normalization works. We expect mean=0 and sd=1.
sapply(LMCI_combined_zscore, mean)
```

```
##          ICSA        AHETPI          AWHI       CIVPART       EMRATIO
## -3.576004e-18 -2.031571e-16  4.899051e-16 -1.064059e-15 -5.643324e-16
##   LNS12032194   LNS13023622   LNS13023706   LNS13025703   LNS17100000
##  5.032859e-17 -1.551042e-17  3.065913e-16 -1.117475e-16  8.720540e-16
##        NAPMEI     TEMPHELPS        U6RATE        UNRATE        USPRIV
## -3.087808e-16  8.882261e-18            NA  1.546173e-16 -4.003761e-16
```

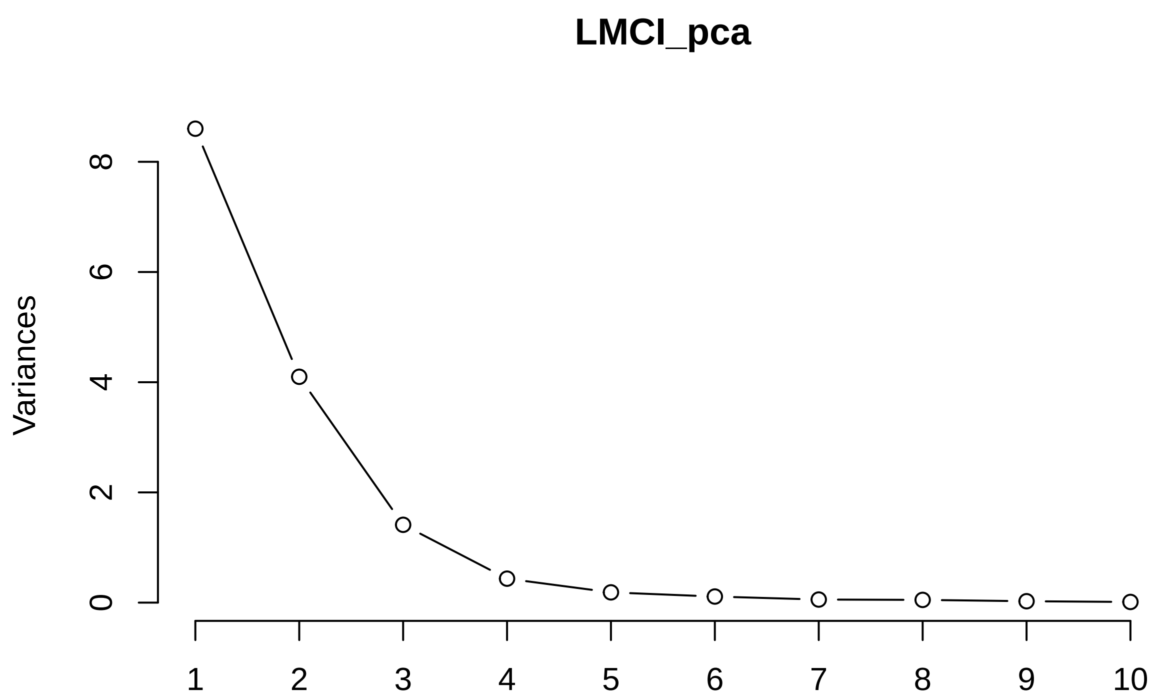
```r
sapply(LMCI_combined_zscore, sd)
```

```
##        ICSA      AHETPI        AWHI     CIVPART     EMRATIO LNS12032194
##           1           1           1           1           1           1
## LNS13023622 LNS13023706 LNS13025703 LNS17100000      NAPMEI   TEMPHELPS
##           1           1           1           1           1           1
##      U6RATE      UNRATE      USPRIV
##          NA           1           1
```

```r
LMCI_pca<-prcomp(na.omit(LMCI_combined_zscore), center=TRUE, scale=TRUE)
summary(LMCI_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4    PC5     PC6    PC7
## Standard deviation     2.9327 2.0245 1.18825 0.65939 0.4313 0.33165 0.2354
## Proportion of Variance 0.5734 0.2732 0.09413 0.02899 0.0124 0.00733 0.0037
## Cumulative Proportion  0.5734 0.8466 0.94075 0.96973 0.9821 0.98947 0.9932
##                            PC8     PC9    PC10    PC11    PC12    PC13
## Standard deviation     0.22224 0.15869 0.10991 0.09545 0.06831 0.03871
## Proportion of Variance 0.00329 0.00168 0.00081 0.00061 0.00031 0.00010
## Cumulative Proportion  0.99645 0.99813 0.99894 0.99954 0.99986 0.99996
##                           PC14    PC15
## Standard deviation     0.02009 0.01626
## Proportion of Variance 0.00003 0.00002
## Cumulative Proportion  0.99998 1.00000
```

```r
screeplot(LMCI_pca, type="lines") #note the first two PC account for 84% of the variance in the data.
```



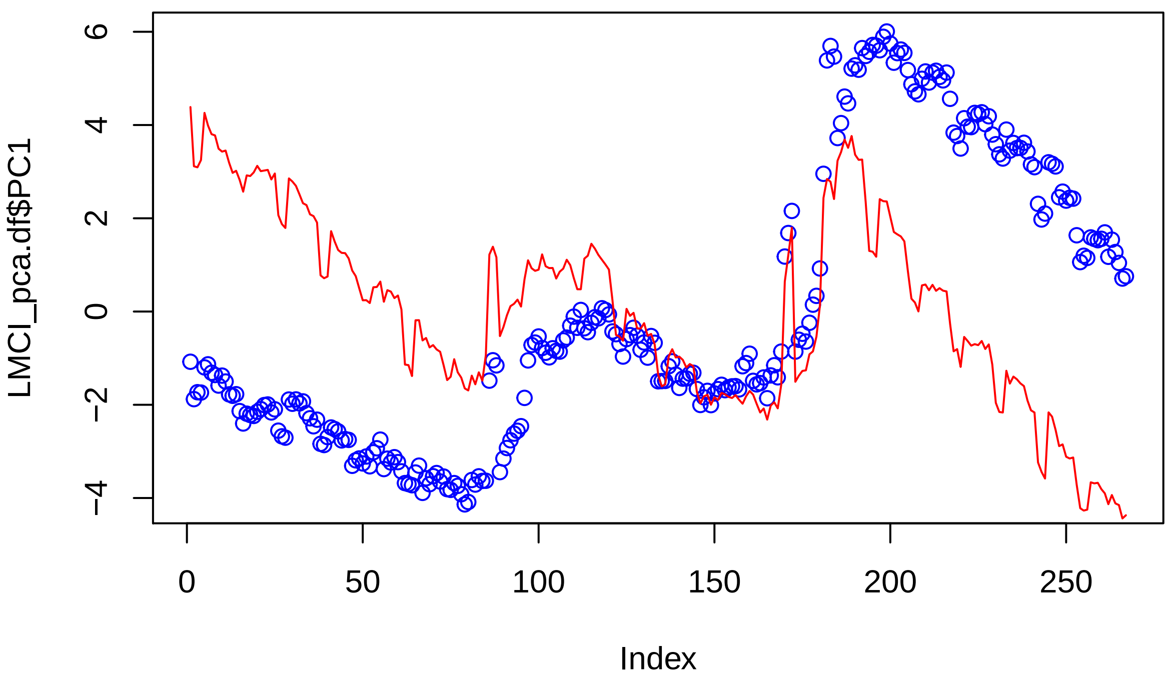

```r
LMCI_pca$rotation[,1]
```

```
##        ICSA      AHETPI        AWHI     CIVPART     EMRATIO LNS12032194
##  0.18337405  0.27090091  0.07926411 -0.26588937 -0.32885299  0.33438954
## LNS13023622 LNS13023706 LNS13025703 LNS17100000      NAPMEI    TEMPHELPS
##  0.28794758 -0.31985989  0.32575518  0.28289429  0.09491689  0.01514861
##      U6RATE      UNRATE      USPRIV
##  0.32618120  0.31500440  0.09907381
```

```r
#PC1 = 0.183*ICSA + 0.271*AHETPI + ...

LMCI_pca$rotation[,2]
```

```
##        ICSA      AHETPI        AWHI     CIVPART     EMRATIO LNS12032194
##  0.29517668 -0.28656349 -0.45623486  0.24760501  0.07101677  0.01186852
## LNS13023622 LNS13023706 LNS13025703 LNS17100000      NAPMEI    TEMPHELPS
##  0.13508390 -0.10895801 -0.02785064  0.17416145 -0.15488718 -0.48445786
##      U6RATE      UNRATE      USPRIV
##  0.12618925  0.17454919 -0.43566200
```

```
#PC2 = 0.30*ICSA - 0.287*AHETPI + ...

LMCI_pca.df<-data.frame(LMCI_pca$x[,1:2])

plot(LMCI_pca.df$PC1, col="blue") #level of activity index?
lines(LMCI_pca.df$PC2, col="red") #rate of change index?
```



In the above PCA example, we omitted the first 24 observations due to the NA's for the variable U6RATE. Omission means we have to deal with less data points, and, thus, less variation. What other data preprocessing techniques we could have used to handle the missing values?

Because We Love the Bakery Dataset

Now think about taking a dataset that has 51 dimensions and reducing it down to two dimensions.

```
library(cluster)
library(fpc)

bakery <- read.csv("bakery-binary.csv")
str(bakery)
```

```
## 'data.frame':    1000 obs. of  51 variables:
##  $ Chocolate.Cake      : int  0 0 1 0 0 0 0 0 1 1 ...
```

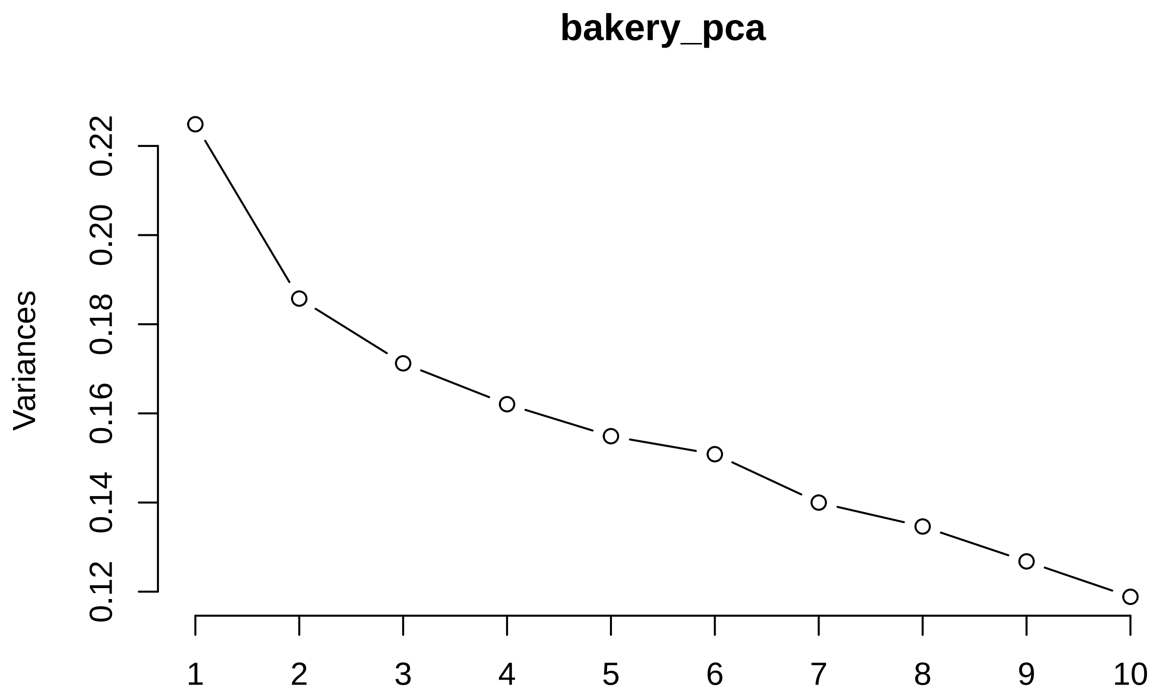
```
## $ Lemon.Cake          : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Casino.Cake         : int  0 0 0 1 1 0 0 0 0 1 ...
## $ Opera.Cake          : int  0 1 0 0 0 0 1 1 0 0 ...
## $ Strawberry.Cake     : int  0 0 1 0 0 0 0 0 0 0 ...
## $ Truffle.Cake        : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Chocolate.Eclair    : int  0 0 0 1 0 0 0 1 0 0 ...
## $ Coffee.Eclair       : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Vanilla.Eclair      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Napoleon.Cake       : int  0 0 1 0 0 1 0 0 0 0 ...
## $ Almond.Tart         : int  0 0 0 0 1 0 0 0 0 0 ...
## $ Apple.Pie           : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Apple.Tart          : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Apricot.Tart        : int  0 0 0 0 0 0 0 1 0 0 ...
## $ Berry.Tart          : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Blackberry.Tart     : int  1 0 0 0 0 0 0 0 0 0 ...
## $ Blueberry.Tart      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Chocolate.Tart      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Cherry.Tart         : int  0 1 0 0 0 0 0 0 0 0 ...
## $ Lemon.Tart          : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Pecan.Tart          : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Ganache.Cookie      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Gongolais.Cookie    : int  0 0 0 1 1 0 0 0 0 0 ...
## $ Raspberry.Cookie    : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Lemon.Cookie        : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Chocolate.Meringue  : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Vanilla.Meringue    : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Marzipan.Cookie     : int  0 0 0 0 0 1 0 0 0 0 ...
## $ Tuile.Cookie        : int  0 0 0 0 0 1 0 0 0 0 ...
## $ Walnut.Cookie       : int  0 0 0 0 0 0 1 0 0 0 ...
## $ Almond.Croissant    : int  0 0 1 0 0 0 0 0 0 0 ...
## $ Apple.Croissant     : int  0 0 0 0 1 0 0 0 0 0 ...
## $ Apricot.Croissant   : int  0 0 0 1 0 0 0 0 0 0 ...
## $ Cheese.Croissant    : int  0 0 1 0 1 0 0 0 0 0 ...
## $ Chocolate.Croissant: int  1 0 0 0 0 0 0 0 0 0 ...
## $ Apricot.Danish      : int  0 1 0 0 0 0 0 0 0 0 ...
## $ Apple.Danish        : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Almond.Twist        : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Almond.Bear.Claw    : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Blueberry.Danish    : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Lemon.Lemonade      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Raspberry.Lemonade : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Orange.Juice        : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Green.Tea           : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Bottled.Water       : int  0 0 0 0 0 0 1 0 0 0 ...
## $ Hot.Coffee          : int  0 0 0 0 0 1 0 0 0 0 ...
## $ Chocolate.Coffee    : int  0 0 0 0 1 0 0 0 0 1 ...
## $ Vanilla.Frappuccino: int  1 0 0 0 0 0 0 0 0 0 ...
## $ Cherry.Soda         : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Single.Espresso     : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Weekend             : int  0 0 1 1 0 0 0 0 0 0 ...
```

```
#View(bakery)

#pca using raw data
```

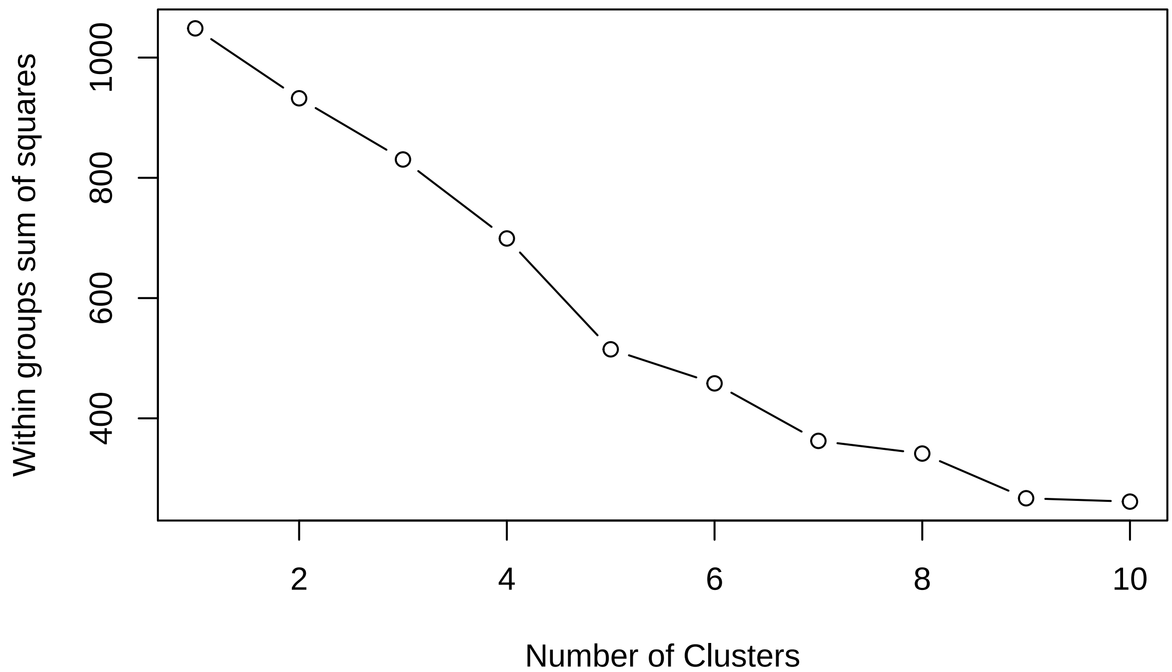
```r
set.seed(12345)
bakery_pca<-prcomp(bakery)
summary(bakery_pca)
```

```
## Importance of components:
##                            PC1     PC2    PC3     PC4     PC5     PC6
## Standard deviation     0.47420 0.43099 0.4138 0.40257 0.39356 0.38838
## Proportion of Variance 0.06448 0.05326 0.0491 0.04647 0.04441 0.04325
## Cumulative Proportion  0.06448 0.11774 0.1668 0.21331 0.25772 0.30098
##                            PC7    PC8     PC9    PC10    PC11   PC12
## Standard deviation     0.37415 0.3669 0.35609 0.34475 0.33271 0.3304
## Proportion of Variance 0.04014 0.0386 0.03636 0.03408 0.03174 0.0313
## Cumulative Proportion  0.34112 0.3797 0.41608 0.45016 0.48190 0.5132
##                           PC13    PC14    PC15    PC16    PC17    PC18
## Standard deviation     0.31357 0.31076 0.24869 0.24380 0.24180 0.24090
## Proportion of Variance 0.02819 0.02769 0.01773 0.01704 0.01677 0.01664
## Cumulative Proportion  0.54139 0.56908 0.58682 0.60386 0.62063 0.63727
##                           PC19   PC20    PC21    PC22    PC23    PC24
## Standard deviation     0.23443 0.2310 0.22539 0.22112 0.22027 0.21703
## Proportion of Variance 0.01576 0.0153 0.01457 0.01402 0.01391 0.01351
## Cumulative Proportion  0.65302 0.6683 0.68289 0.69691 0.71083 0.72433
##                           PC25   PC26    PC27    PC28    PC29    PC30
## Standard deviation     0.21555 0.2145 0.21145 0.20861 0.20669 0.20520
## Proportion of Variance 0.01332 0.0132 0.01282 0.01248 0.01225 0.01207
## Cumulative Proportion  0.73766 0.7509 0.76367 0.77615 0.78840 0.80047
##                           PC31    PC32    PC33    PC34    PC35    PC36
## Standard deviation     0.20246 0.20055 0.19835 0.19736 0.19552 0.19402
## Proportion of Variance 0.01175 0.01153 0.01128 0.01117 0.01096 0.01079
## Cumulative Proportion  0.81223 0.82376 0.83504 0.84621 0.85717 0.86796
##                           PC37    PC38   PC39    PC40   PC41    PC42
## Standard deviation     0.19207 0.18889 0.1876 0.18631 0.1820 0.18001
## Proportion of Variance 0.01058 0.01023 0.0101 0.00995 0.0095 0.00929
## Cumulative Proportion  0.87854 0.88877 0.8989 0.90882 0.9183 0.92761
##                           PC43    PC44    PC45    PC46    PC47    PC48
## Standard deviation     0.17821 0.17741 0.17396 0.16974 0.16956 0.16786
## Proportion of Variance 0.00911 0.00903 0.00868 0.00826 0.00824 0.00808
## Cumulative Proportion  0.93672 0.94574 0.95442 0.96268 0.97093 0.97900
##                           PC49    PC50    PC51
## Standard deviation     0.16422 0.15728 0.14668
## Proportion of Variance 0.00773 0.00709 0.00617
## Cumulative Proportion  0.98674 0.99383 1.00000
```

```r
screeplot(bakery_pca, type="lines")
```

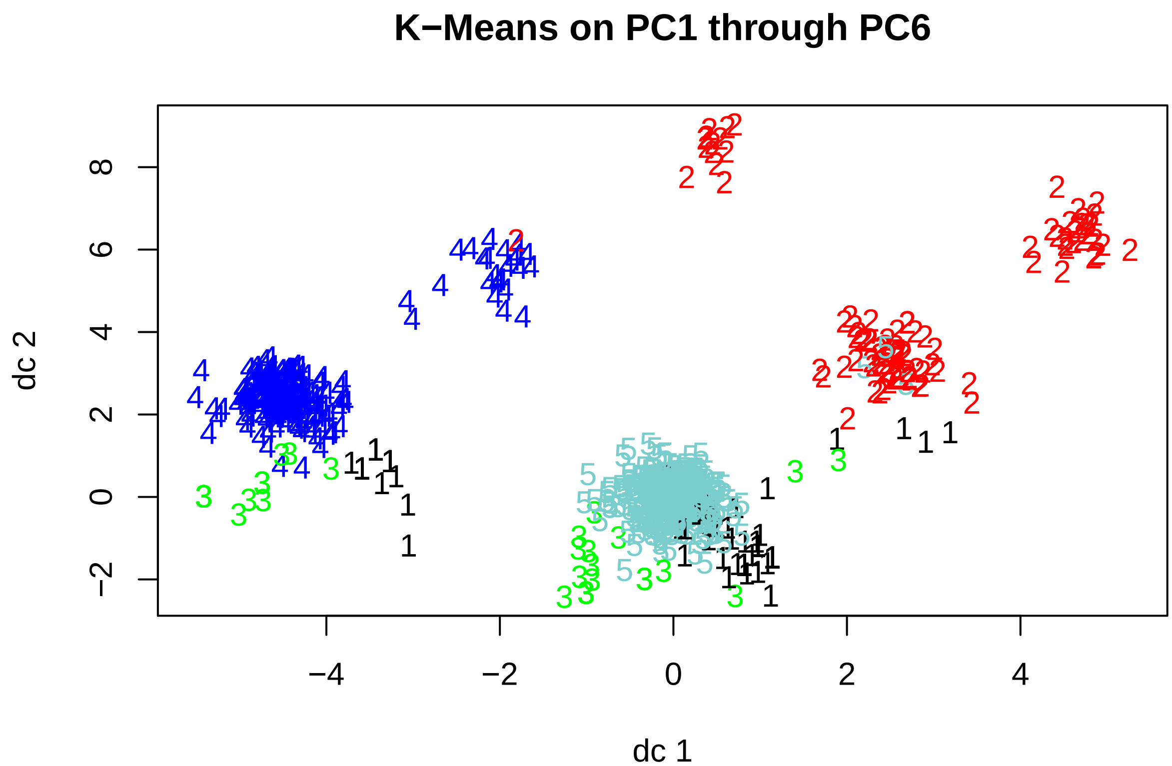



```r
bakery_pca.df<-data.frame(bakery_pca$x[,1:6]) #first 6 PCs


#Using the elbow method. Maybe 5 groups?
set.seed(12345)
wss <- (nrow(bakery_pca.df)-1)*sum(apply(bakery_pca.df,2,var))
for (i in 2:10) wss[i] <- sum(kmeans(bakery_pca.df,
                                     centers=i)$withinss)
plot(1:10, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
```



Clustering the first six PCAs to come up with cluster assignments.

```
set.seed(12345)
bakery_cluster_pca<-kmeans(bakery_pca.df, center=5)
plotcluster(bakery, bakery_cluster_pca$cluster, main="K-Means on PC1 through PC6")
```



Now we take the cluster assignments and attach to the bakery dataset.

```
bakery$cluster<-bakery_cluster_pca$cluster

table(bakery$cluster)
```

```
##
##   1   2   3   4   5
##  88 120  50 263 479
```