Topic 3
Data Preparation and Preprocessing

Xiaogang Su, Ph.D.
Department of Mathematical Sciences
University of Texas at El Paso (UTEP)
xsu@utep.edu

Data coming from real world are imperfect. They can be incomplete (with missing attribute values or certain attributes of interest), noisy (containing errors or outliers which deviate from the expected), and inconsistent (e.g., containing discrepancies from the codebook used to categorize items). Data preparation or preprocessing refers to steps that one takes to make data finally ready for a supervised or unsupervised task. Recall the CRISP model or the SEMMA model for data mining. They all start with data preprocessing.

The specific steps in data preparation include:

1. *Data Integration* – Data sets from multiple sources are integrated by merging or concatenating into a data warehouse or a data cube.
2. *Sampling* – Take random samples from the original database via simple random sampling or stratified sampling; partition the whole sample into separate sets, which will serve for different analytic purposes.
3. *Data Cleaning* – Identify and remove or correct wrongly recorded values or outliers.
4. *Missing Value Handling* – Most procedures cannot deal with missing values. Missing values can be either removed or imputed, depending on the missing percentage and other characteristics.
5. *Data Reduction* – The amount of data can be reduced in both size and dimension by aggregating appropriately, eliminating redundant features, defining linear combinations via principal components analysis, or clustering, for instance.
6. *Exploratory Data Analysis* – Various numerical summary statistics are computed and graphical plots are made to get acquainted with and gain insight into the data under study.
7. *Variable Transformation* – Variables can be transformed in different ways, to serve better for the analytic purposes. Examples include normalization or categorization of a continuous variable and level merging for a categorical variable.

As a matter of fact, data preparation and preprocessing often make up about 90% of the total time spent in a real-world data mining project. The procedures involved are tedious and time-consuming, yet critical for the subsequent data mining tasks. At the same time, data preprocessing takes both experiences for statisticians and knowledge in the subject matter. In this chapter, we shall briefly walk through those procedures via hands-on examples with R programming. Illustrations are also made via a simulated data from the admission database in the School of Nursing (SON) at University of Alabama at Birmingham (UAB).

The University Admission Data

There are two simulated datasets involved in this project, both in the csv format. The first dataset, "applicants.csv", contains info on 986 applicants to the school of nursing in a university in three years. It contains 25 variables, as described in Table 1.

The objective is to profile the applicants and those who got admitted and see how admitted students performed in the course of later study. In particular, it is of keen interest to compare UAB-native students and students who were transferred from another college or university. It is also intriguing to see how students transferred from two-year colleges are compared to those from four-year colleges. For this purpose, a new variable is to be created using the info given in another file called "college-level.csv". Besides, our primary interest is on applicants to the basic degree in nursing rather than those who seek a second-degree in nursing.

Table 1 Variable description for the applicant data set.

Col	Name	Description
1	LAST ACT	Date of last active event with the student
2	Program	Basic; BASIC-2ND for second degree in nursing
3	SEM	Semester of Application
4	YR	Year of Application
5	INITIAL TERM ADM	Initial term of admission (if admitted)
6	Last	Last Name (removed for privacy reasons)
7	First	First Name (removed for privacy reasons)
8	BooNumber	ID
9	COLL1	The first college the student went to.
10	PNUR GPA	GPA for Pre-Nursing period
11	CUM GPA	Cumulative GPA
12	MTH.Yr	Expected graduation date
13	BSN # TERMS TO GRAD	Number of semesters taken to graduate
14	GRAD-GPA	GPA upon graduation
15	NATIVE.UAB	UAB native student or transferred?
16	APL.STATUS	Application Status - admitted or denied?
17	Current Student Status	Current student status
18	NCLEX 1	Passed the first NCLEX exam?
19	NUR HONORS	Taken any honors scourse in nursing?
20	FAILURES ACADEMIC HISTORY	Number of course failures
21	W ACADEMIC HISTORY	Number of Course Withdrawals
22	REPEATS - ACAD HISTORY	Number of repeated courses
23	1st GENERATION	The first generation in college in his/her family
24	RACE	Race
25	Gender	Gender

Data Integration

We have learned that data mining mostly works with observational data. Here, the word "observational" should be interpreted liberally. It merely means that the data to be analyzed already exist. In terms of their original sources, they may come from observational studies, national surveys, or even designed experiments (e.g., in a secondary analysis for answering additional research questions or for generating new research hypotheses).

In a data mining project, the datasets to be analyzed sometimes may come in handy; in other times, data miners need to dig them out from difference data warehouses or resources. Then datasets obtained from different sources are combined together into one data set for a specific project or a specific research question. Most of the time, the data set to be analyzed or modeled would be a data matrix, in which each row represents observed values or one set of observations taken on one subject and each column represents one variable. Variables are also called *attribute* or *feature* in some data mining / machine learning (ML) languages. Some ML authors also distinguish between attribute and feature. Attributes are variables originally provided in the raw data while features are derived or selected variables obtained in the feature generation/extraction and feature selection process.

There are typically two types of data manipulations at this stage: *concatenation* or *merging*. The first type, concatenation, appends one data set to the end of the other, which is useful for updating data. The latter operation, merging, column-wise joins two data sets that provide different information for the same set of subjects. Merging is done via some common key variable such as patient ID, medical record ID, etc. Clearly, it involves issues such as how to deal with the unmatched and duplicated cases.

In R, the function `rbind()` is particularly useful for concatenating two data frames. For merging, `cbind()` does the basic merging while `merge()` does more advanced merging by offering more options. You may check out its options by typing `?merge` in the R console window.

Sampling

In the Sampling step, we either take a sample of the dataset or partition the data into several sets. In scenarios where the data are massive and a single calculation of sample means takes minutes, one convenient approach is to analyze a random sample or subset of the data.

Moreover, for fitting certain models such as decision trees and neural networks, it is an important and common practice to split data into three sets: *the training or learning set*, *the validation set*, and *the test set*. The partitioning ratio may be 2:1:1. In these model fitting procedures, the training sample will be first used to construct a number of candidate models; the final model or a tentative best model is then selected via the validation set; finally, the developed model is deployed to the test set, supplying a basis on which competitive models (e.g., the best logistic model and the final tree structure) can be compared.

The main reason for the above-described operation is that significance testing is no longer reliable when dealing with huge data. This is because the study would be over-powered owing to the enormous sample size. As a result, nearly everything becomes statistically significant, even those scientifically negligible results. Comparatively, the idea of validation plays a more critical role in modeling massive data, with the underlying basic philosophy that "a good model or true signals

should generalize well to new data." Following this idea, we start with over-fitted models that contain both signals and noises and then remove the noises by validation. In practice, validation can be executed practically via an independent validation sample or by resorting to various resampling techniques such as v-fold cross validation, jackknife, or bootstrapping, depending on the sample size available.

Data Cleaning

Data cleaning involves identifying wrongly or inappropriately recorded values. This is always a necessary and time-consuming step in nearly all real-world data analysis projects. The input from field experts can be helpful.

The issue might be also relevant to what package is to use for analysis. For example, Note that R is case-sensitive while SAS is not. Consider the answer choices in a survey or instrument question. The answer "Not at all" may be written as "NOT AT ALL" or "Not At ALL"; they will be treated as different values in R, which would lead to erroneous subsequent analysis if those values are handled.

Data cleaning clearly requires the aid of exploratory analysis results, which vary depending on the measurement type of each variable. For categorical variables, it is useful to obtain its frequency and relative frequency tables. In R, the functions `table(), margin.table(), prop.table()` are useful for this purpose. Here is some illustration of their usage,

```
> NCLEX <- c('PASS', 'PASS', 'FAIL', 'Fail', NA, 'PASS', 'FAIL', 'PASS')
> table(NCLEX)

NCLEX
Fail FAIL PASS
   1    2    4
```

We should correct the value "Fail" to "FAIL".

```
> NCLEX[NCLEX=="Fail"] <- "FAIL"
> table(NCLEX)

NCLEX
FAIL PASS
   3    4
```

The option `useNA="ifany"` of function table() gives the frequency in missing NA category.

```
> table(NCLEX,  useNA = "ifany")

NCLEX
FAIL PASS <NA>
   3    4    1
```

It should be straightforward to inspect two-way contingency tables resulting from the following codes lines.

```
status <- gl(2, 4, labels = c("NATIVE", "TRANSFER"))
tbl0 <- table(NCLEX, status,  useNA = "ifany"); tbl0

# Marginal Tables
margin.table(tbl0, 1)
margin.table(tbl0, 2)
```


```
# Proportions
prop.table(tbl0, 1)
prop.table(tbl0, 2)
```

To identify outlying or irregular values for a continuous variable, it is convenient to use the box-plot besides other means. Box-plot is also called the five-number summary plot, which gives the minimum, the first quartile Q_1, the median, the third-quartile Q_3, and the maximum. For an example, we simulate 100 observations from the exponential distribution with mean 10.

```
x <- rexp(100, rate=.1) # Note that mean = 1/rate
boxplot(x,boxwex=0.15,ylab='x', col="orange")
rug(jitter(x),side=2)
abline(h=mean(x,na.rm=T),lty=2)
```

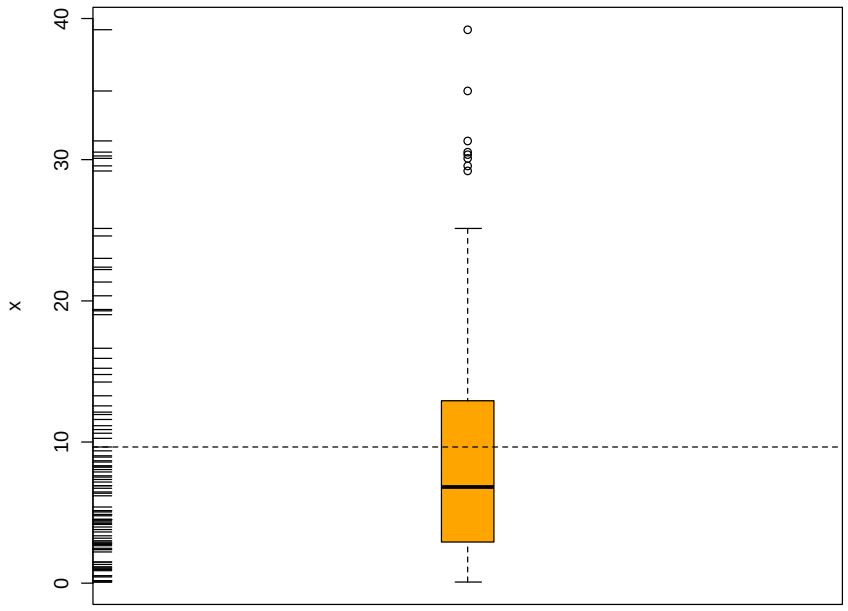

Figure 1: Box-Plot for data from Exponential (10).

Box plots provide a quick summarization of some key properties of the variable distribution. The limits of the box are Q_1 and Q_3. This box has a horizontal line inside that represents the median value of the variable. The height of the box equals to be the interquartile range IQR, which is one measure of variation or spread.

The small horizontal dash (called whisker) above the box is the largest observation that is less or equal to $\left(Q_3 + 1.5 \times \text{IQR}\right)$, while the small horizontal dash below the box is the smallest observation that is greater than or equal to $\left(Q_1 - 1.5 \times \text{IQR}\right)$. The circles below or above these whiskers represent observations that are unusually low (or high) compared to all others, and are usually considered

outliers. Box plots supply information regarding not only the central value and spread of the variable but also on eventual outliers. In addition, the `abline()` function to draw an horizontal line at the mean value of the variable.

To identify erroneous observations, another important way is to set all sorts of *filtering* conditions based on the study context. For example, the SON admission data have two binary variables indicating a student's application status (whether he/she got admitted into the nursing program) and graduation status (whether he/she successfully graduated from the nursing program).
It is impossible that a student graduated without being admitted. As another example, you may check to find out that there are 52 students who had "COLL1" different from UAB but were specified as UAB-Native. To explain this, these students only took less than 12 credit hours or less than 3 courses from other colleges before they moved to UAB. Thus it is reasonable to consider them as UAB native rather than transferred. With common senses and good understanding of subject matter, we can identify or check for mistakenly recorded values.

Occasionally, you might want to rename variables in a dataframe, which can be performed using the following function. In the following example, we renamed columns "last" to "Last.Name" and renamed "first" to "First.Name".

```
rename <- function(dat, oldname, newname) {
    if (length(oldname) != length(newname))
        stop("Oldname and newname do not have same length!!")
    for (j in 1:length(oldname)) {
        names(dat)[names(dat)==oldname[j]]<- newname[j]
    }
    return(dat)
}
dat <- rename(dat, oldname=c("last", "First"),
       newname=c("Last.Name", "First.Name"))
```

Missing Values

Missing values are common in data. The way of denoting missing values varies among computing packages. It is important to figure out how missing values are denoted and whether it is different for categorical variables and continuous variables when using a computing package. In SAS, a missing value is represented by a single blank enclosed in quotes (' ') for categorical variables and denoted by a single period (.) for numerical variable. In R, a missing value is always represented by NA.

It is important to make sure that missing values in the original data file are all read correctly as missing values in R; otherwise, it will take hassles to handle those mislabeled values. The `na.string=` option in `read.table()` function, for example, can help do the job. For example, in the SON admission data file, both blank ('') and 'NA" are used to denote missing values.

```
applicants <- read.csv("applicants.csv", header=T, na.strings = c("", "NA"))
```

The following program calculates the missing percentage for every column or variable in the dataframe dat. To be safe, we also count number of values that are blank or denoted by string "NA". Also output from this program is whether the variable takes character or numeric value and home many distinct levels or values it has.

```
vnames <- colnames(dat); vnames
```

```
n <- nrow(dat)
out <- NULL
for (j in 1: ncol(dat)){
    vname <- colnames(dat)[j]
    x <- as.vector(dat[,j])
    n1 <- sum(is.na(x), na.rm=T)
    n2 <- sum(x=="NA", na.rm=T)
    n3 <- sum(x=="", na.rm=T)
    nmiss <- n1 + n2 + n3
    ncomplete <- n-nmiss
    out <- rbind(out, c(col.number=j, vname=vname,
      mode=mode(x), n.levels=length(unique(x)),
      ncomplete=ncomplete, miss.perc=nmiss/n))
}
out <- as.data.frame(out)
row.names(out) <- NULL
out
write.csv(out, file = "brief-info.csv", row.names=F)
```

The above command creates tabulated results stored in a file called `brief-info.csv`, which can be found in your R working directory. To find out where is your current R working directory, try `getwd()`. The comma-separated-values (CSV) formatted file can be read into and further beautified using MS/Excel for presentation or publication. See Table 2 for example.

Before going into other missing value related issues, it is important to clarify on this. Missing values could occur to either the outcome or the predictor variables or both. Statistical missing value research often focuses on the missing outcome measurements in a longitudinal setting. For example, in a longitudinal depression study we are unable to get the depression score at a particular time point for a patient due to dropout, death, lost follow-up, missed appointment, sensitivity of the question, and other reasons. Thus missing values are also called nonresponses. However, in data mining, we are generally concerned about the latter case – missing predictor values.

In statistical approaches to missing value problems, the missing mechanism, i.e., how missing values are formed, is critical. There are generally three categories of missing mechanism, as listed below.

i. Missing Completely at Random (MCAR) when the nonresponse is independent of both observed and unobserved data.
ii. Missing at Random (MAR) when, conditional on observed values, the nonresponse is independent of unobserved data.
iii. Missing Not at Random (MNAR) when the missing mechanism depends on the missed value and such dependence does not vanish given or conditional on observed data. For example, a patient's depression score is missing because he/she was too depressed at the time of observations.

MAR satisfies most practical scenarios seen in longitudinal study, in which situation techniques such as multiple imputation or likelihood methods (with ignorable missing) are commonly applied. There are ad hoc ways of determining whether the missing process is MAR or MCAR. Comparatively, Existence of MNAR is mostly judged by the field expert and handled by two Bayesian approach oriented methods – the selection models (Little and Rubin, 1987) or pattern mixture models (PPM; Little, 1993, 1994).

Since most of the time data miners deal with large independent data and are only concerned about missing on predictors, the multiple imputation method has not found many applications yet (perhaps due to increased computation cost). However, the three types of missing mechanism could be of


Table 2: Missing value information for SON Admission data (Spring 2008 – Fall 2010).

Column Number	Variable Name	Mode	Number of Levels	Number Complete Cases	Missing Percentage
1	LAST.ACT	character	10	670	32.05%
2	Program	character	2	986	0.00%
3	SEM	character	2	986	0.00%
4	YR	numeric	3	986	0.00%
5	INITIAL.TERM.ADM	numeric	6	986	0.00%
6	Last	logical	1	0	100.00%
7	First	logical	1	0	100.00%
8	BooNumber	character	966	966	2.03%
9	COLL1	character	133	823	16.53%
10	PNUR.GPA	numeric	174	985	0.10%
11	CUM.GPA	numeric	156	986	0.00%
12	Mth.Yr	character	9	645	34.58%
13	TERMS.TO.GRADUATE	numeric	6	286	70.99%
14	GRAD.GPA.1	numeric	120	290	70.59%
15	NATIVE.UAB	character	2	986	0.00%
16	APL.STATUS	character	2	986	0.00%
17	Current.Student.Status	character	3	986	0.00%
18	NCLEX.1	character	3	173	82.45%
19	NUR.HONORS	numeric	2	986	0.00%
20	FAILURES	numeric	17	712	27.79%
21	WITHDRAWALS	numeric	18	711	27.89%
22	REPEATS	numeric	10	667	32.35%
23	FRIST.GENERATION	character	3	969	1.72%
24	Race	character	4	986	0.00%
25	Gender	character	3	984	0.20%
26	college.level	character	3	846	14.20%
27	GRP	character	4	846	14.20%
28	negative.events.UAB	numeric	24	986	0.00%
29	any.negative.UAB	numeric	2	986	0.00%
30	diff.Grad.GPA	numeric	109	290	70.59%
31	GPA.Grad.drop	numeric	3	290	70.59%
32	five.terms.grad	numeric	3	286	70.99%

partial interest in data mining practice, despite the fact that MCAR is often assumed. For this reason, missing value indicator and missing value pattern (MVP) are created and treated as predictor to address issues in non-MCAR situations to some extent.

There are several ways of dealing with missing value on predictors in data mining, depending on the missing rate and missing mechanisms among other factors.

1. Checked and filled – Some missing values can be filled by going to their original sources or via other ways. For example, variable INITIAL TERM ADM indicates which semester of which year a student was admitted into the nursing program, containing a number of missing

values. After some inspection, it can be found that this information can be equivalently gained by joining the other two variables SEM and YR.

2. Removed. When the missing percentage of a variable is really small (say less than 2%) or really high (say, over 98%), it is perhaps a reasonable action by simply removing the variable from modeling procedures.

3. Imputed. When the missing rate is moderately low, missing values can be replaced by some derived values or categories. In this approach, all the missing values on one variable can be replaced by, e.g., simply by its mean, mode, or with its predicted values after fitting a model. For example, if ACT score contains missing values, we build a model for ACT, treating ACT as response and all other variables (excluding the true target or outcome) as predictors using the completed cases that do not have missing values on ACT. Then the fitted model is used to supply fitted values for the missing ACT scores. This way of imputation is called s*imple imputation*, as opposed *to multiple imputation* (MI).

The idea of MI can be briefly described as follows. The MI approach contains several Monte Carlo runs. Within each run, the missing values are imputed by a simulated value from its distribution. Each of the complete datasets after imputation is analyzed by standard methods, and the estimation results from all runs are combined to produce estimates and confidence intervals that incorporate missing-data uncertainty, a method proposed by Rubin (1987) and referred to as "repeated imputation" inference. MI generally assumes MAR; but it might not be appealing in mining massive data due to the computational cost.

It is noteworthy that imputation might introduce into inference and analysis results additional biases that are hard to discover. It is also worth noting that some methods such as trees can automatically handle missing values with the aid of surrogate splits, a concept that we shall discuss further in the later study of tree methods. In other words, one does not have to do anything for missing values when fitting trees in some statistical packages such as SAS Enterprise Miner and CART. The R package `rpart` provides the same functionality.

Data Reduction and Dimension Reduction

The amount of data can be reduced in both size (number of rows or observations) and dimension (number of variables or columns) by eliminating redundant observations or features, aggregating appropriately, defining linear combinations via principal components analysis, clustering, or variable screening, for instance.

Detecting and deleting redundant rows is a necessary step in data cleaning. *Redundant rows* or observations are mis-recorded information and could be identified using the ID variable. For example, one may compare to see if the total of rows is the same as the total number of distinct IDs.

```
c(nrow(dat), length(sort(unique(dat$ID)))
```

Redundant features or columns are variables that are exactly the same or almost the same as other variables or are perfect (or near to perfect) linear combinations of other features. Redundant features can be identified by computing correlation matrix among features and the variance inflation factor (VIF), a common measure of multi-colinearity in linear regression. In R, the function `vif()` in package "`design`" computes VIFs from the estimated covariance matrix of parameter estimates.

For spatially oriented data, *data aggregation* is important to reduce observations. In this approach, data are first partitioned into many small pieces or cubes using univariate grids for example and the averaged info is computed for each piece or cube. The subsequent analysis will be based on the summarized information, instead of the original data. In this way, the total number of rows is equal to number of pieces or cubes. At the same time, the resultant analyses are of weighted version, where the weights are developed from the frequency of observations falling into each piece or cube.

Principal components analysis seeks orthogonal (or independent) linear combinations that explain the variation among original features. In predictive modeling, it would be helpful for some modeling methods (e.g., neural networks) to use the first *m* principal components to replace the *p* original variables without loss of much information, where m is much smaller than p. In decision trees, some authors also consider splitting on principal components, which, however, complicates interpretation.

Clustering is formally referred to forming clusters of observations. But here we are talking about clustering variables. There are many different clustering methods. It is convenient to consider hierarchical clustering. To do so, a proximity matrix among variables is desired and their correlation matrix is commonly used. Hierarchical clustering with average/complete/single linkage can then be applied smoothly. Once variable clusters are formed, one may select one or two from each cluster to represent the entire cluster so that dimension reduction can be achieved. This above-described method can be seen in a few textbooks. Nevertheless, I see neither its applications often, nor good reasons for doing so.

Variable screening is a critical step in data mining. Basically we remove or exclude from further consideration those variables that we really think are not very useful in predicting the outcome. There are two groups of methods for variable screening: (1) The first group is based on bivariate association exploration or simple regression models. For example, when the response or outcome is continuous, one may fit a simple linear regression model with one predictor included each time and report the resultant p-value. Then apply a threshold, e.g., 0.25, to eliminate those variables that are not strongly associated with the outcome. This kind of simple screening methods has been frequently used and recently studied by Fan and Lv (*JRSSB*, 2008) who established the sure screening property. (2) The second group of methods relates to the concept of variable importance or variable relevance. Variable importance or relevance is based on collectively computing the predictive ability of a variable under the joint influence of other variables. Popular methods for computing variable importance are random forests and RELIEF. We shall come back to these methods in later chapters.

Exploratory Data Analysis

Exploratory data analysis provides preliminary looks from different angles at and supplies important insights into the data. The specific tools used in EDA depend on types of variables. At the same time, there are both graphical and numerical methods available.

Suppose that we not only have an outcome variable, but also have a binary treatment variable for which we would like to assess its effect on the outcome. The following two functions are written to provide a variety of summary statistics for continuous and categorical variables respectively.

The first function `describe.interval()` computes the first four moments or cumulants for each variable (i.e., mean, sd, skewness, and kurtosis) for the whole data set and separately for either treatment group. Also, info on two-sample *t* test and the nonparametric alternative – Wilcoxon rank-sum test is also supplied. The arguments `cols=` asks for the column positions in the data set for

continuous variables and the argument `by.col=` asks for the column position of the treatment variable.

```
describe.interval <- function(dat, cols=1:ncol(dat), by.col=0, wilcox.exact=F) {
 library("moments")
 n <- nrow(dat)
 vnames <- colnames(dat)
 result <- NULL
 if (by.col !=0) {
       by.var <- dat[, by.col]
       if (length(unique(by.var))!=2) stop("The BY variable is not binary?!!")
       value1 <- sort(unique(by.var))[2]
       print(value1)
  }
  for (j in cols) {
    print("====================================")
    print(cbind(j, vnames[j]))
              x <- dat[,j]
              # Obtain Missing Info
    n1 <- sum(is.na(x), na.rm=T)
    n2 <- sum(x=="NA", na.rm=T)
    n3 <- sum(x=="", na.rm=T)
    # print(cbind(n1, n2, n3))
    nmiss <- n1 + n2 + n3
    ncomplete <- n-nmiss

    # Normality Checking
    varname <- vnames[j]
    pvalue.norm <- shapiro.test(x)$p.value
    out <- c(varname=varname, ncomplete=ncomplete, miss.perc=nmiss/n,
          mu=mean(x, na.rm=T), sd=sd(x, na.rm=T), kurto=kurtosis(x, na.rm=T),
          skew=skewness(x, na.rm=T), pvalue.norm=pvalue.norm)
    if (by.col !=0) {
       by.var <- dat[, by.col]
       if (length(unique(by.var))!=2) stop("The BY variable is not binary?!!")
       for (k in unique(by.var)){
              x1 <- x[by.var==k]
              # n1 <- length(complete.cases(x1))    # DOES NOT WORK
              n.1 <- sum(is.na(x1), na.rm=T)
              n.2 <- sum(x1=="NA", na.rm=T)
              n.3 <- sum(x1=="", na.rm=T)
              # print(cbind(var=j, n=length(x1), n.1, n.2, n.3))
              n.miss <- n.1 + n.2 + n.3
              n.complete <- length(x1) -n.miss
              out1 <- c(n.complete, mean(x1, na.rm=T), sd(x1, na.rm=T),
                     kurtosis(x1, na.rm=T), skewness(x1, na.rm=T))
              names(out1) <- paste(c("n.", "mean.", "sd.", "kurto.", "skew."),
                     k, sep="")
              out <- c(out, out1)
       }
       # THE TWO-SAMPLE T TEST
       ttest <- t.test(x ~ by.var, var.equal=T, na.action=na.omit)
       out <- c(out, c(ttest$statistic,  ttest$parameter, p.value=ttest$p.value))

       # NONPARAMETRIC TEST - WILCOXON RANK-SUM TEST
       wilcox <- wilcox.test(x ~ by.var, exact=wilcox.exact, correct=T,
              na.action=na.omit)
       out <- c(out, c(wilcox$statistic, pvalue.wilcoxon=wilcox$p.value))
       }
       result <- rbind(result, out)
    }
```


```
    result <- as.data.frame(result)
    row.names(result) <- NULL
    return(result)
}

cols.interval <- c(10, 11, 20:22, 28)
col.trt <- 16
subset0 <- dat$Program == "BASIC" &  dat$NATIVE.UAB =="NO"
summary0 <- describe.interval(dat[subset0,], cols=cols.interval, by.col=col.trt,
wilcox.exact=F)
summary0
write.csv(summary0, file = "sum-interval-2.csv", row.names=F)
```

Exemplary results from running this function are shown in Table 3.

Table 3: Summary of Continuous Variables for UAB Native Applicants who were Seeking **BASIC** degree in Nursing

		Missing			ADMITTED			DENIED			Two-Sample	Wilcoxon
Variable	n	%	Mean	SD	n	Mean	SD	n	Mean	SD	t Test	Rank-Sum
PNUR.GPA	303	0.33%	3.282	0.354	262	3.338	0.344	41	2.923	0.145	0.0000	0.0000
CUM.GPA	304	0.00%	3.287	0.518	263	3.359	0.449	41	2.830	0.678	0.0000	0.0000
FAILURES	211	30.59%	0.621	1.386	183	0.486	1.114	28	1.500	2.380	0.0003	0.0022
WITHDRAWALS	211	30.59%	0.730	1.726	183	0.585	1.360	28	1.679	3.104	0.0016	0.0736
REPEATS	191	37.17%	0.183	0.601	164	0.122	0.411	27	0.556	1.188	0.0004	0.0084
negative.events.UAB	304	0.00%	1.053	2.485	263	0.821	1.960	41	2.537	4.359	0.0000	0.0197

The function `describe.cat()` computes various frequencies for each level of each categorical variable. Pearson's chi-squared test and Fisher's exact test are also made available for detecting their association with the treatment variable. Put in another way, we want to see if there is any imbalance in each covariate between the two treatment groups.

```
describe.cat <- function(dat, cols=1:ncol(dat), by.col=0) {
       vnames <- colnames(dat)
       result <- NULL
       if (by.col !=0) {
              by.var <- dat[, by.col]
              if (length(unique(by.var))!=2) stop("The BY variable
                     is not binary?!!")
              value1 <- sort(unique(by.var))[2]
              print(value1)
       }
       for (j in cols) {
              x <- dat[,j]
              varname <- vnames[j]
              # Obtain Missing Info
              n1 <- sum(is.na(x), na.rm=T)
              n2 <- sum(x=="NA", na.rm=T)
              n3 <- sum(x=="", na.rm=T)
              nmiss <- n1 + n2 + n3
              n <- length(x)
              ncomplete <- n-nmiss
              n1 <- sum(!is.na(x) & !is.element(x, c("NA", ""))
                     & by.var==value1, na.rm=T)
              result <- rbind(result, c(varname=varname, ncomplete,
                     n1, n1/ncomplete))
```

```
            # Chi-Squared Test and Fisher's Exact Test
            xlevels <- sort(unique(x))
            print(cbind(j, varname))
            print(xlevels)
            chi2.stat <- pvalue.chi2 <- fisher <- NA
            if (length(xlevels) >1) {
              print(table(x, by.var, useNA="ifany"))
              chi2 <- chisq.test(x, by.var)
              fisher <- fisher.test(x, by.var, hybrid =T)$p.value
              chi2.stat <- chi2$statistic
              pvalue.chi2 <- chi2$p.value
            }
            result <- rbind(result, c(varname=varname, chi2.stat=chi2.stat,
                  pvalue.chi2=pvalue.chi2, fisher=fisher))
            for (i in xlevels) {
                  n <- sum(x==i, na.rm=T)
                  n1 <- sum(x==i & by.var==value1, na.rm=T)
                  prop1 <- n1/n
                  out <- c(level=i, n=n, n1=n1, prop1=prop1)
                  result <- rbind(result, out)
            }
      }
      result <- as.data.frame(result)
      rownames(result) <- NULL
      colnames(result) <- c("Variable", "Total Count", paste("Num of ",
            value1, sep=""),
       paste("Percent of ", value1, sep=""))
      return(result)
}


cols.cat <- c(2, 15, 26, 27)
col.trt <- 16
summary1 <- describe.cat(dat, cols=cols.cat, by=col.trt)
write.csv(summary1, file = "sum1.csv", row.names=F)
```

Exemplary results from running this function are shown in Table 4.

Graphical tools include univariate graphs such as histograms, boxplots, bar plots and bivariate association plots such as scatterplots, parallel boxplots. Illustration of these techniques will be made available in appropriate places in our later study.

Variable Transformation

We transform variables for several purposes. First, sometimes we may want to standardize variables or put variable values onto the same range, e.g., [0, 1]. Typically, this is done by computing the z-scores

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

Table 4: Frequency Distribution with **Native UAB** Students Seeking **BASIC Degree** in Nursing.

		(a) UAB Native Applicants					(b) Transfer Students				
		Total	DENIED		Chi	Fisher's	Total	DENIED		Chi	Fisher's
Variable		Count	Count	Percent	Squared	Exact	Count	Count	Percent	Squared	Exact
SEM		304	41	13.49%	0.2262	0.2219	459	153	33.33%	0.0000	0.0000
	FALL	193	30	15.54%			256	116	45.31%		
	SPRING	111	11	9.91%			203	37	18.23%		
YR		304	41	13.49%	0.8659	0.8656	459	153	33.33%	0.0016	0.0013
	2008	93	13	13.98%			124	26	20.97%		
	2009	115	14	12.17%			147	51	34.69%		
	2010	96	14	14.58%			188	76	40.43%		
NUR.HONORS		304	41	13.49%	0.0512	0.0200	459	153	33.33%	0.0385	0.0269
	0	275	41	14.91%			443	152	34.31%		
	1	29	0	0.00%			16	1	6.25%		
FRIST.GENERATION		297	41	13.80%	0.4186	0.5298	456	152	33.33%	0.0029	0.0013
	NA	7	0	0.00%			3	1	33.33%		
	NO	183	23	12.57%			284	78	27.46%		
	YES	114	18	15.79%			172	74	43.02%		
Gender		304	41	13.49%	0.6539	0.6468	458	153	33.41%	0.9030	1.0000
	FEMALE	256	36	14.06%			386	129	33.42%		
	MALE	48	5	10.42%			72	24	33.33%		
any.negative.UAB		304	41	13.49%	0.0971	0.0953	459	153	33.33%	0.0000	0.0000
	0	215	24	11.16%			265	64	24.15%		
	1	89	17	19.10%			194	89	45.88%		

Where $(\bar{x}_j, s_j)$ denote the mean and standard deviation for the j-th variable X_j or rescaling by transformation

$$x'_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)}.$$

Such a transformation is an important for, e.g., normalizing the distribution, fitting neural network models or applying the LASSO typed variable selection technique.

Furthermore, it is important to derive new variables that carry better interpretation. For example, BMI is computed as

$$\text{BMI} = \frac{\text{Weight (kg)}}{[\text{Height (m)}]^2}.$$

In the UAB SON admission project, it would be interesting to define an indicator variable indicating whether a student's GPA has dropped from admission to graduation.

Summary

In this chapter, we have gone over a number of issues in preprocessing data in real world applications. It is important to note that data preparation and the subsequent EDA and modeling analyses sometimes are an iterative process. For example, more problems could be identified in EDA or emerge in modeling. We have to go back to perform more data modifications and then continue with modeling. Furthermore, the specific variable screening technique that should be used depends on what kind of modeling procedure, e.g., decision tree or logistic regression, is to be employed.

The messy nature of data, plus other practical requirements in modeling, encountered in the data mining practice calls for a need of "*off-the-shelf*" mining tools. According to Hastie, Tibshirani, and Friedman (HTF, 2008), an "off-the-shelf" method is one that can be directly applied to the data without requiring a great deal of time consuming data preprocessing or careful tuning of the learning procedure. After reviewing a number of available tools, they pointed out that, "of all the well-known learning methods, decision trees come closest to meeting the requirements for serving as an off-the-shelf procedure for data mining."

References:

- Fan, J. and Lv, J. (2008). Sure independence screening for ultra-high dimensional feature space (with discussion). *Journal of Royal Statistical Society B*, **36**: 849-911.
- Hastie, T., Tibshirani, R., and Friedman, J. H. (2008), *Elements of Statistical Learning*, 2nd Edition, Chapman and Hall. ISBN-13: 978-0387848570
- Little, R. J. A. and Rubin, D. B. (1987) *Statistical Analysis with Missing Data*. J. Wiley & Sons, New York.
- Pyle, D. (1999). *Data Preparation for Data Mining*. Morgan Kaufmann: San Francisco, CA.
- Refaat, M. (2006). *Data Preparation for Data Mining Using SAS*. SAS Institute. ISBN 10: 012-373577-7.
- Rubin, D.B. (1987) *Multiple Imputation for Nonresponse in Surveys*. J. Wiley & Sons, New York.
- Svolba, G. (2007). *Data Preparation for Analytics using SAS*. SAS Institute. ISBN-10: 159-994047-7.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso *Journal of Royal Statistical Society B*, **58**: 267-288.