Practical Issues in Data Analysis

Ruoqing Zhu

Last Updated: September 26, 2024

Contents

Data Pre-processing

When we perform an analysis, data pre-processing is an important step. The data may contain missing values, and have highly skewed or non-standard distribution, etc. Some of these issues always need to be addressed, while some others depends on your particular goal, and also your experience with similar/other scientific problems. We will discuss some of them and provide some simple solutions. However, there is a much broader literature on these issues, so feel free to read and explore yourself.

Missing Data

This is a very common issue in practice. For a comprehensive discussion of statistical issues under this topic, I recommend the book *Statistical Analysis With Missing Data* by Little and Rubin. However, we will only discuss some simple solutions. Please note that most R functions will automatically ignore missing observations, so if you have a lot of missing entries, then you could potentially miss a lot of information. Let's use an artificial example, which is derived from the prostate data. You should consider removing the outcome variable when you perform the imputation. This is because for prediction purpose, if you have missing values in your testing data, you cannot utilize the outcome variable to impute them. Some statistical method still utilize the outcome because they only care about the parameter estimate, but not prediction.

```
library(ElemStatLearn)
data(prostate)
n = nrow(prostate)

# construct missing values
# remove the train/test label and also the outcome variable
prostate_miss = prostate[, 1:8]

# randomly set 10 observations to be random
set.seed(1)
prostate_miss$lbph[sample(1:n, size = 10)] = NA
prostate_miss$lcp[sample(1:n, size = 20)] = NA
```

Now we have this data. The first step to check for any missing values (usually `NA` in `R`) is to use the `is.na()` function. We can see that they should match our construction.

```
  as.matrix(colSums(is.na(prostate_miss)))
##         [,1]
## lcavol     0
## lweight    0
## age        0
## lbph      10
## svi        0
## lcp       20
## gleason    0
## pgg45      0
```

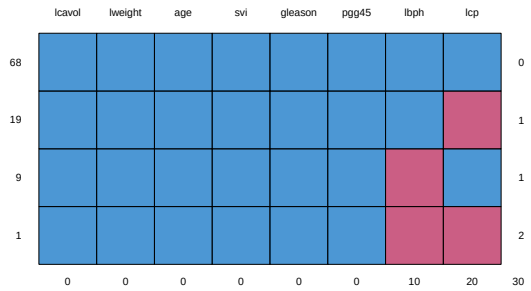
There are some popular approaches:

- **Discard any observations with missing values**. This is common, but usually not preferred unless there are only a few observations with missing values. However, removing observations with missing outcome variable is very common.
- **Discard the variable with missing values**. This is also not preferred, unless most of observations are missing this variable.
- **Impute with mean**. Calculate the mean value from the non-missing ones of this variable, and replace the missing entries with the mean.
- **Random Imputation**. Randomly draw values from the non-missing ones to replace the missing ones.
- **Regression Imputation**. The idea is to use other covariates to predict this variable with missing value. There are two common types: Deterministic and Stochastic. A deterministic regression imputation will just replace the missing ones with the predicted value from using other variables as covariates, while a Stochastic regression imputation will also consider the random error terms of such regression and add additional variation to the imputed values.

The `mice` package is a popular one for performing imputation. The methods implemented there can be far more advanced than what we introduced in this course, but we will just use some simple features. Use the `mice()` function to perform this. Note that the argument `m` specifies the number of times you want to perform the imputation. And we only need to perform this once with `maxit = 1`. You can explore other features yourself.

```
  library(mice)

  # This functions shows the missing pattern
  md.pattern(prostate_miss)
```



```
##    lcavol lweight age svi gleason pgg45 lbph lcp
## 68      1       1   1   1       1     1    1   1  0
## 19      1       1   1   1       1     1    1   0  1
## 9       1       1   1   1       1     1    0   1  1
## 1       1       1   1   1       1     1    0   0  2
##         0       0   0   0       0     0   10  20 30
```

```
  # Imputation with mean value
  imp <- mice(prostate_miss, method = "mean", m = 1, maxit = 1)
##
##  iter imp variable
##   1   1  lbph  lcp

  # Imputation with randomly sampled value
  imp <- mice(prostate_miss, method = "sample", m = 1, maxit = 1)
##
##  iter imp variable
##   1   1  lbph  lcp

  # Deterministic regression imputation
  imp <- mice(prostate_miss, method = "norm.predict", m = 1, maxit = 1)
##
##  iter imp variable
##   1   1  lbph  lcp

  # Stochastic regression imputation
  imp <- mice(prostate_miss, method = "norm.nob", m = 1, maxit = 1)
##
##  iter imp variable
##   1   1  lbph  lcp

  # after performing the imputation, use this function to extract the imputed data
  prostate_imp <- complete(imp)

  # we can check the missingness
  any(is.na(prostate_imp))
## [1] FALSE
```

For further reading, you could look at page 77 of the documentation of the `mice` package. There are some Vignettes that provides use examples.

Variable Transformations

When covariates or outcome variables are highly skewed or have non-standard distribution, we may consider performing variable transformation. There are some motivations for performing them and two of them are most obvious:

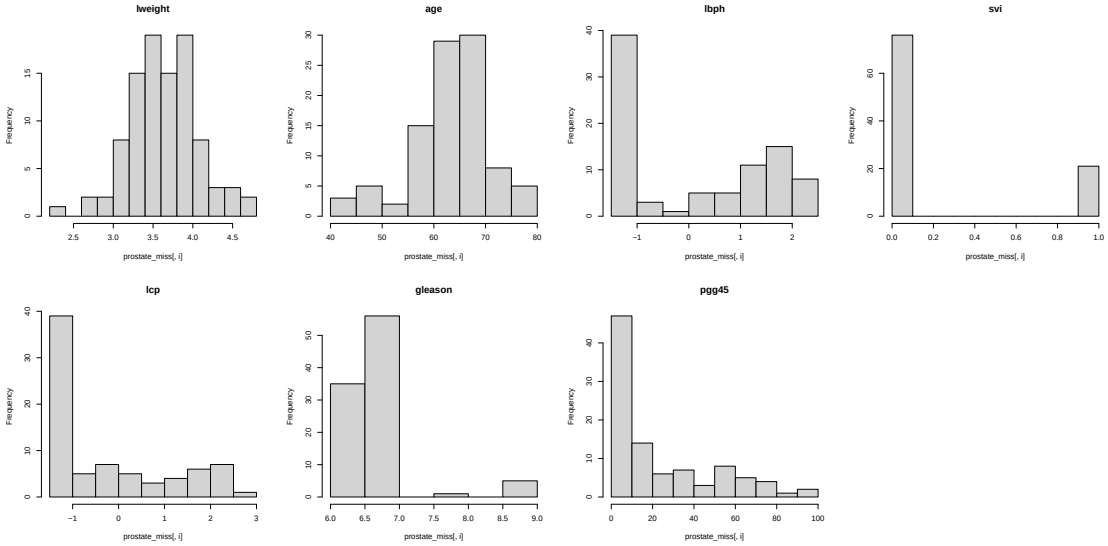
- When fitting a linear model, the linear trend will not likely hold when the covariate has extremely large values. Performing variable transformation may keep a variable at reasonable range so that linear model approximate the data pattern better.
- For many machine learning algorithms, having normally distributed or uniformly distributed (bounded) data makes the model more stable.

However, there is NO guarantee that performing univariate transformations would improve your model fitting. I will always recommend to first perform model fittings on the original variables and then explore potential improvements using variable transformation. For our course, the ultimate decision would still be the prediction performance, while in statistics, sometimes the goal is to estimate the parameters in reliable way. Nonetheless, we will discuss some approaches.

- Discretization
- Univariate functions
- Quantile transformation

Before we proceed with these methods, we could utilize univariate histograms to visualize their distributions and identify anything unusual.

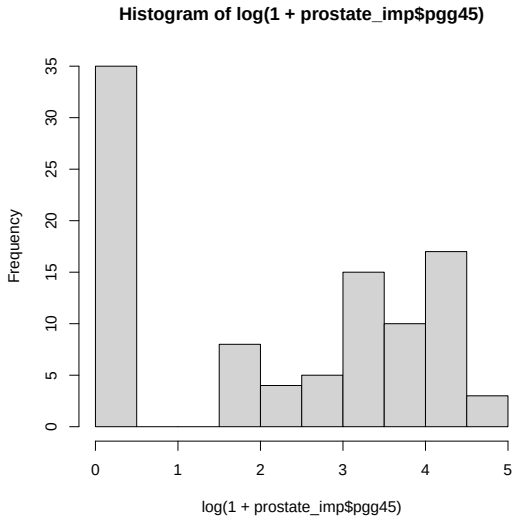
```
par(mfrow = c(2, 4))
for (i in 2:ncol(prostate_imp))
  hist(prostate_miss[,i], breaks = 10, main = colnames(prostate_imp)[i])
```



For examples:

- Discritization could be done to variables that has two mode, such as `lbph`.
- `pgg45` have pretty long tail towards the right hand side and could benefit from a log transformation. Note that this is only possible when the variable is non-negative. If it contains zero, we can use $\log(1+x)$.

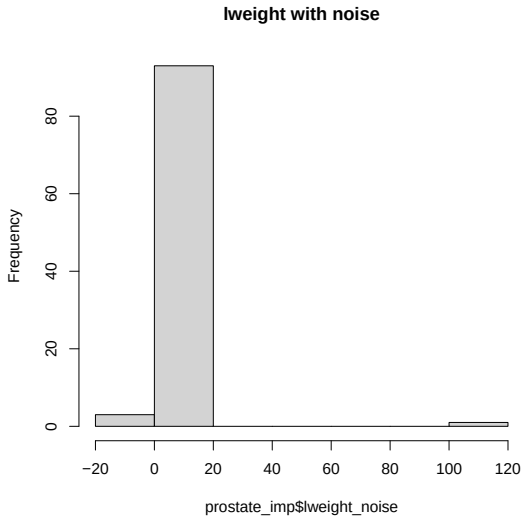
```
hist(log(1 + prostate_imp$pgg45))
```



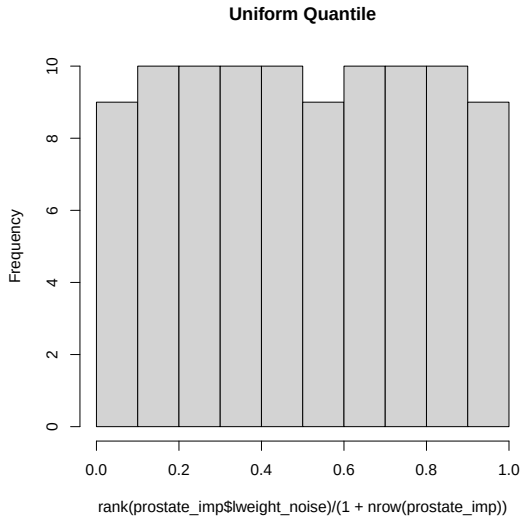
After performing this transformation, it seems that the variable has two clusters, one is zero and the other one ranges around 1.5 to 5. This gets rid of long tails, however, you have to perform the regression model to see if it benefits.

- Quantile transformation can be useful when a variable contains heavy tails on both sides. The essential idea is to use the rank of the value among all observations instead of using the raw value. This would transform it to a uniformly distributed variable. We can also further transform this rank value to quantiles of other, better distributions such as Gaussian. For example, consider the following contaminated data with large outliers:

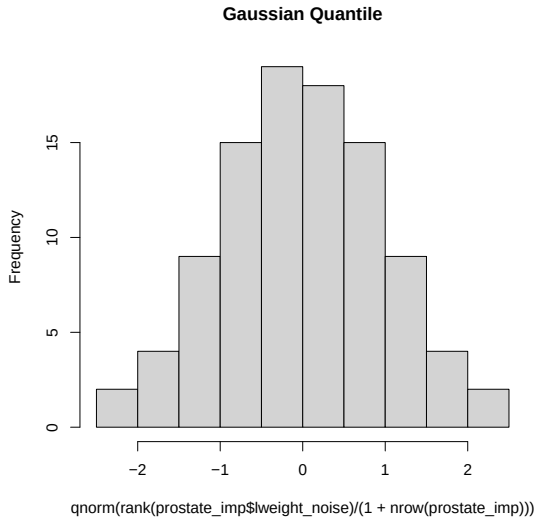
```r
# set up a contamination
set.seed(1)
prostate_imp$lweight_noise = prostate_imp$lweight
prostate_imp$lweight_noise[1:5] = 10 * rt(5, 1)
hist(prostate_imp$lweight_noise, main = "lweight with noise")
```



```r
# perform quantile transformation
# because rank ranges from 1 to n, we can transform them to (0, 1)
hist(rank(prostate_imp$lweight_noise) / (1 + nrow(prostate_imp)),
     main = "Uniform Quantile")
```




```r
# this can be further transformed into Gaussian quantiles
hist(qnorm(rank(prostate_imp$lweight_noise) / (1 + nrow(prostate_imp))),
     main = "Gaussian Quantile")
```



However, this does not seem to be necessary for the `lweight` variable because it does not really contain heavy tails. In practice, you will need to make the judgement yourself.