Data PreProcessing with R

Normalising Data

Let us examine how we can generate summary statistics for a variable. You will also see how to carry out Min-Max normalisation and Z Score Standardisation using R.

```
#Input dataset numeric weather into a dataframe (from last week)

weathernum <-read.table(file ="c:/weathernumeric.txt", stringsAsFactors=FALSE, sep =",", header=TRUE)

#Show the first 10 records and all the columns
weathernum[1:10,]

#Examine the 5 number summary statistics with mean for temperature and humidity. From this calculate the interquartile range
summary(weathernum$temperature)
summary(weathernum$humidity)
```

For you to do!

- By hand, calculate the IQR fro these dimesnsions. Are there any outliers for these variables?
- Use the IQR method to detect outliers.

Sol for temp

```
temp <- weathernum$temperature
IQRt <- 78.75 -69.25
OutlierMaxtemp <- 78.75 + (1.5*IQRt)
OutlierMaxtemp
[1] 93
OutlierMintemp <- 69.25 - (1.5*IQRt)
OutlierMintemp
[1] 55
sort(weathernum[,2])
 [1] 64 65 68 69 70 71 72 72 75 75 80 81 83 85
```

```
#Min-Max normalisation for the weathernum$humidity attribute

mmnorm.humidity <- (weathernum$humidity - min(weathernum$humidity))/(max(weathernum$humidity) - min(weathernum$humidity))

summary(mmnorm.humidity)

#zScore Standardisation

zscore.humidity <-(weathernum$humidity - mean(weathernum$humidity))/sd(weathernum$humidity)
summary(zscore.humidity)
```

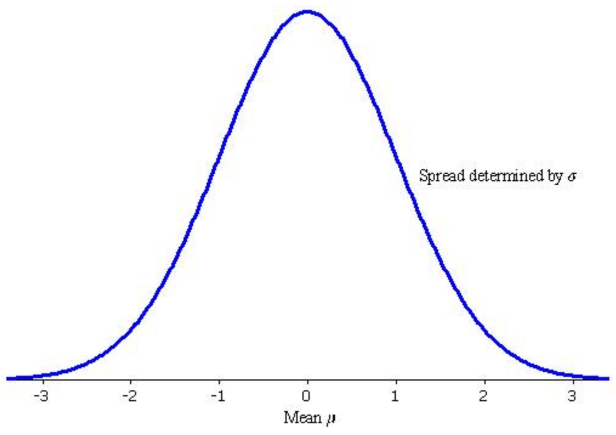
For you to do!

- What are the ranges for this variable for both normalisation methods? Do you think there are any outliers?
- Write the R formula to decimal scale weathernum$humidity attribute. Hint use nchar function. Examine the summary statistics afterwards. Note any observations

```
Sol
decScaling <- weathernum$temperature /10^nchar(weathernum$temperature)
> weathernum$temperature
 [1] 85 80 83 70 68 65 64 72 69 75 75 72 81 71
> decScaling
 [1] 0.85 0.80 0.83 0.70 0.68 0.65 0.64 0.72 0.69 0.75 0.75 0.72 0.81 0.71
```

Transforming the data in an attempt to achieve normality#

Some data mining and statistical methods require that the variables be normally distributed. The normal distribution is a continuous probability distribution commonly known as the bell curve, which is symmetric. It is centred at the mean and spread is determined by the standard deviation. Standard normal Z distribution has a mean of 0 and a standard deviation of 1.



Standard normal Z-distribution
with μ=0 and σ=1

However, variables are of often skewed to the right (+) or left (-). We can measure use the following statistic to measure the skewness of a distribution

$$Skewness = \frac{3(mean\ - median)}{standard\ deviation}$$

For perfectly symmetric data, the mean, median and mode = 0

We can transform the distribution in different ways in an attempt to reduce skewness and achieve a symmetric distribution.

```
#Read in the cars2 dataset and transform weightlbs in different ways
cars2 <- read.csv(file="c:/cars2.txt", stringsAsFactors=TRUE)

#Natural Log Transformation
natlog.weightlbs <- log(cars2$weightlbs)
natlog.weightlbs

#Square Root Transformation
sqrt.weightlbs <- sqrt(cars2$weightlbs)
sqrt.weightlbs

#Inverse Square Root  Transformation
invsqrt.weightlbs <- 1/sqrt(cars2$weightlbs)
invsqrt.weightlbs
```

For you to do!!

- **Using R calculate the skewness for the car weightlbs attribute**

```
weightlbs.skew <-(3*(mean(cars2$weightlbs) - median(cars2$weightlbs)))/sd(cars2$weightlbs)
weightlbs.skew    #positive skew
```

- **Using R find the skewness for the natural log transformation of the car weightlbs attribute**

```
natlog.weightlbs.skew <-(3*(mean(natlog.weightlbs) - median(natlog.weightlbs)))/sd(natlog.weightlbs)
natlog.weightlbs.skew #positive skew but reducing
```

- **Using R find the skewness for the square root transformation of the car weightlbs attribute**

```
sqrt.weightlbs.skew <-(3*(mean(sqrt.weightlbs) - median(sqrt.weightlbs)))/sd(sqrt.weightlbs)
sqrt.weightlbs.skew #positive skew
```

- **Using R find the skewness for the inverse square root transformation of the car weightlbs attribute**

```
invsqrt.weightlbs.skew <-(3*(mean(invsqrt.weightlbs) - median(invsqrt.weightlbs)))/sd(invsqrt.weightlbs)
invsqrt.weightlbs.skew
```

- **Now re-calculate skewness using 3 (normality) transformation methods but with car weightlbs transformed using Min-Max Normalisation**

SOl

# Min-Max normalisation

```
mmnorm.weightlbs <- (cars2$weightlbs - min(cars2$weightlbs))/(max(cars2$weightlbs) -
min(cars2$weightlbs))
mmnorm.weightlbs
```

# Min-Max normalisation Skew

```
mmnorm.weightlbs.skew <-(3*(mean(mmnorm.weightlbs) - median(mmnorm.weightlbs)))/sd(mmnorm.weightlbs)
mmnorm.weightlbs.skew    # normalisation has no  effect on the skew value
```

```
# Min-Max normalisation With NatLog Skew

mmnnatlog.weightlbs <- log(mmnorm.weightlbs)
mmnorm.weightlbs.natLogskew <-(3*(mean(mmnnatlog.weightlbs) -
median(mmnnatlog.weightlbs)))/sd(mmnnatlog.weightlbs)
mmnorm.weightlbs.natLogskew  #NaN  due to –Inf value



# Min-Max normalisation With SquareRoot Skew

mmnormsqrt.weightlbs <- sqrt(mmnorm.weightlbs)

mmnormsqrt.weightlbs

mmnormsqrt.weightlbs.skew <-(3*(mean(mmnormsqrt.weightlbs
) - median(mmnormsqrt.weightlbs )))/sd(mmnormsqrt.weightlbs)

sqrt.weightlbs.skew #positive skew # normalisation has no  effect on the skew value

# Min-Max normalisation With Inverse Square Root Transformation Skew

mmninvsqrt.weightlbs <- 1/sqrt(mmnorm.weightlbs)
mmninvsqrt.weightlbs


mmnorminvsqrt.weightlbs.skew <-(3*(mean(mmninvsqrt.weightlbs
) - median(mmninvsqrt.weightlbs)))/sd(mmninvsqrt.weightlbs)

mmnorminvsqrt.weightlbs.skew


# ? Interpret your results.  What do you concur?
```

Binning Data

Let us look at equal frequency , equal- binning and k-means clustering as binning strategies

Here are some examples

```
#Create a vector of values for binning
xdata <-c(1,11,2,1,1,2,12,11,44,2,12,1)
#get the sample size of the variable
n <- length(xdata)
#Declare number of bins and bin indicator
nbins <- 3
whichbin <- c(rep(0,n))

#Equal Frequency (equal-depth)
freq <- n/nbins
#sort the data
xsorted <- sort(xdata)
for(i in 1:nbins) {
   for(j in 1:n) {
       if ((i-1)*freq <j && j<=i*freq)
          whichbin[j] <- i
   }
}
whichbin
xsorted

#equal-width
#note bin 2 has 0 values while bin 3 has 1 value
range.xdata <- max(xdata) - min(xdata) +1
binwidth <- range.xdata/nbins
 for(i in 1:nbins) {
   for(j in 1:n) {
       if ((i-1)*binwidth < xdata[j] && xdata[j] <= (i)*binwidth)
          whichbin[j] <- i
   }
}
whichbin
xdata

#K-means clustering as a binning strategy where k = 3
kmeansclustering <- kmeans(xdata,centers=nbins)
whichbin <- kmeansclustering$cluster
whichbin

# What is the best binning strategy from above? Explain your reasoning
```

How to create a Histogram and a Scatteplot

```
#Read in the cars2 dataset if you have not done so already
cars2 <- read.csv(file="c:/cars2.txt", stringsAsFactors=TRUE)

#Create a histogram
#Set up the plot area
par(mfrow=c(1,1))

#Examine the weightlbs histogram and write down your observations
hist(cars2$weightlbs,
   breaks=30,
   xlim= c(0,5000),
   col="blue",
   border="black",
   ylim=c(0,40),
   xlab="Weight in lbs",
   ylab="Counts",
   main="Histogram of Car Weights")
box(which="plot", lty="solid", col="black")
```

FOR YOU TO DO!

Create a histogram of the variable MPG

#Use the following command before creating the weight and z score of weight histograms if you want to see them size by side.

```
par(mfrow = c(1,2))
```

REM SOL

```
summary(cars2$mpg)
hist(cars2$mpg,
  breaks=30,
  xlim= c(0,550),
  col="blue",
  border="black",
  ylim=c(0,40),
  xlab="MPG Score lbs",
  ylab="Counts",
  main="Histogram of Car MPgs")
zscore.mpg <-( cars2$mpg - mean(cars2$mpg))/sd(cars2$mpg)
summary(zscore.mpg)
hist(zscore.mpg,
  breaks=30,
  xlim= c(-4,+17),
  col="blue",
  border="black",
  ylim=c(0,40),
  xlab="MPG ZScore lbs",
  ylab="Counts",
  main="Histogram of Car MPgs (zScore)")
```

#Let us create a ScatterPlot of MPG by Weight. Examine the plot. What do you concur? Are there outliers? Would they be deemed outliers for the individual variables.

```
plot(cars2$weightlbs, cars2$mpg,
   xlim= c(0,5000),
   ylim=c(0,600),
   xlab="Weight",
   ylab="MPG",
   main="Scatter Plot of MPG by Weight",
   type ="p",
   pch=16,
   col="blue")
points(cars2$weightlbs,
    cars2$mpg,
    type="p",
       col="red")
#?Choose 2 other numeric variables and produce a scatter plot. Can you identify any outliers? Are they
correlated?
```

Create a Histogram with fitted Normal Distribution and Normal Probability Plot

# A histogram inverse square root of weight

```
invsqrt.weightlbs <- 1/sqrt(cars2$weightlbs)
 invsqrt.weightlbs

x <- rnorm(1000000,
mean=mean(invsqrt.weightlbs),
sd=sd(invsqrt.weightlbs))

par(mfrow=c(1,1))
hist(invsqrt.weightlbs,
     breaks=30,
     xlim= c(0.0125,0.0275),
     col="lightblue",
     prob= TRUE,
     border="black",
     xlab="Inverse Square Root of Weightlbs",
     ylab="Counts",
     main="Histogram of Inverse Square Root of Car Weightlbss")
lines(density(x),
      col="red")

box(which="plot",
    lty="solid",
    col="black")

lines(density(x),
      col="red")
```

# Normal Probability plot that indicates non-normality

```
 par(mfrow= c(1,1))
```

```
qqnorm(invsqrt.weightlbs,
       datax =TRUE,
       col="red",
       ylim=c(0.01,0.03),
      main ="Normal Q-Q Plot of inverse Square Root of Weightlbs")

qqline(invsqrt.weightlbs,
      col="blue",
      datax=TRUE)
```

We can also test for normality is a formal way.

Using the Shapiro-Wilks test, we can examine the p-value. The p-value tells you what the chances are that the sample comes from a normal distribution. The lower this value, the smaller the chance it comes from a normal distribution. Statisticians typically use a value of 0.05 as a cutoff. When the p-value is less than 0.05, one can conclude that the sample deviates from normality.

```
result <- shapiro.test( invsqrt.weightlbs)
```

```
result$p.value
```

A Note of transforming the data and duplicate records

If you have transformed the data using Z-Score, Min-Max Normalisation, Inverse Square Root Transform etc, you will have to detransform the data at some stage to see the actual values here is an example

Transforming the data

```
x <- cars2$weightlbs[1]; x
[1] 4209
h1 <-head(cars2$weightlbs); h1
[1] 4209 1925 3449 3761 2051 3900
```

```
#Transform x using the inverse sqrt
y <- 1/sqrt(x); y
[1] 0.01541383

z <- 1/sqrt(h1); z
[1] 0.01541383 0.02279212 0.01702760 0.01630603 0.02208092
[6] 0.01601282
```

```
# Detransform x using 1/(y)^2
detransx <- 1/y^2; detransx
[1] 4209

detransz <- 1/z^2; detransz
[1] 4209 1925 3449 3761 2051 3900
```

Finding duplicate Records in a data frame

```
#Finding Duplicate Records in a data frame
anyDuplicated(cars2)
[1] 0
duplicated(cars2)
# Note: TRUE Indicates a record which is a duplicate of a previous record
# FALSE indicates a record which is not a duplicate of a previous record

# Duplicate the first record to make a new dataset
new.cars2 <- rbind(cars2,cars2[1,])


anyDuplicated(new.cars2)
[1] 262

# The 262nd record is duplicated
duplicated(new.cars2)
#TRUE Indicates a record which is a duplicate of a previous record
```

EXERCISES

Using the churn dataset

1. Explore whether there are any missing values for any of the variables
2. Use a graph to visually determine whether there are any outliers among the number of calls to customer service
3. Identify the range of customer service calls that should be considered outliers using:
   i. Z-Score Method
   ii. IQR
4. Transform the day minutes attribute using Z-Score standardisation
5. Work with skewness as follows.
   i. Calculate the skewness of day minutes
   ii. Then calculate the skewness of the Z-score standardised day minutes. Comment
   iii. Based on the skewnes value, would you consider day minutes to be skewed or nearly perfectly symmetric