Lab 2 Multivariate Data Screening

# Set up R session

## Install and load packages.

Note that for a RMarkdown document to knit, you need to comment out these ‘install.packages’ lines of code before knitting. This is because you cannot knit when your document needs to connect to an external web server. Right now these lines of code are commented out.

# install.packages("mvnormtest")  
# install.packages("MVN")  
# install.packages("MVA")  
# install.packages("psych")  
# install.packages("Hmisc")  
# install.packages("vegan")  
# install.packages("StatMatch")  
# install.packages("MASS")  
# install.packages("raster")  
# install.packages("cluster")

library(mvnormtest)  
library(MVN)  
library(MVA)  
library(psych)  
library(Hmisc)  
library(vegan)  
library(StatMatch)  
library(MASS)  
library(raster)  
library(cluster)

## Importing Data

We will be using the USairpollution and the usAir\_mod.csv data sets for these examples.

The MVA US air pollution data set:

usAir<-USairpollution

The modified USairpollution data set from your working directory is a csv file. Note you will need to modify you working directory if it is different.

usAir\_mod<- read.csv("G:/Shared drives/MultivariateStatistics/Data/LabData/Lab2/usAir\_mod.csv", row=1, header=TRUE)

# Data screening

Your first move when conducting a multivariate analysis (or any analysis) is to screen the data. You are looking for data errors, missing data, and outliers that may influence your analysis.

## Data errors

One way to check for data errors is to examine the summary statistics for your data set.

First look at the summary statistics for usAir:

describeBy(usAir)

***Question 1: Do you see any unrealistic values? (5 pts) Note please answer all questions with points related to them.***  The standard deviation values for manu and popul as they are much larger that the mean and median values, indicating some large spread. The range of values for these two also seem pretty extreme so that brings some suspicion. Wind values also seem to be pretty low but also given the context of the cities, Phoenix has the lowest wind value which makes sense since it is generally flat and there are not many tall buildings compared to Chicago with a much higher wind value.

Now look at the summary statistics for usAir\_mod:

describeBy(usAir\_mod)

Look at the max for temperature. It will be easier to look for data errors when it is your own data.

## Missing Data

When you have missing entries in your data sheet, R replaces them with “NA”. You can check if you have any missing variables in *usAir\_mod*:

describe(usAir\_mod)

The *describe* function provides some of the same information as *describeBy*, but importantly shows you which variables have missing values.

We talked about two methods for dealing with missing values in lecture; **Complete Case and Imputation**. We will look at **complete case and imputation** for now.

**Complete Case** involves the removal of samples (in this case cities) with missing data:

usAir\_mod [complete.cases(usAir\_mod),]

**Imputation** involves filling in missing values with plausible data. Let’s replace NAs with the mean of the variable.

#First, let’s calculate the mean of each variable (column) with the NA removed:  
  
meanz<-colMeans(usAir\_mod,na.rm=T)  
  
#`na.rm=T`, means that you want to remove NAs  
  
#To replace your NAs with the means you just calculated you will use the following function:  
  
naFunc<-function(column) {   
 column[is.na(column)] = round(mean(column, na.rm = TRUE),2)  
 return(column)   
}  
  
#and “apply” it to the usair\_mod data set  
  
Impute<-apply(usAir\_mod,2,naFunc)

Check out the new Impute data object and make sure that the NA’s have been replaced.

We will not go into this advanced function too much. However, know that *apply* allows us to perform a function on all the rows and/or columns in a data frame of matrix. As we spoke about in lecture, there are many types of imputation methods. We can explore further methods for your specific missing data.

# Multivariate Normal Distribution

Many of the analyses we will do in this course have an assumption of multivariate normality. While there are many tests of multivariate normality, they tend to be overly conservative. If we strictly followed these tests, we may never run a multivariate analysis with ecological or agricultural data. Here we will look at two multivariate tests of normality.

## Shapiro-Wilks test

Shapiro-Wilks tests if the distribution of the observed data differs from multivariate normal distribution. So, we are looking for p-values > 0.05.

mshapiro.test(t(usAir))

##   
## Shapiro-Wilk normality test  
##   
## data: Z  
## W = 0.59549, p-value = 2.025e-09

## Mardia test

Mardia’s test looks at multivariate extensions of Skewness and Kurtosis. In both cases, we are looking for p-values > 0.05 to show that our data do not deviate from the expectations of multivariate normal Skewness and Kurtosis. For the observed data to be considered multivariate normal, p-values from both the Skewness and Kurtosis statistics must be > 0.05. This function also tests for univariate normality of residuals using the Shapiro-Wilk statistic.

mvn(usAir, mvnTest = "mardia")

## $multivariateNormality  
## Test Statistic p value Result  
## 1 Mardia Skewness 226.612731693166 4.82491336101954e-15 NO  
## 2 Mardia Kurtosis 3.97754689564216 6.96298933924311e-05 NO  
## 3 MVN <NA> <NA> NO  
##   
## $univariateNormality  
## Test Variable Statistic p value Normality  
## 1 Anderson-Darling SO2 2.3841 <0.001 NO   
## 2 Anderson-Darling temp 0.9633 0.0136 NO   
## 3 Anderson-Darling manu 4.2925 <0.001 NO   
## 4 Anderson-Darling popul 3.4292 <0.001 NO   
## 5 Anderson-Darling wind 0.3784 0.3911 YES   
## 6 Anderson-Darling precip 0.8742 0.0228 NO   
## 7 Anderson-Darling predays 0.5175 0.1783 YES   
##   
## $Descriptives  
## n Mean Std.Dev Median Min Max 25th 75th Skew  
## SO2 41 30.048780 23.472272 26.00 8.00 110.0 13.00 35.00 1.584112608  
## temp 41 55.763415 7.227716 54.60 43.50 75.5 50.60 59.30 0.822975684  
## manu 41 463.097561 563.473948 347.00 35.00 3344.0 181.00 462.00 3.484603302  
## popul 41 608.609756 579.113023 515.00 71.00 3369.0 299.00 717.00 2.941257977  
## wind 41 9.443902 1.428644 9.30 6.00 12.7 8.70 10.60 0.002675131  
## precip 41 36.769024 11.771550 38.74 7.05 59.8 30.96 43.11 -0.692518149  
## predays 41 113.902439 26.506419 115.00 36.00 166.0 103.00 128.00 -0.550092270  
## Kurtosis  
## SO2 2.25541093  
## temp 0.09066032  
## manu 14.33200058  
## popul 10.57605759  
## wind 0.06015407  
## precip 0.49578021  
## predays 0.72033969

# Data transformation

The next step is preparing your data for analysis is transforming the data. Today we will look at the log, square root, and arcsine square root transformations.

## Log transformation:

Several common transformations have built-in functions in R. While you can build transformation functions on your own, we will use the ones R has developed today. First, let’s look at a histogram of our first variable, SO2, to determine if transformation is necessary:

Remember, to extract the SO2 column:

usAir$SO2   
  
#or   
  
usAir[,1]   
  
  
#Next you can simply wrap either of those commands in the histogram function:  
  
hist(usAir$SO2)   
  
#or   
  
hist(usAir[,1])

To log transform each value in our data frame:

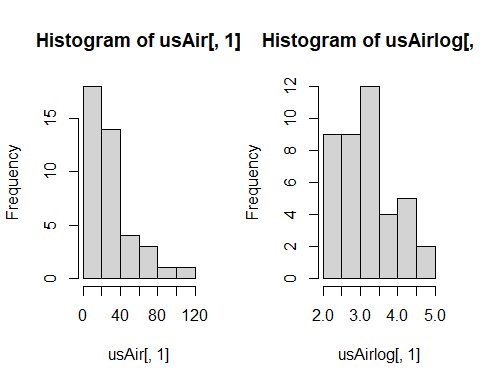
usAirlog<-log1p(usAir)

and the histogram:

hist(usAirlog$SO2)   
  
#or   
  
hist(usAirlog[,1])

You can compare the histograms side by side using the par function followed by hist:

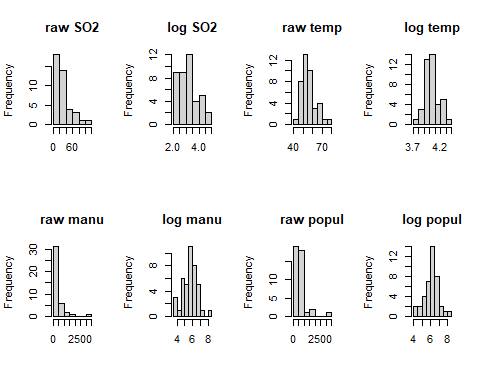
par(mfrow=c(1,2))  
hist(usAir[,1])   
hist(usAirlog[,1])



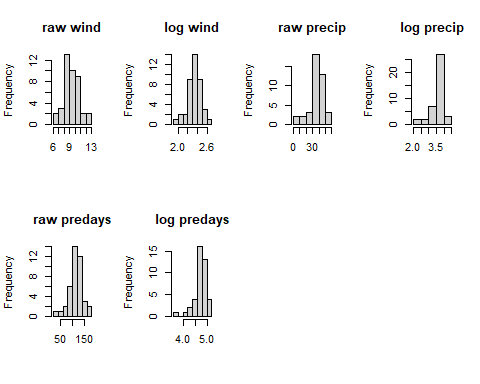
Placing 1, 2 in parentheses after the c (which stands for concatenate) in the par function indicates that you want you plots arranged in 1 row and two columns. Note this plotting is done in base R as opposed to using the ggplot functions of Tidyverse. It is helpful to know base R and Tidyverse to be able to read and trouble shoot code with a wide range of collaborators. In ggplot this code would be similar to what the *facet* function does.

Compare histograms for the raw data and the log transformed data for each variable.

par(mfrow=c(2,4))  
#SO2  
hist(usAir$SO2, main = "raw SO2", xlab="")  
hist(usAirlog$SO2, main = "log SO2", xlab="")  
  
#temp  
hist(usAir$temp, main = "raw temp", xlab="")  
hist(usAirlog$temp, main = "log temp", xlab="")  
  
#manu  
hist(usAir$manu, main = "raw manu", xlab="")  
hist(usAirlog$manu, main = "log manu", xlab="")  
  
#popul  
hist(usAir$popul, main = "raw popul", xlab="")  
hist(usAirlog$popul, main = "log popul", xlab="")



#wind  
hist(usAir$wind, main = "raw wind", xlab="")  
hist(usAirlog$wind, main = "log wind", xlab="")  
  
#precip  
hist(usAir$precip, main = "raw precip", xlab="")  
hist(usAirlog$precip, main = "log precip", xlab="")  
  
#predays  
hist(usAir$predays, main = "raw predays", xlab="")  
hist(usAirlog$predays, main = "log predays", xlab="")



**Question 2: Which variable might not need to be log transformed? (5 pts)** The wind and predays seem to not need log transformation since they appear to be normal. Temp and precip may also not need log but it seems they are slightly improved.

## Square root transformation:

To square root transform each value in our data frame:

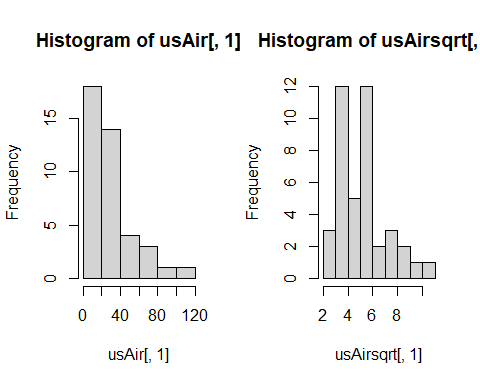
usAirsqrt<-sqrt(usAir)

and the histogram:

hist(usAirsqrt$SO2)  
  
#or   
  
hist(usAirsqrt[,1])

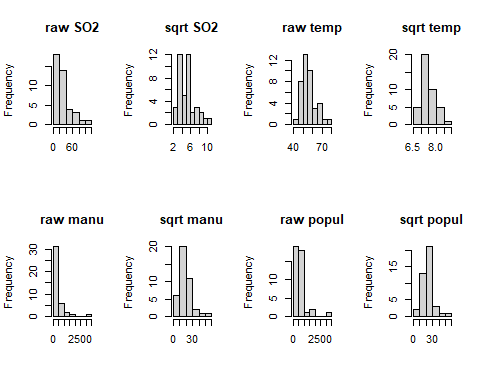
Compare the histograms side by side using the par function followed by hist:

par(mfrow=c(1,2))  
  
hist(usAir[,1])   
hist(usAirsqrt[,1])

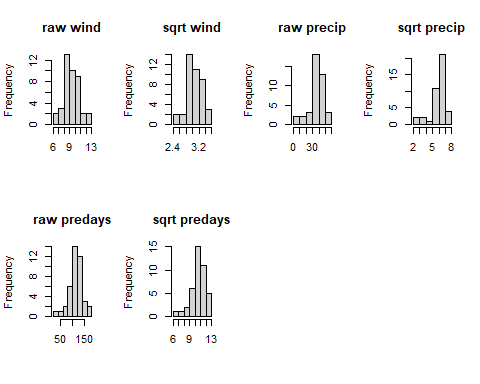


Compare histograms for the raw data and the square root transformed data for each variable…

par(mfrow=c(2,4))  
#SO2  
hist(usAir$SO2, main = "raw SO2", xlab="")  
hist(usAirsqrt$SO2, main = "sqrt SO2", xlab="")  
  
#temp  
hist(usAir$temp, main = "raw temp", xlab="")  
hist(usAirsqrt$temp, main = "sqrt temp", xlab="")  
  
#manu  
hist(usAir$manu, main = "raw manu", xlab="")  
hist(usAirsqrt$manu, main = "sqrt manu", xlab="")  
  
#popul  
hist(usAir$popul, main = "raw popul", xlab="")  
hist(usAirsqrt$popul, main = "sqrt popul", xlab="")



#wind  
hist(usAir$wind, main = "raw wind", xlab="")  
hist(usAirsqrt$wind, main = "sqrt wind", xlab="")  
  
#precip  
hist(usAir$precip, main = "raw precip", xlab="")  
hist(usAirsqrt$precip, main = "sqrt precip", xlab="")  
  
#predays  
hist(usAir$predays, main = "raw predays", xlab="")  
hist(usAirsqrt$predays, main = "sqrt predays", xlab="")



Remember that square root transformations are best used on count data.

## Arcsine square root transformation: = arcsine

If you remember arcsine square root transformations are for percentage data. So, the values for your variable must range between 0 and 1. None of the variables in usAir are appropriate for this transformation. Let’s draw some random numbers between 0 and 1 so we can use the arcsine square root transformation.

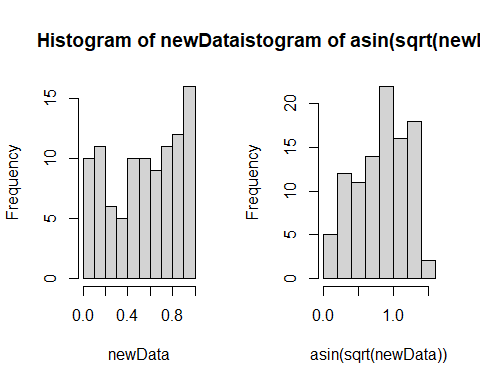
newData<- runif(100, 0, 1)

You just chose 100 random values between 0 and 1. Now let’s transform:

asin(sqrt(newData))

and compare histograms:

par(mfrow=c(1,2))  
  
hist(newData)  
hist(asin(sqrt(newData)))



# Data standardization

Column standardization adjusts for differences among variables. The focus is on the profile across a sample unit. Row standardization adjusts for differences among sample units, wherein the focus is on the profile within a sample unit. Row standardization is good when variables are measured in the same units (e.g. species). You will more often than not be using column standardization.

## Coefficient of Variation (cv)

Let’s first see if the air pollution data set needs standardization by calculating the *coefficient of variation* **(cv)** for column totals. Remember, the **cv** is the ratio of the standard deviation to the mean (σ/μ):

First calculate the column **sums**:

cSums<-colSums(usAir)

Then calculate the **standard deviation** and **mean** for the column sums:

Sdev<-sd(cSums)  
M<-mean(cSums)

Finally, calculate the **cv**:

Cv<-Sdev/M\*100

Our rule of thumb for cv is that if **cv> 50**, data standardization is necessary.

***Questin 3: Is standardization necessary for the USairpollution data? (5 pts)*** Standardization is necessary for the the USairpollution dataset (cv = 129.3).

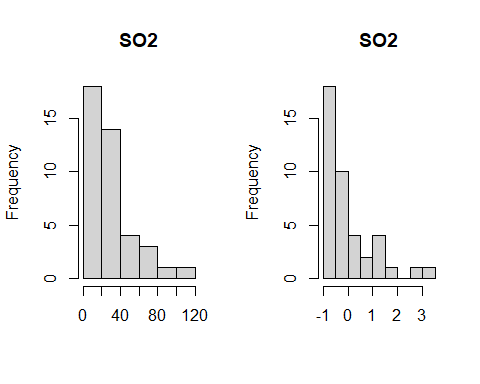
## Z- standardization = (

Your goal here is to equalize the variance for variables measured on different scales. There is a built-in function scale that will do this for you:

scaledData<-scale(usAir)

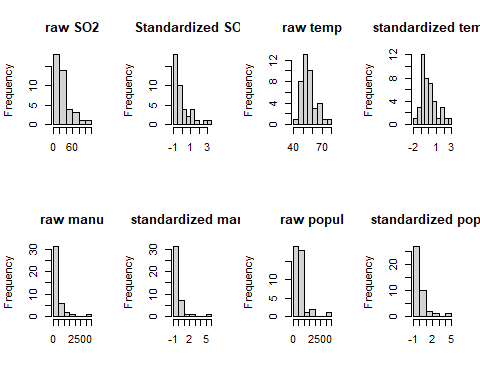
Let’s look at histograms for the scaled and unscaled data for the first variable, SO2:

par(mfrow=c(1,2))  
  
hist(usAir[,1] ,main=colnames(usAir)[1],xlab=" ")  
hist(scaledData[,1] ,main=colnames(usAir)[1],xlab=" ")

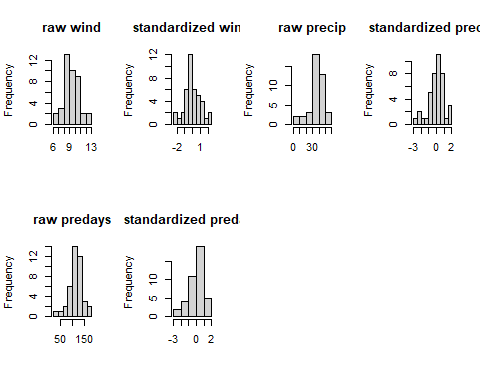


Compare the raw and standardized histograms for all of the variables.

par(mfrow=c(2,4))  
#SO2  
hist(usAir$SO2, main = "raw SO2", xlab="")  
hist(scaledData[,1], main = "Standardized SO2", xlab="")  
  
#temp  
hist(usAir$temp, main = "raw temp", xlab="")  
hist(scaledData[,2], main = "standardized temp", xlab="")  
  
#manu  
hist(usAir$manu, main = "raw manu", xlab="")  
hist(scaledData[,3], main = "standardized manu", xlab="")  
  
#popul  
hist(usAir$popul, main = "raw popul", xlab="")  
hist(scaledData[,4], main = "standardized popul", xlab="")



#wind  
hist(usAir$wind, main = "raw wind", xlab="")  
hist(scaledData[,5], main = "standardized wind", xlab="")  
  
#precip  
hist(usAir$precip, main = "raw precip", xlab="")  
hist(scaledData[,6], main = "standardized precip", xlab="")  
  
#predays  
hist(usAir$predays, main = "raw predays", xlab="")  
hist(scaledData[,7], main = "standardized predays", xlab="")



***Question 4: Are you convinced that the variances are equalized? Just to check, calculate the mean and variance for each of the standardized variables. (10 pts)***  
Based off of the calculations the standard deviations seem pretty equalized despite the apparent skewness of the histograms.

matmean = colMeans(scaledData)   
matsd = c(sd(scaledData[1,]), sd(scaledData[2,]), sd(scaledData[3,]), sd(scaledData[4,]), sd(scaledData[5,]), sd(scaledData[6,]), sd(scaledData[7,]))  
scaledSprd = rbind(matmean, matsd)  
rownames(scaledSprd) <- c("mean", "sd")  
scaledSprd

## SO2 temp manu popul wind  
## mean 7.951169e-17 -2.446383e-16 -2.704688e-17 -3.046344e-18 3.423752e-16  
## sd 7.494264e-01 9.401069e-01 5.254691e-01 3.126387e-01 1.288528e+00  
## precip predays  
## mean 2.751864e-16 -1.262011e-16  
## sd 1.064159e+00 2.452911e+00

**Z standardization is very common in life sciences.**

# Detecting Outliers

Outliers are recorded values of measurements or observations that are outside the range of the bulk of the data. Outliers can inflate variance and lead to erroneous conclusions.

## Univariate outliers

One way to deal with outliers in multivariate data is to examine each variable separately. You will standardize your data into standard deviation units (z –standardization) and look for values that fall outside of three standard deviations.

First the z-standardization:

scaledData<-scale(usAir)

Next we will create histograms to look for values > than 3 sd. However, this time we will use the *par fu*nction to look at all seven histograms at once.

par(mfrow=c(2,4))  
hist(scaledData [,1] ,main=colnames(usAir)[1],xlab=" ")  
hist(scaledData [,2] ,main=colnames(usAir)[2],xlab=" ")   
hist(scaledData [,3] ,main=colnames(usAir)[3],xlab=" ")   
hist(scaledData [,4] ,main=colnames(usAir)[4],xlab=" ")  
hist(scaledData [,5] ,main=colnames(usAir)[5],xlab=" ")   
hist(scaledData [,6] ,main=colnames(usAir)[6],xlab=" ")   
hist(scaledData [,7] ,main=colnames(usAir)[7],xlab=" ")

Finally, you can identify the outlier(s) for each variable:

scaledData [,1][scaledData [,1]>3]   
scaledData [,2][scaledData [,2]>3]   
scaledData [,3][scaledData [,3]>3]   
scaledData [,4][scaledData [,4]>3]  
scaledData [,5][scaledData [,5]>3]  
scaledData [,6][scaledData [,6]>3]   
scaledData [,7][scaledData [,7]>3]

Alternatively, you could use the apply function, less typing!

For the histogram function (hist):

par(mfrow=c(2,4))  
mapply(hist,as.data.frame(usAir),main=colnames(usAir),xlab=" ")

Here is a new function for detecting outliers called out.

out<-function(x){  
lier<-x[abs(x)>3]  
return(lier)  
}

Let’s apply that function:

apply(scaledData,2,out)

## $SO2  
## Chicago   
## 3.406199   
##   
## $temp  
## named numeric(0)  
##   
## $manu  
## Chicago   
## 5.112752   
##   
## $popul  
## Chicago   
## 4.766583   
##   
## $wind  
## named numeric(0)  
##   
## $precip  
## named numeric(0)  
##   
## $predays  
## named numeric(0)

**Question 5: Do you detect any outliers? For which variables? (5 pts)** Outliers were detected for the following: SO2, temp, manu, popul.

## Multivariate outliers

**we will come back to this…**

# Distance and Dissimilarity

As we know from lecture, multivariate data with *p* variables are visually represented by a collection of points forming a data cloud in *p*-dimensional space. The shape, clumping, and dispersion of the data cloud contains information we seek to describe. Several distance and dissimilarity measures are used to calculate the distance between data points.

## Euclidean Distance:

**Euclidean** distance is one of the most commonly used distance measures. It is normally preceded by column standardization (e.g. z standardization). Let’s calculate Euclidean distance for the US air pollution data set. You will use the function *vegdist* from the *vegan* (vegetation analysis) package. Look up *vegdist* to see the different indices available in this package.

?vegdist

First, z standardization:

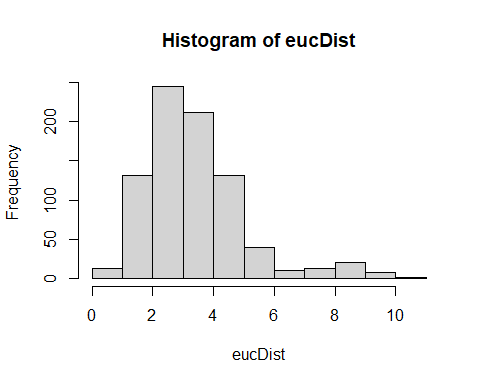
scaledData<-scale(usAir)

Then calculate distance:

#Euclidean distance: common distance & for cont data   
eucDist<- vegdist(scaledData,"euclidean")

Let’s look at a histogram of distances:

hist(eucDist)



**Question 6: What does this frequency distribution tell you about pollution conditions across these 41 cities? (5 pts)** The distance between the majority of the pollution values across cities are close to each other, with a few groups that have much different pollution conditions.

Euclidean Distance can be weird. Let look at the data matrix below:

We want to determine how similar these farms are in the production of strawberries, peaches, and raspberries.

Fruit <-rbind(c(1,0,1,1),c(2,1,0,0), c(3,0,4,4))  
colnames(Fruit)<-c("Farm","Strawberry","Peach", "Rasberry")  
Fruit

## Farm Strawberry Peach Rasberry  
## [1,] 1 0 1 1  
## [2,] 2 1 0 0  
## [3,] 3 0 4 4

Calculating Euclidean distance on these data:

eucDist<- vegdist(Fruit[,-1], "euclidean")

Gives us this distance matrix (R gives you the triangular matrix without the diagonal): | |1|2| |2|1.73| | |3|4.24|5.74|

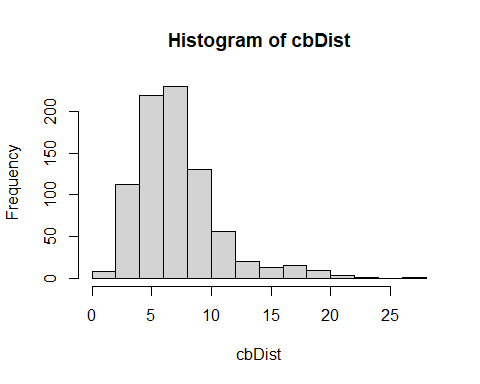
The distance between farms 1 and 2, which grow none of the same fruits:

Is **less** (i.e., these farms are more similar in their fruit production) than farms 1 and 3, which grow the same fruit:

Euclidean distance is not a jack-of-all-trades and is not appropriate for all data sets. Our next distance metric, Manhattan distance would also rank Farms 1 and 2 more similar than 1 and 3.

## City-block (Manhattan) distance

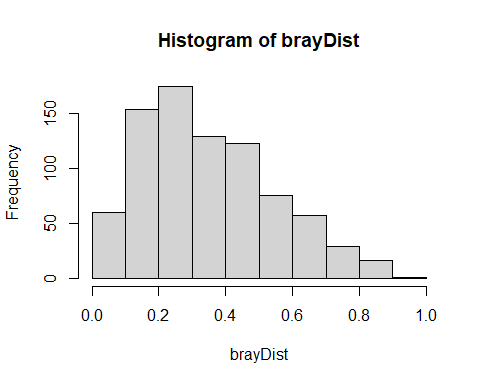
cbDist<- vegdist(scaledData,"manhattan")  
  
#Let’s look at a histogram of distances:  
hist(cbDist)



**Question 7: How does this distribution compare to Euclidean distance? (5 pts)** The city blok distribution gives a similar shape to the Euclidean distance, but the scales of the distances differ (city block has a larger range).

## Bray-Curtis dissimilarity

brayDist<- vegdist(usAir,"bray")  
  
#Histogram:  
hist(brayDist)



Let’s quickly look at our fruit farm data with Bray-Curtis:

brayFruit<- vegdist(Fruit[,-1], "bray")  
brayFruit

## 1 2  
## 2 1.0   
## 3 0.6 1.0

That makes more sense! Farms 1 and 2 (and 2 and 3) are at maximum dissimilarity and farms 1, 3 are more similar.

**Back to multivariate outliers!**

Your goal here is to examine deviations of the sample average distances to other samples. We will use **Bray-Curtis** distance:

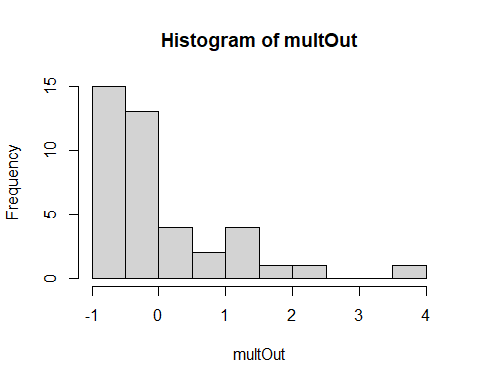
brayDist<- vegdist(usAir,"bray")

Next, calculate column means. These column means represent the average dissimilarity of each city to all other cities. You want to know if any cities are on average more than 3 standard deviation units (z scores). To achieve this, z-transform the averages:

multOut<-scale(colMeans(as.matrix(brayDist)))

Plot a histogram and look for observations >3 sd units:

hist(multOut)



You can find the cities that are outliers with:

multOut [multOut >3,]

## Chicago   
## 3.678029

Another possibility is to determine which observation are > 3 standard deviations from the mean. Using Bray-Curtis distance again:

Calculate column means:

colBray<-colMeans(as.matrix(brayDist))

Calculate the mean of the column means:

mBray<-mean(colBray)

Calculate the standard deviation:

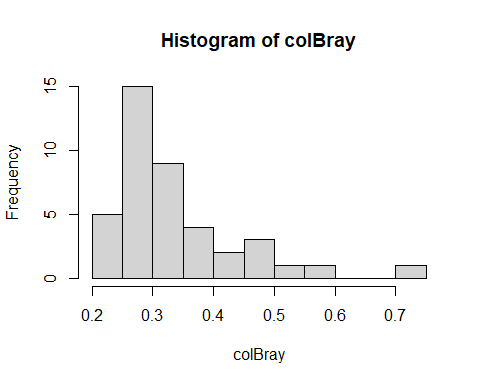
stdBray<- sd(colBray)

… 3 standard deviations

threeSD<-stdBray \* 3 + mBray

plot a histogram and look for observations >3 sd:

hist(colBray)



Find the outliers:

colBray [colBray >threeSD]

## Chicago   
## 0.7113063

# Working through my dataset

**Question 8: NOW, RUN THROUGH THE ABOVE EXERCISES WITH YOUR OWN DATA! (55 pts)**

#read in neon-npn phenology dataset with row names appointed to observation IDs  
phe = read.csv("G:/Shared drives/MultivariateStatistics/Data/StudentDataSets/AmadorL\_NeonNpn\_OpenFLowers\_conus.CSV", row=1, header = TRUE)  
  
library(tidyverse, quietly = TRUE)

## Warning: package 'tidyverse' was built under R version 4.2.3

## Warning: package 'tibble' was built under R version 4.2.3

## Warning: package 'tidyr' was built under R version 4.2.3

## Warning: package 'dplyr' was built under R version 4.2.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ lubridate 1.9.2 ✔ tibble 3.2.1  
## ✔ purrr 1.0.1 ✔ tidyr 1.3.0  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ ggplot2::%+%() masks psych::%+%()  
## ✖ ggplot2::alpha() masks psych::alpha()  
## ✖ tidyr::expand() masks Matrix::expand()  
## ✖ tidyr::extract() masks raster::extract()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ tidyr::pack() masks Matrix::pack()  
## ✖ dplyr::select() masks raster::select(), MASS::select()  
## ✖ dplyr::src() masks Hmisc::src()  
## ✖ dplyr::summarize() masks Hmisc::summarize()  
## ✖ tidyr::unpack() masks Matrix::unpack()  
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors

#remove extra columns  
phe = phe %>%  
 select(-c("update\_datetime", "phenophase\_id", "phenophase\_description", "kingdom", "common\_name", "elevation\_in\_metersStat", "phenophase\_status", "intensity\_category\_id", "intensity\_value", "abundance\_value", "elevation\_in\_meters", "first\_yes\_julian\_date", "numdays\_since\_prior\_no", "last\_yes\_julian\_date", "numdays\_until\_next\_no"))

Main variables are columns 8, 12, 16, 18 (day\_of\_year, first\_yes\_doy, last\_yes\_doy, mean\_first\_yes\_doy). Each have corresponding years to use.

## Data screening

describeBy(phe)

## Warning in describeBy(phe): no grouping variable requested

## vars n mean sd median trimmed mad  
## site\_id 1 9435 27604.33 11140.14 31256.00 28598.51 7676.90  
## state\* 2 9435 21.13 13.93 21.00 20.78 19.27  
## species\_id 3 9435 673.50 625.03 702.00 618.98 901.42  
## genus\* 4 9435 139.75 100.68 148.00 137.85 137.88  
## species\* 5 9435 252.22 143.91 275.00 256.20 185.32  
## individual\_id 6 9435 166273.73 79760.03 188674.00 170062.19 93697.35  
## observation\_date\* 7 9435 1059.60 453.29 1083.00 1098.01 613.80  
## day\_of\_year 8 9435 144.94 60.67 131.00 138.04 44.48  
## first\_yes\_year 9 9435 2017.87 2.47 2018.00 2018.06 2.97  
## first\_yes\_month 10 9435 5.03 1.89 5.00 4.81 1.48  
## first\_yes\_day 11 9435 15.24 8.68 15.00 15.15 10.38  
## first\_yes\_doy 12 9435 136.61 57.09 124.00 130.22 43.00  
## last\_yes\_year 13 9435 2017.87 2.47 2018.00 2018.06 2.97  
## last\_yes\_month 14 9435 5.39 1.96 5.00 5.18 1.48  
## last\_yes\_day 15 9435 15.80 8.66 16.00 15.86 10.38  
## last\_yes\_doy 16 9435 148.38 59.43 133.00 141.43 43.00  
## first\_yes\_sample\_size 17 9435 5.88 8.57 2.00 3.76 1.48  
## mean\_first\_yes\_doy 18 9435 141.25 58.87 127.00 134.51 44.48  
## data\_name\* 19 9435 1.70 0.46 2.00 1.75 0.00  
## DomainID\* 20 9435 6.95 5.28 6.00 6.58 7.41  
## longitude 21 9435 -93.18 17.24 -88.16 -92.28 20.65  
## latitude 22 9435 38.90 4.81 39.06 38.96 5.16  
## min max range skew kurtosis se  
## site\_id 3.00 45611.00 45608.00 -0.72 -0.49 114.69  
## state\* 1.00 46.00 45.00 0.08 -1.32 0.14  
## species\_id 1.00 2100.00 2099.00 0.47 -1.17 6.43  
## genus\* 1.00 317.00 316.00 0.01 -1.39 1.04  
## species\* 1.00 474.00 473.00 -0.19 -1.32 1.48  
## individual\_id 635.00 293161.00 292526.00 -0.37 -1.04 821.13  
## observation\_date\* 1.00 1698.00 1697.00 -0.52 -0.85 4.67  
## day\_of\_year 4.00 365.00 361.00 1.02 0.76 0.62  
## first\_yes\_year 2009.00 2021.00 12.00 -0.52 -0.18 0.03  
## first\_yes\_month 1.00 12.00 11.00 1.01 1.03 0.02  
## first\_yes\_day 1.00 31.00 30.00 0.07 -1.15 0.09  
## first\_yes\_doy 8.00 354.00 346.00 1.06 1.07 0.59  
## last\_yes\_year 2009.00 2021.00 12.00 -0.52 -0.18 0.03  
## last\_yes\_month 1.00 12.00 11.00 1.03 0.88 0.02  
## last\_yes\_day 1.00 31.00 30.00 -0.07 -1.17 0.09  
## last\_yes\_doy 14.00 362.00 348.00 1.07 0.86 0.61  
## first\_yes\_sample\_size 1.00 56.00 55.00 2.55 7.64 0.09  
## mean\_first\_yes\_doy 4.00 358.00 354.00 1.04 0.85 0.61  
## data\_name\* 1.00 2.00 1.00 -0.87 -1.24 0.00  
## DomainID\* 1.00 16.00 15.00 0.47 -1.26 0.05  
## longitude -124.20 -67.81 56.40 -0.41 -1.21 0.18  
## latitude 27.66 48.92 21.26 -0.12 -0.88 0.05

describe(phe) #No missing values

## phe   
##   
## 22 Variables 9435 Observations  
## --------------------------------------------------------------------------------  
## site\_id   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 1543 1 27604 12239 5952 9024   
## .25 .50 .75 .90 .95   
## 20206 31256 35896 37282 42661   
##   
## lowest : 3 6 19 30 38, highest: 45424 45522 45542 45574 45611  
## --------------------------------------------------------------------------------  
## state   
## n missing distinct   
## 9435 0 46   
##   
## lowest : AL AR AZ CA CO, highest: VT WA WI WV WY  
## --------------------------------------------------------------------------------  
## species\_id   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 619 1 673.5 696 3 12   
## .25 .50 .75 .90 .95   
## 82 702 1187 1601 1740   
##   
## lowest : 1 2 3 4 6, highest: 2088 2092 2094 2096 2100  
## --------------------------------------------------------------------------------  
## genus   
## n missing distinct   
## 9435 0 317   
##   
## lowest : Acer Achillea Adenostoma Aesculus Agastache   
## highest: Yeatesia Yucca Zinnia Zizia Ziziphus   
## --------------------------------------------------------------------------------  
## species   
## n missing distinct   
## 9435 0 474   
##   
## lowest : absinthifolia absinthium acanthocarpa acaule acerifolium   
## highest: vulgare wislizeni woodsii wrightii yedoensis   
## --------------------------------------------------------------------------------  
## individual\_id   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 9435 1 166274 90657 24148 49877   
## .25 .50 .75 .90 .95   
## 96827 188674 218677 266975 276250   
##   
## lowest : 635 637 638 1069 1272, highest: 291173 291174 291176 291181 293161  
## --------------------------------------------------------------------------------  
## observation\_date   
## n missing distinct   
## 9435 0 1698   
##   
## lowest : 2009-04-16 2009-04-21 2009-04-23 2009-04-28 2009-05-02  
## highest: 2021-12-03 2021-12-05 2021-12-26 2021-12-28 2021-12-31  
## --------------------------------------------------------------------------------  
## day\_of\_year   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 340 1 144.9 65.29 70 81   
## .25 .50 .75 .90 .95   
## 105 131 170 240 275   
##   
## lowest : 4 7 10 11 12, highest: 355 356 360 362 365  
## --------------------------------------------------------------------------------  
## first\_yes\_year   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 13 0.98 2018 2.774 2014 2015   
## .25 .50 .75 .90 .95   
## 2016 2018 2020 2021 2021   
##   
## Value 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019  
## Frequency 21 41 68 55 178 543 686 1287 1213 1182 1286  
## Proportion 0.002 0.004 0.007 0.006 0.019 0.058 0.073 0.136 0.129 0.125 0.136  
##   
## Value 2020 2021  
## Frequency 906 1969  
## Proportion 0.096 0.209  
##   
## For the frequency table, variable is rounded to the nearest 0  
## --------------------------------------------------------------------------------  
## first\_yes\_month   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 12 0.958 5.025 2.004 3 3   
## .25 .50 .75 .90 .95   
## 4 5 6 8 9   
##   
## Value 1 2 3 4 5 6 7 8 9 10 11  
## Frequency 53 304 1302 2692 2292 1059 723 338 359 192 104  
## Proportion 0.006 0.032 0.138 0.285 0.243 0.112 0.077 0.036 0.038 0.020 0.011  
##   
## Value 12  
## Frequency 17  
## Proportion 0.002  
##   
## For the frequency table, variable is rounded to the nearest 0  
## --------------------------------------------------------------------------------  
## first\_yes\_day   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 31 0.999 15.24 10.02 2 3   
## .25 .50 .75 .90 .95   
## 8 15 23 27 29   
##   
## lowest : 1 2 3 4 5, highest: 27 28 29 30 31  
## --------------------------------------------------------------------------------  
## first\_yes\_doy   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 329 1 136.6 61.24 65 76   
## .25 .50 .75 .90 .95   
## 100 124 162 217 260   
##   
## lowest : 8 10 11 12 14, highest: 341 345 346 352 354  
## --------------------------------------------------------------------------------  
## last\_yes\_year   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 13 0.98 2018 2.774 2014 2015   
## .25 .50 .75 .90 .95   
## 2016 2018 2020 2021 2021   
##   
## Value 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019  
## Frequency 21 41 68 55 178 543 686 1287 1213 1182 1286  
## Proportion 0.002 0.004 0.007 0.006 0.019 0.058 0.073 0.136 0.129 0.125 0.136  
##   
## Value 2020 2021  
## Frequency 906 1969  
## Proportion 0.096 0.209  
##   
## For the frequency table, variable is rounded to the nearest 0  
## --------------------------------------------------------------------------------  
## last\_yes\_month   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 12 0.956 5.394 2.081 3 3   
## .25 .50 .75 .90 .95   
## 4 5 6 8 10   
##   
## Value 1 2 3 4 5 6 7 8 9 10 11  
## Frequency 29 157 886 2328 2762 1158 746 485 378 297 161  
## Proportion 0.003 0.017 0.094 0.247 0.293 0.123 0.079 0.051 0.040 0.031 0.017  
##   
## Value 12  
## Frequency 48  
## Proportion 0.005  
##   
## For the frequency table, variable is rounded to the nearest 0  
## --------------------------------------------------------------------------------  
## last\_yes\_day   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 31 0.999 15.8 9.987 2 4   
## .25 .50 .75 .90 .95   
## 8 16 23 27 29   
##   
## lowest : 1 2 3 4 5, highest: 27 28 29 30 31  
## --------------------------------------------------------------------------------  
## last\_yes\_doy   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 332 1 148.4 63.66 77 88   
## .25 .50 .75 .90 .95   
## 109 133 173 241 275   
##   
## lowest : 14 15 16 17 18, highest: 352 354 356 361 362  
## --------------------------------------------------------------------------------  
## first\_yes\_sample\_size   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 29 0.941 5.88 7.235 1 1   
## .25 .50 .75 .90 .95   
## 1 2 5 22 25   
##   
## lowest : 1 2 3 4 5, highest: 27 28 29 30 56  
## --------------------------------------------------------------------------------  
## mean\_first\_yes\_doy   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 312 1 141.2 63.37 68.0 79.4   
## .25 .50 .75 .90 .95   
## 102.0 127.0 168.0 227.0 266.0   
##   
## lowest : 4 11 12 13 14, highest: 339 341 354 355 358  
## --------------------------------------------------------------------------------  
## data\_name   
## n missing distinct   
## 9435 0 2   
##   
## Value NEON NPN  
## Frequency 2835 6600  
## Proportion 0.3 0.7  
## --------------------------------------------------------------------------------  
## DomainID   
## n missing distinct   
## 9435 0 16   
##   
## Value D01 D02 D03 D05 D06 D07 D08 D09 D10 D11 D12  
## Frequency 1924 841 447 641 793 985 427 175 162 113 84  
## Proportion 0.204 0.089 0.047 0.068 0.084 0.104 0.045 0.019 0.017 0.012 0.009  
##   
## Value D13 D14 D15 D16 D17  
## Frequency 342 974 171 559 797  
## Proportion 0.036 0.103 0.018 0.059 0.084  
## --------------------------------------------------------------------------------  
## longitude   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 1489 1 -93.18 19.54 -122.33 -120.04   
## .25 .50 .75 .90 .95   
## -110.83 -88.16 -78.07 -72.84 -71.25   
##   
## lowest : -124.203 -124.153 -123.824 -123.401 -123.356  
## highest: -68.4814 -68.4753 -68.1692 -68.1458 -67.8074  
## --------------------------------------------------------------------------------  
## latitude   
## n missing distinct Info Mean Gmd .05 .10   
## 9435 0 1495 1 38.9 5.52 31.58 32.23   
## .25 .50 .75 .90 .95   
## 35.46 39.06 42.52 45.41 46.23   
##   
## lowest : 27.6648 28.0578 28.1238 28.1262 28.1269  
## highest: 48.7295 48.7304 48.7307 48.7549 48.9199  
## --------------------------------------------------------------------------------

Day of year values seem to be within a reasonable range. No missing values.

## Multivariate Normal Distribution

All columns must be numerical for mshapiro.test and mvn function to work, so subsetting in the meantime. Keeping identifier columns, ignore in tests. Look at the normality of variables for the first 5,000 observations

phe.mvn = phe %>%  
 select(c("site\_id", "species\_id", "individual\_id", "day\_of\_year", "first\_yes\_doy", "last\_yes\_doy", "mean\_first\_yes\_doy"))  
  
c = t(phe.mvn[0:5000, 4:7])  
mshapiro.test(c) #specifying rows bc of sample size limit & need to specify column

##   
## Shapiro-Wilk normality test  
##   
## data: Z  
## W = 0.57693, p-value < 2.2e-16

Not normal, p-value < 0.05.

mvn(phe.mvn, mvnTest = "mardia")

## $multivariateNormality  
## Test Statistic p value Result  
## 1 Mardia Skewness 53062.0287074993 0 NO  
## 2 Mardia Kurtosis 433.223588866461 0 NO  
## 3 MVN <NA> <NA> NO  
##   
## $univariateNormality  
## Test Variable Statistic p value Normality  
## 1 Anderson-Darling site\_id 362.0686 <0.001 NO   
## 2 Anderson-Darling species\_id 460.1525 <0.001 NO   
## 3 Anderson-Darling individual\_id 203.2785 <0.001 NO   
## 4 Anderson-Darling day\_of\_year 246.4899 <0.001 NO   
## 5 Anderson-Darling first\_yes\_doy 225.3881 <0.001 NO   
## 6 Anderson-Darling last\_yes\_doy 272.0426 <0.001 NO   
## 7 Anderson-Darling mean\_first\_yes\_doy 236.8177 <0.001 NO   
##   
## $Descriptives  
## n Mean Std.Dev Median Min Max 25th 75th  
## site\_id 9435 27604.3277 11140.13927 31256 3 45611 20206 35896  
## species\_id 9435 673.5011 625.02686 702 1 2100 82 1187  
## individual\_id 9435 166273.7304 79760.03309 188674 635 293161 96827 218677  
## day\_of\_year 9435 144.9408 60.66665 131 4 365 105 170  
## first\_yes\_doy 9435 136.6109 57.09012 124 8 354 100 162  
## last\_yes\_doy 9435 148.3837 59.42743 133 14 362 109 173  
## mean\_first\_yes\_doy 9435 141.2450 58.87056 127 4 358 102 168  
## Skew Kurtosis  
## site\_id -0.7238152 -0.4900502  
## species\_id 0.4704712 -1.1698713  
## individual\_id -0.3706951 -1.0369196  
## day\_of\_year 1.0172878 0.7647108  
## first\_yes\_doy 1.0581875 1.0727093  
## last\_yes\_doy 1.0653142 0.8626998  
## mean\_first\_yes\_doy 1.0364176 0.8463127

None of the variables are normal, p-value < 0.05.

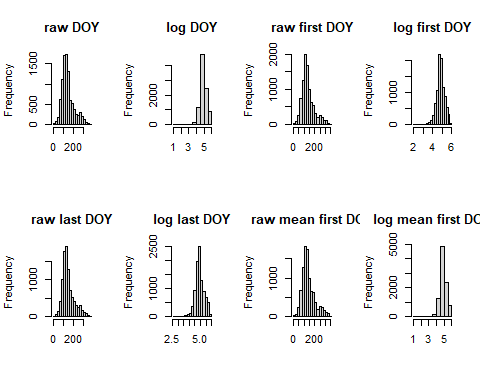
## Transformations

### Log transformation on the following: day\_of\_year, first\_yes\_doy, last\_yes\_doy, mean\_first\_yes\_doy

phe.log = phe.mvn %>%  
 mutate(day\_of\_year = log(day\_of\_year), first\_yes\_doy = log(first\_yes\_doy), last\_yes\_doy = log(last\_yes\_doy), mean\_first\_yes\_doy = log(mean\_first\_yes\_doy))

Plot the raw vs log transformed data

par(mfrow=c(2,4))  
#day of year (DOY)  
hist(phe.mvn$day\_of\_year, main = "raw DOY", xlab = " ")  
hist(phe.log$day\_of\_year, main = "log DOY", xlab = " ")  
  
#first day of year  
hist(phe.mvn$first\_yes\_doy, main = "raw first DOY", xlab = " ")  
hist(phe.log$first\_yes\_doy, main = "log first DOY", xlab = " ")  
  
#last day of year  
hist(phe.mvn$last\_yes\_doy, main = "raw last DOY", xlab = " ")  
hist(phe.log$last\_yes\_doy, main = "log last DOY", xlab = " ")  
  
#average first day of year  
hist(phe.mvn$mean\_first\_yes\_doy, main = "raw mean first DOY", xlab = " ")  
hist(phe.log$mean\_first\_yes\_doy, main = "log mean first DOY", xlab = " ")

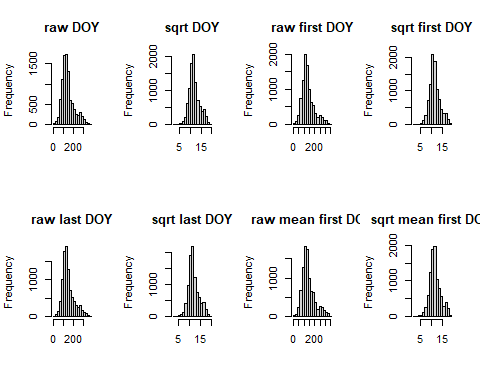


### Square root transformation on the following: day\_of\_year, first\_yes\_doy, last\_yes\_doy, mean\_first\_yes\_doy

phe.sqrt = phe.mvn %>%  
 mutate(day\_of\_year = sqrt(day\_of\_year), first\_yes\_doy = sqrt(first\_yes\_doy), last\_yes\_doy = sqrt(last\_yes\_doy), mean\_first\_yes\_doy = sqrt(mean\_first\_yes\_doy))

Plot the raw vs square root transformed data

par(mfrow=c(2,4))  
#day of year (DOY)  
hist(phe.mvn$day\_of\_year, main = "raw DOY", xlab = " ")  
hist(phe.sqrt$day\_of\_year, main = "sqrt DOY", xlab = " ")  
  
#first day of year  
hist(phe.mvn$first\_yes\_doy, main = "raw first DOY", xlab = " ")  
hist(phe.sqrt$first\_yes\_doy, main = "sqrt first DOY", xlab = " ")  
  
#last day of year  
hist(phe.mvn$last\_yes\_doy, main = "raw last DOY", xlab = " ")  
hist(phe.sqrt$last\_yes\_doy, main = "sqrt last DOY", xlab = " ")  
  
#average first day of year  
hist(phe.mvn$mean\_first\_yes\_doy, main = "raw mean first DOY", xlab = " ")  
hist(phe.sqrt$mean\_first\_yes\_doy, main = "sqrt mean first DOY", xlab = " ")



The square root transformation worked the best. Not performing arcsine square root transformation because my variables are integer/count data (i.e. days into the year until open flower observed).

## Data standardisation

### Coefficienct of variance

Checking to see if phenology data set needs standardization by calculating the *coefficient of variation* **(cv)** for column totals.

#column sums - only for the four target columns (the first three are identifier info)  
cSums <- colSums(phe.mvn[, 4:7])  
#stadard deviation  
sd <- sd(cSums)  
#mean  
m <- mean(cSums)

Cv<-sd/m\*100  
Cv>50

## [1] FALSE

Our rule of thumb for cv is that if **cv> 50**, data standardization is necessary. Standardization is not necessary for the phenology data set (cv = 3.54). If it was necessary then we would scale the data using something like Z-standardization (going to do this anyway to detect outliers).

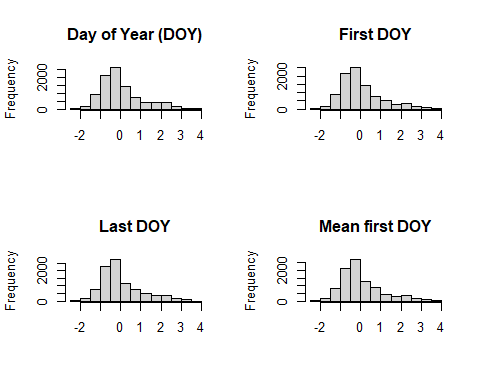
## Detecting outliers

Let’s detect univariate outliers. We will Z-standardize the variables (day\_of\_year, first\_yes\_doy, last\_yes\_doy, mean\_first\_yes\_doy) and detect any values outside of 3 standard deviations.

#scaling individually to keep identifier columns  
phe.scaled = phe.mvn %>%  
 mutate(day\_of\_year = scale(day\_of\_year), first\_yes\_doy = scale(first\_yes\_doy), last\_yes\_doy = scale(last\_yes\_doy), mean\_first\_yes\_doy = scale(mean\_first\_yes\_doy))

Let’s take a look at the scaled data

par(mfrow=c(2,2))  
  
hist(phe.scaled$day\_of\_year, main = "Day of Year (DOY)", xlab = " ")  
hist(phe.scaled$first\_yes\_doy, main = "First DOY", xlab = " ")  
hist(phe.scaled$last\_yes\_doy, main = "Last DOY", xlab = " ")  
hist(phe.scaled$mean\_first\_yes\_doy, main = "Mean first DOY", xlab = " ")



Use the out function for detecting outliers called out.

apply(phe.scaled[,4:7],2,out)

## $day\_of\_year  
## 4670872 12561148 15539083 28069159 27937280 18927283 18927302 18927261   
## 3.000977 3.149329 3.033944 3.017461 3.198779 3.149329 3.149329 3.149329   
## 18927265 18786817 18799047 18798860 18809530 24812465 24812757 24812552   
## 3.149329 3.248230 3.149329 3.149329 3.083395 3.066911 3.165812 3.165812   
## 24812553 24812575 24812556 24812953 24813081 24813092 24812912 24812830   
## 3.165812 3.165812 3.165812 3.314164 3.314164 3.314164 3.314164 3.314164   
## 24817174 24817195 24817314 24817289 24817633 28738019 21145692 24408983   
## 3.149329 3.149329 3.149329 3.149329 3.478999 3.116362 3.182296 3.297681   
## 28078607 28078618 12536016 24247640 29021299 29138168 3093223 6772032   
## 3.627351 3.627351 3.066911 3.000977 3.132845 3.577900 3.231747 3.017461   
## 9523856 12629258 15532254 28054347 28054510 27854851 3050969 3091999   
## 3.297681 3.363615 3.017461 3.544933 3.544933 3.050428 3.050428 3.231747   
## 3093176 4697677 6771791 6809126 6809778 6809008 6809162 9493679   
## 3.231747 3.132845 3.017461 3.132845 3.132845 3.132845 3.132845 3.000977   
## 9474370 9515373 12659248 24315063 27916351 27916342   
## 3.182296 3.264714 3.462516 3.149329 3.165812 3.165812   
##   
## $first\_yes\_doy  
## 9279210 15392875 15392876 15392893 16834693 20826349 20920294 9227983   
## 3.107177 3.002080 3.002080 3.002080 3.019596 3.002080 3.037112 3.002080   
## 27937280 18773921 18773993 18773872 18773941 18773955 18773908 18773808   
## 3.545081 3.019596 3.019596 3.019596 3.019596 3.019596 3.019596 3.019596   
## 18773748 18774255 18774084 18774114 18774137 18927283 18927261 18785980   
## 3.019596 3.142209 3.142209 3.142209 3.142209 3.492532 3.492532 3.142209   
## 18785932 18786114 18785970 18785979 18786817 18949107 18796614 18799047   
## 3.142209 3.019596 3.264822 3.019596 3.142209 3.492532 3.019596 3.492532   
## 18798860 18809530 18822359 18822408 24811482 24811916 24811645 24812099   
## 3.492532 3.422468 3.282338 3.159725 3.019596 3.037112 3.037112 3.142209   
## 24811935 24812134 24812123 24812465 24812757 24812552 24812553 24812575   
## 3.159725 3.037112 3.264822 3.159725 3.404951 3.159725 3.282338 3.037112   
## 24812556 24812953 24813081 24813092 24812912 24817174 24817195 24817314   
## 3.159725 3.019596 3.019596 3.159725 3.159725 3.667694 3.159725 3.492532   
## 24817289 24817633 28737927 28738019 21042719 21890782 24408983 12536016   
## 3.019596 3.019596 3.019596 3.457500 3.299854 3.807823 3.650178 3.404951   
## 24110161 15475116 15475096 27807227 19816541 20965862 27647645 27666835   
## 3.037112 3.054628 3.054628 3.264822 3.264822 3.019596 3.002080 3.037112   
## 29110878 29111019 6685764 12395100 15189087 3093223 8189588 6772032   
## 3.229790 3.054628 3.089660 3.019596 3.019596 3.580113 3.194757 3.352403   
## 9044744 9523856 10463235 11573061 11575818 12444936 12472961 12484143   
## 3.142209 3.650178 3.334887 3.527565 3.334887 3.054628 3.124693 3.247306   
## 17906860 20862990 22113379 22536434 24309785 24222833 25826105 27642519   
## 3.229790 3.282338 3.072144 3.037112 3.089660 3.264822 3.299854 3.002080   
## 27721676 27748959 27748964 3050969 3091999 3093176 4506894 4557736   
## 3.054628 3.177241 3.177241 3.387435 3.580113 3.580113 3.352403 3.317371   
## 4598011 4687505 4655114 4655581 5399917 6086496 9493679 16033381   
## 3.124693 3.159725 3.299854 3.299854 3.772791 3.124693 3.334887 3.054628   
## 15610498 15461120 23894057 24315063   
## 3.054628 3.037112 3.334887 3.492532   
##   
## $last\_yes\_doy  
## 4670872 27840158 27937280 18773941 18927283 18927261 18785980 18949107   
## 3.005621 3.005621 3.375820 3.157066 3.157066 3.157066 3.157066 3.157066   
## 18799047 18798860 18809530 24811645 24812465 24812757 24812552 24812556   
## 3.157066 3.157066 3.089757 3.325339 3.072930 3.173893 3.173893 3.173893   
## 24812953 24813081 24813092 24812912 24812830 24817174 24817195 24817314   
## 3.325339 3.325339 3.325339 3.325339 3.325339 3.577747 3.325339 3.157066   
## 24817289 28738019 12178754 21890782 24408983 12536016 15475096 29021299   
## 3.173893 3.123412 3.106584 3.459957 3.308511 3.072930 3.325339 3.140239   
## 15399034 21766458 3093223 8189588 6772032 7600881 9523856 10463235   
## 3.123412 3.072930 3.241202 3.594574 3.022448 3.325339 3.308511 3.005621   
## 11573061 11575818 12484143 13737364 14776649 17906860 22536434 26215032   
## 3.190721 3.005621 3.056103 3.022448 3.022448 3.123412 3.106584 3.493611   
## 26219752 25701130 26757588 27523822 27748959 27748964 3050969 3091999   
## 3.224375 3.157066 3.459957 3.140239 3.123412 3.291684 3.056103 3.241202   
## 3093176 4506894 4557736 4687505 5399917 5990230 6771791 9493679   
## 3.241202 3.022448 3.056103 3.089757 3.426302 3.308511 3.140239 3.241202   
## 9474370 13690857 16033381 17906630 23894057 23894041 24315063 27713741   
## 3.190721 3.089757 3.207548 3.224375 3.241202 3.241202 3.157066 3.089757   
##   
## $mean\_first\_yes\_doy  
## 1862322 27483819 27937280 21890782 12536016 10367032 10367227 10373784   
## 3.681890 3.070379 3.359148 3.613945 3.223257 3.070379 3.070379 3.070379   
## 10403181 10403225 10403480 10403557 10472759 10472770 10472850 10603118   
## 3.070379 3.070379 3.070379 3.070379 3.070379 3.070379 3.070379 3.070379   
## 10721630 10601880 10601823 10902136 10966919 27807227 25298884 3093223   
## 3.070379 3.070379 3.070379 3.070379 3.070379 3.087366 3.223257 3.393121   
## 3459956 7392853 7634091 10302901 10463235 10658807 10681493 11573061   
## 3.342162 3.053393 3.053393 3.053393 3.155311 3.155311 3.257230 3.342162   
## 11575818 15143286 14815933 16933729 20813841 20862953 20862990 21405791   
## 3.155311 3.257230 3.155311 3.121338 3.257230 3.104352 3.104352 3.104352   
## 22364982 22536434 24222833 24631752 25826105 26847452 26847445 26847438   
## 3.155311 3.155311 3.104352 3.257230 3.121338 3.155311 3.155311 3.155311   
## 26784423 27748959 27748964 3050969 3665853 3665868 3665870 3867428   
## 3.155311 3.002433 3.002433 3.206270 3.155311 3.155311 3.155311 3.002433   
## 4557736 7059745 7461552 7686528 7887724 7930615 7939257 8014880   
## 3.206270 3.087366 3.087366 3.223257 3.223257 3.223257 3.223257 3.002433   
## 8093201 9515373 9660597 11096075 11096088 11583499 13793330 12959393   
## 3.223257 3.121338 3.002433 3.223257 3.223257 3.121338 3.121338 3.087366   
## 16181642 17575870 21917627 24315063 24598840 24755689   
## 3.087366 3.223257 3.630931 3.308189 3.121338 3.121338

Outliers were detected for all variables. The most were found in first\_yes\_doy but all had outliers later in the year. There are plants opening their flowers late in the year.

## Distance & Dissimilarity

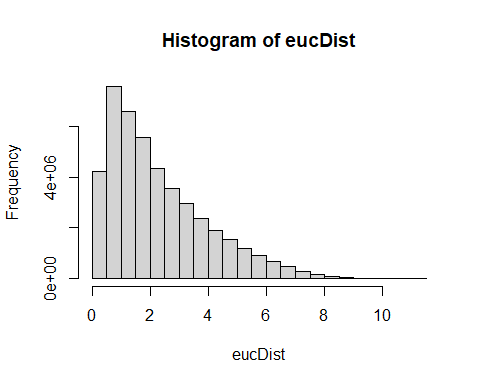
### Euclidean distance

For continuous data, my data is more discrete but let’s take a gander. First Z-standardizing

#scaling individually to keep identifier columns  
phe.scaled = phe.mvn %>%  
 mutate(day\_of\_year = scale(day\_of\_year), first\_yes\_doy = scale(first\_yes\_doy), last\_yes\_doy = scale(last\_yes\_doy), mean\_first\_yes\_doy = scale(mean\_first\_yes\_doy))

Then calculate distance & view histogram of distances:

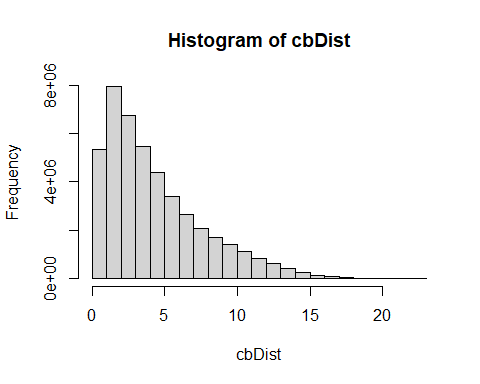
#Euclidean distance: common distance & for cont data   
eucDist<- vegdist(phe.scaled[4:7],"euclidean")  
#histogram of distances  
hist(eucDist)

 The majority of the raw data points are relatively close to each other.

Note: depending on the data & transformation would need to include the transformed data into the dissimilarty/distance metrics – especially to identify outliers.

### City-block (Manhattan) distance

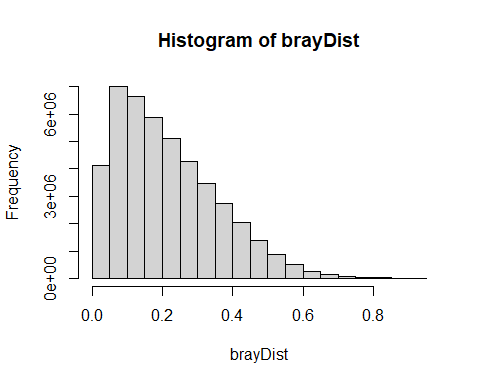
cbDist<- vegdist(phe.scaled[4:7],"manhattan")  
  
#Let’s look at a histogram of distances:  
hist(cbDist)

 The city blok distribution gives a similar shape to the Euclidean distance, but the scales of the distances differ (city block has a larger range).

### Bray-Curtis dissimilarity

Let’s look at the bray-curtis dissimilarity. In addition, examine deviations of the sample average distances to other samples using **Bray-Curtis** distance.

brayDist<- vegdist(phe.mvn[4:7],"bray")  
  
#Histogram:  
hist(brayDist)

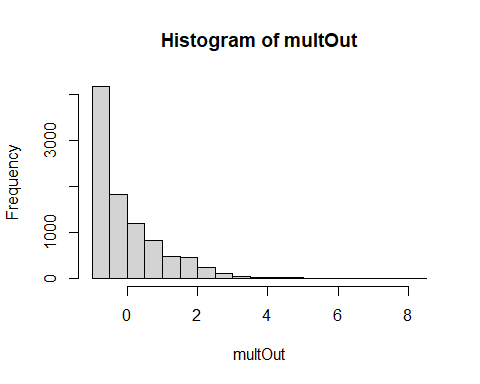


Calculate column means. These column means represent the average dissimilarity of each observation to all other day of year metric. You want to know if any observations are on average more than 3 standard deviation units (z scores). To achieve this, z-transform the averages:

multOut<-scale(colMeans(as.matrix(brayDist)))

Plot a histogram and look for observations >3 sd units:

hist(multOut)

 You can find the observations that are outliers with:

multOut [multOut >3,]

## 21704635 15886244 15933257 18852166 24844216 3281916 4918309 4918413   
## 3.898912 3.394985 3.079147 4.993236 3.347836 4.020318 4.790223 4.332700   
## 9842514 9842528 9862760 9862771 9880127 9880338 9941977 12945053   
## 3.393015 3.393015 3.260520 3.260520 3.238989 3.238989 3.033543 3.662031   
## 12906275 12910920 12906416 12948867 12865629 12899027 12964877 12912365   
## 4.288570 4.808541 4.499051 4.084446 5.663679 4.143246 3.285473 4.114566   
## 12930963 15755941 15815251 15813481 15834972 15834737 15834723 15852302   
## 3.420142 6.583303 6.014388 4.566451 4.938757 5.231250 3.973552 5.740865   
## 15885118 15915250 15915178 15915257 15935875 15943209 15943969 21318899   
## 4.113812 4.415469 3.417559 3.579999 3.915318 3.259521 3.339502 4.629051   
## 21318965 21319092 21324808 21356654 21356687 21371926 21388484 21421815   
## 4.629051 4.629051 6.166417 4.097868 4.097868 6.716091 3.246935 4.652658   
## 21464239 21464250 21536609 21540552 21540568 21933501 21890782 22881401   
## 4.907476 4.907476 3.446277 3.408659 3.408659 5.546143 3.017883 3.417770   
## 24578733 24634506 24649555 24649565 25117526 24709067 25017831 25017609   
## 4.478735 5.295937 4.343836 4.343836 3.274418 4.123482 3.435334 4.763197   
## 28078607 28078618 33196975 21456961 6937469 6937438 6937440 9866359   
## 4.184578 4.329055 4.518353 4.499051 8.147982 7.974038 7.974038 3.105501   
## 10010957 15992102 21511211 8218051 21571077 1778888 3216710 3216717   
## 5.171644 3.245406 3.365565 3.023205 3.087316 4.778313 3.760316 3.399459   
## 4959121 4959138 4959156 8290545 9947805 20862953 21063414 21405791   
## 3.940505 3.940505 3.940505 5.033276 3.363027 3.842120 3.589595 3.519721   
## 21423959 21481559 21542867 21542873 24682902 24682897 24691035 24740896   
## 3.883800 3.331079 3.185560 3.185560 3.521937 3.521937 3.893619 3.606432   
## 24741895 24741890 7110210 13116151 13116203 12901501 12901509 13115294   
## 3.159079 3.497949 3.054375 3.043759 3.043759 3.113535 3.113535 3.039195   
## 12916207 12932396 21454714 3224756 3258488 4659932 4830888 4940910   
## 4.289332 4.202260 3.008561 3.341695 3.288515 3.754505 5.646121 4.998737   
## 6809162 7059745 7098909 9649101 9676452 9736476 9842786 13160765   
## 3.034882 3.641259 3.421805 6.875373 6.418214 5.129062 3.579999 3.821409   
## 12900499 12900291 12900279 12900353 12900416 12830224 12959393 13196549   
## 3.925443 3.768098 3.768098 4.320159 4.320159 4.836609 3.153175 3.814513   
## 15872784 16106903 21300131 21531444 21531670 24598840 24617793 25167562   
## 3.939648 3.729745 5.313576 4.293416 4.378624 5.070584 3.073434 3.754760

Another possibility is to determine which observation are > 3 standard deviations from the mean. Using Bray-Curtis distance again:

Calculate column means:

colBray<-colMeans(as.matrix(brayDist))

Calculate the mean of the column means:

mBray<-mean(colBray)

Calculate the standard deviation:

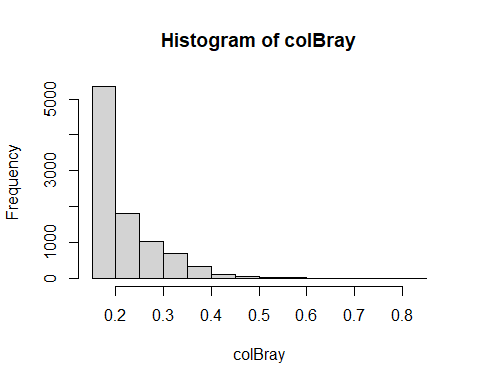
stdBray<- sd(colBray)

… 3 standard deviations

threeSD<-stdBray \* 3 + mBray

plot a histogram and look for observations >3 sd:

hist(colBray)



Find the outliers:

colBray [colBray >threeSD]

## 21704635 15886244 15933257 18852166 24844216 3281916 4918309 4918413   
## 0.5009283 0.4639681 0.4408031 0.5811909 0.4605100 0.5098328 0.5663010 0.5327443   
## 9842514 9842528 9862760 9862771 9880127 9880338 9941977 12945053   
## 0.4638236 0.4638236 0.4541058 0.4541058 0.4525266 0.4525266 0.4374583 0.4835544   
## 12906275 12910920 12906416 12948867 12865629 12899027 12964877 12912365   
## 0.5295076 0.5676446 0.5449452 0.5145362 0.6303642 0.5188488 0.4559360 0.5167453   
## 12930963 15755941 15815251 15813481 15834972 15834737 15834723 15852302   
## 0.4658132 0.6978135 0.6560867 0.5498886 0.5771952 0.5986479 0.5064027 0.6360253   
## 15885118 15915250 15915178 15915257 15935875 15943209 15943969 21318899   
## 0.5166901 0.5388149 0.4656237 0.4775378 0.5021316 0.4540325 0.4598987 0.5544800   
## 21318965 21319092 21324808 21356654 21356687 21371926 21388484 21421815   
## 0.5544800 0.5544800 0.6672372 0.5155206 0.5155206 0.7075527 0.4531094 0.5562114   
## 21464239 21464250 21536609 21540552 21540568 21933501 21890782 22881401   
## 0.5749009 0.5749009 0.4677301 0.4649710 0.4649710 0.6217436 0.4363098 0.4656392   
## 24578733 24634506 24649555 24649565 25117526 24709067 25017831 25017609   
## 0.5434551 0.6033924 0.5335610 0.5335610 0.4551252 0.5173993 0.4669275 0.5643188   
## 28078607 28078618 33196975 21456961 6937469 6937438 6937440 9866359   
## 0.5218803 0.5324769 0.5463609 0.5449452 0.8125740 0.7998162 0.7998162 0.4427360   
## 10010957 15992102 21511211 8218051 21571077 1778888 3216710 3216717   
## 0.5942761 0.4529973 0.4618103 0.4367001 0.4414023 0.5654275 0.4907631 0.4642962   
## 4959121 4959138 4959156 8290545 9947805 20862953 21063414 21405791   
## 0.5039790 0.5039790 0.5039790 0.5841276 0.4616241 0.4967629 0.4782417 0.4731168   
## 21423959 21481559 21542867 21542873 24682902 24682897 24691035 24740896   
## 0.4998200 0.4592810 0.4486079 0.4486079 0.4732793 0.4732793 0.5005401 0.4794766   
## 24741895 24741890 7110210 13116151 13116203 12901501 12901509 13115294   
## 0.4466657 0.4715199 0.4389862 0.4382076 0.4382076 0.4433253 0.4433253 0.4378728   
## 12916207 12932396 21454714 3224756 3258488 4659932 4830888 4940910   
## 0.5295634 0.5231772 0.4356261 0.4600596 0.4561591 0.4903369 0.6290764 0.5815944   
## 6809162 7059745 7098909 9649101 9676452 9736476 9842786 13160765   
## 0.4375565 0.4820309 0.4659352 0.7192352 0.6857051 0.5911530 0.4775378 0.4952439   
## 12900499 12900291 12900279 12900353 12900416 12830224 12959393 13196549   
## 0.5028742 0.4913339 0.4913339 0.5318244 0.5318244 0.5697032 0.4462327 0.4947381   
## 15872784 16106903 21300131 21531444 21531670 24598840 24617793 25167562   
## 0.5039161 0.4885208 0.6046861 0.5298630 0.5361125 0.5868640 0.4403841 0.4903556