**Wine Review Analysis**

**By John Brunzell, Chris Ewald, Neal Sakash, Jasmine Smallwood**

**Abstract.** Using data gathered from *Wine Enthusiast*, Zack Thoutt tried to create a predictive model to identify wines through blind tasting like a master sommelier would. His goal was to create a model that could identify the variety, winery, and location of a wine based on a description that a sommelier would give. We decided to use some of the same data that Zack used to try and make a predictive model that could predict the price of a wine based on some predictor variables such as country or region of origin and point value assessed by a sommelier.

**I. Brief Overview**

*Wine Enthusiast* wine tasters and contributing editors tasted hundreds of thousands of wines from around the world. Each wine was given a robust verbal description along with a wine rating and paired with the cost of the wine. The wine rating was based on a scale developed by Robert M. Parker Jr. which uses a scale from 50 - 100. This roughly translates to receiving a grade from A through F, with a 100 being the top and 50 being the bottom. Our project focused on the analysis of a wine review dataset imported from the data science and machine learning hosting site Kaggle. The dataset was curated by a Kaggle user after scraping the review website *Wine Enthusiast*. Before preprocessing, the raw dataset used contained just under 130,000 entry rows and 10 descriptive columns of distinct wines found from all over the world. Prominent wine producing countries included the United States, France, Italy, Spain, Argentina, and Australia. Other descriptive columns included information relating to region, province, winery, as well as its US price and the rating/review given by each *Wine Enthusiast* taster.

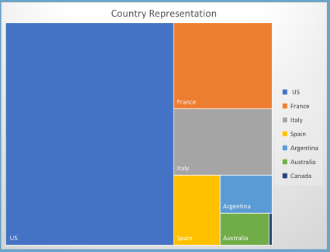
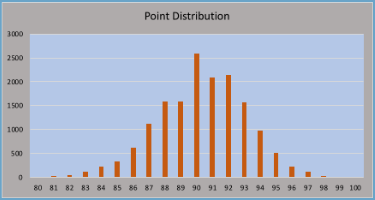
**II. Initial Data Analysis**

We began our project by cleaning and manipulating the dataset in order to help facilitate meaningful analysis. We removed all entries with N/A or null values. During this phase we did some transformations that allowed us to parse out a wine’s vintage. To factor in the length of review, we also added a column that contained a numeric value that represented the character length of the review. Lastly, we gathered the latitude, longitude and elevation from each distinct winery and merged this information to make our final data frame.

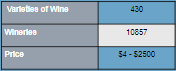


(Sample of the cleaned dataset without the addition of the latitude-longitude data)

Once we had all of the data we needed, we moved onto the data exploration phase. To better help understand the story, we attempted to visualize as much of it as we could. Some of our initial analysis including looking at how the data was grouped and what types of distributions we might see.

 (Point distribution of all wine in the dataset)

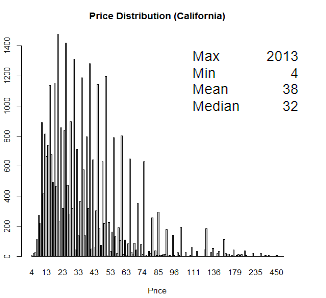
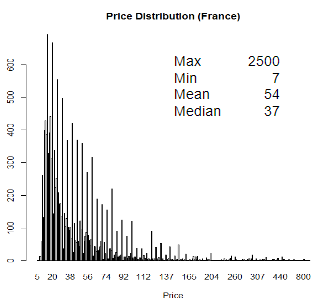
(Distribution of wine by country)



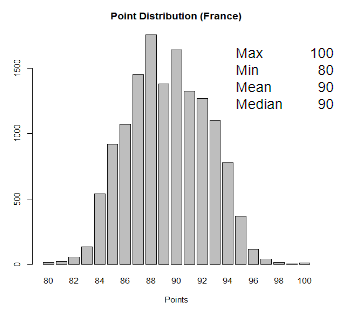
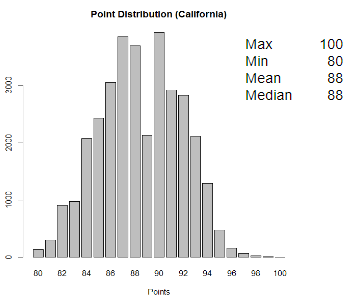
(Basic information from dataset)

We made the decision to narrow our focus to French and California wines. With the understanding of the long battle between the two areas, we wanted to see if we could discover anything in particular that made one province stick out over the other. This led us into narrowing our visuals down to contain only data from the two areas.

Looking at the distributions of the price of French and California wines, we see there distributions are very similar. This was important to note in that we were comparing wines with similar price distributions and not high-priced wines from one region against low-priced wines from another region.

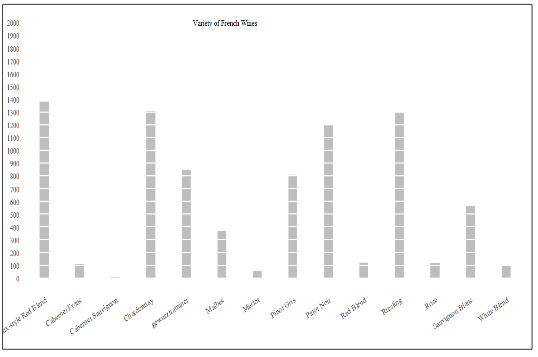


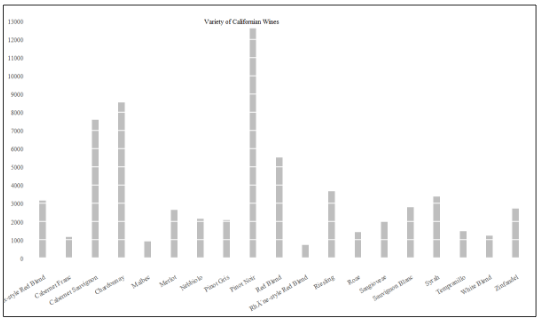
The case was similar when comparing the distributions of wine point values of French wines with California wines. Again, the distributions were similar, so we were not comparing highly rated wines from one region against low rated wines from another region.



One point to note with the data from the point distributions is that French wines have an average rating two-points higher than the average rating of California wines. This could be a result of French wines being a little better on average than California wines, but it could also be a bias in the ratings given by the tasters who knew what wine they were tasting ahead of time. However, when comparing the average ratings and prices of the four tasters, there appears to be no bias as the average points for all tasters was 84 for all wines, French wines, and California wines and the average price of the wine tasted for all tasters was $35 for all wines, French wines, and California wines.

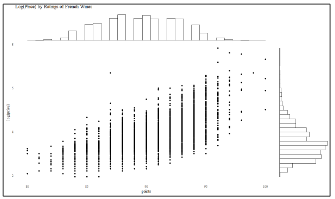
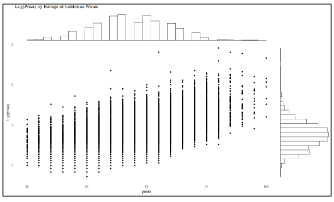
We also ran distributions on the varieties of wines from both France and California to see if there were any overlaps in the types of wines being produced by each region.





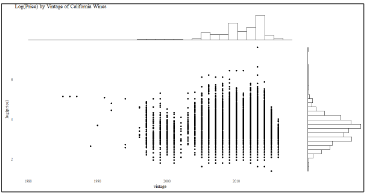
These distributions for the varieties produced in France and California do show that there are wine varieties that are produced by both regions, so it would be possible to compare like wine types by region as well. There bar graphs of wine produced also show that France produces a lot of specific wines whereas California produces a wide range of wines and only mass producing 2-3 types. This leads us to wonder whether the main difference between the two has something to do with market goals or if the wines that are being produced in large amounts are the wines scoring the highest or are they cheapest? We believe further analysis would be able to answer some of these questions.

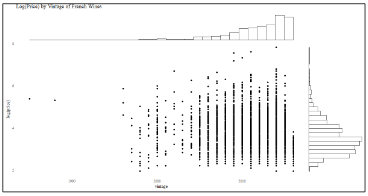
While observing the price distributions, we noticed that it appeared that prices increased logarithmically with respect to their point values. So, we looked at these to find this to be true.

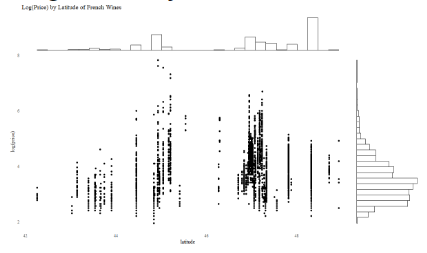
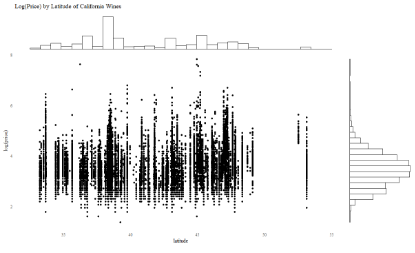


(Log-price by rating, French wines) (Log-price by rating, California wines)

With this in mind, we looked at the log-price with respect to the vintage and found no correlation with the price with respect to the wine’s vintage. We also looked at the log-price with respect to latitude and found no correlation in the price with respect to the latitude.



 (Log-price by vintage, French wines) (Log-price by vintage, California wines)

 (Log-price by latitude, France) (Log-price by latitude, California)

After making some initial observations based on the data, we moved forward to exploratory data analysis. Our team set out to find associations with the price of various wines relative to their taster ratings, growing region, variety, and vintage. Machine learning techniques used for our analysis included random forest and tree classification, and stepwise, partial-least-squares, and lasso regression.

**III. Applying Statistical Models against the Dataset**

**Method France California**

PLS Mean Outsample Error 0.3009265 Mean Outsample Error 0.247108

Components 3 Components 3

Lasso Mean Outsample Error 0.3007747 Mean Outsample Error 0.2477594

CV Lasso Coefs CV Lasso Coefs

(Intercept) -1.291215e+01 (Intercept) -6.308227e+00

points 1.617092e-01 points 9.996102e-02

review\_length -1.222271e-03 review\_length 6.566235e-04

elevation 4.038411e-04 elevation 6.897244e-05

abs\_latitude 4.863186e-02 abs\_latitude 1.949269e-02

Step Mean Outsample Error 0.8089517 Mean Outsample Error 0.5525468

Step Coefs Step Coefs

(Intercept) -1.366806e+01 (Intercept) -6.6027099086

points 1.693462e-01 points 0.1013407916

abs\_latitude 5.180491e-02 abs\_latitude 0.0233231990

review\_length -1.594942e-03 review\_length 0.0007419274

elevation 4.839848e-04 elevation 0.0001137839

**Method France California**

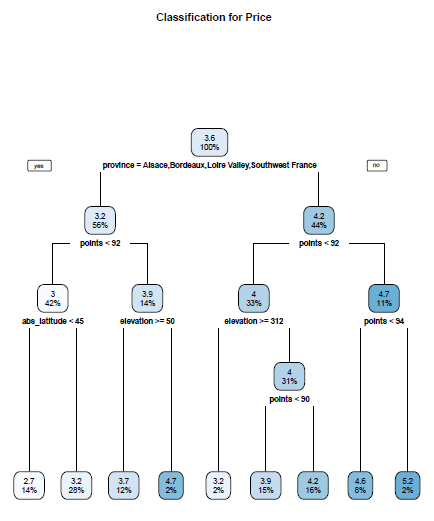
TreeCV Mean Outsample Error 0.1931049 Mean Outsample Error 0.215169

Random Mean Outsample Error 0.1138528 Mean Outsample Error 0.1517635

Forest

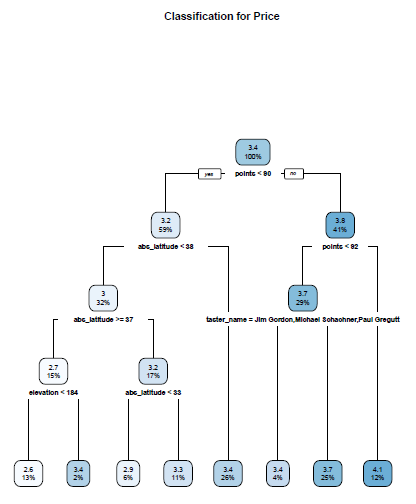
With the models applied to both French and California wines, we notice that our sample error with the step dropped significantly to almost suspicious levels. This most likely means that our tree models are creating a mess of error. Lasso performs well, with points remaining as the largest explainer of variability in log price. If we run the tree models, our mean outsample squared error drops significantly. This likely indicates that there were too many levels in the categorical data, thus causing us to limit the number of unique categorical variables.

When running the TreeCV model, we obtained the following trees:



(TreeCV – French wines predicting log-price)

Interestingly enough, French wines have the most effective split beginning with region.  Our cross-validated tree for classifying log of price suggests that wines within Alsace, Bordeaux, Loire Valley, and Southwest France are less expensive (Burgandy and Champagne are the others).  More importantly, this is a better classifier than first splitting with points, which we continuously see as a strong indicator of price in other models.



(TreeCV – California wines predicting log-price)

The highest split for California wines is at points < 90, which makes sense since one of our best predictors of log\_price is points.  One thing to note though is that on the left side of the tree, we split by abs\_latitude three times, which may be overfitting the groups.  These groups translate to northern California (38th), the San Francisco area (> 37th, < 38th), and southern California (LA, San Diego) (<33rd).   It indicates that wines in northern California are more expensive, but like I said, maybe overfitting.  On the right branch we see taster\_name as a split for wines with points < 92. This is interesting to suggest that the taster themselves may indicate the price of wine.  For example, only the top wines are tasted by the top tasters.

Random forest barely reduces the outsample error and loses the interpretability of the cv tree model, so this doesn’t help us much.

IV. Findings and Conclusions.

The random forest produced the best results for determining price; however, the tree using cross validation was very close and took less time while providing better interpretability than did the random forest. Both the random forest and the cross validated tree were much better at predicting price which is what would be expected with the use of many categorical variables that do not follow a linear model.

From the TreeCV on French wines, the highest node purity came on province, with this being the first split. The highest wine prices came from the two French regions, Champagne and Burgundy. This leads us to believe that higher priced French wines may be based on region. The next separation was based on points, which is what we would expect that higher rated wines would produce higher costs.

From the TreeCV on California wines, the highest node purity came on points, with this being the first split. This is what we initially would expect that higher rated wines would cost more. This leads us to believe that higher priced California wines are based on quality. This tree also separated the California wines based on region, Northern California, San Francisco, and Los Angeles.

After comparing the TreeCV on French and California wines, our initial question might be, “Are French wines priced on where they come from whereas California wines are priced based on quality?” Overall, we were satisfied with the performance of the TreeCV and Random Forest as a predictive model for determining the price of a French or California wine.

With more time and additional analysis, we might be able to determine if French wines are truly priced based on locations and if California wines are priced based on rating. Another area that could be further analyzed would be comparing similar varieties from these two regions to determine, for example, if French Pinot Noir’s are of better quality and/or higher priced than California Pinot Noir’s.

Resources

**Kaggle – Wine Review**

<https://www.kaggle.com/zynicide/wine-reviews>

**R Software** -  free software environment for statistical computing and graphics

**R Script**

library(stringr)

library(jsonlite)

library(plyr)

library(tree) # trees

library(randomForest) # randomForest

library(gbm) # gradient boosting

library(e1071) # svm

library(glmnet) # ridge/lasso

library(pls) # pls

setwd('/Users/christopherewald/Documents/Math541/Project/')

developer\_key = 'AIzaSyDk743GZpF2BUasrshdc7vxTp-uw8NRTgs'

wine.raw = read.csv('winemag-data-130k-v2.csv')

column\_locations = c("country","province","region\_1","region\_2")

wine = cleanWineData(wine.raw, c("country", "points", "price", "province", "region\_1", "region\_2", "variety", "winery", "taster\_name"))

wine = joinLatLngElev(wine, column\_locations,'region\_1') # merge lat lng values by province

# limit the data set to the most prevalent categorical values

variety\_counts = count(wine$variety)

province\_counts = count(wine$province)

highest\_freq\_variety\_counts = tail(variety\_counts[order(variety\_counts$freq),],20)$x

highest\_freq\_provinces = tail(province\_counts[order(province\_counts$freq),],20)$x

# wine with the most most prevalent varieties and provinces

wine = wine[(wine$variety %in% highest\_freq\_variety\_counts) & (wine$province %in% highest\_freq\_provinces),]

wine = droplevels(wine) # hate R...so much hate

wine$price = log(wine$price)

# select a subset by location

wine.cali = wine[wine$province == 'California',]

wine.france = wine[wine$country == 'France',]

### Interesting Plots

plot(wine.cali$points, wine.cali$price)

plot(wine.cali$points, wine.cali$log\_price)

plot(wine.cali$review\_length, wine.cali$log\_price) # the number of characters in the review, no relation

plot(wine.cali$province, wine.cali$price) # doesn't look like any relation between price and province

plot(wine.cali$variety, wine.cali$price) # doesn't look like any relation between price and variety

plot(wine.cali$east\_of\_atlantic, wine.cali$price)

data = droplevels(wine.france[c("points","price","province","taster\_name","review\_length","elevation","abs\_latitude")])

### Splitting test and training sets

set.seed(1)

n = dim(data)[1]

train\_ix = sample(seq\_len(n), size=0.75\*n)

test\_ix = (-train\_ix)

### Predicting Price with other quantitative variables ###

getStep(data[train\_ix,], data[test\_ix,])

getPLS(data[train\_ix,], data[test\_ix,])

getLasso(data[train\_ix,], data[test\_ix,])

getRandomForest(data[train\_ix,], data[test\_ix,])

data.train = data[train\_ix,] # not sure why this needs to be here, looks like R bug

getTreeCV(data[train\_ix,], data[test\_ix,])

#####################################

# Only need to use this if we want to pull in more latlng data

unique\_locations = getUniqueLocations(wine.clean, column\_locations)

collectLatLngData(unique\_locations)

########### Analysis Functions ###############

getStep = function(data.train, data.test)

{

null = lm(price~1, data=data.train)

full = lm(price~., data=data.train)

fwd.lm = step(null, scope=list(lower=null, upper=full), direction="forward")

bwd.lm = step(full, direction="backward")

#summary(fwd.lm)

#summary(bwd.lm)

step.pred = predict(fwd.lm, newx=data.test[,-1])

error = mean((step.pred-data.test[,1])^2)

cat("Min Outsample Error", error, "\n")

cat("Step Coefs","\n")

print(fwd.lm$coefficients)

}

getRidge = function(data.train, data.test)

{}

getLasso = function(data.train, data.test)

{

grid = 10^seq(10,-2,length=100)

lasso.mod = glmnet(as.matrix(data.train[,-1]),data.train[,1],alpha=1,lambda=grid)

cv.out = cv.glmnet(as.matrix(data.train[,-1]),data.train[,1],alpha=1)

bestlam = cv.out$lambda.min

lasso.pred = predict(lasso.mod,s=bestlam,newx=as.matrix(data.test[,-1]))

error = mean((lasso.pred-data.test[,1])^2)

out = glmnet(as.matrix(data.train[,-1]),data.train[,1],alpha=1,lambda=grid)

lasso.coef = predict(out,type="coef",s=bestlam)[1:length(data.train),]

plot(lasso.mod,lwd=2)

plot(cv.out)

cat("Min Outsample Error", error, "\n")

cat("CV Lasso Coefs","\n")

print(lasso.coef)

}

getPLS = function(data.train, data.test)

{

pls.fit = plsr(price~.,data=data.train,validation="CV",scale=T)

#summary(pls.fit)

validationplot(pls.fit,val.type="MSEP")

k = dim(data.train)[2]-1

val.errors.pls<-rep(NA,k)

for (i in 1:k)

{

pls.pred = predict(pls.fit,data.test[,-1],ncomp=i)

val.errors.pls[i] = mean((pls.pred-data.test[,1])^2)

}

j = which.min(val.errors.pls)

min\_error = val.errors.pls[j]

cat("Min Outsample Error", min\_error, "Components", j, "\n")

data.test['predicted\_price'] = pls.pred

}

getSVMLinear = function(data, cost, verbose=FALSE)

{

tuning\_costs = c(.1,1,10,100,1000)

tuning\_gammas = c(0.5,1,2,3,4)

wine.svm<-svm(country~.,data=data,kernel="linear",cost=1,scale=T)

tune.out<-tune(svm,country~.,data=data,kernel="linear",ranges=list(cost=tuning\_cost))

return(tune.out)

}

getSVMRadial = function(data, cost, gamma, verbose=FALSE)

{

tuning\_costs = c(.1,1,10,100,1000)

tuning\_gammas = c(0.5,1,2,3,4)

pima.tr.svm<-svm(type~.,data=data,kernel="radial",cost=1,gamma=1,scale=T)

tune.out<-tune(svm,type~.,data=data,kernel="radial",ranges=list(cost=tuning\_cost,gamma=tuning\_gamma))

return(tune.out)

}

getTreeCV = function(data.train, data.test)

{

data.tree = tree(price~., data=data.train)

data.tree.cv = cv.tree(data.tree, FUN=prune.tree)

data.tree.prune = prune.tree(data.tree, best=data.tree.cv$size[which.min(data.tree.cv$dev)])

plot(data.tree.prune, main="Classification for Price")

text(data.tree.prune, all=TRUE, cex=.8)

data.tree.prune.pred = predict(data.tree.prune, data.test, type='vector')

error = mean((data.tree.prune.pred-data.test[,1])^2)

cat("Min Outsample Error", error, "\n")

}

getRandomForest<-function(data.train, data.test, mtry=3, importance=FALSE)

{

data.rf<-randomForest(price~.,data=data.train,mtry=mtry,importance=importance)

data.rf.pred<-predict(data.rf,newdata=data.test)

error = mean((data.rf.pred-data.test[,1])^2)

cat("Min Outsample Error=", error, "\n")

}

# better for categorical response? allows for larger set categorical indep vars

# this doesn't seem to work the way I would expect it to...

getGBM<-function(data.train, data.test, interaction=1, shrinkage=0.001)

{

data.gbm<-gbm(price~.,data=data.train,

distribution="gaussian",interaction.depth=interaction,

shrinkage=shrinkage)

data.gbm.pred<-predict(object=data.gbm,newdata=data.test,

n.trees=100,type="response")

return(data.gbm.pred)

}

getKMeans = function(data)

{}

############# Data Manipulation ################

cleanWineData = function(data, column\_names)

{

data.out = data[,column\_names]

data.out['review\_length'] = apply(matrix(wine.raw$description), MARGIN=1, FUN=nchar)

data.out = na.omit(data.out)

removed = dim(data)[1]-dim(data.out)[1]

cat("Removed",removed,"NaNs",removed/dim(data)[1],"percent")

return(data.out)

}

joinLatLngElev = function(data, column\_locations, query\_location\_str)

{

if (!match(query\_location\_str, column\_locations))

{

return(NA)

}

latlng = read.csv('latlng.prod.csv')

merged\_data = merge(x=data, y=latlng, by.y='location', by.x=query\_location\_str, all.x=TRUE)

merged\_data = na.omit(merged\_data)

merged\_data$unique\_loc\_ix = NULL

merged\_data$abs\_latitude = abs(merged\_data$latitude)

merged\_data$east\_of\_atlantic = merged\_data$longitude < -30

removed = dim(data)[1]-dim(merged\_data)[1]

#cat("Removed",removed,"NaNs",removed/dim(data)[1],"percent")

return(merged\_data)

}

getWineNumericalValues = function(data, quantitative\_variables)

{

data = data[, quantitative\_variables]

data$points = as.numeric(data$points)

return(data)

}

getWineCategoricalValues = function(data)

{}

numExtract = function(string){

return(as.numeric(str\_extract(string, "2[0-9]{3}")))

}

getLatLng = function(location)

{

html\_loc = gsub("\\s+", "+", location)

html\_str = sprintf('https://maps.google.com/maps/api/geocode/json?address=%s&sensor=false&key=%s', html\_loc, developer\_key)

cat(html\_str, "\n")

data = fromJSON(html\_str)

lat = as.numeric(data$result$geometry$location$lat[1])

lng = as.numeric(data$result$geometry$location$lng[1])

return(c(lat,lng))

}

getAltitude = function(latlng)

{

html\_str = sprintf('https://maps.googleapis.com/maps/api/elevation/json?locations=%s,%s&key=%s', latlng[1], latlng[2], developer\_key)

cat(html\_str, latlng, "\n")

if (is.na(latlng[1]) || is.na(latlng[2]))

{

elevation = NA

}

else

{

data = fromJSON(html\_str)

elevation = as.numeric(data$results$elevation)

}

return(elevation)

}

getUniqueLocations = function(data, cols\_with\_locations)

{

unique\_locations<-c()

for (col in cols\_with\_locations)

{

values = as.vector(unlist(unique(data[col])))

unique\_locations = c(unique\_locations, values)

}

return(sort(unique\_locations[unique\_locations != ""]))

}

collectLatLngData = function(unique\_elements, filename='latlng.csv', max\_errors=3)

{

i = 1

errors = 0

data<-NULL

while (i <= length(unique\_elements))

{

latlng = getLatLng(unique\_elements[i])

elevation = getAltitude(latlng)

str = paste(i, unique\_elements[i], latlng[1], latlng[2], elevation, sep=',')

cat(str,"\n")

Sys.sleep(.1)

if (length(latlng) == 0) # query came up empty, sleep and try again

{

errors = errors + 1

if (errors > max\_errors) # giveup if we exceed max tries

{

i = i + 1

data<-c(data,c(i,latlng))

errors = 0

}

Sys.sleep(1)

next

}

data<-c(data,c(i, latlng))

write(str, file=filename, append=TRUE)

i = i+1

errors = 0

}

return(data)

}