1 Exploratory Data Analysis

1. **Relatively how many terms appear in exactly one document?**
   Total number terms that appear once is: 48821/79653 or about 61% of the terms.
2. **Relatively how many terms appear at least 5 times?**
   There are 12635/79653 or 15% of terms appear more than five times
3. **Show the 20 most frequent words. Words like “and”, “to”, “the” aren’t especially meaningful. Which is the first word that you feel may be meaningful for characterizing the nonprofit? Why?**
   The top 20 I got are “and to the of provid for in educ a communiti servic is promot organ support none with see hous through”. The first few words make sense, they are all very common, others make sense in this context like communiti, servic, support all which describe functions of non-profits. The first work I think would be provid, being that these are non-profits it makes sense for this to be very common, as it is what they do.
4. **How many documents contain “sacramento”?**
   'Sacramento' shows up in 152 documents
5. **What’s one element in `irs990extract` where the mission contains “sacramento”?**
   Element #498's mission contains the word "sacramento", it is for the Sacramento Public Library Foundation
6. **How many non-profits have revenue over $100 million?**
   3612 non-profits have revenue above $100000000

2 Selecting a Subset

What do you do when your program doesn’t run? Try using a subset of the data, the most important subset.

1. **Use one or more of the fields in `irs990extract` to define and pick the 10,000 largest nonprofits.**
2. **What’s the largest nonprofit based on your definition? Does it seem reasonable?**
   Based on my definition Kaiser is the largest non-profit by revenue, which seems completely reasonable with the cost of healthcare in the US.
3. **Drop all words that don’t appear at least twice in this subset.**
   I found 4995 words with 2 or more occurrences

We’ll use this subset for the remainder of the assignment.

3 Principal Components Analysis

Fit the first 10 principal components, i.e. project the data down into a 10 dimensional subspace.

1. **Interpret the principal ratio. What does it mean?**
   My principle ratio of 0.5524287887912542 means that about 55% of the variance was kept when reducing 10000 dimensions to 10 dimensions. So there are 10 transformed dimensions made up of all the words that can be used to distinguish most of the variance in non-profits.
2. **Plot the variances of the first 10 principal components as a function of the principal component number. What do you observe?**
   ```
   1  0.018677254311293717
   2  0.006369477630804123
   3  0.003155583797237752
   4  0.0022243210679529475
   5  0.0017170860248846575
   6  0.0015222223433784144
   7  0.0011264142597841758
   8  0.0009674686194267264
   9  0.0008739713585524194
   10 0.0006767934257638501
   ```
   I couldn’t graph this out on AWS EC2, however, I was able to determine from this table that the first few principal components are holding most of the variance for the PCA.
3. **Which words have the relatively largest loadings in the first principal component? (These the absolute values of the entries of `projection()`.) Are these the kinds of words you expected? Explain.**
   "Teacherscholar" "fourpart" "1906" "middlesex" "multiservic" "nonresidenti" "abort" "steadi" "1962" "attorney" "genom" "depaul" "r" "degener" "macular" "qualtex" "registri" "fifth" "1904" "pro".
   These are terms that I would expect, they are very specific, like teacherscholar which would appear in many school related terms but is fairly general.

4 Clustering

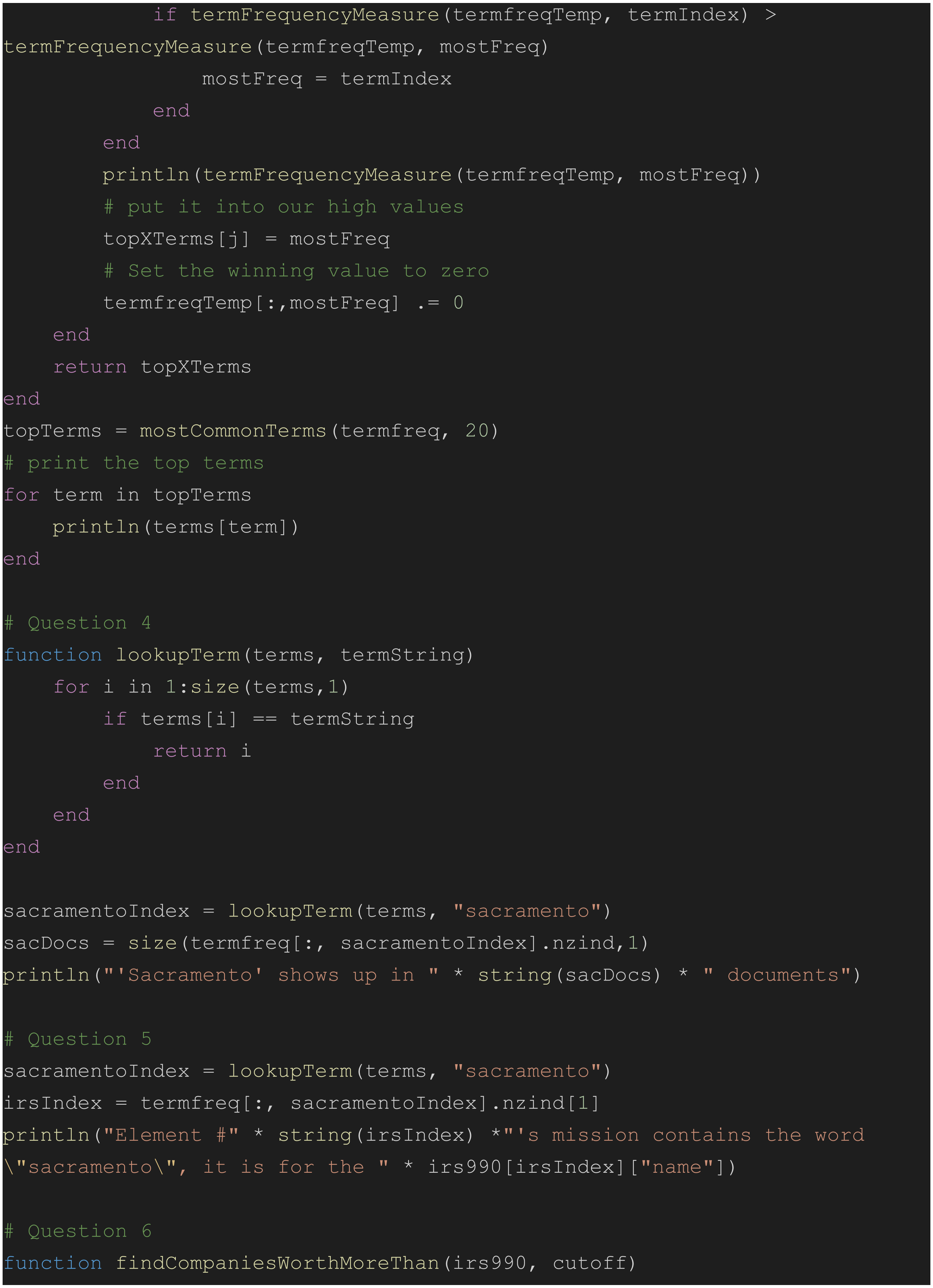
Apply k means with k = 3 to the principal components of the subset of data. This means you should be fitting k means to a data matrix with 10,000 observations, and 10 features, which are the scores for each of the 10 principal components.

1. **How many elements are in each group?**
   Size of group 1 is 9354

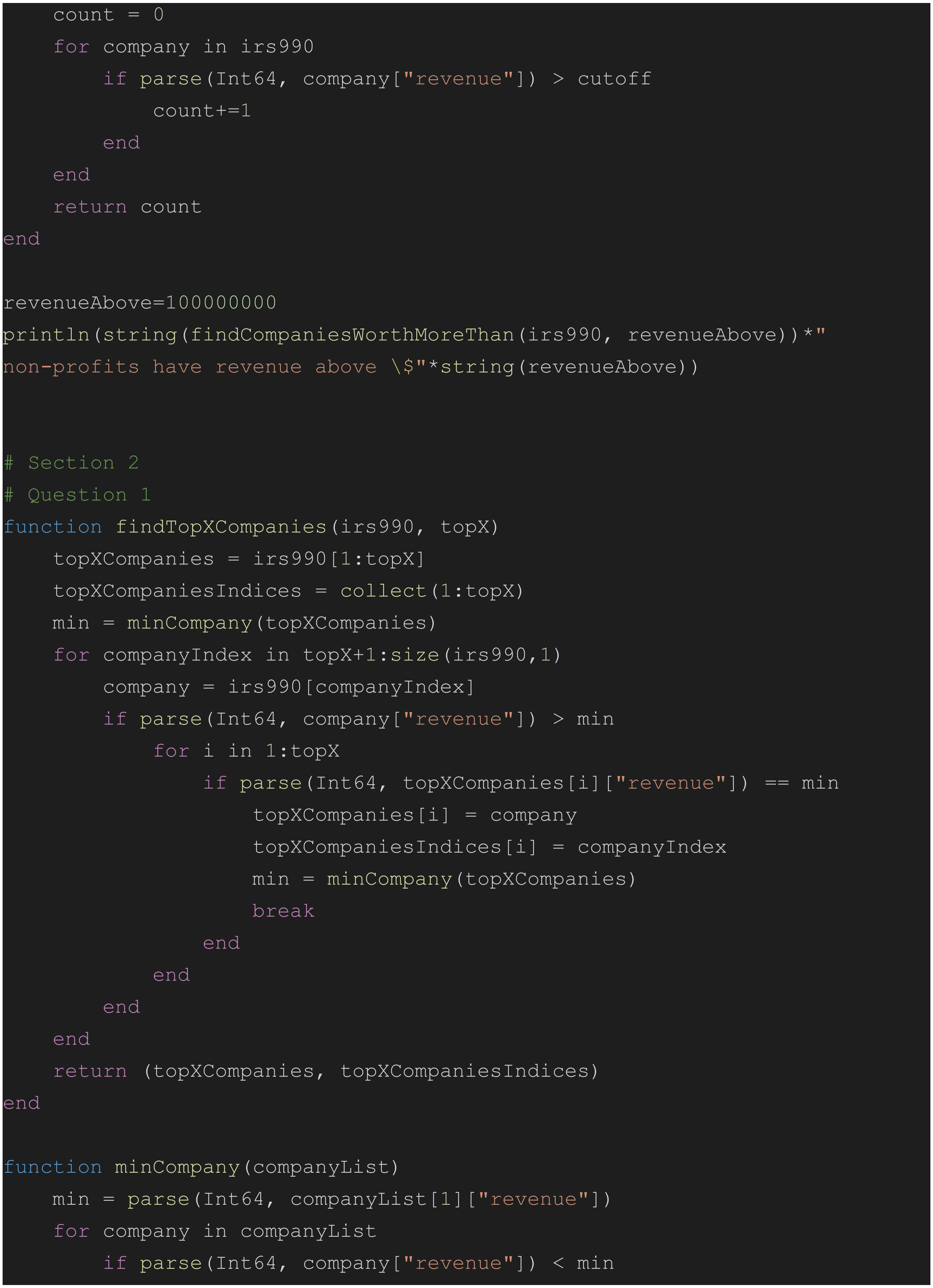
Size of group 2 is 63
Size of group 3 is 583

2. **Which nonprofits are closest to the centroids? Feel free to use the function below.**
   "Hospice of the Comforter Inc" "PEDIATRIC ACADEMIC ASSOCIATION INC" and "Planned ParenthoodOrange &"
3. **k means should find a group of mission statements that are very similar. What happened? Is it reasonable? If we were to continue this analysis, what would you do next?**
   The groups seem to be based on length of the mission and the case. For group one it has a full length mission statement with generally lower case words. For the second group it was a longer statement with uppercase words. For the final group it was made up of very short mission statements.

```
using Serialization
using TextAnalysis
using MultivariateStats
using Plots
using Distributions
using Clustering
terms = Serialization.deserialize("terms.jldata")
termfreq = Serialization.deserialize("termfreq.jldata")
irs990 = Serialization.deserialize("irs990extract.jldata")

# Section 1
# Question 1
function termsThatAppearXTimes(termfreq, x)
    termAppearances = 0
    for i in 1:size(termfreq, 2)
        term = termfreq[:, i]
        if size(term.nzval,1) == 1
            termAppearances += 1
        end
    end
    return termAppearances
end

println("Total number terms that appear once is: " *
string(termsThatAppearXTimes(termfreq, 1)))

# Question 2
function howManyTermsAppear(termfreq, moreThanX)
    totalTerms = 0
    for i in 1:size(termfreq, 2)
        col = termfreq[:, i]
        if size(col.nzval,1) > moreThanX
            totalTerms += 1
        end
    end
    return totalTerms
end

println("There are " * string(howManyTermsAppear(termfreq, 5)) * " terms
that appear more than five times")

# Question 3
function termFrequencyMeasure(termfreq, termIndex)
    term = termfreq[:, termIndex]
    total = 0
    for i in term.nzval
        total += i
    end
    return total
end

function mostCommonTerms(termfreq, topX)
    termfreqTemp = termfreq
    # get the first topX values from termfreq
    topXTerms = collect(1:topX)
    for j in 1:topX
        # loop through every element to find the highest and remove it
        mostFreq = 1
        for termIndex in 1:size(termfreqTemp, 2)
            term = termfreqTemp[:, termIndex]
```

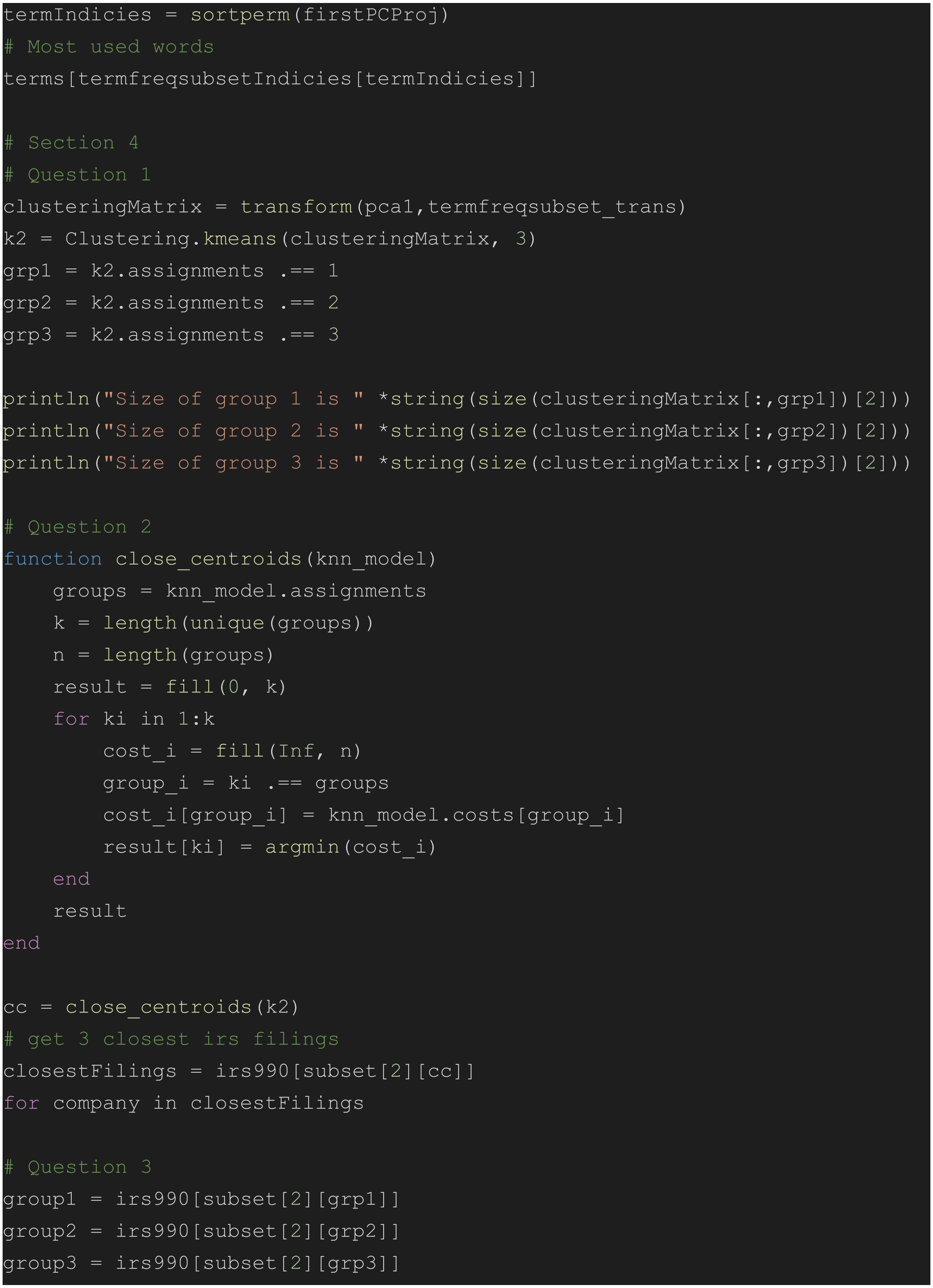


```
            if termFrequencyMeasure(termfreqTemp, termIndex) >
termFrequencyMeasure(termfreqTemp, mostFreq)
                mostFreq = termIndex
            end
        end
        println(termFrequencyMeasure(termfreqTemp, mostFreq))
        # put it into our high values
        topXTerms[j] = mostFreq
        # Set the winning value to zero
        termfreqTemp[:,mostFreq] .= 0
    end
    return topXTerms
end
topTerms = mostCommonTerms(termfreq, 20)
# print the top terms
for term in topTerms
    println(terms[term])
end

# Question 4
function lookupTerm(terms, termString)
    for i in 1:size(terms,1)
        if terms[i] == termString
            return i
        end
    end
end

sacramentoIndex = lookupTerm(terms, "sacramento")
sacDocs = size(termfreq[:, sacramentoIndex].nzind,1)
println("'Sacramento' shows up in " * string(sacDocs) * " documents")

# Question 5
sacramentoIndex = lookupTerm(terms, "sacramento")
irsIndex = termfreq[:, sacramentoIndex].nzind[1]
println("Element #" * string(irsIndex) *"'s mission contains the word
\"sacramento\", it is for the " * irs990[irsIndex]["name"])

# Question 6
function findCompaniesWorthMoreThan(irs990, cutoff)
```



```
    count = 0
    for company in irs990
        if parse(Int64, company["revenue"]) > cutoff
            count+=1
        end
    end
    return count
end

revenueAbove=100000000
println(string(findCompaniesWorthMoreThan(irs990, revenueAbove))*"
non-profits have revenue above \$"*string(revenueAbove))


# Section 2
# Question 1
function findTopXCompanies(irs990, topX)
    topXCompanies = irs990[1:topX]
    topXCompaniesIndices = collect(1:topX)
    min = minCompany(topXCompanies)
    for companyIndex in topX+1:size(irs990,1)
        company = irs990[companyIndex]
        if parse(Int64, company["revenue"]) > min
            for i in 1:topX
                if parse(Int64, topXCompanies[i]["revenue"]) == min
                    topXCompanies[i] = company
                    topXCompaniesIndices[i] = companyIndex
                    min = minCompany(topXCompanies)
                    break
                end
            end
        end
    end
    return (topXCompanies, topXCompaniesIndices)
end

function minCompany(companyList)
    min = parse(Int64, companyList[1]["revenue"])
    for company in companyList
        if parse(Int64, company["revenue"]) < min
```

```
            min = parse(Int64, company["revenue"])
        end
    end
    return min
end

function maxCompany(companyList)
    max = parse(Int64, companyList[1]["revenue"])
    for company in companyList
        if parse(Int64, company["revenue"]) > max
            max = parse(Int64, company["revenue"])
        end
    end
    return max
end

subset = findTopXCompanies(irs990, 10000)
function findLargestCompany(irs990temp)
    largest = maxCompany(irs990temp)
    largestCompany = irs990temp[1]
    for company in irs990temp
        if parse(Int64, company["revenue"]) == largest
            largestCompany = company
        end
    end
    return largestCompany
end
println("largest company is " *
string(findLargestCompany(irs990)["name"]))

# Question 3
termfreq = Serialization.deserialize("termfreq.jldata")
termfreqsubset = termfreq[subset[2],:]
function removeTermsThatAppearLessThanX(termfreqtemp, lessThanX)
    keepTerms = []
    matrixSize = size(termfreqtemp,2)
    for i in 1:size(termfreqtemp,2)
        term = termfreqtemp[:, i]
        if size(term.nzval,1) >= 2
            push!(keepTerms,i)
        end
    end
    # return keepTerms
    return termfreqtemp[1:end, keepTerms]
end
function removeTermsThatAppearLessThanXIndex(termfreqtemp, lessThanX)
    keepTerms = []
    matrixSize = size(termfreqtemp,2)
    for i in 1:size(termfreqtemp,2)
        term = termfreqtemp[:, i]
        if size(term.nzval,1) >= 2
            push!(keepTerms,i)
        end
    end
    return keepTerms
end

termfreqsubsetIndicies =
removeTermsThatAppearLessThanXIndex(termfreqsubset, 2)
termfreqsubset = removeTermsThatAppearLessThanX(termfreqsubset, 2)
println("I found " * string(size(termfreqsubset,2)) * " words with 2 or
more occurrences")

# Part 3
# Question 1
# convert to a dense matrix
termfreqsubset = collect(termfreqsubset)
# Transform the data
termfreqsubset_trans = transpose(termfreqsubset)
# Create PCA1
pca1 = fit(PCA, termfreqsubset_trans, maxoutdim = 10)

# Question 2


# Question 3
# create projection of absolute values
proj = projection(pca1)
absProj = abs.(proj)
firstPCProj = absProj[1:end, 1]
```



```
termIndicies = sortperm(firstPCProj)
# Most used words
terms[termfreqsubsetIndicies[termIndicies]]

# Section 4
# Question 1
clusteringMatrix = transform(pca1,termfreqsubset_trans)
k2 = Clustering.kmeans(clusteringMatrix, 3)
grp1 = k2.assignments .== 1
grp2 = k2.assignments .== 2
grp3 = k2.assignments .== 3

println("Size of group 1 is " *string(size(clusteringMatrix[:,grp1])[2]))
println("Size of group 2 is " *string(size(clusteringMatrix[:,grp2])[2]))
println("Size of group 3 is " *string(size(clusteringMatrix[:,grp3])[2]))

# Question 2
function close_centroids(knn_model)
    groups = knn_model.assignments
    k = length(unique(groups))
    n = length(groups)
    result = fill(0, k)
    for ki in 1:k
        cost_i = fill(Inf, n)
        group_i = ki .== groups
        cost_i[group_i] = knn_model.costs[group_i]
        result[ki] = argmin(cost_i)
    end
    result
end

cc = close_centroids(k2)
# get 3 closest irs filings
closestFilings = irs990[subset[2][cc]]
for company in closestFilings

# Question 3
group1 = irs990[subset[2][grp1]]
group2 = irs990[subset[2][grp2]]
group3 = irs990[subset[2][grp3]]
```