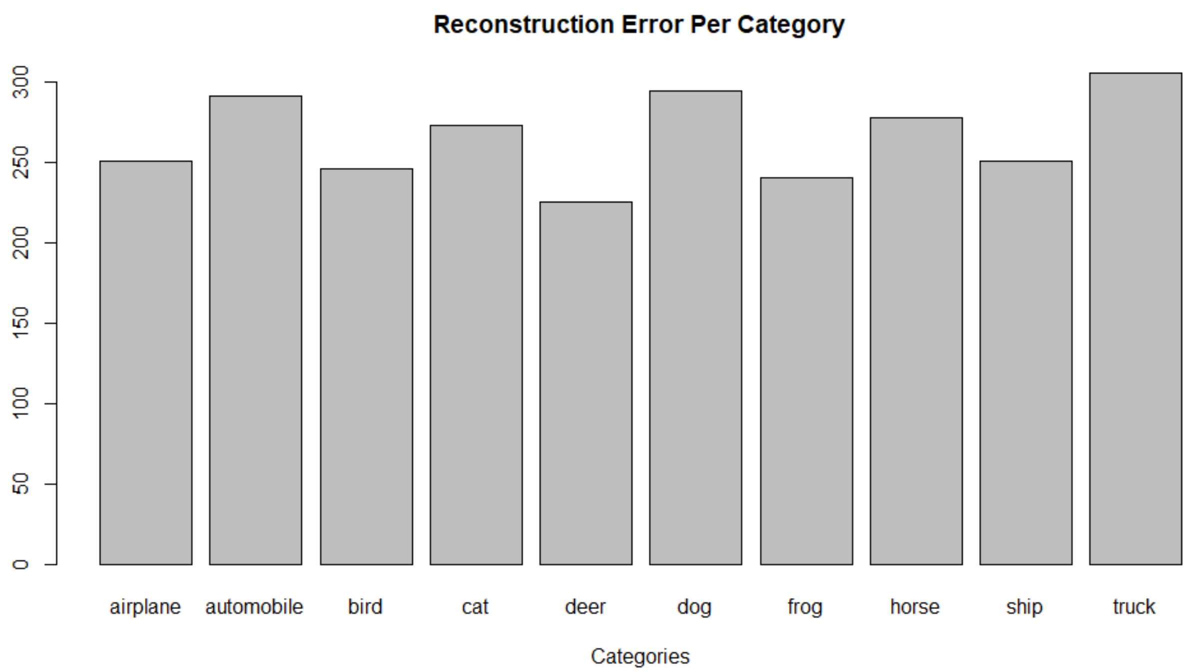
Data Preprocessing

I used an online available code to read the data with some tweaking, code for pre-processing the data is in ReadImages.R file. After reading the training and testing files, data is stored in one file with 1 row per each image, first column represents the label, and then the data in 3072 columns.

Problem

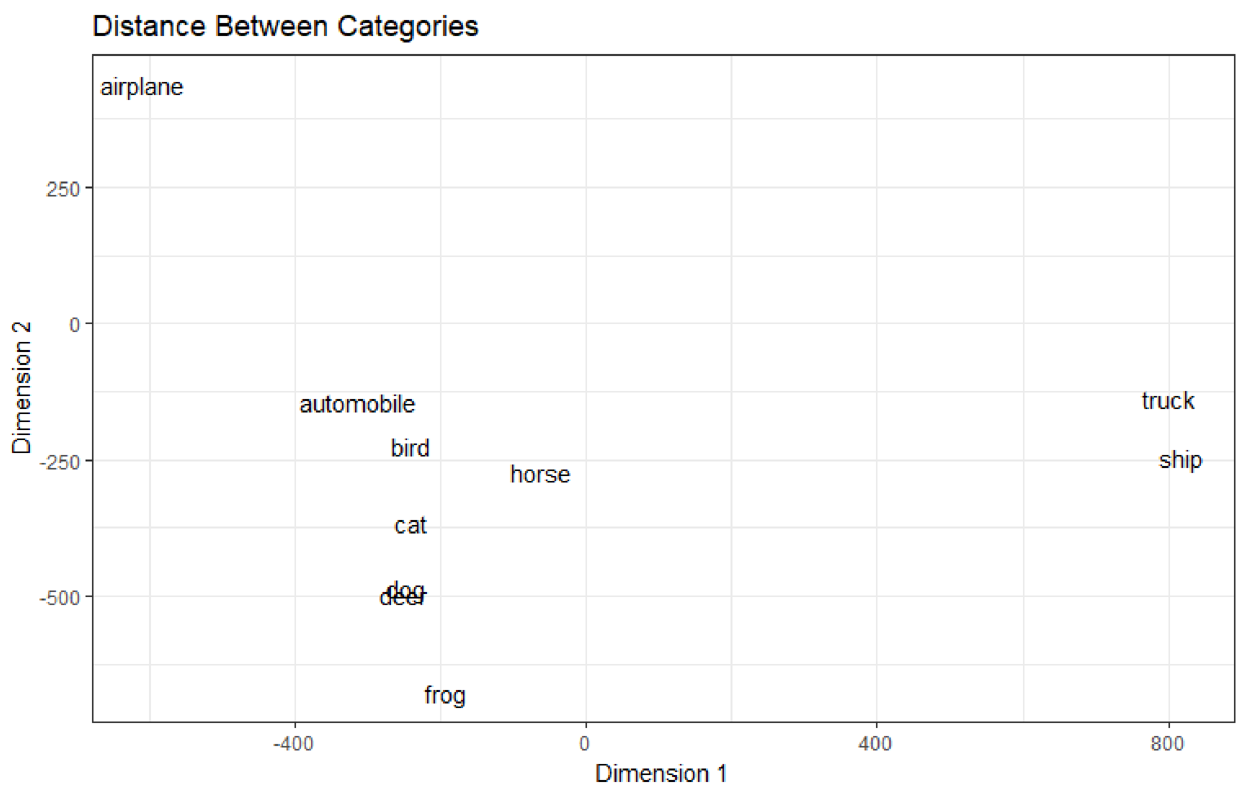
1- PartA.R contains code for first requirement. Using princomp function in R, the reconstruction error is identical between the two possible methods, that's why I used it in part C as well. Princomp, unlike prcomp, is using eigen() to compute the principal components rather than SVD.

Below are the errors using the summation of the eigen values bigger than 21.



2- PartB.R includes the code for this problem, it's logical that cat, dog, and deer are somehow close to each other, however, it seems that the cat had somehow higher weight in this representation to make both dog and deer on an equal distance from her, although their own distance is not that close to each other.

Below is the 2-D diagram and the distance table for this part.



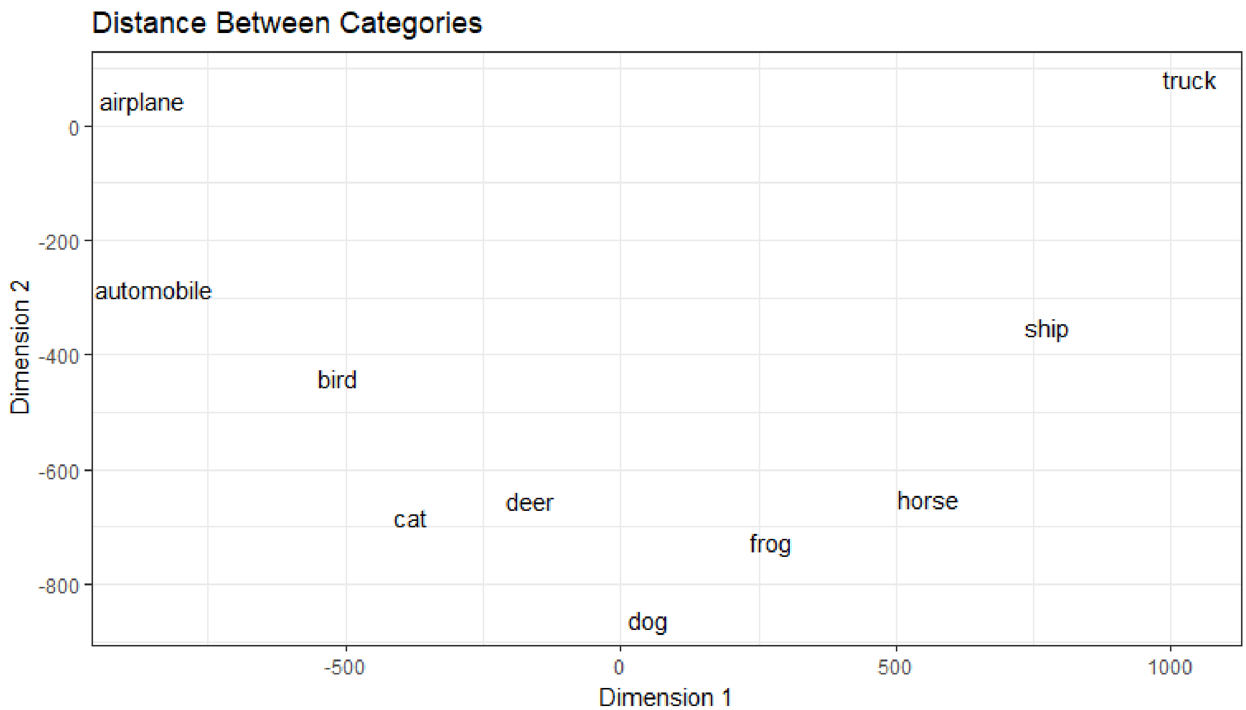
	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0	1683.635	1605.0243	1905.5353	2148.7634	1965.2215	2445.6797	1663.6459	945.5411	1449.0949
automobile	0	0	886.2367	1027.6498	1143.0814	1216.0794	1191.192	950.7861	1303.4665	949.9958
bird	0	0	0	517.3115	601.2503	701.4682	913.7475	418.2763	1557.715	1416.6747
cat	0	0	0	0	469.7917	412.1817	677.492	596.3767	1851.2145	1676.4679
deer	0	0	0	0	0	617.6971	460.5109	684.3469	2065.6217	1830.7409
dog	0	0	0	0	0	0	828.5811	843.6721	1897.5918	1880.2438
frog	0	0	0	0	0	0	0	948.704	2249.1998	1913.2409
horse	0	0	0	0	0	0	0	0	1660.2681	1347.3341
ship	0	0	0	0	0	0	0	0	0	1066.9416
truck	0	0	0	0	0	0	0	0	0	0

3- PartC.R still uses princomp for getting the principal components, note that centering happens automatically and it's available in the output $center, eigen vectors are available in the output $loadings

Notes about solution:

- I ignore principal components bigger than 20 using: temp$loadings[,21:3072] <- 0
- I am using the following line of code to compute the constructed images using another class PCs and the class category mean (tweaking to equation in page 75):
  irestoration <- t(pcas[[j]]$loadings %*% t((as.matrix(l_m_categoriesData[[i]]) - pcas[[i]]$center) %% pcas[[j]]$loadings)) + pcas[[i]]$center*
  Note that if I replace j with i (use same category PCs), and didn't set its higher PCs dimension to 0 (bigger than 20), it will result in the original images (tested that)
- **As an answer to the problem question**, results using this technique are different than using the mean in previous technique, and I think this is due to a few reasons:
  - Using the mean of a category's images as a representation is a bit erogenous, increases its similarity with other categories
  - Representing one category's image using another category's PCs is logical if we want to check how good the reconstruction will be
  - Practical reasons: From the two charts, the second technique was able to make the categories more scattered. Automobile is no longer that close to birds, neither trucks and ships, however, categories that are a bit similar, like dog and deer, are still relatively close.

Below are the 2-D graph and the distances matrix using this technique:



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0	2079.141	1770.067	1910.49	1736.29	1954.714	1883.643	1974.757	1935.983	2254.246
automobile	0	0	2020.04	2097.014	1989.471	2157.909	2074.909	2203.615	2138.522	2359.826
bird	0	0	0	1753.694	1625.787	1773.776	1734.45	1861.326	1903.432	2166.892
cat	0	0	0	0	1772.183	1855.302	1835.15	1960.679	2000.4	2226.993
deer	0	0	0	0	0	1798.909	1725.463	1840.398	1866.581	2150.324
dog	0	0	0	0	0	0	1870.553	1994.736	2067.504	2288.634
frog	0	0	0	0	0	0	0	1962.098	1984.915	2235.082
horse	0	0	0	0	0	0	0	0	2093.87	2342.49
ship	0	0	0	0	0	0	0	0	0	2313.733
truck	0	0	0	0	0	0	0	0	0	0