Visualizing Diversity of Genetic Ancestry Effectively Through High Dimensional Clustering and Data Science

Sydney Greeno

Mentored by Steven Eschrich and Anders Berglund
Moffitt Cancer Center

Although there is a clear scientific difference between the concepts of “race” and “ancestry,” many in the field misunderstand the importance of distinguishing between the two. Particularly for cancer research, highlighting ancestry when presenting data plays a significant role. When determining a solid goal for my project, it became clear that visualization was the key to representing ancestry effectively. This is why my mentors and I decided to create an algorithm in order to come up with solutions.

In order to conceptualize the need for this visual, it is essential to first understand the difference between race and ancestry. Figure 1 outlines the various differences between them that is often confused by viewers. As shown, ancestry is determined by an individual’s genome, which is the preferred method of organization in the case of cancer research.

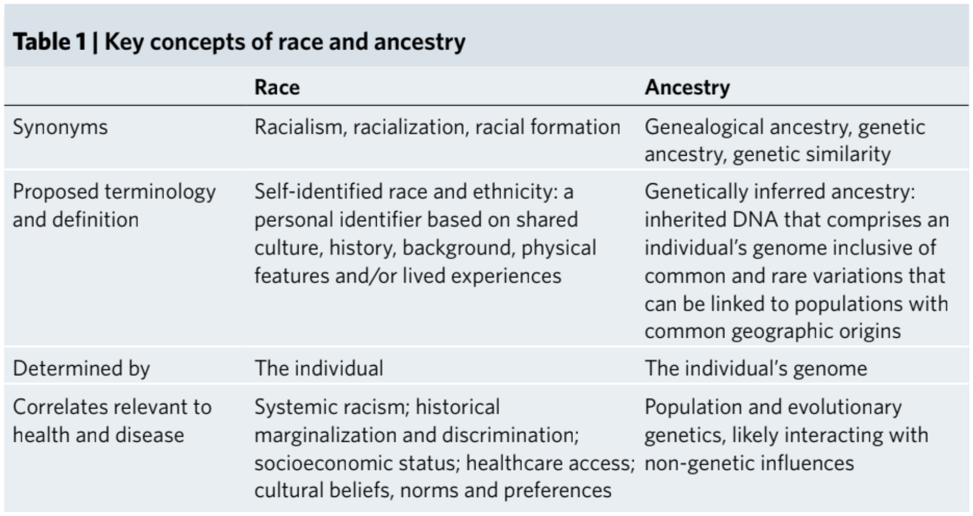


Table 1 | Key concepts of race and ancestry

	Race	Ancestry
Synonyms	Racialism, racialization, racial formation	Genealogical ancestry, genetic ancestry, genetic similarity
Proposed terminology and definition	Self-identified race and ethnicity: a personal identifier based on shared culture, history, background, physical features and/or lived experiences	Genetically inferred ancestry: inherited DNA that comprises an individual’s genome inclusive of common and rare variations that can be linked to populations with common geographic origins
Determined by	The individual	The individual’s genome
Correlates relevant to health and disease	Systemic racism; historical marginalization and discrimination; socioeconomic status; healthcare access; cultural beliefs, norms and preferences	Population and evolutionary genetics, likely interacting with non-genetic influences

Figure 1 Rebbeck, T.R., Mahal, B., Maxwell, K.N. *et al.* The distinct impacts of race and genetic ancestry on health. *Nat Med* 28, 890–893 (2022). https://doi.org/10.1038/s41591-022-01796-1

Moreover, there are not any existing models that demonstrate the distinct impacts of genetic ancestry on health. Figure 2 is an example of a type of model used typically which is often seen as convoluted and difficult to understand. One of the main goals for my project involves the subject of data visualization. Data visualization focuses heavily on models that are easy to understand for as many people as possible. I took this into consideration when deciding how to model the data.

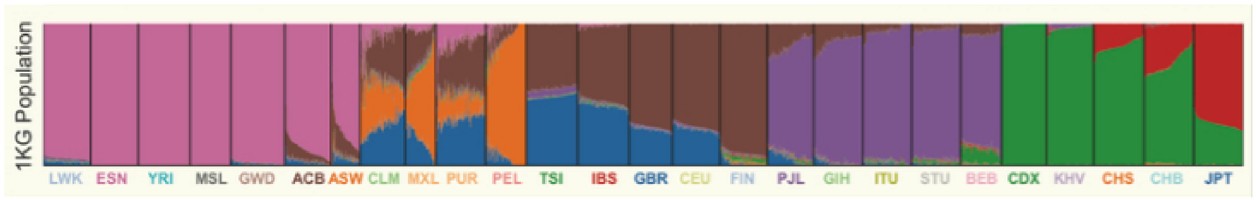
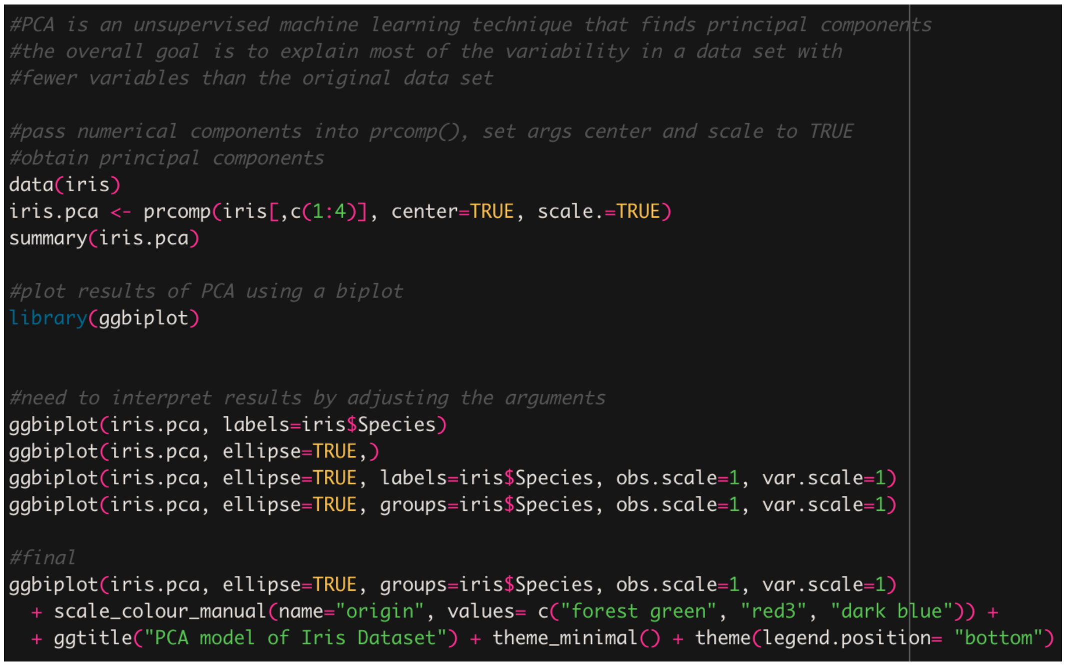


Figure 2 "*An Interactive Resource to Probe Genetic Diversity and Estimated Ancestry in Cancer Cell Lines Julie Dutil, Zhihua Chen, Alvaro N. Monteiro, Jamie K. Teer and Steven A. Eschrich Cancer Res March 20 2019 DOI: 10.1158/0008-5472.CAN-18-2747*

While attempting to find an effective machine learning algorithm, it was necessary to utilize various data sets in order to test the efficiency of each model. The final model would use the 1000 Genomes data set, but first I tested different algorithms with the Iris data set.



```r
#PCA is an unsupervised machine learning technique that finds principal components
#the overall goal is to explain most of the variability in a data set with
#fewer variables than the original data set

#pass numerical components into prcomp(), set args center and scale to TRUE
#obtain principal components
data(iris)
iris.pca <- prcomp(iris[,c(1:4)], center=TRUE, scale.=TRUE)
summary(iris.pca)

#plot results of PCA using a biplot
library(ggbiplot)


#need to interpret results by adjusting the arguments
ggbiplot(iris.pca, labels=iris$Species)
ggbiplot(iris.pca, ellipse=TRUE,)
ggbiplot(iris.pca, ellipse=TRUE, labels=iris$Species, obs.scale=1, var.scale=1)
ggbiplot(iris.pca, ellipse=TRUE, groups=iris$Species, obs.scale=1, var.scale=1)

#final
ggbiplot(iris.pca, ellipse=TRUE, groups=iris$Species, obs.scale=1, var.scale=1)
  + scale_colour_manual(name="origin", values= c("forest green", "red3", "dark blue")) +
  + ggtitle("PCA model of Iris Dataset") + theme_minimal() + theme(legend.position= "bottom")
```

Figure 3

Above is a figure that outlines the exact code I utilized to test whether or not a PCA model would suffice for the goal of the project. PCA is a technique that uses principal components to create new axes in order to minimize the number of variables in a high dimension data set.

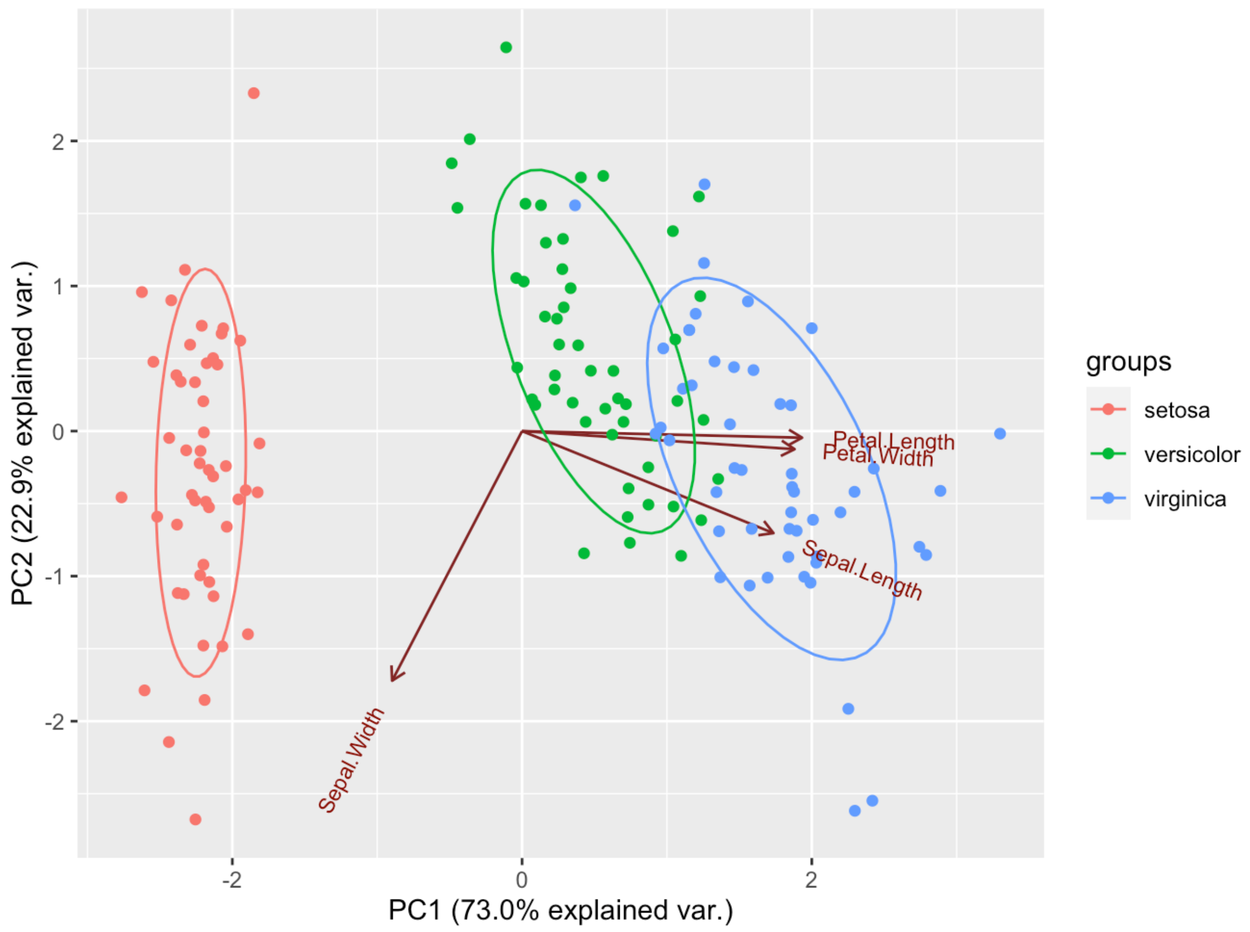
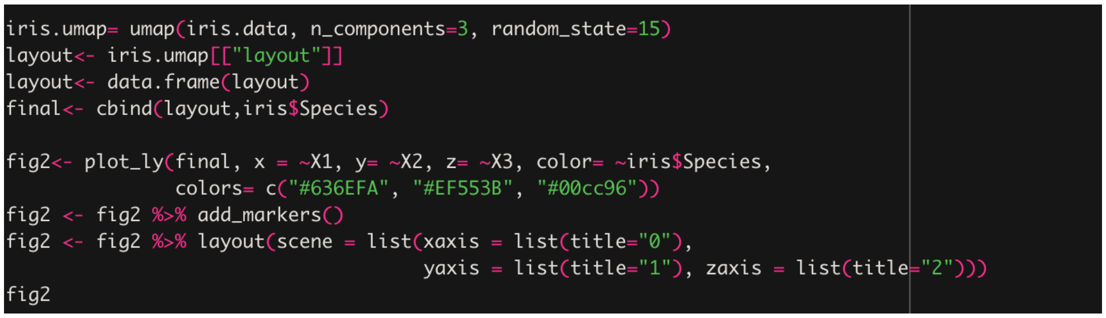


Figure 4

As shown above, the PCA of the iris data set was effective for distinguishing between these three groups. However, I projected that they would not have the same efficiency when representing different populations because of the overlapping of data points. This is shown in the blue and green points in figure 4, which would be overwhelming to look at if given a wider variety of data points.

After ruling that PCA would not be the most suitable algorithm for the best results, I decided to test UMAP, which is a clustering algorithm that runs typically using a combination of PCA and t-SNE.



```
iris.umap= umap(iris.data, n_components=3, random_state=15)
layout<- iris.umap[["layout"]]
layout<- data.frame(layout)
final<- cbind(layout,iris$Species)

fig2<- plot_ly(final, x = ~X1, y= ~X2, z= ~X3, color= ~iris$Species,
               colors= c("#636EFA", "#EF553B", "#00cc96"))
fig2 <- fig2 %>% add_markers()
fig2 <- fig2 %>% layout(scene = list(xaxis = list(title="0"),
                                     yaxis = list(title="1"), zaxis = list(title="2")))
fig2
```

Figure 5

UMAP was considered due to its exceptionally fast runtime and potential for 3D representations as seen below with the Iris data set. It was the newest method and a strong contender for this project.

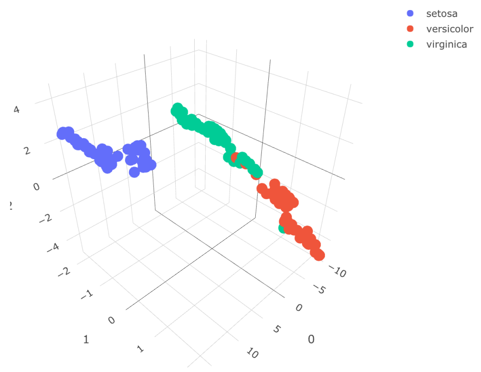


Figure 6

The ultimate reason for not using UMAP became apparent in the use of the final 1000 Genome data set. The visual clustered extremely poorly as seen through the long lines where the algorithm forced neighboring points to be clustered together. Overall, it was clear that the parameters would need to be heavily adjusted for the UMAP model to be valid. This was not the case with the final algorithm, which was used for the final visual.

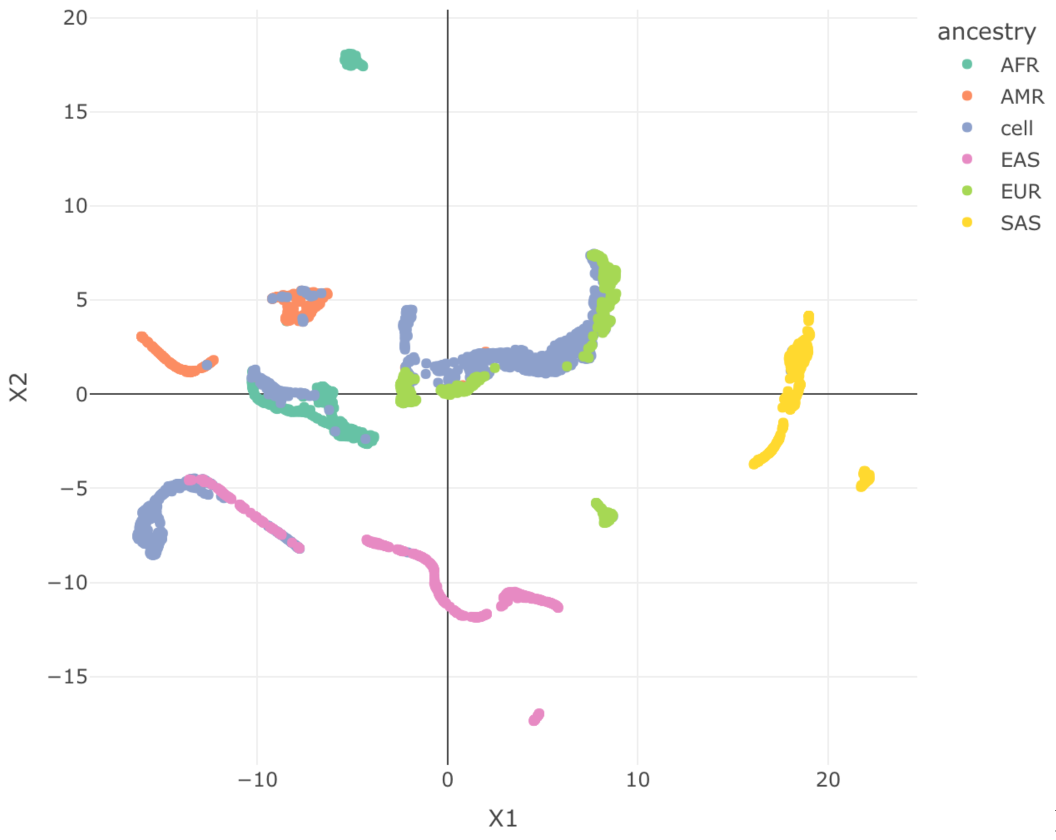
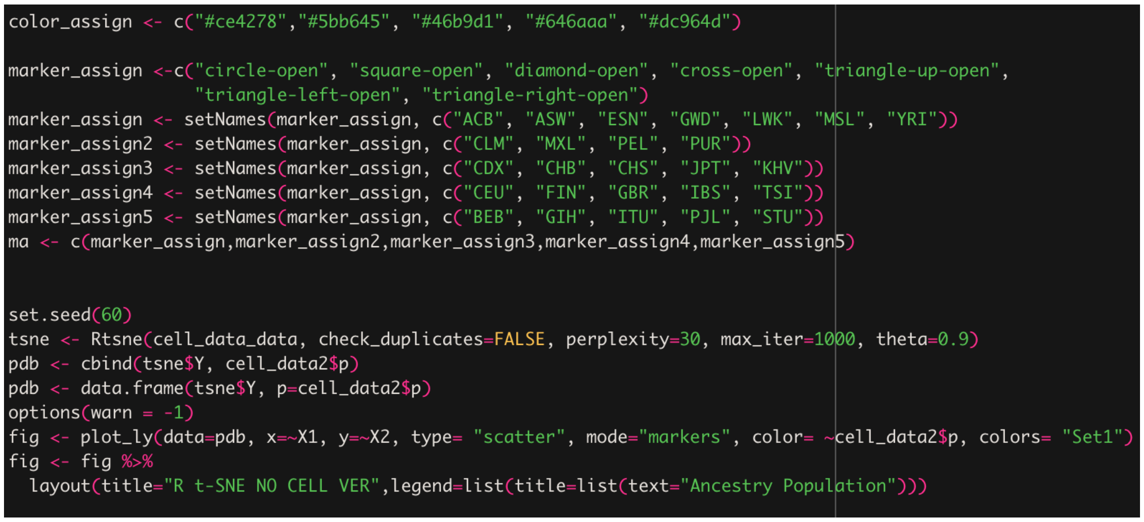


Figure 7

The t-SNE algorithm ended up being the most useful algorithm when attempting to visualize this data. There are a variety of reasons why I chose t-SNE over the other algorithms. The main reason is due to the better looking clustering which would require less parameter changes with the algorithm itself. There is also much more documentation and recorded use of t-SNE within scientific research over UMAP.

After deciding on t-SNE being the main algorithm for the clustering, there were a few aspects of the data that I had to adjust before starting to code one of the final visuals. Setting the seed is a method within R that allows for each run of the algorithm to result in the same iterations. This means that it was essential to test various different seeds until I found the optimal starting place for the one thousand iterations that t-SNE would perform. I tested around one hundred different seeds through trial and error and found that 60 had the best looking graph with minimal forced nearest neighbors. I also had to test different R packages that would perform the t-SNE. Factors I searched for were runtime, parameters and documentation. I found that the optimal package ran through a compiler rather than the tedious R interpreter, which would run about an hour faster than the slowest package.



```
color_assign <- c("#ce4278","#5bb645", "#46b9d1", "#646aaa", "#dc964d")

marker_assign <-c("circle-open", "square-open", "diamond-open", "cross-open", "triangle-up-open",
                  "triangle-left-open", "triangle-right-open")
marker_assign <- setNames(marker_assign, c("ACB", "ASW", "ESN", "GWD", "LWK", "MSL", "YRI"))
marker_assign2 <- setNames(marker_assign, c("CLM", "MXL", "PEL", "PUR"))
marker_assign3 <- setNames(marker_assign, c("CDX", "CHB", "CHS", "JPT", "KHV"))
marker_assign4 <- setNames(marker_assign, c("CEU", "FIN", "GBR", "IBS", "TSI"))
marker_assign5 <- setNames(marker_assign, c("BEB", "GIH", "ITU", "PJL", "STU"))
ma <- c(marker_assign,marker_assign2,marker_assign3,marker_assign4,marker_assign5)


set.seed(60)
tsne <- Rtsne(cell_data_data, check_duplicates=FALSE, perplexity=30, max_iter=1000, theta=0.9)
pdb <- cbind(tsne$Y, cell_data2$p)
pdb <- data.frame(tsne$Y, p=cell_data2$p)
options(warn = -1)
fig <- plot_ly(data=pdb, x=~X1, y=~X2, type= "scatter", mode="markers", color= ~cell_data2$p, colors= "Set1")
fig <- fig %>%
  layout(title="R t-SNE NO CELL VER",legend=list(title=list(text="Ancestry Population")))
```

Figure 8

After performing the t-SNE with these factors taken into consideration, I was able to adjust different parameters to my liking to make the visual as appealing as possible. Not only did I experimented with different perplexity values, which would expand or contract the clusters. In addition, I evaluated the theta value, which can be changed in order to adjust accuracy. I also decided to keep the maximum iterations at the default of one thousand because the preciseness of the visual plateaued at around nine hundred iterations.

Moreover, I became aware of a package with the method of DBSCAN. DBSCAN uses the K Nearest Neighbors algorithm which measures clusters through distance. DBSCAN allows for the labeling of subpopulations which were included in the 1000 Genomes data set. This differentiation between super and subpopulations was a challenge to represent effectively in one visual. At this point it was an experiment of trial and error with different methods.

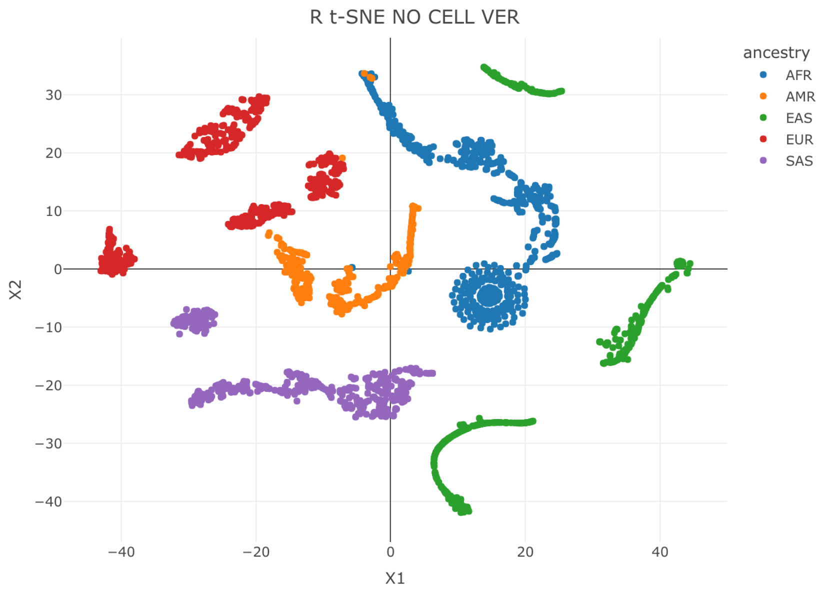


Figure 9

Figure 9 demonstrates the original t-SNE with no adjustment of parameters or consideration over the visual aesthetic. When compared with the analyzed version of the data in figure 10, it becomes clear how important data visualization is in order to properly understand what the data scientist is meaning to convey. Each of the symbols and colors were altered in order to find the maximal differences in color and shape to differentiate super populations and subpopulations. I utilized various resources in order to choose these. One of which is the open source project iWantHue, which is designed to generate color palettes for different audiences. I handpicked the optimal colors from this resource and compiled them together to complete the visual in figure 10.

In figure 10 the alternating symbols represent subpopulations within super populations that are represented through color. For example, AFR represents the super population of Africa and ACB is African Caribbean in Barbados specified through a pink circle.

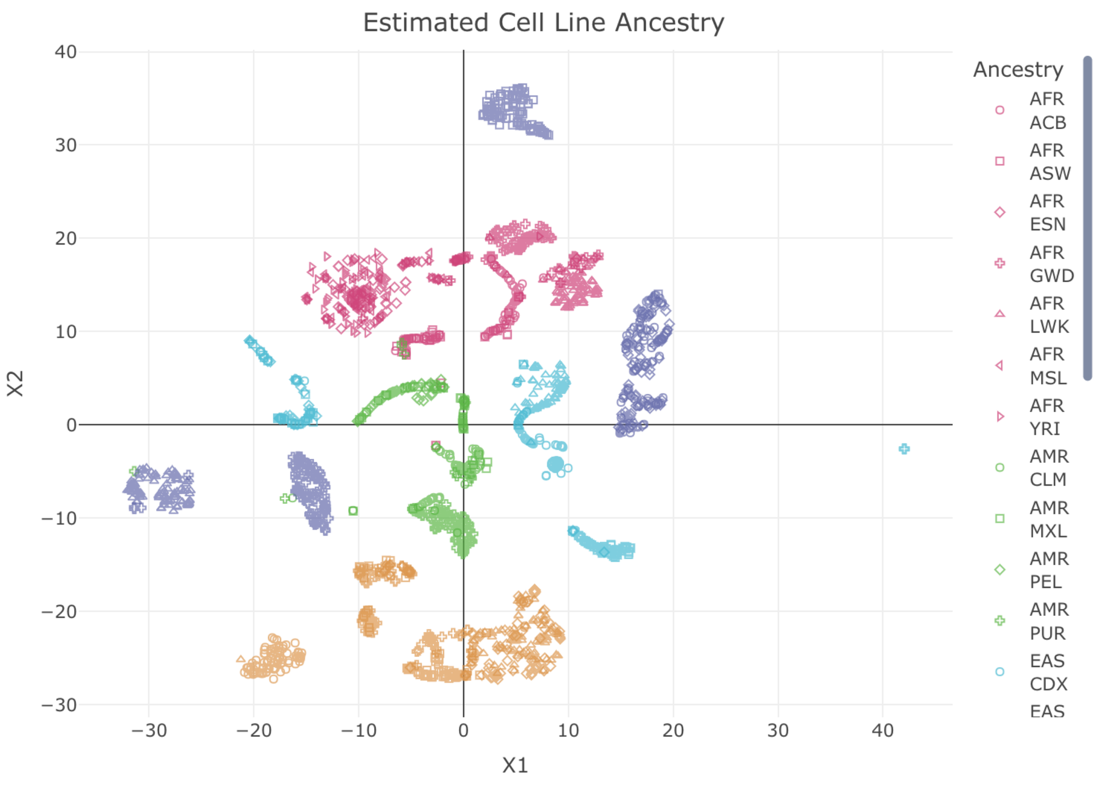
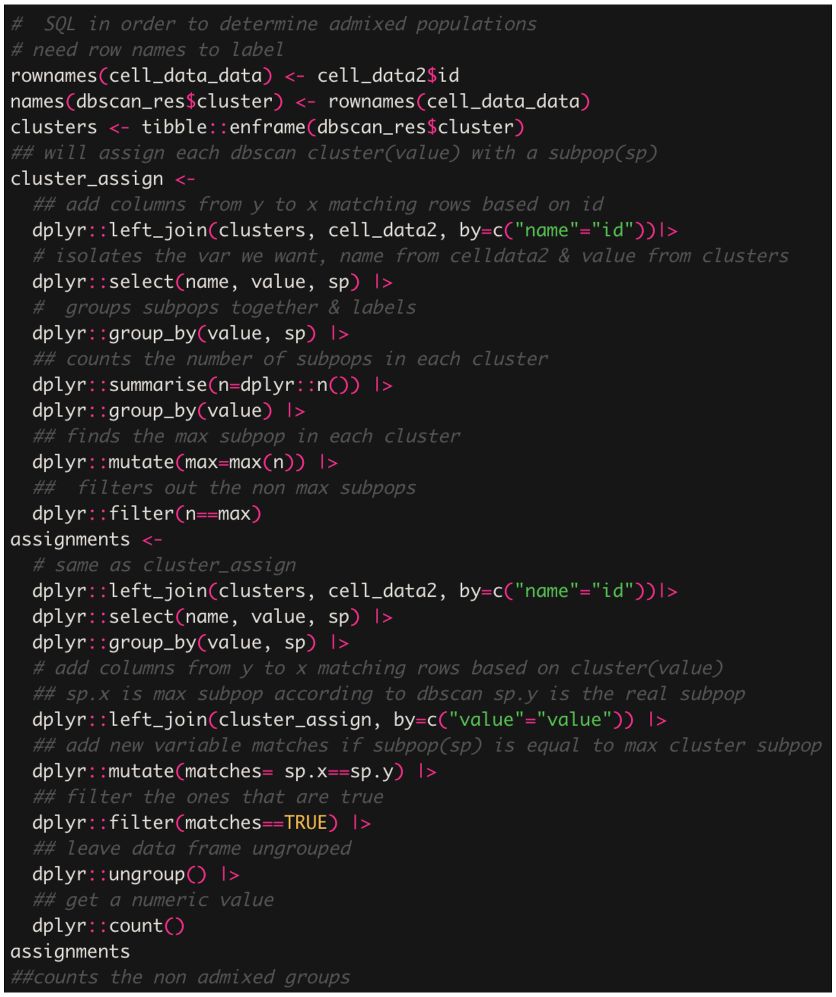


Figure 10

This representation effectively demonstrates how populations can be sorted into clusters represented through genetic data collected through samples. However, I knew the algorithms that I used could be put to further use for determining mixed populations.

I used the R package tidyverse in order to simulate SQL using dplyr and become able to manipulate the data frames in order to measure whether a population in a cluster is admixed. I was able to use the dplyr code in order to find the number of samples who were correctly sorted using DBSCAN.



```
#  SQL in order to determine admixed populations
# need row names to label
rownames(cell_data_data) <- cell_data2$id
names(dbscan_res$cluster) <- rownames(cell_data_data)
clusters <- tibble::enframe(dbscan_res$cluster)
## will assign each dbscan cluster(value) with a subpop(sp)
cluster_assign <-
  ## add columns from y to x matching rows based on id
  dplyr::left_join(clusters, cell_data2, by=c("name"="id"))|>
  # isolates the var we want, name from celldata2 & value from clusters
  dplyr::select(name, value, sp) |>
  #  groups subpops together & labels
  dplyr::group_by(value, sp) |>
  ## counts the number of subpops in each cluster
  dplyr::summarise(n=dplyr::n()) |>
  dplyr::group_by(value) |>
  ## finds the max subpop in each cluster
  dplyr::mutate(max=max(n)) |>
  ##  filters out the non max subpops
  dplyr::filter(n==max)
assignments <-
  # same as cluster_assign
  dplyr::left_join(clusters, cell_data2, by=c("name"="id"))|>
  dplyr::select(name, value, sp) |>
  dplyr::group_by(value, sp) |>
  # add columns from y to x matching rows based on cluster(value)
  ## sp.x is max subpop according to dbscan sp.y is the real subpop
  dplyr::left_join(cluster_assign, by=c("value"="value")) |>
  ## add new variable matches if subpop(sp) is equal to max cluster subpop
  dplyr::mutate(matches= sp.x==sp.y) |>
  ## filter the ones that are true
  dplyr::filter(matches==TRUE) |>
  ## leave data frame ungrouped
  dplyr::ungroup() |>
  ## get a numeric value
  dplyr::count()
assignments
##counts the non admixed groups
```

Figure 11

Using this code I discovered that the number of correctly sorted samples out of 2,514 was 1,584. At first I assumed that this meant my original code lacked accuracy and needed to be adjusted to account for possible outliers. I then realized the flaw in this thinking. I had not considered the possibility of admixed populations where the samples have combined ancestries from different populations. I decided to slightly change the method that I used to count the admixed populations so that I could visualize the populations on my original model that were composed of these groups. Figure 12 is the model that demonstrates which subpopulations were considered admixed through the t-SNE and DBSCAN algorithms in combination. Gray samples were non-admixed and red samples were. The graph provides incredibly insightful information on the connection between admixed hispanic and South Asian populations which are extremely prevalent in the visual.

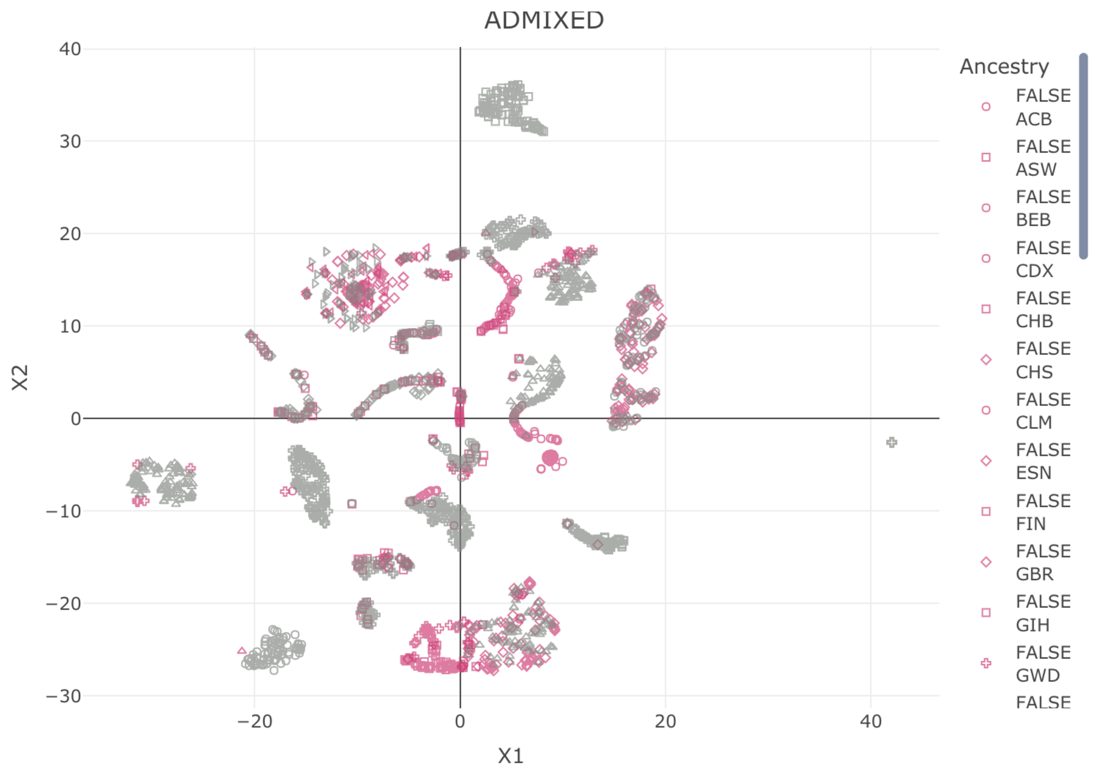


Figure 12

Through the analysis of the admixed populations, it became clear that there were potential correlations that were missing from both of these visuals that needed one more. It came to my attention that it was difficult to tell the difference between DBSCAN clusters because in the two visuals they were not represented through colors or shapes. This posed a need for a third visual that pointed out all twenty one clusters as a spectrum of colors. I visualized this through figure 13.

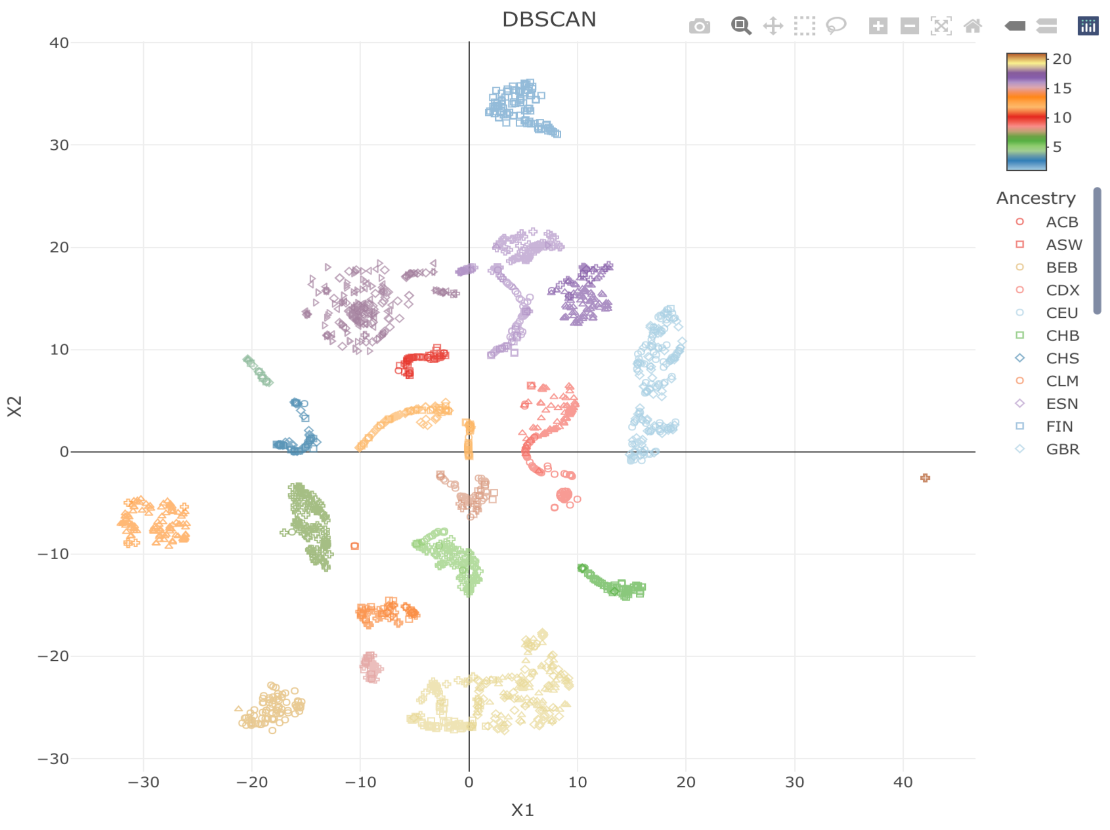


Figure 13

Ultimately, each of the three visuals that I created demonstrates the diversity of genetic ancestry through the combination of various algorithms and methods. Color, shape, opacity and spacing were all heavily taken into consideration and experimented with to optimize the viewer's experience and understanding. Not only can these methods be streamlined to bigger data sets, but the amalgamation of projects from numerous institutions will indeed be used to represent ancestry in a way that is clearly understood by the scientific world.