HDS 10. Nonlinear Dimension Reduction with t-SNE and UMAP

Matti Pirinen, University of Helsinki

30.11.2021

We have seen how PCA extracts such linear combinations of the p original variables that are maximally informative among all linear combinations. Often the leading PCs have a clear and interpretable structure and therefore PCA is a widely-used method to visualize and reduce high-dimensional data.

PCs are **global linear** functions of data and hence the leading PCs tend to capture such directions from the input space on which the distant data points remain distant from each other also in the leading PCs as such directions maximize the variance of the projected points. However, for high-dimensional data that happen to be structured in some non-linear way on some lower dimensional subspace, it would also be important to keep the similar samples close together in the low-dimensional representation, which may not be possible by any global linear function such as a PC.

Many methods for dimension reduction that try to capture more of the local structure are non-linear and are not guaranteed to yield a globally optimal solution (result may change with the seed of the random number generator that initializes the algorithm).

Here we study two methods: **t-SNE** and **UMAP**. Let's first see what they produce in practice and then come back to what is going on under the hood.

1000 Genomes data

The 1000 Genomes Project has produced genotype data from across the world. Here we consider a subset of $n = 1092$ individuals from the following 14 **populations**, divided into 4 continental **groups**,

- ASW [AFR] (61) - African Ancestry in Southwest US
- CEU [EUR] (85) - Utah residents (CEPH) with Northern and Western European ancestry
- CHB [ASN] (97) - Han Chinese in Beijing, China
- CHS [ASN] (100) - Southern Han Chinese
- CLM [AMR] (60) - Colombian in Medellin, Colombia
- FIN [EUR] (93) - Finnish from Finland
- GBR [EUR] (89) - British from England and Scotland
- IBS [EUR] (14) - Iberian population in Spain
- JPT [ASN] (89) - Japanese in Toyko, Japan
- LWK [AFR] (97) - Luhya in Webuye, Kenya
- MXL [AMR] (66) - Mexican Ancestry in Los Angeles, CA
- PUR [AMR] (55) - Puerto Rican in Puerto Rico
- TSI [EUR] (98) - Toscani in Italia
- YRI [AFR] (88) - Yoruba in Ibadan, Nigeria

Each individual has been measured on $p = 4212$ genetic variants (each can have value 0, 1 or 2) from chromosomes 15-22.

```r
X = read.table("geno_1000G_phase1_chr15-22.txt", as.is = T, header = T)
dim(X)
```

```
## [1] 1092 4215
```

```r
X[1:4, 1:10]
```

```
##        id population group X1 X2 X3 X4 X5 X6 X7
## 1 HG00096        GBR   EUR  0  0  0  0  2  0  0
## 2 HG00097        GBR   EUR  1  1  0  0  1  0  0
## 3 HG00099        GBR   EUR  0  0  0  0  1  1  0
## 4 HG00100        GBR   EUR  0  0  1  0  2  0  1
```

```r
table(X[,"group"], X[,"population"])
```

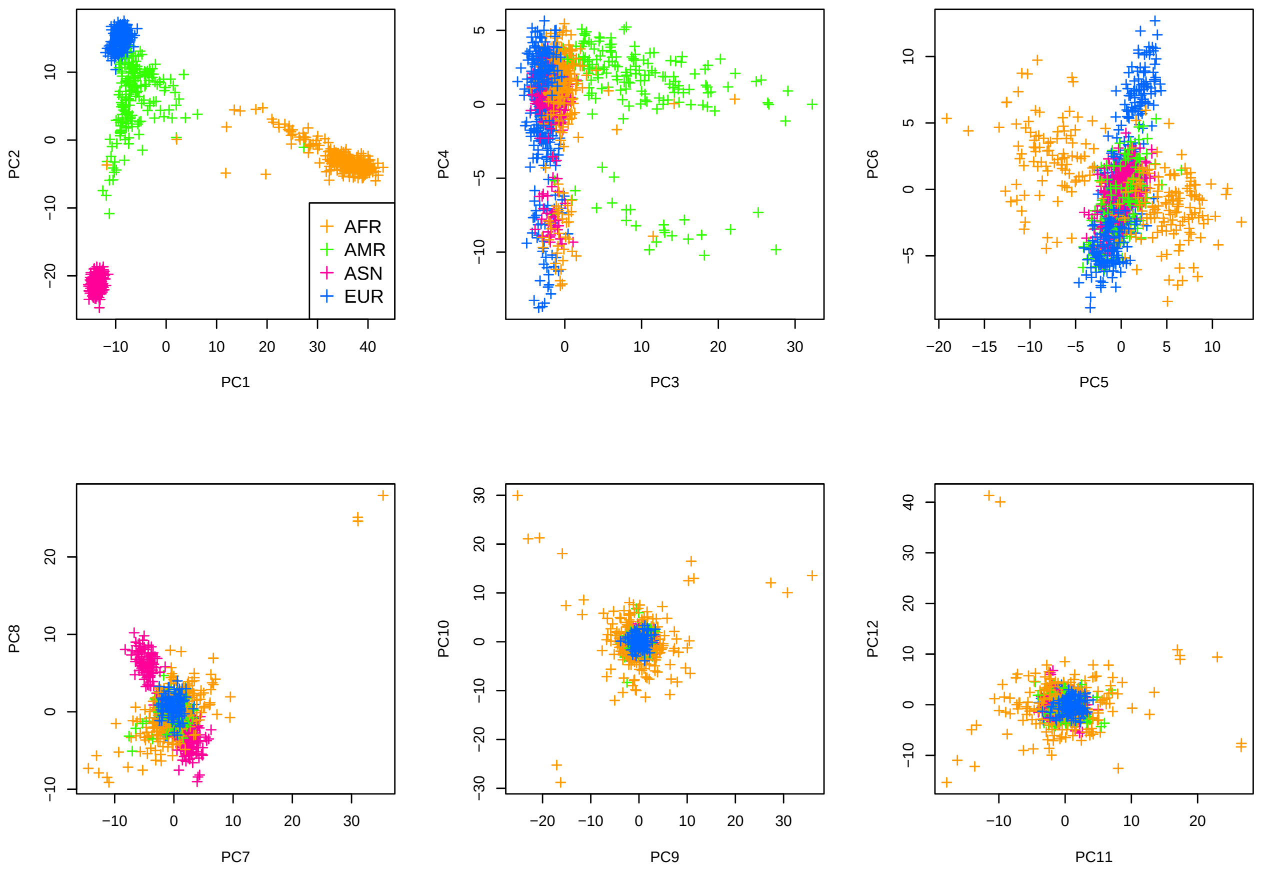
```
##
##       ASW CEU CHB CHS CLM FIN GBR IBS JPT LWK MXL PUR TSI YRI
##   AFR  61   0   0   0   0   0   0   0   0  97   0   0   0  88
##   AMR   0   0   0   0  60   0   0   0   0   0  66  55   0   0
##   ASN   0   0  97 100   0   0   0   0  89   0   0   0   0   0
##   EUR   0  85   0   0   0  93  89  14   0   0   0   0  98   0
```

Let's do PCA and plot 12 leading PCs in pairwise plots coloring each individual by the continental group (Africa, Americas, Asia, Europe) given by the `group` variable.

```r
x = as.matrix(X[, 4:ncol(X)])
n = nrow(x)
p = ncol(x)
pr = prcomp(x, scale = T)

grs = names(table(X$group))
grs.col = hsv(c(0.1, 0.3, 0.9, 0.6), 1, 1) #find distinct colors for groups
cols.gr = rep(NA, n) #color of the group of each individual
for(ii in 1:length(grs)) cols.gr[X$group == grs[ii]] = grs.col[ii]

par(mfrow = c(2,3))
for(ii in 1:6){
  plot(pr$x[,2*ii-1], pr$x[,2*ii], col = cols.gr, pch = 3,
       xlab = paste0("PC", 2*ii-1), ylab = paste0("PC", 2*ii))
  if(ii == 1) legend("bottomright", col = grs.col, leg = grs, pch = 3, cex = 1.3)
}
```



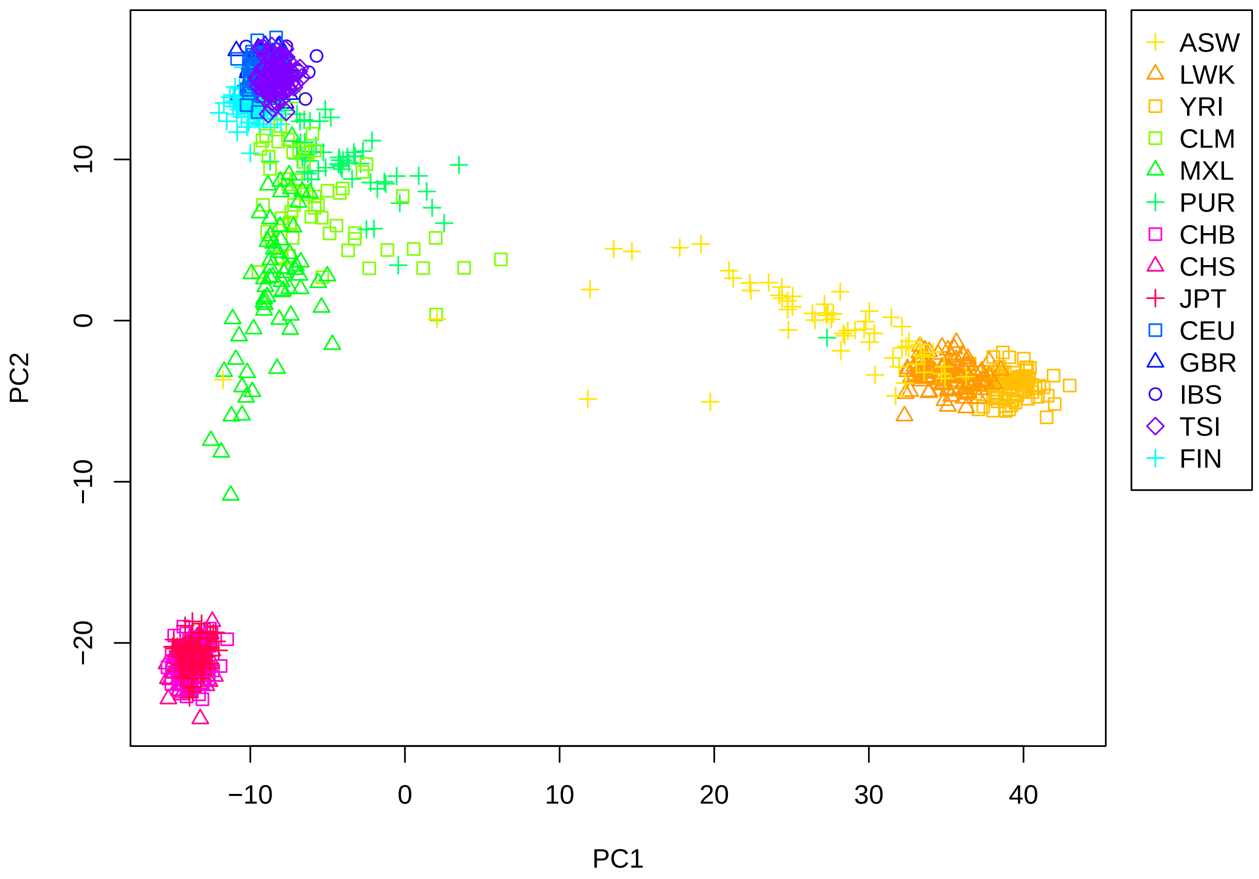
Visually, PCs 1-8 seem to capture broader structure whereas PCs from 9 onwards seem to separate small groups of possibly more closely related pairs or triples.

Let's next color points by population rather than continent.

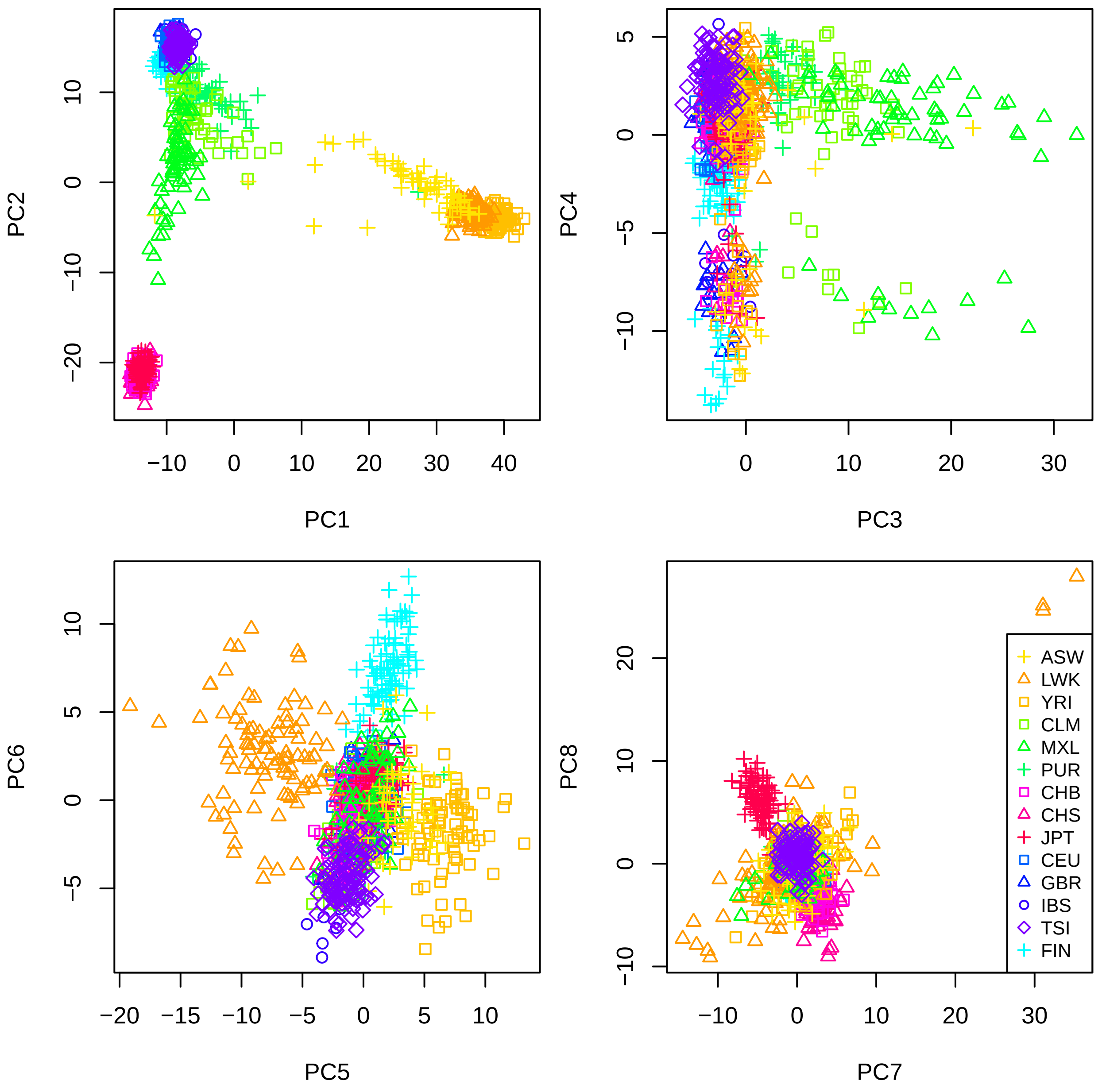
```
#3 AFR, 3 AMR, 3 ASN and 5 EUR populations given colors and plotting symbols
pops = c("ASW","LWK","YRI","CLM","MXL","PUR","CHB","CHS","JPT","CEU","GBR","IBS","TSI","FIN")
pops.col = hsv(c(0.15, 0.1, 0.125, 0.25, 0.35, 0.4,
                 0.85, 0.9, 0.95, 0.6, 0.65, 0.7, 0.75, 0.5), 1, 1)
pops.pch = c(3,2,0,  0,2,3,  0,2,3,  0,2,1,5,3) #plotting symbol for each population

cols.pop = rep(NA, n)
pchs.pop = rep(NA, n)
for(ii in 1:length(pops)) {
  cols.pop[X$population == pops[ii]] = pops.col[ii]
  pchs.pop[X$population == pops[ii]] = pops.pch[ii]
}

par(mar = c(4,4,0.5,8), xpd = TRUE)
plot(pr$x[,1], pr$x[,2], col = cols.pop, pch = pchs.pop, xlab = "PC1", ylab = "PC2")
legend("topright", inset = c(-0.15,0), leg = pops, col = pops.col, pch = pops.pch)
```

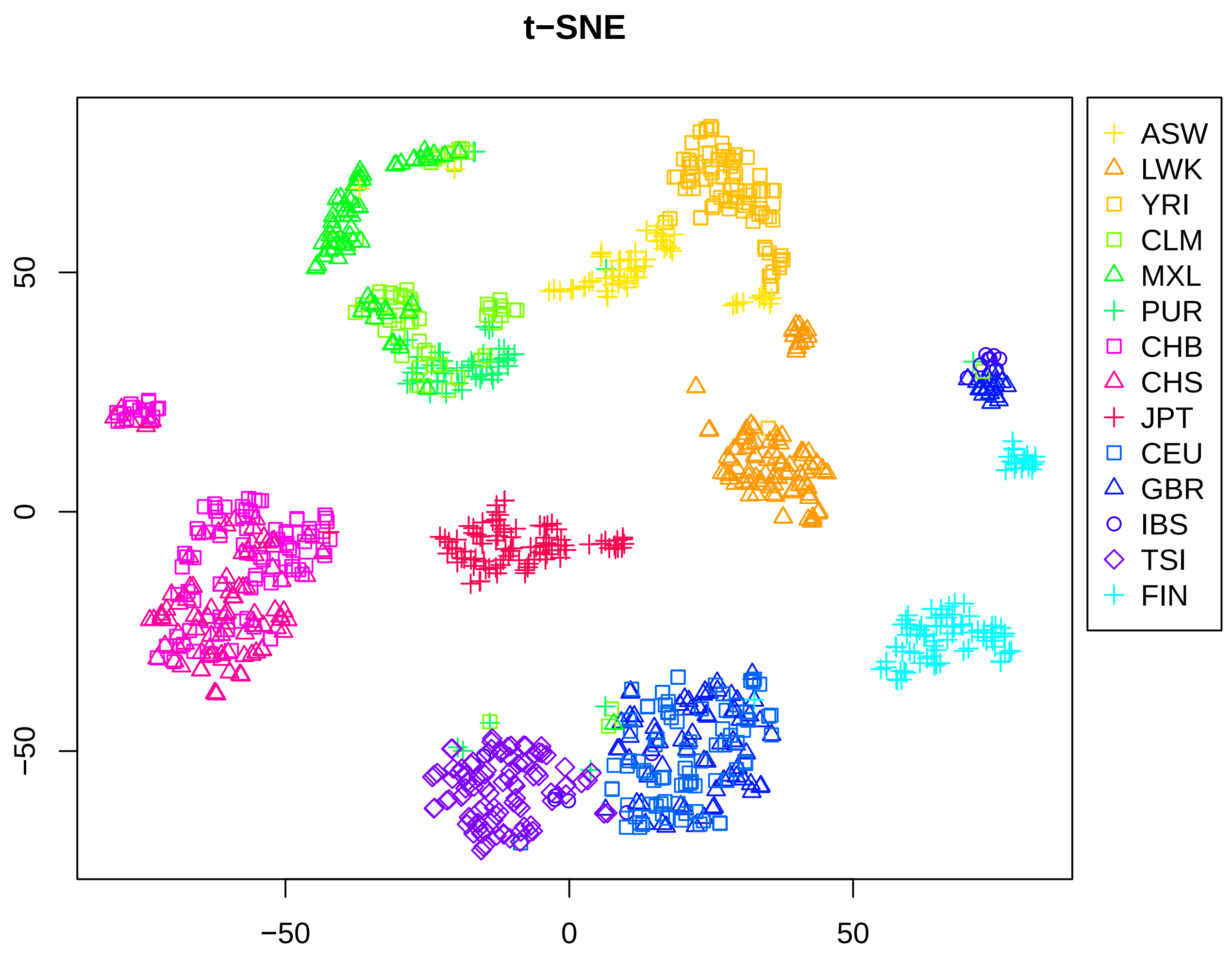


```
par(mfrow=c(2,2), mar = c(4,4,1,0.5))
for(ii in 1:4){
  plot(pr$x[,2*ii-1], pr$x[,2*ii], col = cols.pop, pch = pchs.pop,
       xlab = paste0("PC",2*ii-1), ylab = paste0("PC",2*ii))
  if(ii == 0) legend("bottomright", col = grs.col, leg = grs, pch = 3, cex = 1.3)
}
legend("bottomright", leg = pops, col = pops.col, pch = pops.pch, cex = 0.8)
```

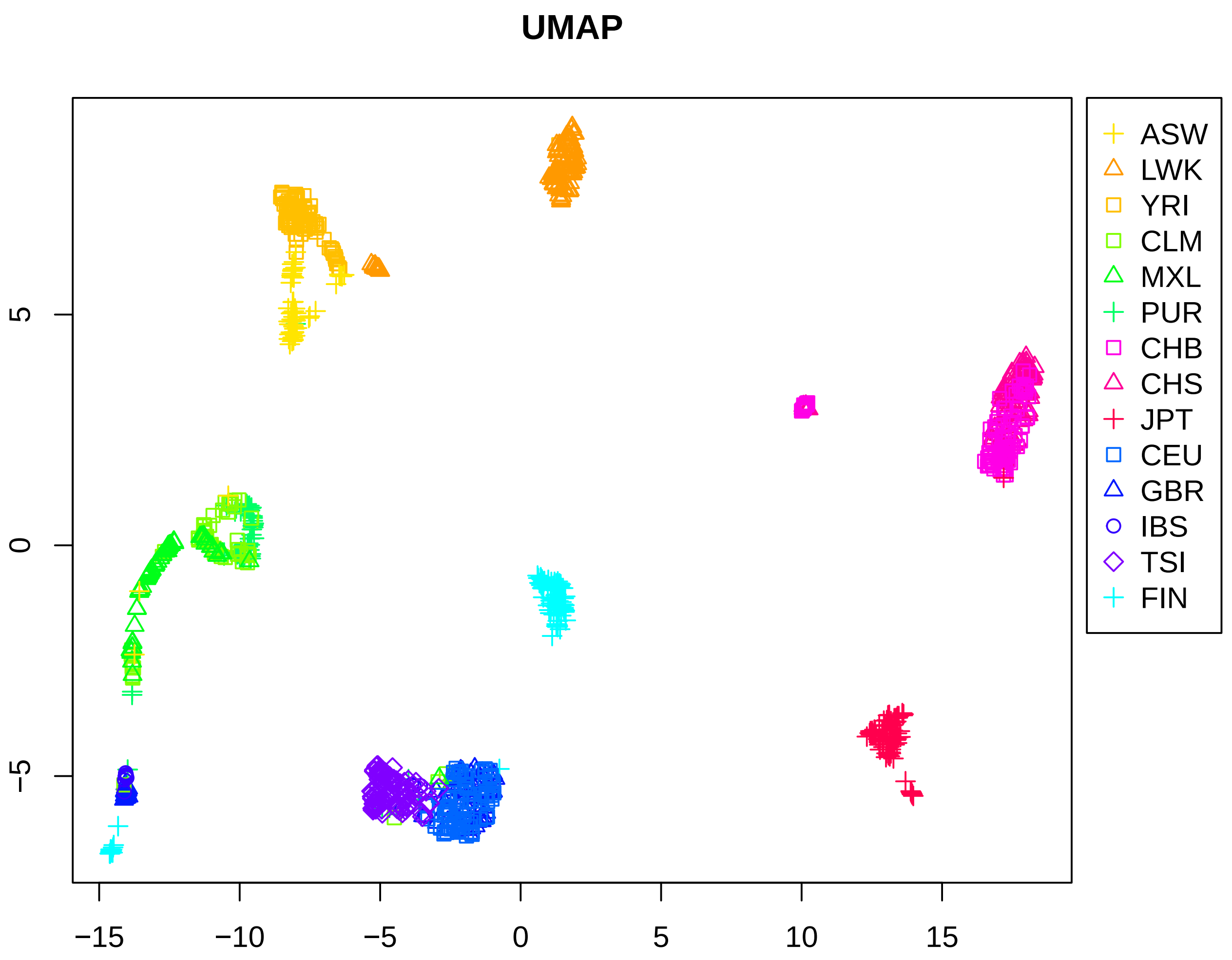


Let's then compare the PCA plots to t-SNE and UMAP, using R packages `Rtsne` and `umap`, respectively, that we apply to compress the first 8 PCs to 2 dimensions.

```
#install.packages("Rtsne")
library(Rtsne)
set.seed(67)
tsne = Rtsne(X = pr$x[,1:8], perplexity = 10, theta = 0.0, pca = FALSE)
par(mar = c(4,4,4,8), xpd = TRUE)
plot(tsne$Y, col = cols.pop, pch = pchs.pop, main = "t-SNE", xlab ="", ylab = "")
legend("topright", inset = c(-0.15,0), leg = pops, col = pops.col, pch = pops.pch)
```



```
#install.packages("umap")
library(umap)
set.seed(67)
umap.res = umap(pr$x[,1:8])
par(mar = c(4,4,4,8), xpd = TRUE)
plot(umap.res$layout, col = cols.pop, pch = pchs.pop, main = "UMAP", xlab ="", ylab = "")
legend("topright", inset = c(-0.15,0), leg = pops, col = pops.col, pch = pops.pch)
```



Wee see that t-SNE and UMAP largely group individuals from a same population close together and separate them from other populations whereas they do not put so much emphasis on making, for example, the two African continental populations YRI and LWK equally distant from all 5 European populations or all 3 Asian populations, as the PCs 1 and 2 did above. Thus, we see that t-SNE and UMAP may indeed preserve more of the local structure around the neighborhood of each sample but consequently cannot be completely faithful to the overall global structure as defined by leading PCs. In a sense, t-SNE and UMAP try to present both the local and global structure in only 2 dimensions, and for this they need to find a trade-off between these two goals.

t-SNE: t-distributed Stochastic Neighbor Embedding

t-SNE was introduce in "Visualizing Data using t-SNE" by van der Maaten & Hinton (2008).

It builds on earlier work on Stochastic Neighbor Embedding (SNE), where the idea is to measure distance in the high-dimensional input space by a conditional probability. If $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are two p-dimensional data points, we can compute a conditional probability $p_{j|i}$ that $\boldsymbol{x}_i$ would pick as its neighbor the point $\boldsymbol{x}_j$ if neighbors would be chosen from a Gaussian distribution centered on $\boldsymbol{x}_i$ and having a same variance σ_i^2 in each dimension (we come back to how σ_i^2 will be chosen later).

$$p_{j|i} \propto \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\,\sigma_i^2}\right) \quad \text{and} \quad \sum_{j\neq i} p_{j|i} = 1.$$

This probability is larger for points that are closer to $\boldsymbol{x}_i$ than for those that are farther away. Similarly, we can define $p_{i|j}$. Finally, we can average (and normalize by n) these to get $p_{ij} = p_{ji} = \frac{1}{2n}(p_{i|j} + p_{j|i})$ to represent the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by a single value.

The goal of SNE is to map the p-dimensional input values $\boldsymbol{x}_i$ to 2-dimensional (or 3-dimensional) points y_i in such a way that the distances q_{ij} defined by a similar density function evaluation in the output space would optimally match the input space distances p_{ij}. Here "optimally" means in terms of the Kullback-Leibler divergence of distribution $Q = (q_{ij})$ from distribution $P = (p_{ij})$:

$$\mathrm{KL}(P\,||\,Q) = \sum_{i<j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right),$$

which is always non-negative and is zero if and only if the distributions are the same. Minimizing this cost function puts more emphasis on making pairs with high p_{ij} to have similarly high q_{ij}, but less emphasis on matching the two when p_{ij} is small. Hence, SNE is expected to preserve particularly well the local structure in the data and pay less attention to what happens with long distances.

What t-SNE adds on top of SNE is that the distribution $Q = (q_{ij})$ is defined using the density function of the **t-distribution with 1 degree of freedom**, also known as the **Cauchy distribution**, rather than by a Gaussian, as was done in SNE. This means that, in the low-dimensional output space, the conditional probabilities are defined as

$$q_{j|i} \propto \left(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2\right)^{-1} \quad \text{and} \quad \sum_{j\neq i} q_{j|i} = 1,$$

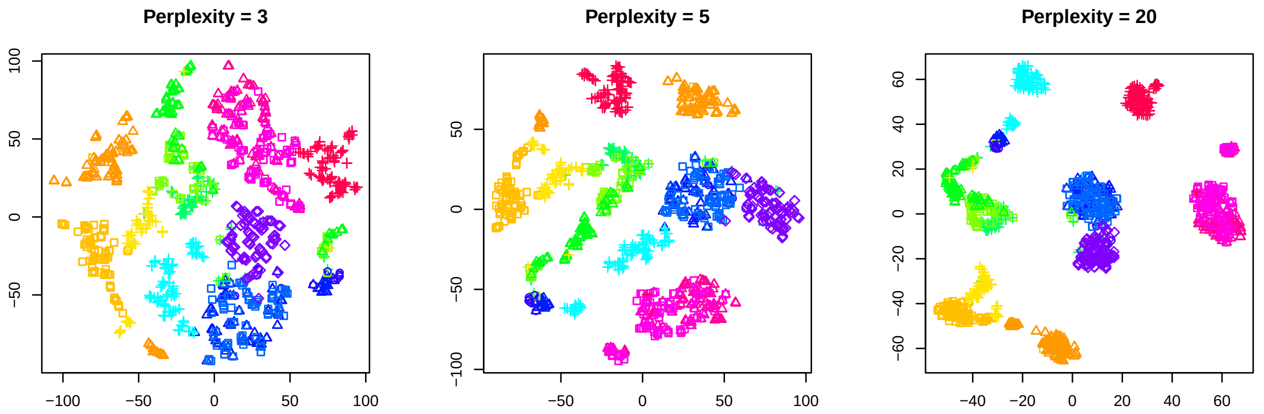
and these are symmetrized and normalized as above: $q_{ij} = q_{ji} = \frac{1}{2n}(q_{i|j} + q_{j|i})$.

The Cauchy distribution has thick tails, and therefore t-SNE can tolerate more discrepancy between the distances in input and output space when it comes to the points that are moderately far from each other in the input space. This helps to avoid *the crowding problem*: in the high-dimensional input space, there are potentially many equidistant points with moderate distance from a particular point, and not all of these can be similarly accounted for in the low-dimensional space. The Cauchy distribution makes sure that some of these points can be more spread out in the output space without a very high penalty.

Perplexity In input space, the distances are defined by a Gaussian density with a data point specific variance σ_i^2. This parameter determines how the t-SNE measure of distance from $\boldsymbol{x}_i$ decays with the Euclidean distance from $\boldsymbol{x}_i$, with larger values of σ_i^2 meaning slower decay and smaller values meaning quicker decay. In order to preserve local structure around each point, t-SNE adjusts σ_i^2 in such a way that all the conditional distributions $(p_{j|i})_{i\leq n}$ have approximately the same *perplexity*, which can be interpreted as an effective number of neighbors. The target perplexity is given as a parameter to the algorithm and is typically between 5 and 50. For example, value 15 means roughly that the distances $p_{j|i}$ from $\boldsymbol{x}_i$ to about 15 of its most closest data points are all large enough so that those points can be considered as "neighbors" but the same is not true for some larger set of points than the 15 closest ones. In other words, the similarity measured from $\boldsymbol{x}_i$ decays with such a rate that about 15 points are "nearby". Effectively about 15 closest neighbors are taken into account when constructing the low-dimensional representation.

Let's see how different perplexities show up in results:

```
par(mfrow=c(1,3))
for(perp in c(3, 5, 20)){
  tsne = Rtsne(X = pr$x[,1:8], perplexity = perp, theta = 0.05, pca = FALSE)
  plot(tsne$Y, col = cols.pop, pch = pchs.pop,
       main = paste0("Perplexity = ",perp), xlab ="", ylab = "")
}
```



theta parameter is a trade-off between accuracy and computation, where 0 means most accuracy and larger values mean more efficient but less accurate approximationa. Default is 0.5.

Resources to learn more about t-SNE

- StatQuest video on t-SNE (11:48)
- How to Use t-SNE Effectively
- How to tune hyperparameters of tSNE
- Roberto Stelling's blog
- "Visualizing Data using t-SNE" paper by van der Maaten & Hinton (2008).

UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

UMAP was introduced in 2018 by L.McInnes, J.Haley, J.Melville. Their abstract summarizes their motivation for UMAP compared to t-SNE: similar quality of visualization with (much) more efficient algorithm. Additionally, UMAP is better able to maintain the global structure than t-SNE, which, on the other hand, may reduce some details of the local structure.

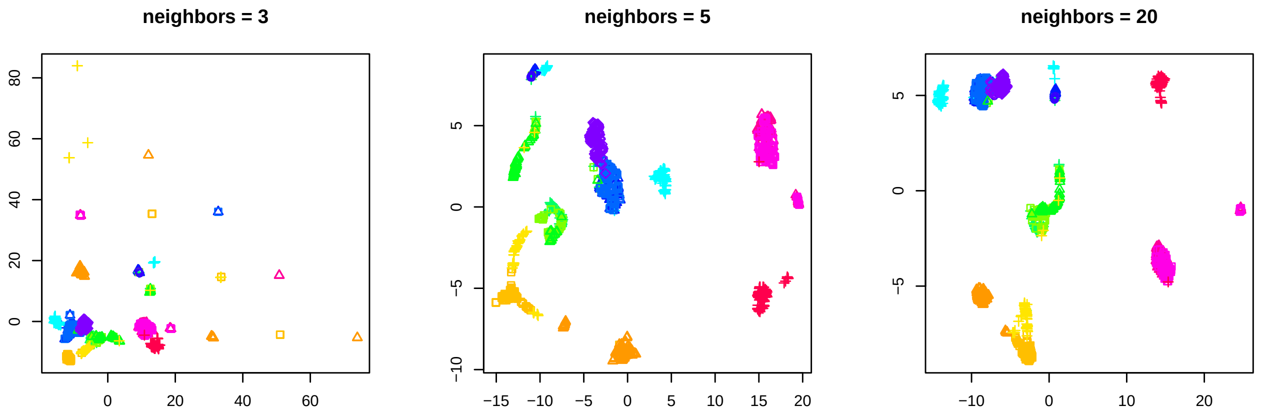
Methodologically, UMAP uses similar ideas as t-SNE although the theoretical derivation is more mathematical. A description of the differences to t-SNE can be found from Appendix C of the UMAP paper. Another description is "How exactly UMAP works?" by Nikolay Oskolkov.

Main parameters of UMAP are `n_neighbors`, the number of closest neighbors that are considered, `min_dist`, the minimum distance of the points in the output space, `n_components`, the output dimension and `metric` that defines the distance of the input space. These are clearly explained at the UMAP website.

An illustrative site about UMAP: https://pair-code.github.io/understanding-umap/

Let's see how `n_neighbors` compares to effect of perplexity in t-SNE (that we saw above).

```r
par(mfrow=c(1,3))
set.seed(81)
umap.config = umap.defaults #umap takes parameters in a config object
for(nbors in c(3, 5, 20)){
  umap.config$n_neighbors = nbors
  umap.res = umap(pr$x[,1:8], config = umap.config)
  plot(umap.res$layout, col = cols.pop, pch = pchs.pop,
       main = paste0("neighbors = ",nbors), xlab ="", ylab = "")
}
```



Discussion

- Most non-linear dimension reduction techniques (including t-SNE and UMAP) lack the strong interpretability of Principal Component Analysis where the dimensions are the directions of greatest variance in the source data. If strong interpretability is needed, PCA is recommended.
- As t-SNE and UMAP are based on the distance between observations rather than the source features, they do not produce easily interpretable loadings per each variable that PCA can provide for each output dimension.
- A core assumptions of UMAP is that there exists manifold structure in the data. Because of this UMAP may find manifold structure within the noise of a dataset, a type of overfitting. As more data is sampled, UMAP becomes more robust. However, care must be taken with small sample sizes of noisy data, or data with only large-scale manifold structure.
- If data are high-dimensional, say $p > 100$, it is often useful to first apply PCA and take some tens of the leading PCs as input for t-SNE or UMAP to further reduce to 2 or 3 dimensions for visualization.