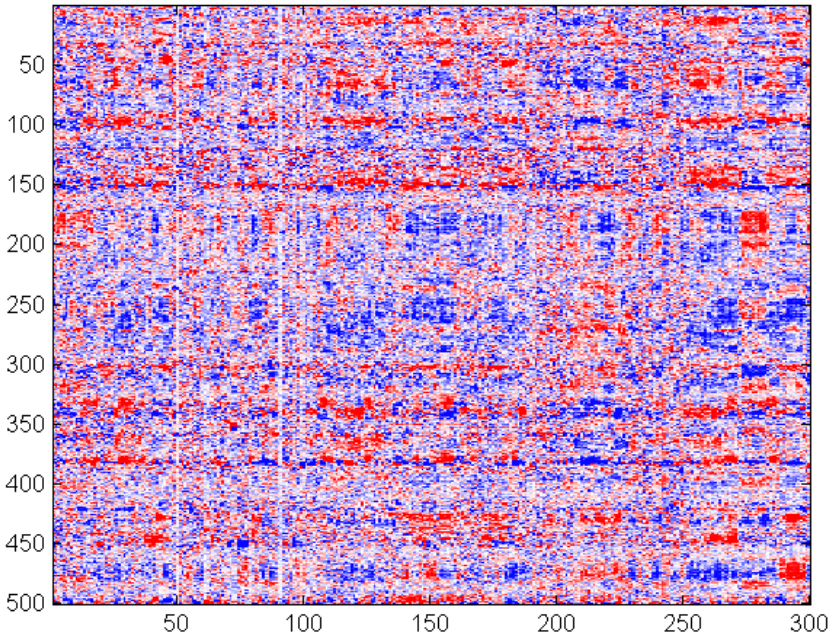
Opinionated Lessons in Statistics

by Bill Press

#47 Low Rank Approximation of Data

Linear “Stuff”: SVD, PCA, Eigenthingies, and all that!

We start with a “data matrix” or “design matrix”: $\mathbf{X} = \{X_{ij}\}$
Let’s use gene expression data as the example.



(This is just to give you something to look it. Typically, for these techniques, you would not be able to see anything in the data “by eye”.)

N rows are data points, here genes 1 – 500
M columns are the responses. View each as a vector in M (here 300) dimensional space
For gene expression data, each column is a separate micro array experiment under a different condition

Let's always assume that the individual experiments (columns of $\mathbf{X}$) have zero mean. (Can always get this by subtracting the mean of each column.)

While we're at it, we might as well also scale each column to unit standard deviation.

And, it's a good idea to eliminate outliers.

Matlab for this (and plotting the previous picture) is:

```
load yeastarray_t2.txt;
size(yeastarray_t2)
ans =
   500   300
yclip = prctile(yeastarray_t2(:),[1,99])
yclip =
  -204   244
data = max(yclip(1),min(yclip(2),yeastarray_t2));
dmean = mean(data,1);
dstd = std(data,1);
data = (data - repmat(dmean,[size(data,1),1]))./repmat(dstd,[size(data,1),1]);
genecolormap = [min(1,(1:64)/32); 1-abs(1-(1:64)/32); min(1,(64-(1:64))/32)]';
colormap(genecolormap);
image(20*data+32)
```

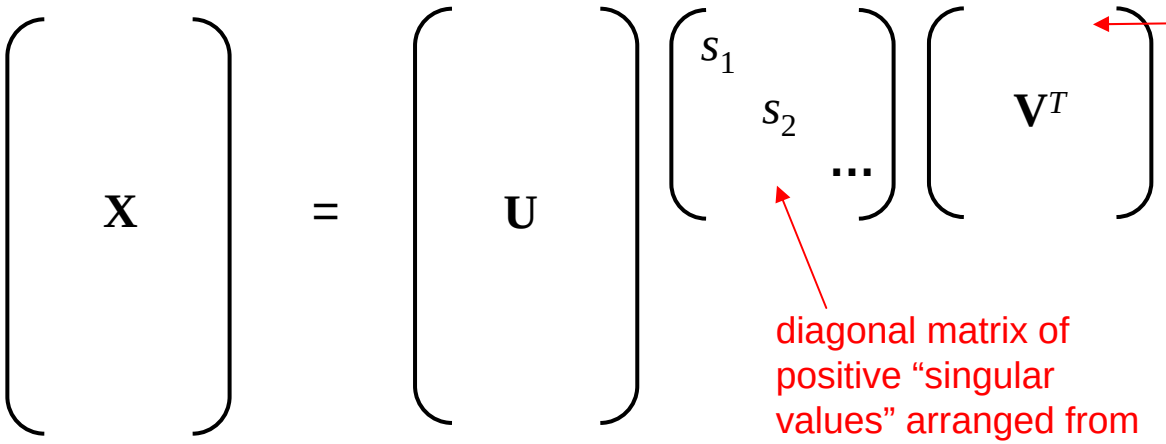
clip outliers by percentile (bottom and top 1%)

this is the arcane Matlab colormap stuff for making a blue-to-white-to-red plot that we saw before

saturate to red or blue at ~1.5 σ

Singular Value Decomposition (SVD)

Any matrix $\mathbf{X}$ (needn't be square) can be decomposed, more-or-less uniquely, as follows:



Degree of freedom (DOF) counting:

MN = MN – M(M+1)/2 + M + M^2 – M(M+1)/2

Decomposition has an efficient algorithm (of order the same workload as inverting a matrix). Matlab, Python, etc., have ready-to-use implementations.

We can write out the (middle) sums over the singular values explicitly. Each column of $\mathbf{U}$ gets paired with the corresponding row of $\mathbf{V}^T$ (or column of $\mathbf{V}$).

$$\mathbf{X} = \sum_{i=1}^{M} s_i \, \mathbf{U}_{\cdot i} \otimes \mathbf{V}_{\cdot i}$$

note: "dot" now does NOT mean sum. It's just a placeholder!

This turns out to be the optimal decomposition of $\mathbf{X}$ into rank-1 matrices, optimal in the sense that the partial sums converge in the "greediest" way in L^2 norm. (I.e., at each stage in the sum, there is no better decomposition.)

Recall: A rank-one matrix has all its columns proportional to each other, which also implies that all the rows are proportional to each other: $C_{ij} = A_i B_j$ So the rows (or columns) lie on a one-dimensional line in the row (or column) dimension space.

$$\begin{aligned}
\sum_i \sum_j |X_{ij}|^2 = \mathrm{Tr}(\mathbf{X}\mathbf{X}^T) &= \sum_i \left[\sum_j X_{ij} X^T_{ji} \right] \\
&= \mathrm{Tr}(\mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}\mathbf{U}^T) \\
&= \mathrm{Tr}(\mathbf{U}\mathbf{S}^2\mathbf{U}^T) \\
&= \mathrm{Tr}(\mathbf{S}^2\mathbf{U}^T\mathbf{U}) \\
&= \mathrm{Tr}(\mathbf{S}^2)
\end{aligned}$$

$$\mathbf{X} = \sum_{i=1}^{M} s_i \, \mathbf{U}_{\cdot i} \otimes \mathbf{V}_{\cdot i}$$

So this "builds up dimensionality" with each term in the sum: adds a new basis vector (in both the U and the V spaces)

If the data actually lie on a lower dimensional (than M) hyperplane that goes through the origin, then only that many s_i's will be nonzero.

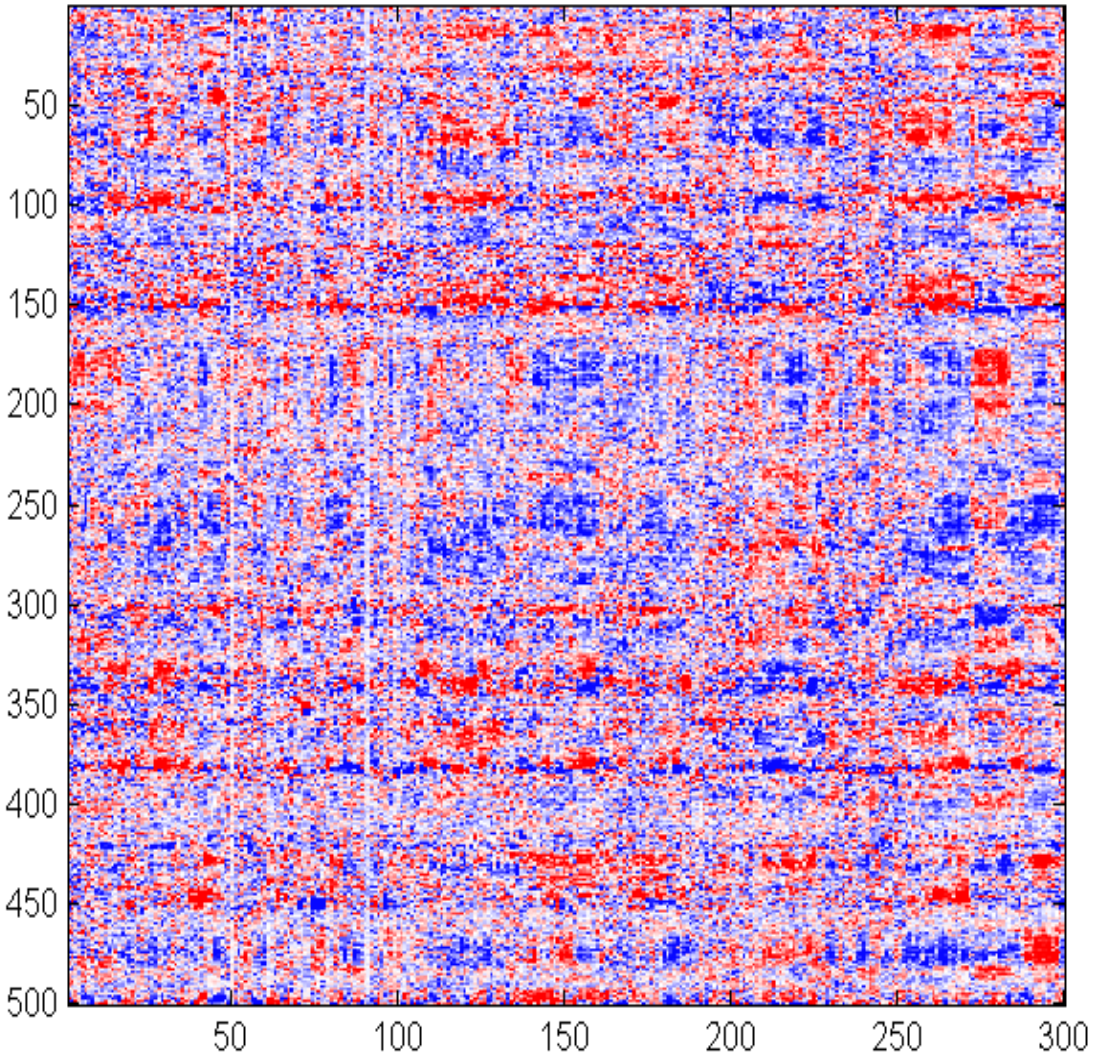
That is why we subtracted the means!

Or, in the general case we can just truncate the sum to get the best lower rank approximation to $\mathbf{X}$. This can be useful for filtering out noise (we will see).

Notice that this captures only a "linear" (hyperplane) view of the world. More complicated functional relationships that might decrease dimensionality are not, in general, identified by SVD or PCA.

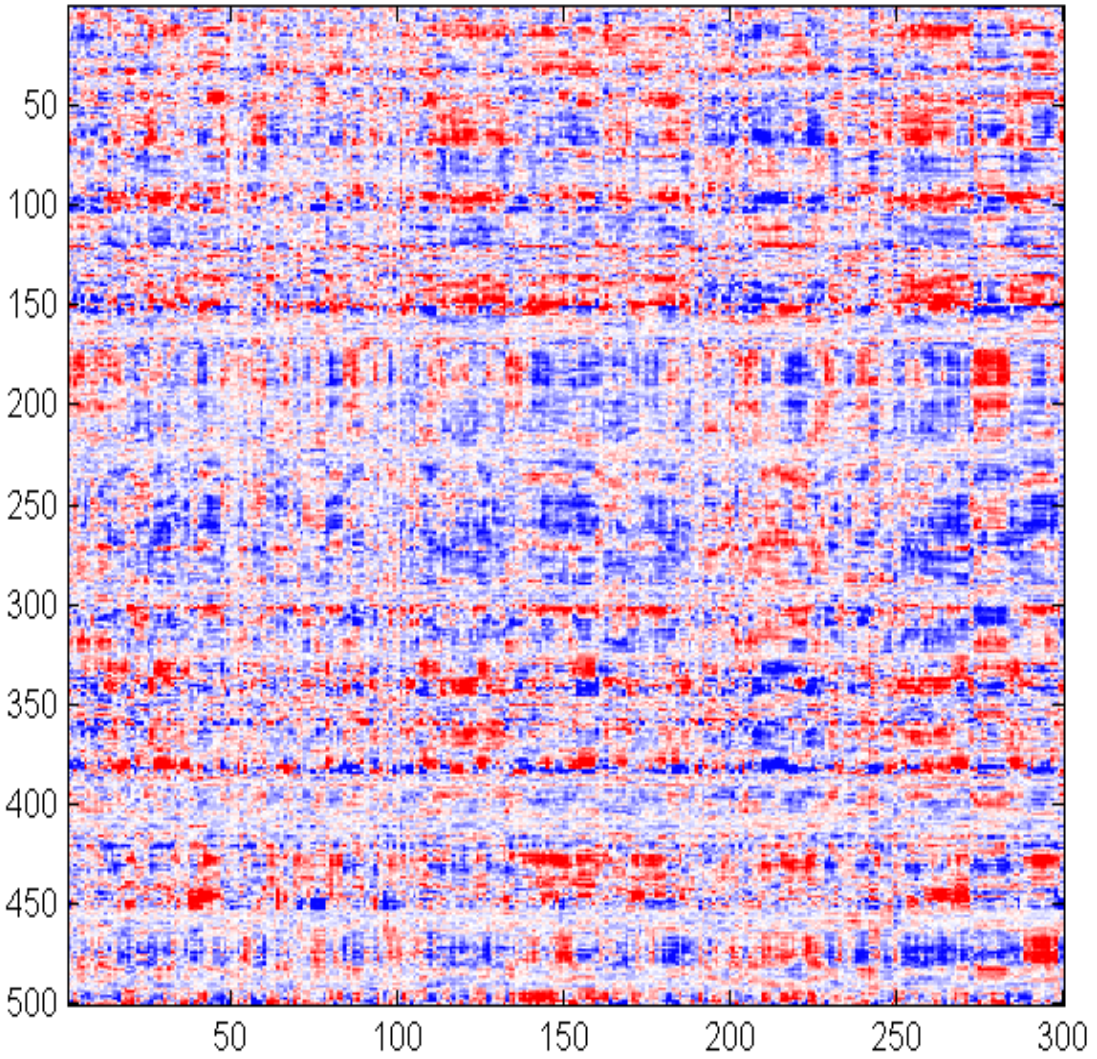
Let's see what some lower rank approximations to the data look like:

Original data set:



Approximated by 20 SV's:

```
strunc = diag(S);
strunc(21:end) = 0;
filtdata = U*diag(strunc)*V';
colormap(genecolormap);
image(20*filtdata+32)
```



Or, just 5 SV's:

```
strunc(6:end) = 0;
filtdata = U*diag(strunc)*V';
colormap(genecolormap);
image(20*filtdata+32)
```