Supplementary Methods:

Cloning and viral infection of cells. In-Fusion Cloning (TaKaRa, Osaka, Japan) was used to construct a murine full length expression cDNA construct containing the endogenous 3' UTR corresponding to the regulatory selenocysteine insertion element (SECIS), which directs incorporation of selenocysteine (SEC) into the stop codon positioned at amino acid 40 near the N-terminus. Essentially, RNA was extracted from murine esophageal tissue (C57BL/6J) and cDNA generated using the superscript II reverse transcription kit (Thermo Fisher, Waltham, MA). cDNA was used as a template for PCR amplification of the full length GPX2 cDNA plus the SECIS using primers with additional overlapping sequence from the lentiviral destination vector pLVX-IRES-Puro (Addgene, Watertown, MA). The GPX2-SECIS amplicon was cloned into pLVX-IRES-Puro digested with XhoI / BamHI to generate pLVX-IRES-mGFPX2 (Supplementary Figure 1), which lacks fluorescent or exogenous tags, and the sequence confirmed by Sanger Sequencing. For *in vitro* GPX2 knockdown experiments, human cells were infected with lentiviral pGIPZ-shRNA (Horizon, Waltham, MA) targeting human GPX2 (V3LHS_368762 with antisense TCCAGGCCACATCTGAGCG) or an empty vector (EV)control.

Lentiviral vectors with insert or devoid of insert (negative controls) were packaged in 293FT by co-transfecting them with helper vectors pMD2.G and psPAX2 (Addgene). Supernatants containing virus were collected and concentrated using Lenti-X Concentrators (Takara Bio, USA) before infecting cells by spinoculation in the presence of polybrene (6-10 µg/ml). To generate stable cell lines, MOC1 infected cells were selected with puromycin (2 µg/ml) initially as polyclonal populations and single cells subsequently seeded into 96 well plates using a BD FACS AriaII cell sorter to derive long term clones subsequently screened for GPX2 protein levels by Western blot analysis. For GPX2 KD, human cells infected with shRNA were sorted for EGFP expression 48 h post-infection and allowed to recover for 2 days before setting up experiments to collect supernatants for ELISA or cell lysates for Western blot analyses. Levels of prostaglandin E2 (PGE2) or interleukin-6 (IL-6) in conditioned media were measured by human or mouse ELISA kits (Cayman chemicals, Ann Arbor, MI) and normalized to total cell number.

Immune Profiling by flow cytometery. All *in vivo* experiments were performed with approval of the Institutional Animal Care and Use Committee (IACUC) at Baylor College of Medicine (BCM) or MD Anderson and followed established protocols. Female C57BL/6J mice (8–10 weeks of age) were purchased from the Jackson Laboratory and housed under specific pathogen-free conditions. Mice were randomly divided into treatment groups and subcutaneously (s.c.) inoculated with $3x10^6$ MOC1/wt, MOC1/pLVX-IRES-puro, or $2.5x10^6$ MOC1/pLVX-IRES-puro-GPX2 cells in the right flank. Tumor area (mm^2) was calculated as L x W, where L is Length and W is Width. To profile tumor immune cell infiltration, MOC1/wt, MOC1/pLVX-IRES-puro, and MOC1/pLVX-IRES-puro-GPX2 tumors were harvested at equal sizes (~0.8 cm width), and analyzed by flow cytometry using the approach we previously described [1 2] with a slight modification. Briefly, tumors were dissociated in RPMI 1640 (Sigma-Aldrich) containing DNase I (20 U/ml; Sigma-Aldrich), Collagenase I (1mg/ml; EMD Millipore) and Collagenase IV (250 U/ml; Worthington Biochemical Corporation) using a gentleMACS Dissociator (Miltenyi Biotec, CA) and then incubated at 37˚C for 30 minutes to complete digestion. Tumor infiltrating leukocytes were enriched from single cell suspensions using Lymphoprep™ (STEMCELL Technologies). Cells were first blocked with anti-mouse CD16/CD32 Fc block (BD Biosciences) and stained using fluorescently tagged antibodies from either the lymphocyte or myeloid panels (Supplementary Methods). DAPI was added to stained leukocytes to exclude dead cells and subsets analyzed with the gating strategy outlined in Supplementary Figure 2. For intracellular staining, cells were fixed and permeabilized with Intracellular Fixation and Permeabilization

Buffer Set (eBioscience) prior to the addition of anti-INOS. Data were acquired on a LSRII and LSRFortessa (BD Biosciences) flow cytometers, for myeloid and T cell panels respectively, and analyzed using FlowJo v10 software (FlowJo, LLC).

Generation of Nrf2 and NF-kB activation gene signatures. We previously described a method for vetting published gene signature lists to identify the most robust targets specific for a given tissue.[3] Essentially, the two by two Pearson correlation values for co-expression among all possible gene pairs are calculated using a cohort of samples with similar histology or the cohort under study and the matrix of cross-correlations is analyzed by two-way hierarchical clustering to produce a heatmap with co-expression modules appearing along the diagonal. Modules are then further evaluated for greatest average cross-correlations or biological significance that matches known functional properties of the pathway being defined to identify the most robust gene list. To derive Nrf2 and NF-kB signatures, we downloaded published lists of 469 genes (NFE2L2.V2) and 84 genes (HINATA NFkB Targets Kerationcytes Up) from the Molecular Signature Database.[4] Cross-correlation of the 469 Nrf2 gene list was examined in pooled TCGA samples (i.e., Supplementary Figure S7) and the 84 NF-kB gene list in a panel of 65 established human HNSCC cell lines (Supplementary Methods Figure SM1) to identify the most robust downstream targets.

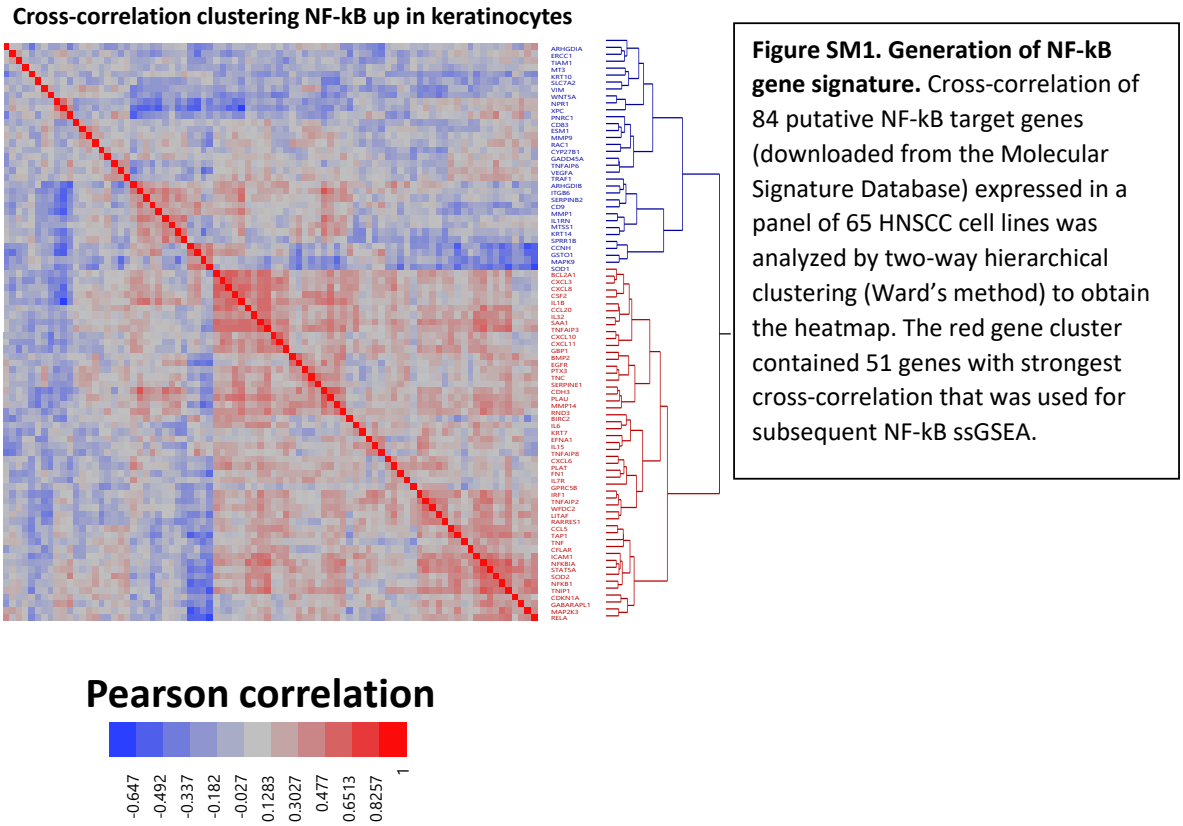

Figure SM1. Generation of NF-kB gene signature. Cross-correlation of 84 putative NF-kB target genes (downloaded from the Molecular Signature Database) expressed in a panel of 65 HNSCC cell lines was analyzed by two-way hierarchical clustering (Ward's method) to obtain the heatmap. The red gene cluster contained 51 genes with strongest cross-correlation that was used for subsequent NF-kB ssGSEA.

Consensus Clustering-We used a modification of the resampling-based method published by Monti et al.[5] to perform hierarchical two-way agglomerative (Ward's linkage) consensus clustering with an in-house Matlab script listed below and publicly available at https://github.com/aif33/Hierarchical-two-way-agglomerative-consensus-clustering

To run the program, 5 Matlab files below (names in bold) are created and saved to the same folder containing input data files. Users run the ClusteringScript.m file and input 3 parameters as "File name.xlsx", "maximum number of clusters to try (i.e., 8)", and "FirstDimension , Second Dimension or an empty string ("") referring to orientation of Z-scores to be calculated. If input values are already Z transformed, then input the empty string. The program always clusters whichever variable is listed vertically by whatever feature is listed horizontally. For example, to cluster samples based on subset scores or gene expression values the input file matrix would have each row representing a different sample and each column a different feature. If data are not Z-scaled the user would input "FirstDimenion " to calculate Z vertically. To reverse and cluster genes or leukocyte subsets by samples, simply transpose the input matrix and input "SecondDimension" so Z -scores will now be calculated horizontally to give equivalent values. The expected architecture for input files is shown by example below. The additional Matlab files contain helper methods and classes to run Clusterinscript.m. The program is hard coded to use 400 re-samplings but that can be changed within the code according to user preference. The program uses an 80% proportion for random re-sampling and applies the identical set of 400 re-samplings across all "N" clusters to facilitate choice of optimal N, which can be chosen from the output graphical plot of Normalized Euclidian Distance (NED) verses number of clusters by selecting a local minimum. The NED is a measure of error that represent the Euclidean distance between a perfect transformed similarity matrix and the calculated transformed similarity matrix at each value of N as we previously described.[3] Thus optimal choice of clusters proceeds independently for the first and second dimension analyses when performing two-way clustering.

FirstDimension input-Z scores calculated vertically

samples	aDC	Bcells	CD8	Cytotoxic	DC	iDC
TCGA-4P-AA8J	3962.401	3861.641	1518.665	1517.294	-2148.8	2101.675
TCGA-BA-4074	3569.409	-1372.16	-3636.74	-2287.41	-1142.81	-1614.04
TCGA-BA-4075	-178.691	-2850.88	-4107.13	-2803.09	807.9017	1071.645
TCGA-BA-5149	6265.759	1085.697	-963.355	-910.477	-2614.58	-936.648
TCGA-BA-5151	6960.077	2522.59	-2192.68	-1244.51	486.7138	291.1622
TCGA-BA-5152	8171.622	3530.965	1547.833	4269.614	2501.658	3267.812
TCGA-BA-5556	8832.716	4422.375	4229.137	5707.508	-432.351	2391.95
TCGA-BA-5557	4666.945	-887.898	588.2251	755.4258	2897.251	2814.198
TCGA-BA-5558	6710.69	1855.682	1482.251	2441.415	-1206.88	537.9367

SeondDimension input- Z scores calculated horizontally

samples	TCGA-4P-AA8J	TCGA-BA-4074	TCGA-BA-4075	TCGA-BA-5149	TCGA-BA-5151	TCGA-BA-5152	TCGA-BA-5556	TCGA-BA-5557	TCGA-BA-5558
aDC	3962.401	3569.409	-178.691	6265.759	6960.077	8171.622	8832.716	4666.945	6710.690
Bcells	3861.641	-1372.162	-2850.881	1085.697	2522.590	3530.965	4422.375	-887.898	1855.682
CD8	1518.665	-3636.743	-4107.126	-963.355	-2192.683	1547.833	4229.137	588.225	1482.251
Cytotoxic	1517.294	-2287.413	-2803.089	-910.477	-1244.508	4269.614	5707.508	755.426	2441.415
DC	-2148.800	-1142.811	807.902	-2614.576	486.714	2501.658	-432.351	2897.251	-1206.876
iDC	2101.675	-1614.045	1071.645	-936.648	291.162	3267.812	2391.950	2814.198	537.937

1. **ClusteringScript.m**

```
clust = clusterSampling("NewBLCAsamples.xlsx", 8, "FirstDimension");
%possible inputs -> "FirstDimension", "SecondDimension", ""
maxGroups = size(clust,2);

x = zeros(1,maxGroups);
y = zeros(1,maxGroups);

for i = 1:maxGroups
    x(1,i) = i;
    y(1,i) = clust(i).EuclideanDistance;

end

plot(x,y);
```

2. clusterSampling.m

```
function clustSampling = clusterSampling(dataFileName, numberOfGroups,
zScoreDimensionType)

    %{

    inputs and outputs:

    dataFileName -->  file name to find the data with sample names in the
    left most
    column, features we are measuring in the top most row, and values in
    the intersections

              feature1   feature2  feature3... featureN
     sample1    a          b        c          d
     sample2    e          f        g          h
        .       .          .        .          .
        .       .          .        .          .
        .       .          .        .          .
     sampleN    i          j        k          l


     numberOfGroups -> number of groups to be used in clustering


     clustSampling will be a list of a class 'ClusterInfo',
     where each instance of 'ClusterInfo' contained in the list represents
     instance number N, for 1 <= N <= numberOfGroups.

     an instance n of 'ClusterInfo' contains:
     (a) percent similarity matrix,
     (b) transformed and sorted percent similarity matrix
     (c) euclidean distance between matrix of ones and (b)

    %}

    clustSampling = clusterN.empty;
```

```
    [NUM,TXT,RAW] = xlsread(dataFileName);

    %eliminate first column and first row labels. then we are going to
    randomly
    %sample 80 percent of the data and calcuate z scores of this randomly
    %selected 80 percent.

    numberOfSamples = size(NUM,1);

    numOfResamplings = 400;

    allRks = zeros(numOfResamplings, numberOfSamples);

    % generate indexes of which samples we selected randomly with
    % 80 % of samples selected
    % r(k) == 0 ? picked sample k : did not pick sample k
    % where k is the index of the sample in the vector of sample names from
    % our data file
    % get 400 pages of z scores --> see numOfResamplings

    numberOfRandomRows = round(numberOfSamples * .8);

    zScoresSampledPages = zeros(numberOfRandomRows, size(NUM,2),
numOfResamplings);

    for cnt = 1:numOfResamplings

        k = randperm(numberOfSamples, numberOfRandomRows);
        r = true(1,numberOfSamples);
        r(k) = false;

        allRks(cnt,:) = r;

        numSampled = double.empty;

        for i = 1:numberOfSamples
            if r(i) == 0 %if r is 0, we picked that sample
                numSampled(end+1,:) = NUM(i,:);
            end
        end


        if zScoreDimensionType == "FirstDimension"

                    zScoresSampled = zscore(numSampled,0,1);
      %z score of randomly sampled data in first dimension

            zScoresSampledPages(:,:,cnt) = zScoresSampled(:,:);

        elseif zScoreDimensionType == "SecondDimension"

                        zScoresSampled = zscore(numSampled,0,2);
            %z score of randomly sampled data in second dimension

            zScoresSampledPages(:,:,cnt) = zScoresSampled(:,:);
        else
```

```
            zScoresSampled = numSampled;  %input is already z scored

            zScoresSampledPages(:,:,cnt) = zScoresSampled(:,:);
        end

    end

    for N = 1:numberOfGroups

        numeratorMatrix = zeros(numberOfSamples, numberOfSamples);

        denominatorMatrix = zeros(numberOfSamples, numberOfSamples);

        for cnt = 1:numOfResamplings

            r = allRks(cnt,:);

            zScoresSampled = zScoresSampledPages(:,:,cnt);

            clusterMatrixOfSampled = clusterMatrix(zScoresSampled,N);

            %we are going to take the randomly sampled groups and create a
            one column
            %matrix with the rest of the groups we did not sample and assign
            "fake"
            %group numbers to the groups that we did not sample

            fakeGroupNumber = 10000;

            indexOfSampled = 1;

            clusterMatrixAll = zeros(i, 1);

            clusterMatrixAllForDenominator = zeros(i,1);


            for k = 1:numberOfSamples

                if r(k) == 0
                     clusterMatrixAll(k) =
clusterMatrixOfSampled(indexOfSampled);
                     clusterMatrixAllForDenominator(k) = 1;
                     indexOfSampled = indexOfSampled + 1;
                else
                     clusterMatrixAll(k) = fakeGroupNumber;
                     clusterMatrixAllForDenominator(k) = fakeGroupNumber;
                     fakeGroupNumber = fakeGroupNumber + 1;
                end

            end

            similarityMatrixNumerator = similarityMatrix(clusterMatrixAll);

            similarityMatrixDenominator =
similarityMatrix(clusterMatrixAllForDenominator);

            numeratorMatrix = numeratorMatrix + similarityMatrixNumerator;
```

```
            denominatorMatrix = denominatorMatrix +
similarityMatrixDenominator;

        end

        sampleNames = TXT(2:end,1);

        percentSimilarityMatrix = numeratorMatrix./denominatorMatrix;

        clusterSimilarity = clusterMatrix(percentSimilarityMatrix,N);

        %now we will compute transformed percent similarity matrix

        sampleIndexAndGroups = zeros(numberOfSamples,2);

        sampleIndexAndGroups(:,1) = 1:numberOfSamples;

        sampleIndexAndGroups(:,2) = clusterSimilarity;

        sampleIndexSortedByGroups = sortrows(sampleIndexAndGroups,2);

        mappedSampleIntersection =
containers.Map('KeyType','char','ValueType','double');

        percentSimilaritySortedAndT = cell(numberOfSamples +1,numberOfSamples
+1);

        for a = 1:numberOfSamples
            %a is our row index

            groupNumberRow = clusterSimilarity(a);

            rowName = string(sampleNames(a));

            percentSimilaritySortedAndT(a+1,1) =
sampleNames(sampleIndexSortedByGroups(a,1));

            for b = 1:numberOfSamples
                %b is our column index

               colName = string(sampleNames(b));

               groupNumberCol = clusterSimilarity(b);

               if (groupNumberRow == groupNumberCol)
                   value = percentSimilarityMatrix(a,b);
               else
                   value = 1 - percentSimilarityMatrix(a,b);
               end

               mappedSampleIntersection(rowName + '_' + colName) = value;

               percentSimilaritySortedAndT(1,b+1) =
sampleNames(sampleIndexSortedByGroups(b,1));

           end
```

```
        end

        for a = 1:numberOfSamples

            rowName = string(percentSimilaritySortedAndT(a+1,1));

            for b = 1:numberOfSamples

                colName = string(percentSimilaritySortedAndT(1,b+1));

                percentSimilaritySortedAndT(a+1,b+1) =
num2cell(mappedSampleIntersection(rowName + '_' + colName));

            end

        end

        percentSimilarityValuesSortedAndT =
cell2mat(percentSimilaritySortedAndT(2:numberOfSamples + 1, 2:numberOfSamples
+ 1));

        %now we will compute the euclidean distance

        matrixOfOnes = ones(numberOfSamples);

        x = matrixOfOnes - percentSimilarityValuesSortedAndT;

        y = x.^2;

        sumFinal = sum(y,'All');

        finalValue = (sqrt(sumFinal) / numberOfSamples) * 100;

        clustSampling(N) = clusterN(percentSimilarityMatrix,
percentSimilaritySortedAndT, finalValue);

    end

end
```

3. clusterN.m

```
classdef clusterN
    properties
        PercentSimilarityMatrix
        PercentSimiliarityMatrixSortedAndT
        EuclideanDistance

    end

    methods
        function obj =
clusterN(percentSimilarity,percentSimilaritySortedAndT, euclideanDistance)
            obj.PercentSimilarityMatrix = percentSimilarity;
            obj.PercentSimiliarityMatrixSortedAndT =
percentSimilaritySortedAndT;
```

```
            obj.EuclideanDistance = euclideanDistance;
        end
    end
end
```

4. similarityMatrix.m

```
function D = similarityMatrix(clusterMatrixAll)

N = size(clusterMatrixAll,1);
D = zeros(N,N);
for i=1:N
 for j=1:i
     if (clusterMatrixAll(i) == clusterMatrixAll(j))
         D(i,j) = 1;
     else
         D(i,j) = 0;
     end

     D(j,i) = D(i,j);% D is the similarity matrix
 end
end
```

5. clusterMatrix.m

```
function clustMatrix = clusterMatrix(inputMatrix, numberOfGroups)

%get cluster matrix from N groups, defined by numberOfGroups

 linkageMatrix = linkage(inputMatrix, 'ward');
 clustMatrix = cluster(linkageMatrix,'MaxClust',numberOfGroups);
end
```

Immune Profiling- Single cell suspensions prepared as described in the main methods were divided into two tubes and stained with fluorescently tagged antibodies to either a lymphocyte or myeloid panel.

Panel	Antibody	Source	Catalogue #	Flurophore	T cells	CD4+ T cells	CD8+ T cells	T regs	NK Cells	B cells	G-MDSC	Monocytes	MΦ	M1 MΦ	M2 MΦ	Dendritic cells
Lymphoid	Anti-mouse CD45 (30-F11)	ThermoFisher	47-0451-82	APC-EF780	+	+	+	+	+	+						
Lymphoid	Anti-mouse TCRB (H57-597)	ThermoFisher	15-5961-82	APC	+	+	+	+								
Lymphoid	Anti-mouse CD4 (GK1.5)	ThermoFisher	15-0041-82	PE-CYN5		+		+								
Lymphoid	Anti-mouse CD8A (KT15)	Santa Cruz	SC-53473	PE			+									
Lymphoid	Anti-mouse FOXP3 (FJK-16S)	ThermoFisher	56-5698-82	FITC				+								
Lymphoid	Anti-NK1.1 (PK136)	ThermoFisher	45-5941-82	PerCP-CYN5.5					+							
Lymphoid	Anti-CD19	ThermoFisher	48-0193-82	EF450						+						
Meyloid	Anti-mouse CD45 (30-F11)	ThermoFisher	47-0451-82	APC-EF780							+	+	+	+	+	+
Meyloid	Anti-mouse CD11b (M1/70)	BD Pharmingen	561960	APC							+	+	+	+	+	
Meyloid	Anti-mouse LY-6G (RB6-8C5)	ThermoFisher	12-5931-82	PE							+					
Meyloid	Anti-mouse LY6C (HK1.4)	ThermoFisher	45-5932-82	PerCP-CYN5.5								+				
Meyloid	Anti-mouse MHCII (iA/1E)	ThermoFisher	56-5321-82	AF700								+	+	+	+/-	+
Meyloid	Anti-mouse F4/80 (BM8)	ThermoFisher	48-4801-82	EF450									+	+	+	
Meyloid	Anti-mouse CD11C (N418)	ThermoFisher	15-0114-82	PE-CYN5												+
Meyloid	Anti-iNOS/NOS	BD Pharmingen	610330	FITC										+		

The table above lists the antibodies to markers used in the immune panels along with which leukocyte

subsets they identify. Both panels included anti-CD45 (pan leukocyte) antibody and DAPI stain to exclude dead cells from analysis.

Supplementary References

1. Veeramachaneni R, Yu W, Newton JM, et al. Metformin generates profound alterations in systemic and tumor immunity with associated antitumor effects. *J Immunother Cancer* 2021;9(7) doi: 10.1136/jitc-2021-002773 [published Online First: 2021/07/08]
2. Newton JM, Hanoteau A, Sikora AG. Enrichment and Characterization of the Tumor Immune and Non-immune Microenvironments in Established Subcutaneous Murine Tumors. *J Vis Exp* 2018(136) doi: 10.3791/57685
3. Frederick M, Skinner HD, Kazi SA, et al. High expression of oxidative phosphorylation genes predicts improved survival in squamous cell carcinomas of the head and neck and lung. *Sci Rep* 2020;10(1):6380. doi: 10.1038/s41598-020-63448-z
4. Liberzon A, Birger C, Thorvaldsdóttir H, et al. The Molecular Signatures Database (MSigDB) hallmark gene set collection. *Cell Syst* 2015;1(6):417-25. doi: 10.1016/j.cels.2015.12.004
5. Monti S, Tamayo P, Mesirov J, et al. Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. *Machine Learning* 2003;52(1):91-118. doi: 10.1023/A:1023949509487