Interpretation with LIME of classification in high-dimensional tabular data

2020

Introduction

TODO

Data Description

The dataset was uploaded to Kaggle by/thanks to UCI Machine Learning Repository:
https://www.kaggle.com/murats/gene-expression-cancer-rnaseq

Source: Samuele Fiorini, samuele.fiorini@dibris.unige.it, University of Genoa, redistributed under Creative Commons license.

The data consists of 801 patients.

The 20532 independent variables are all RNA sequencing gene expression levels measured by a sequencing platform (Illumina HiSeq).

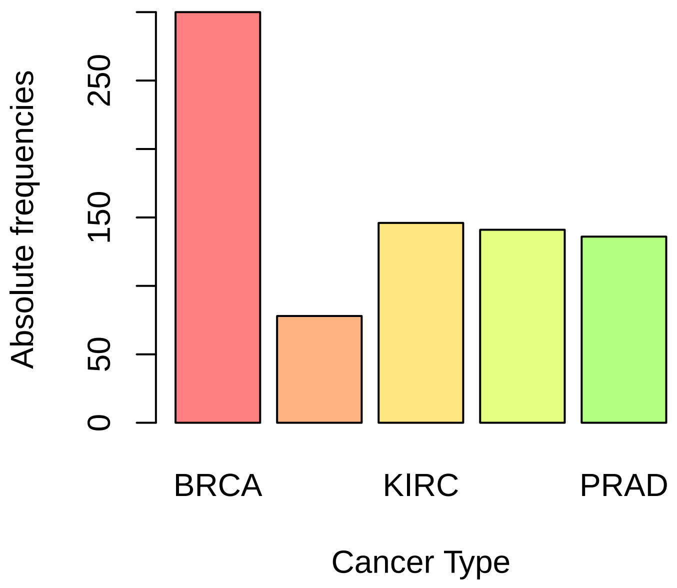
The dependent categorical variable represent primary tumors occurring in different parts of the body, covering 5 tumor types including:

- lung adenocarcinoma (LUAD)
- breast carcinoma (BRCA)
- kidney renal clear-cell carcinoma (KIRC)
- colon adenocarcinoma (COAD)
- prostate adenocarcinoma (PRAD)

A machine learning model will be trained to predict the type of tumor for new patients.
TODO: buscar un dataset similar pero que contenga tambien pacientes sanos.

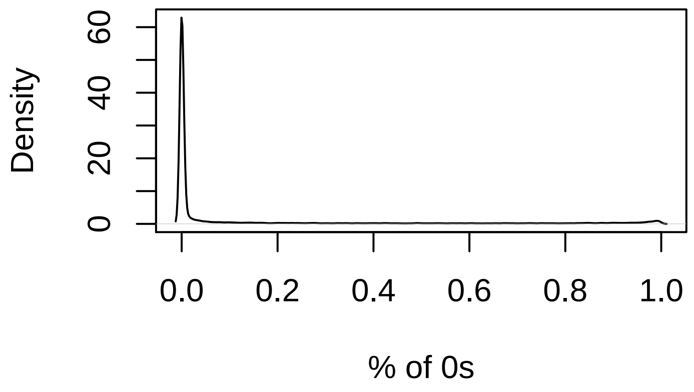
Data Analysis and Preprocessing

Absolute frequencies of the response:

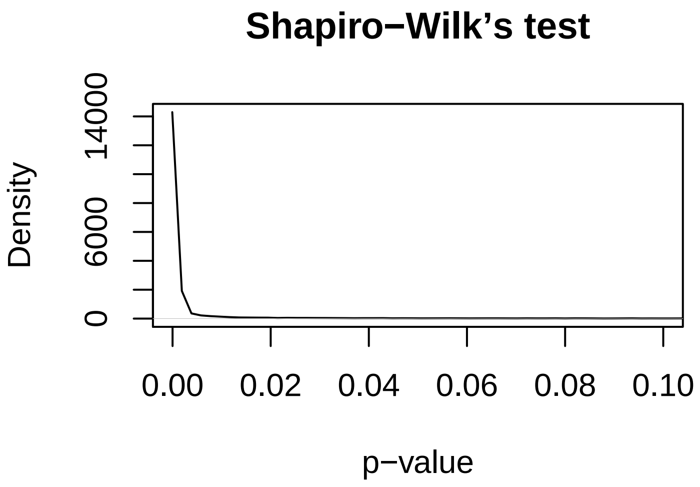


268 gene expression variables contain only 0s are removed.

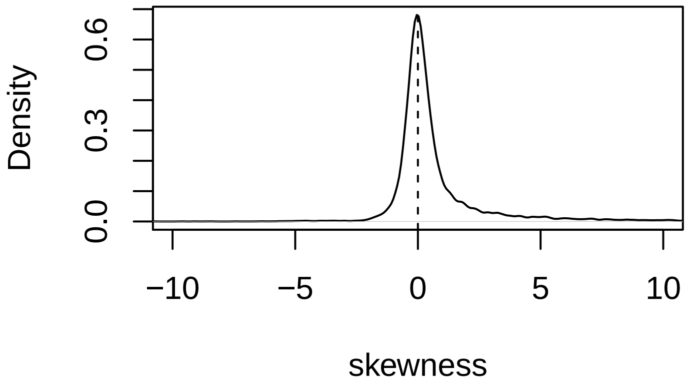
From the remaining 20264 gene expressions, 670 have at least 95% of 0s:

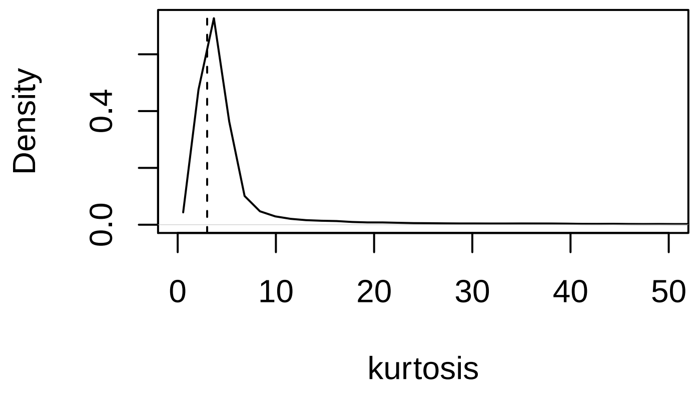


The predictors don't follow distributions close to normal. Nornality cannot be assumed:

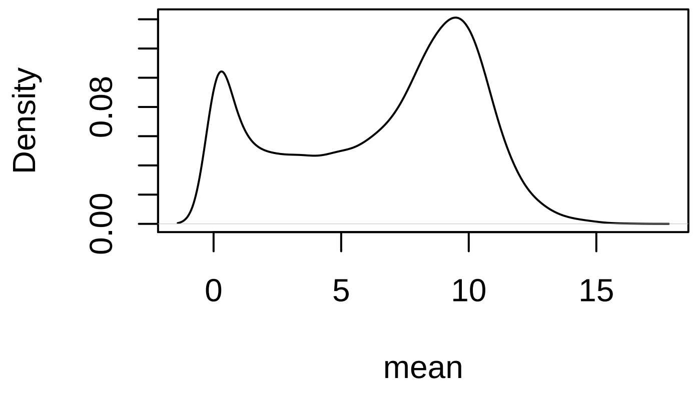


Symmetry and kurtosis:





The distribution of means suggests there are two categories of gene expression:

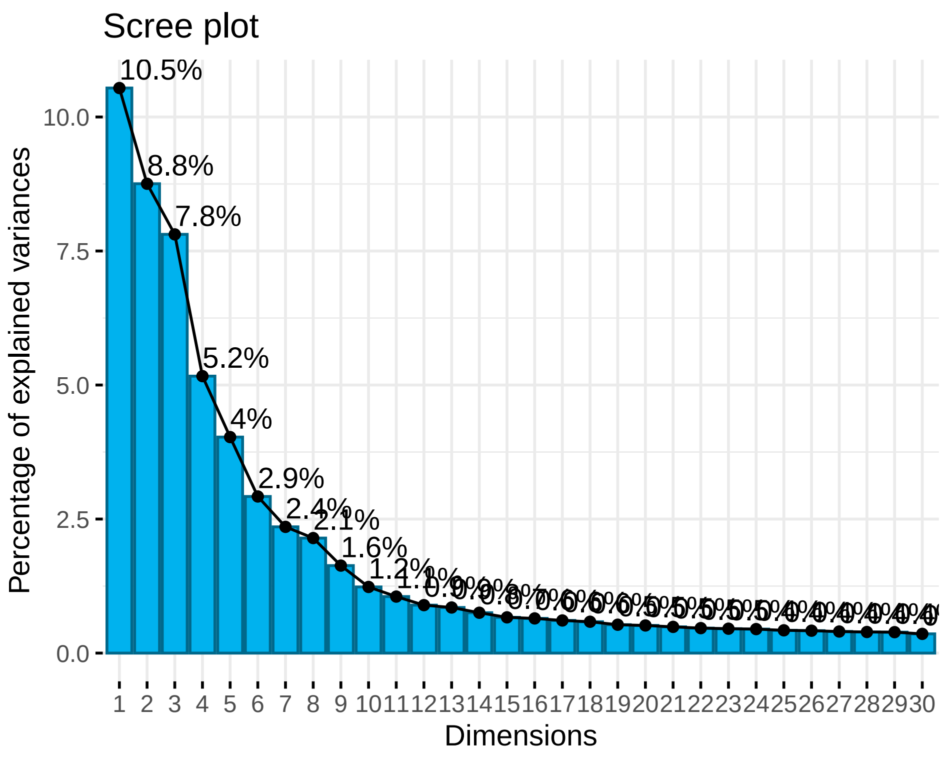


Outliers:

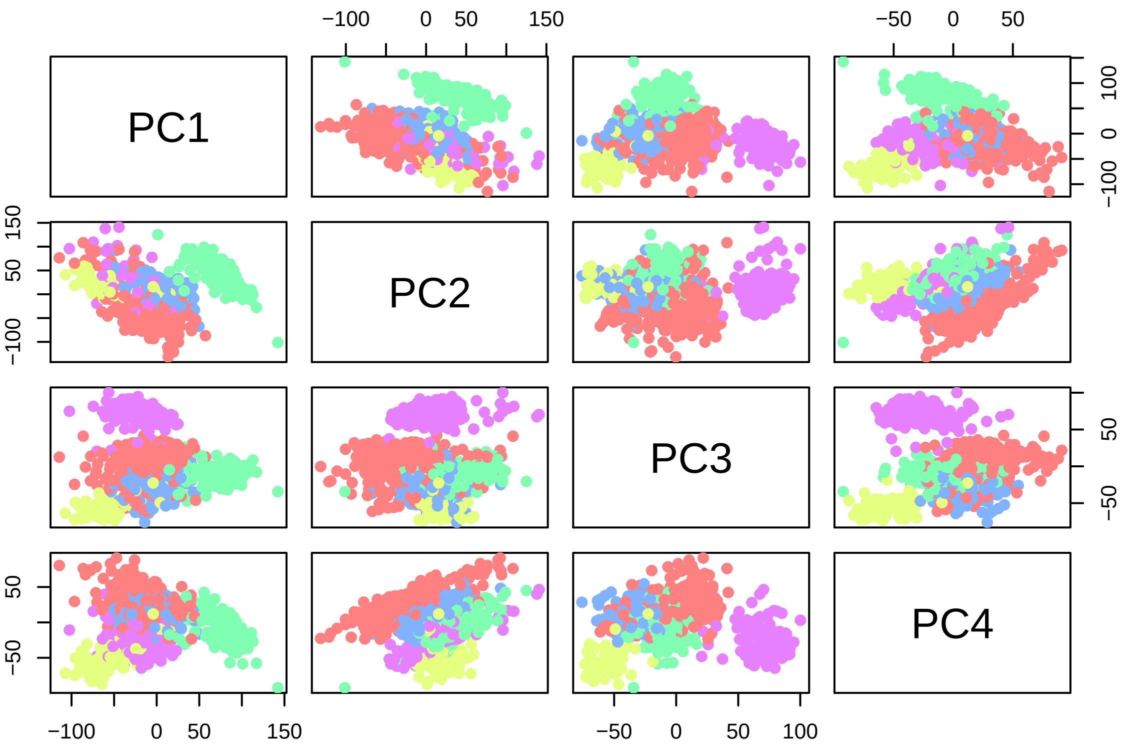
TODO

The data is scaled to help with the training of the prediction model.

At least 45% of the variability of the data is explained by the first 10 PCA components:

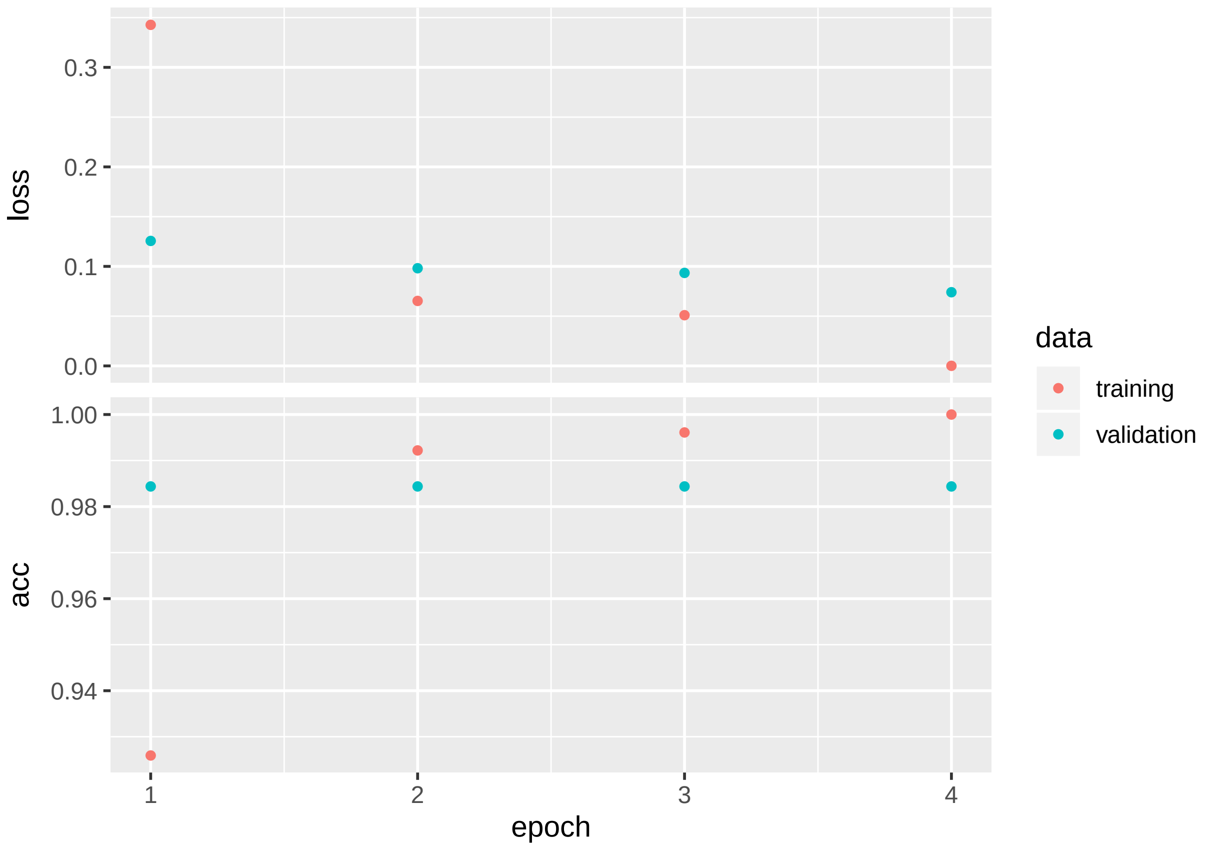


First 4 PCs look enough to classify the types of cancer despite only accounting for 32.27% of the variability of the data, which suggests high collinearity:


PC1
PC2
PC3
PC4
−100
0
50
150
−50
0
50
100
−100
50
150
−50
50
−50
0
50
100
0
−100

Predictions with a fully connected network

A fully connected network is trained with 80% of the data:



The accuracy of the network is close to 1 (sometimes 1 depending on the random initialization of the network):

```
## $loss
## [1] 0.01714346
##
## $acc
## [1] 0.99375
```

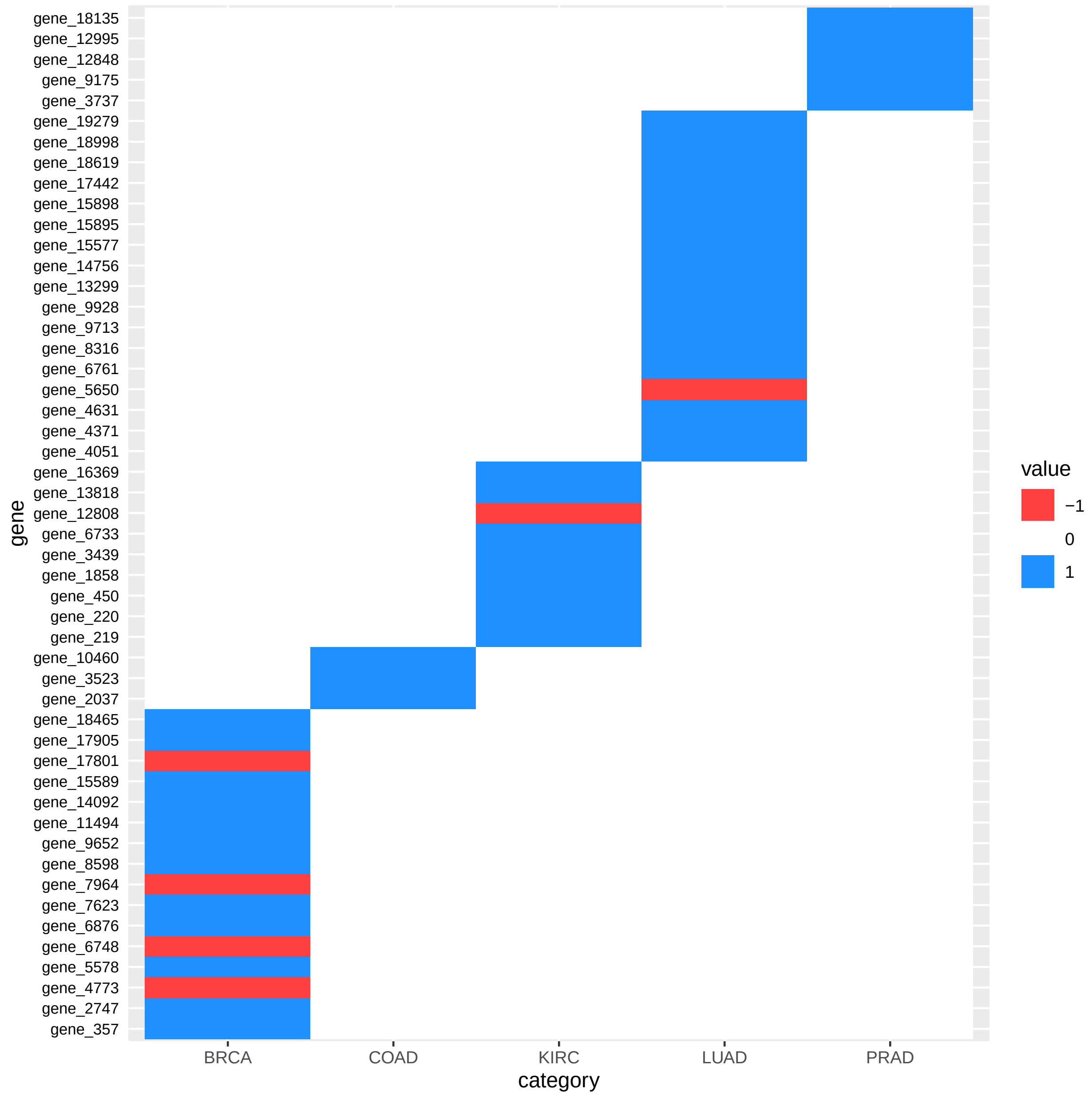
Global interpretation with a surrogate model

Neural networks use to be powerful prediction tools but they come with a cost in terms of interpretability of the predictions. They contain a huge number of parameters making difficult the identification of features that are influential in the response of the model.

To tackle the interpretability issue of the prediction model (which we'll call *black box* from now on), a surrogate interpretable model will be fitted in parallel with the training data that was used to fit the black box. This surrogate model will help us to undestand the relationship between the features and the response at a global level (i.e. for no particular prediction).

For the interpretable surrogate model we'll use lasso regularization, as it performs feature selection efficiently for high-dimensional data and it's very easy to understand.

Heatmap showing the more influential genes (selected by the lasso with optimal lambda):

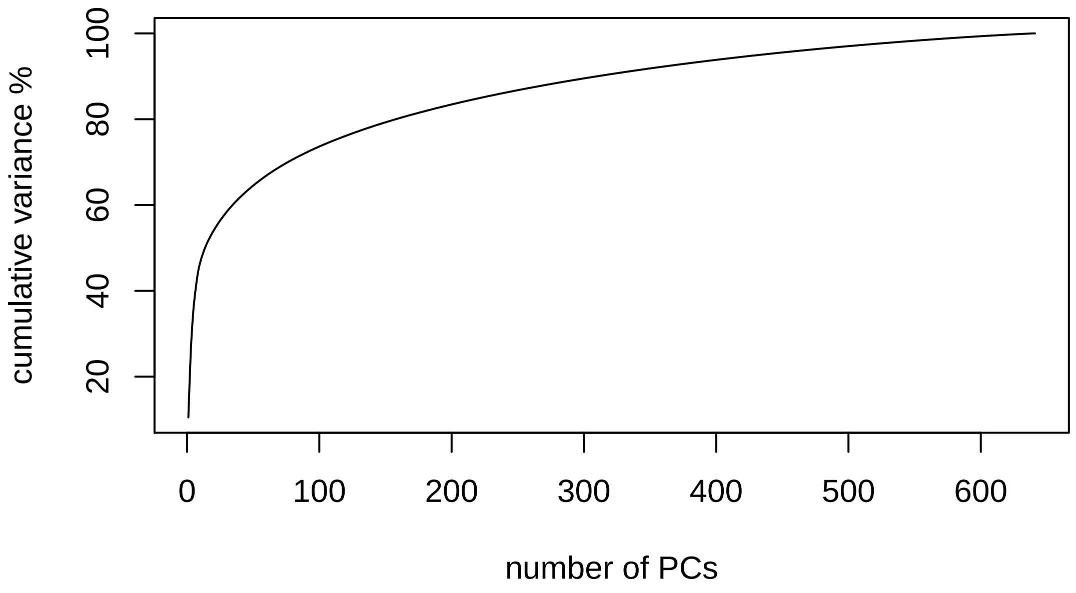


The genes selected by the lasso give a global sense of the more influential features in driving the model. However a potential issue with high-dimensional data is the high variance in the interpretations, depending on the selected observation the influential genes will be completely different even for the same category, specially in regions of the data space that are complex (non-linear).

The data representation for the interpretations can be different from the data representation used in the predictions. We could use whatever is more convenient for the interpretation, as long as we keep a mapping between both data representations.

To get a more interpretable data representation, we do dimension reduction with PCA. This also will help with the high variance issue. The expert in the domain has to advise if the representation would be useful for interpretation. The expert might be able to understand the meaning of the main components.

The number of components in the interpretable data representation could be reduced to a percentage of the explained variance, since probably only a few dozens of them will be really influencial. We'll start with the first 100 PCs for now.



Because the purpose of this model is interpretation, we'll add a parameter on top of lasso to select the number of features we want to interpret the black box model. We'll fit several models with different lambdas and pick the lambda with the desired number of selected features (or close to that number, it will depend on the given lambda range). This is done for each category. With a fixed number of features we'll be able to fairly compare the surrogate model with other models with the same number of features as we'll see later with LIME.

In addition, the output in the lasso fitting won't be the predicted categories but the predicted probabilities, so we can better understand how the black box makes predictions.

The 10 more influential features will be selected.

```
get_surrogate_model <- function(category, x, y = black_box_train_pred, plot_feat_sel = TRUE){

  feat_selection = glmnet(x, y[,category+1], alpha = 1, nlambda = 300)

  # code I picked up from the LIME package: https://github.com/thomasp85/lime/blob/49df0a131deee4919a29
  has_value <- apply(coef(feat_selection)[-1,], 2, function(x) x != 0)
  f_count <- apply(has_value, 2, sum)  # number of parameters for each lambda (columns in lasso_sparse)
  # In case that no model with correct n_feature size was found return features <= n_features
  lambda_index <- rev(which(f_count <= n_features))[1]
  features <- which(has_value[, lambda_index])

  if (plot_feat_sel)
  {
    plot(feat_selection, xvar="lambda", label = TRUE, main = c("BRCA","COAD","KIRC","LUAD","PRAD")[categ
    abline(v = log(feat_selection[["lambda"]][rev(which(f_count <= n_features))[1]]), lty=2)
  }

  glmnet(x[,features], y[,category+1], alpha = 0, lambda = 2 / nrow(x))
}
```

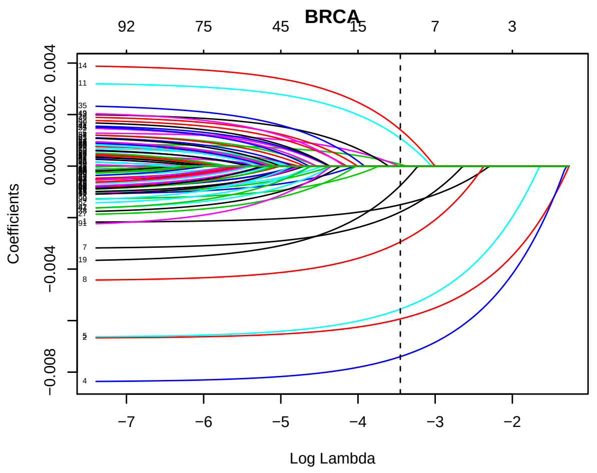
```
##
## R^2 for BRCA: 0.950763
## MSE for BRCA: 0.011477
```

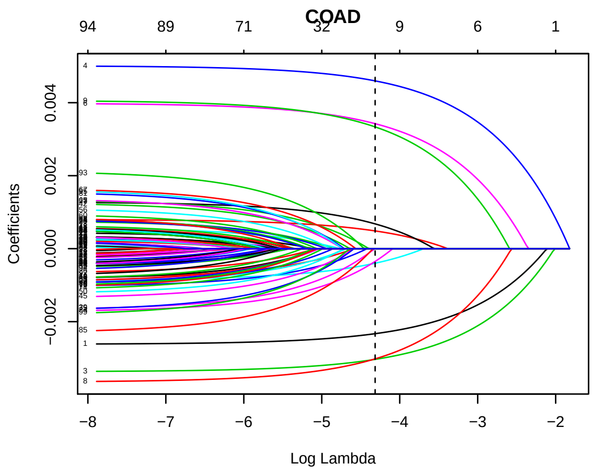
```
##
## R^2 for COAD: 0.938616
## MSE for COAD: 0.005363
```

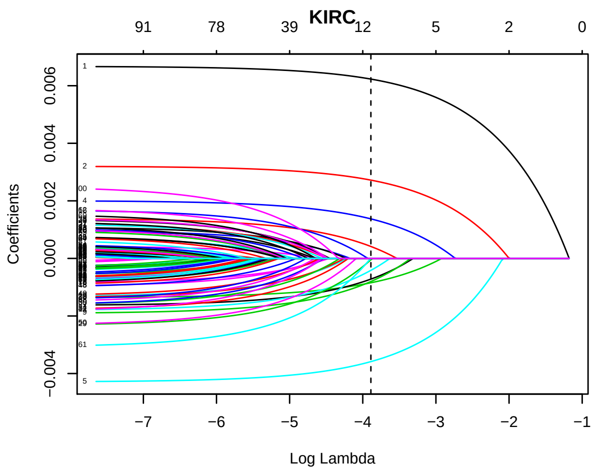
```
##
## R^2 for KIRC: 0.952313
## MSE for KIRC: 0.007020
```

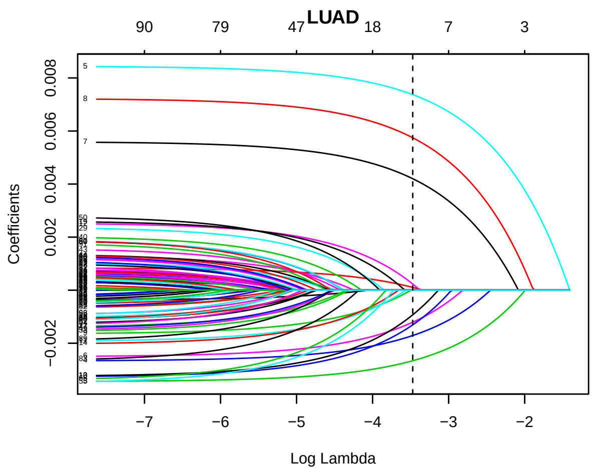
```
##
## R^2 for LUAD: 0.908083
## MSE for LUAD: 0.013662
```

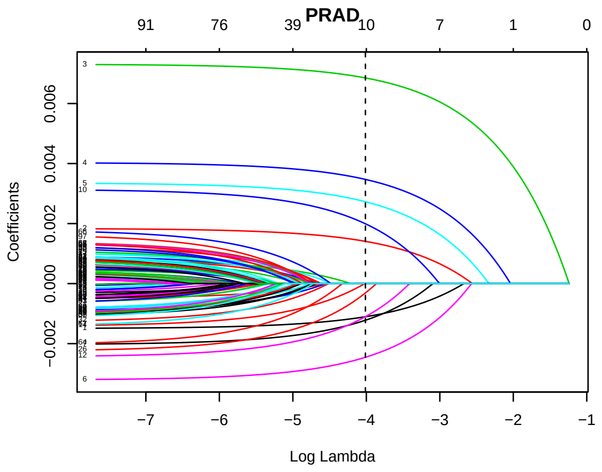
```
##
## R^2 for PRAD: 0.956345
## MSE for PRAD: 0.006061
```











The plots above show the coefficients of the selected features for each category. The higher the absolute value, the higher the influence (globally) in the output.

R^2 values are high for all the categories, but the big assumption here is that the data is linear.

The vertical lines represent the computed lambdas 10 features (or close to 10 features). Because the number of features is fixed, the selected values for lambda are not optimal for the fitting. Not a problem since the MSE is very low for all the categories.

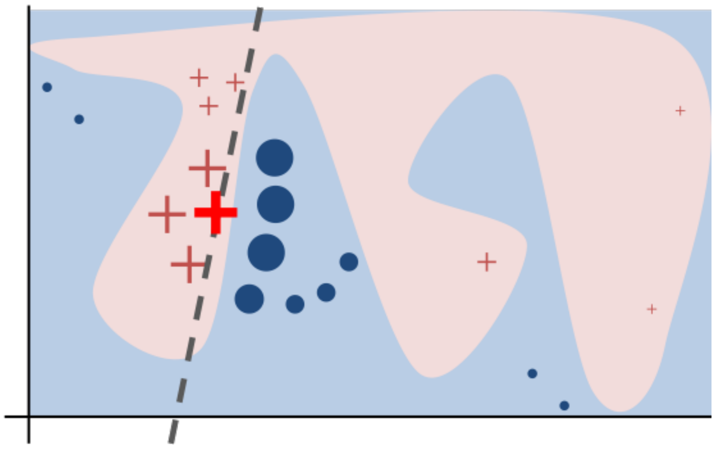
TODO: que pasa si el accuracy/MSE es mucho peor que el accuracy/MSE del black box model?

LIME

Intuition

TODO

(image taken from https://github.com/marcotcr/lime)



Formulation

TODO

Algorithm

- In order to fit a local linear model around an observation which classification we want to explain, we need enough data in the surroundings of that observation. To achieve that, kernel densities estimations are computed for each variable and sampled to "simulate" new data, reducing the sparsity of the data and therefore the variance in the local fit for the explanation.

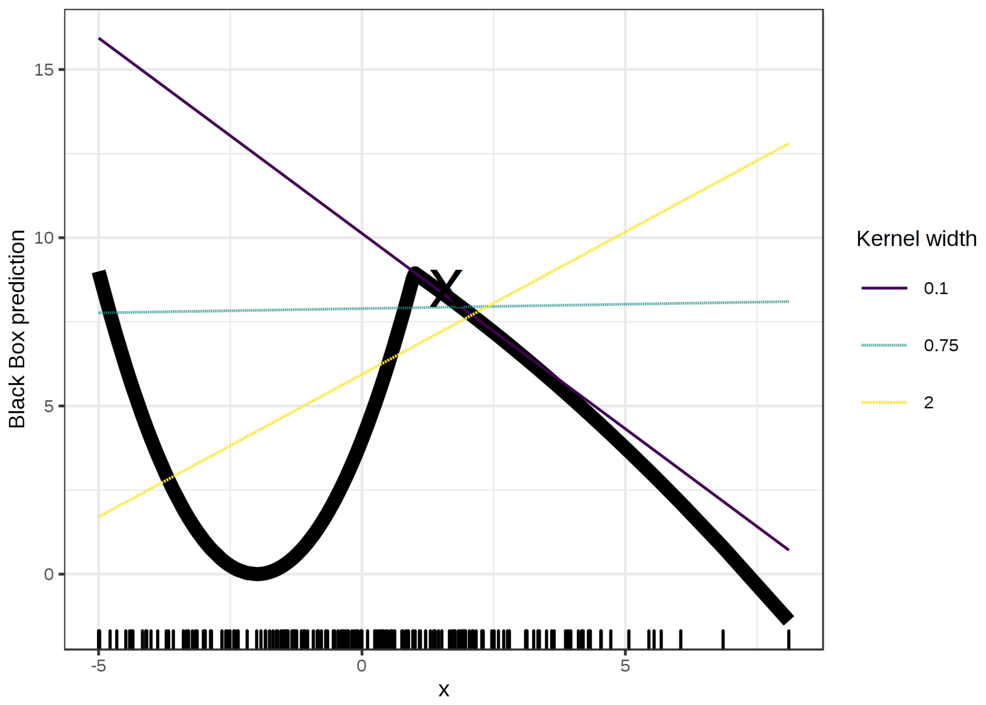
The parameter to tune in this step is the number of "simulated" data points (**n_permutations**).
The risk if the parameter is too low is the high variance of the simulated data each time the prediction for a same observation is interpreted. If each time the observation is explained the simulated data is too different, the variance in the interpretation will also be high, making the interpretations inconsistent and untrustworthy.

TODO: no veo inconveniente en incrementar este parametro al maximo hasta que las limitaciones en CPU/memoria lo permitan.

TODO: Los estimated kernel densities de cada variable no tienen en cuenta las relaciones entre las variables lo cual puede generar data points "irreales". Idea -> usar multivariate kernel density estimation para generar un data set mas real (aplicar dimensionality reduction si es demasiado costoso).

- Of data points simulated by sampling from the features kdes, we are specially interested in those in the surroundings to the observation of interest, since we are fitting a local model. So we need to give more weight to the data close to the instance of interest and this is done with a smoothing exponential kernel. The width of the kernel is a parameter of the LIME model (**kernel_width**) and probably the more tricky one as shown in the following example of a 1-dimensional dataset:

(image taken from https://christophm.github.io/interpretable-ml-book/lime.html)



In this example changes in the kernel width lead to drastic changes in the local linear fit for the instance of interest (the cross in the plot). Adding more dimensions would increase the sensitivity of the width parameter even more.

In the *lime* package, the default value is $0.75\sqrt{p}$ - the more number of dimensions the more the data space is sparse and therefore the kernel width is increased depending on p.
The appropriate value seems to depend on the surroundings of the data point being explained so there isn't a clear rule of thumb to follow for this parameter.

Too small values could lead to insufficient data to fit the local model and too much variance in different interpretations for the same observation to explain.
Too high values could lead to loosing "locality" and hence incorrect interpretations.

TODO: Buscar algun criterio, quizas basado en la complejidad (clasificaciones con menos o mas certidumbre) para seleccionar el kernel width.

Another parameter is the distance function that measures the proximity between the instance of interest and the simulated data points. The default option is Gower's distance but others like Euclidean or Manhattan (see *?dist()* for details) can also be used.

TODO: Elegir la funcion mas adecuada teniendo en cuenta:
- la alta dimensionalidad
- la alta colinearidad
- variables no-normales (pero tampoco exponenciales)
- las variables han sido standardizadas

- The next step is to feed the black box with the simulated data to get the classifications required to fit the local model.

- With the simulated data and the corresponding predictions, a linear model is fit. Several local models are available (**feature_select**). We'll choose lasso for two reasons: the high-dimensionality of the data and to get a better comparison with the global surrogate lasso model. Lasso will use the weights from the smoothing kernel to give more influence to the neighborhood to the original observation to be explained.

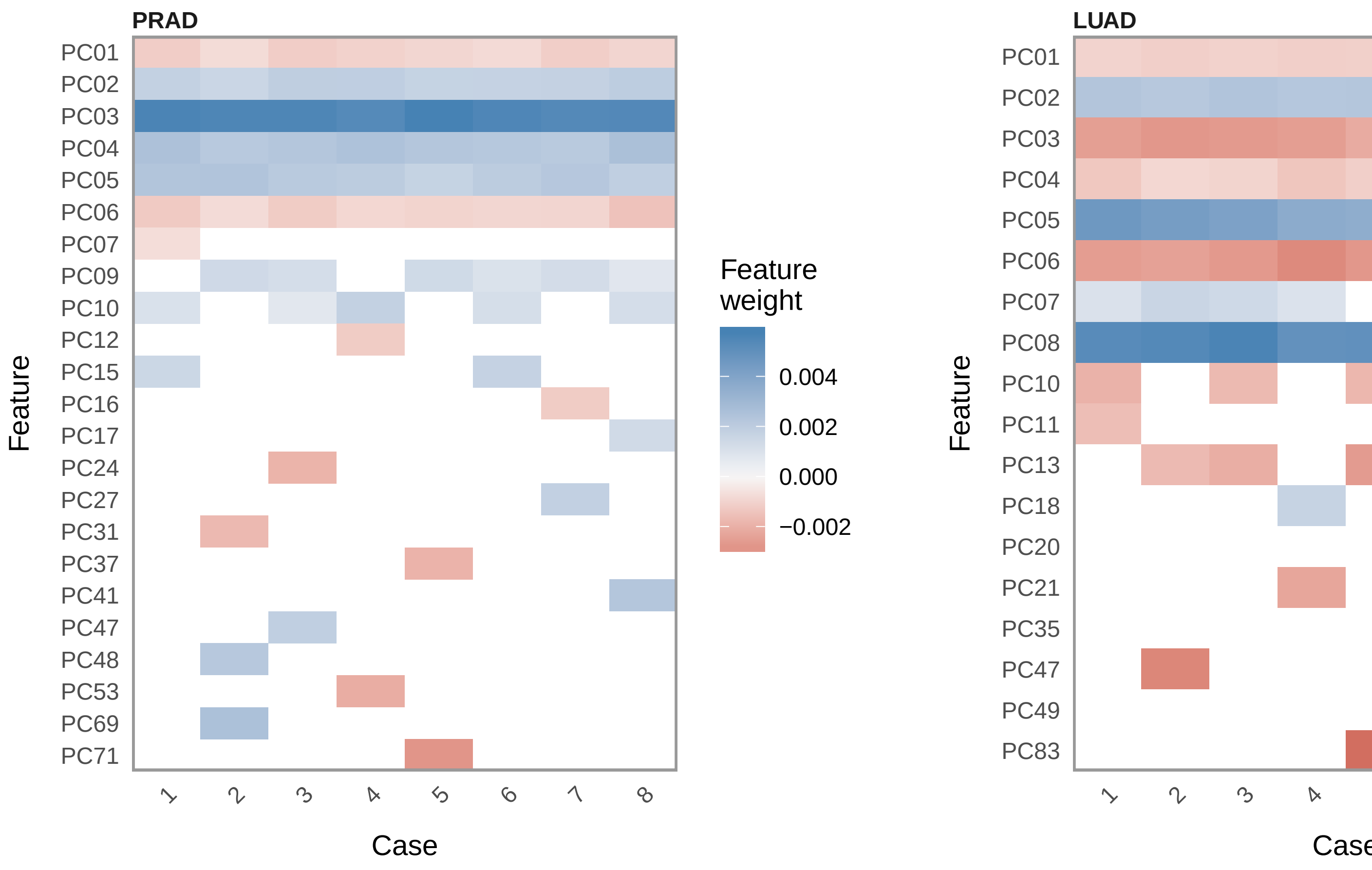
- Finally the coefficients with higher absolute values are selected (**n__features**) to explain the output (**n__labels**, *1* if we are just interested in the selected category).

Interpretation with PCA

Let's use LIME to make interpretations of the predictions of individual observations in the training data with PCA reduction.

```r
get_PCA_explanation <- function(datapoints_index, n_permutations = 2000, kernel_width = 0.75)
{
  preprocessing <- function(x){

    # reversing PCA (with the remaining info) to feed the black box which only understands the original
    a = as.matrix(x) %*% t(x_train_pca$rotation[,1:pcs_n])
    b = t(a) + x_train_pca$center
    t(b)
  }

  explainer_PCA <- lime(
    x = as.data.frame(x_train_pca$x)[,1:pcs_n],
    model = black_box,
    use_density = TRUE,
    preprocess = preprocessing,
    bin_continuous = FALSE
  )

  lime::explain(
    # converting train data coordinates to the PCA space:
    x = as.data.frame(x_train[datapoints_index,] %*% x_train_pca$rotation[,1:pcs_n]),
    explainer = explainer_PCA,
    n_permutations = n_permutations,
    #kernel_width = kernel_width,
    feature_select = "lasso_path",
    n_features = 10,
    n_labels = 1
  )
}
```

We analyse the consistency in the interpretations by plotting the more influential components eight times for the same observation. The test is done with two categories: PRAD (this category barely overlaps with other categories) and LUAD (this category overlaps with other categories as seen in the PCA plots - it's harder to predict).



Some observations:

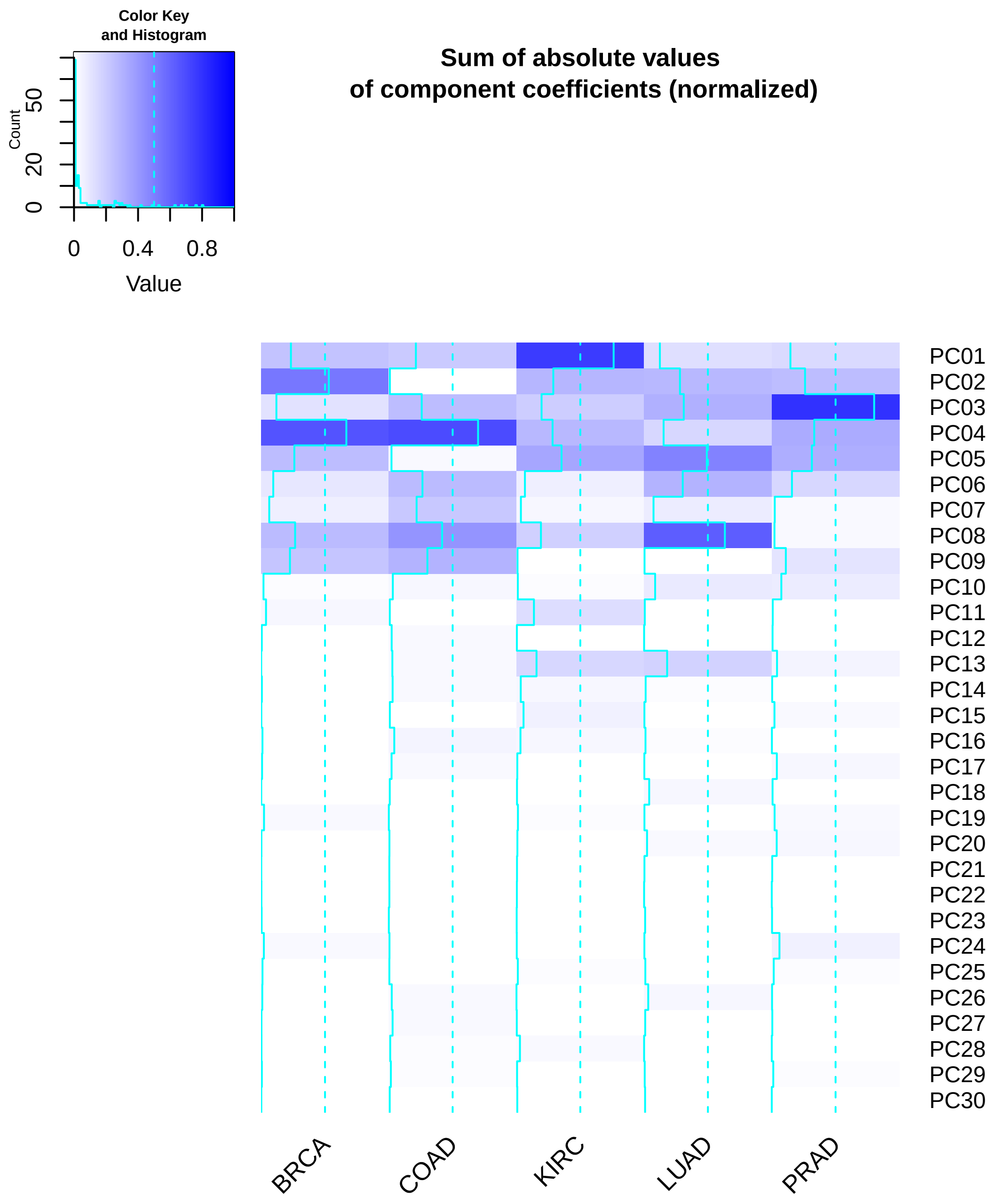
- The plots show that the interpretations are consistent.
- There is more variance in the selection of the components for PRAD (easier to predict), probably because the influence is concentrated in just one component (PC3) compared to LUAD.
- R^2 is high in both categories, specially with category PRAD:

```
##
## R^2 mean for PRAD:

## [1] 0.4905

##
## R^2 mean LUAD:

## [1] 0.3167
```

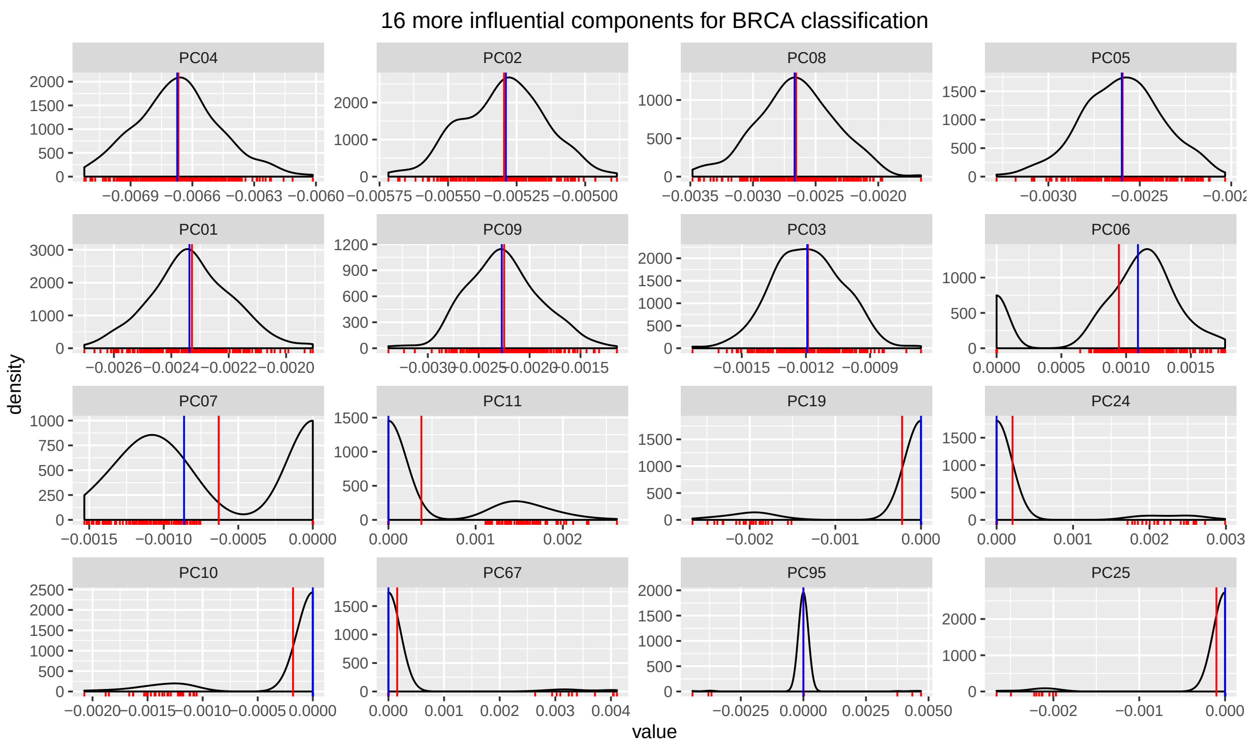
We now check the correlation between the global surrogate model and the local lime models of all the observations in all the categories, using the training data and 2000 permutations for each observation.

The coefficients of the selected features in the interpretations of all the observations for the same category are summarized in a single list of component coefficients. Each summarized component coefficient will be computed with a statistic describing the central tendency of the coefficient across the observations.

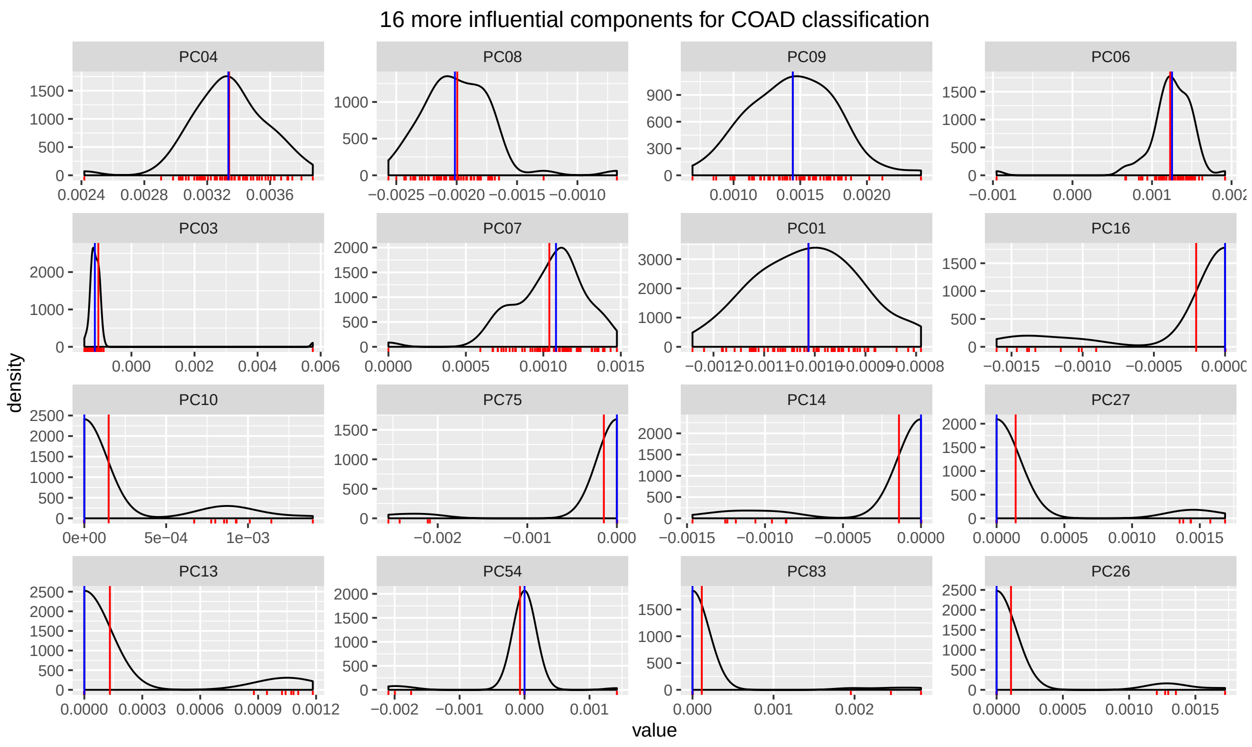
To decide which statistic to use we analyse the more influential features for each category. To measure the influence we use the sum of the absolute values of the coefficients of all the observations for each component.



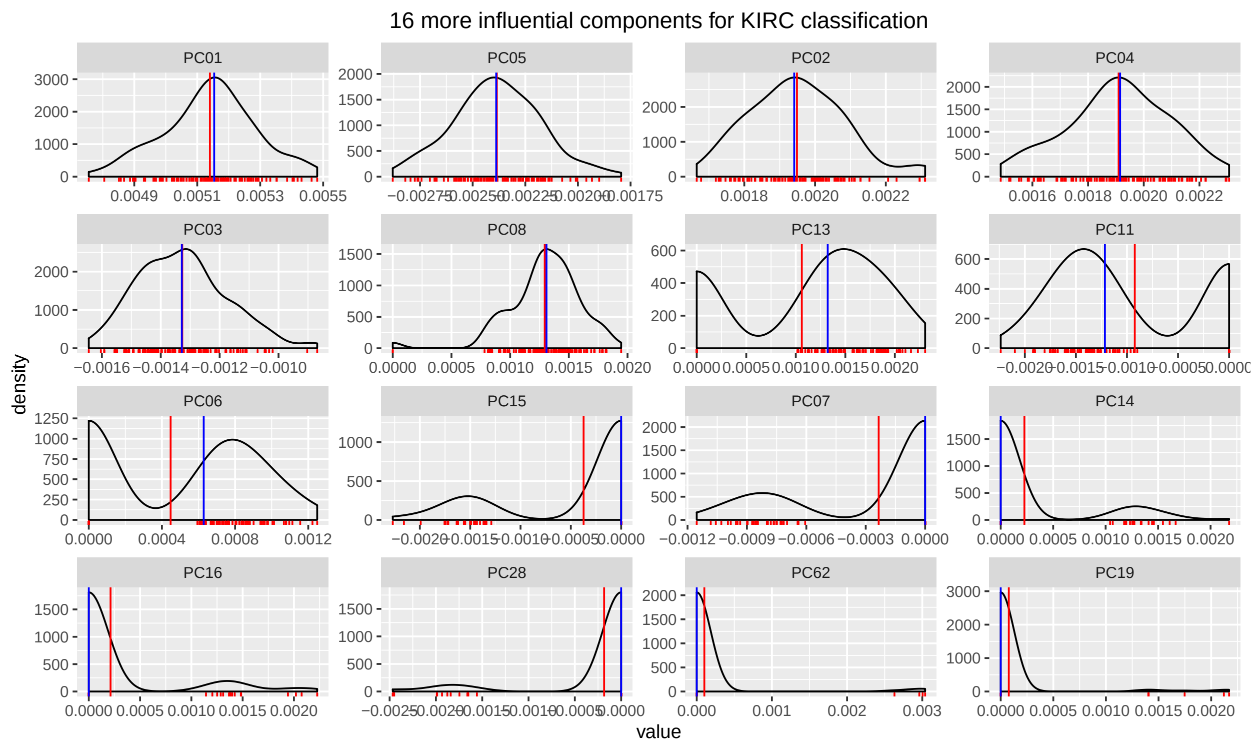
Below are displayed the distributions of the 16 more influential components for all the categories. The vertical red line represents the mean, the blue one represents the median.

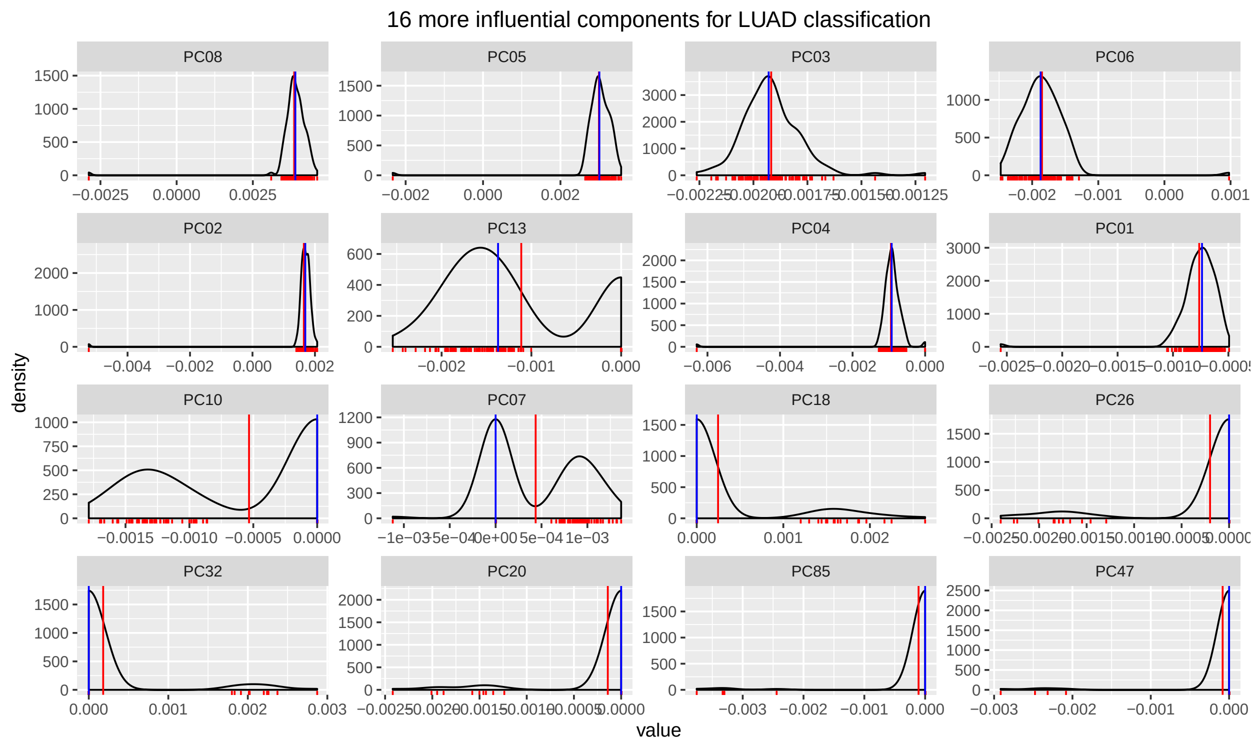


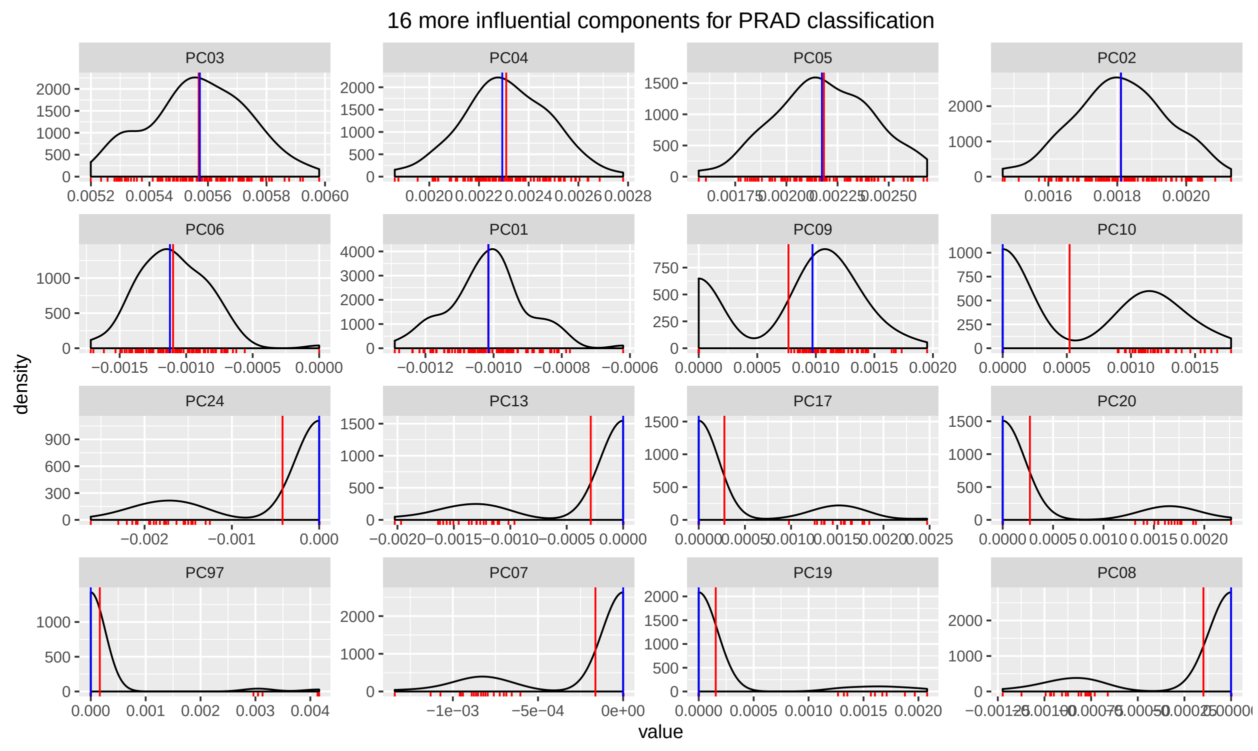
16 more influential components for BRCA classification
PC04
PC02
PC08
PC05
PC01
PC09
PC03
PC06
PC07
PC11
PC19
PC24
PC10
PC67
PC95
PC25
density
value



16 more influential components for COAD classification
PC04
PC08
PC09
PC06
PC03
PC07
PC01
PC16
PC10
PC75
PC14
PC27
PC13
PC54
PC83
PC26
density
value


16 more influential components for KIRC classification
PC01
PC05
PC02
PC04
PC03
PC08
PC13
PC11
PC06
PC15
PC07
PC14
PC16
PC28
PC62
PC19
density
value

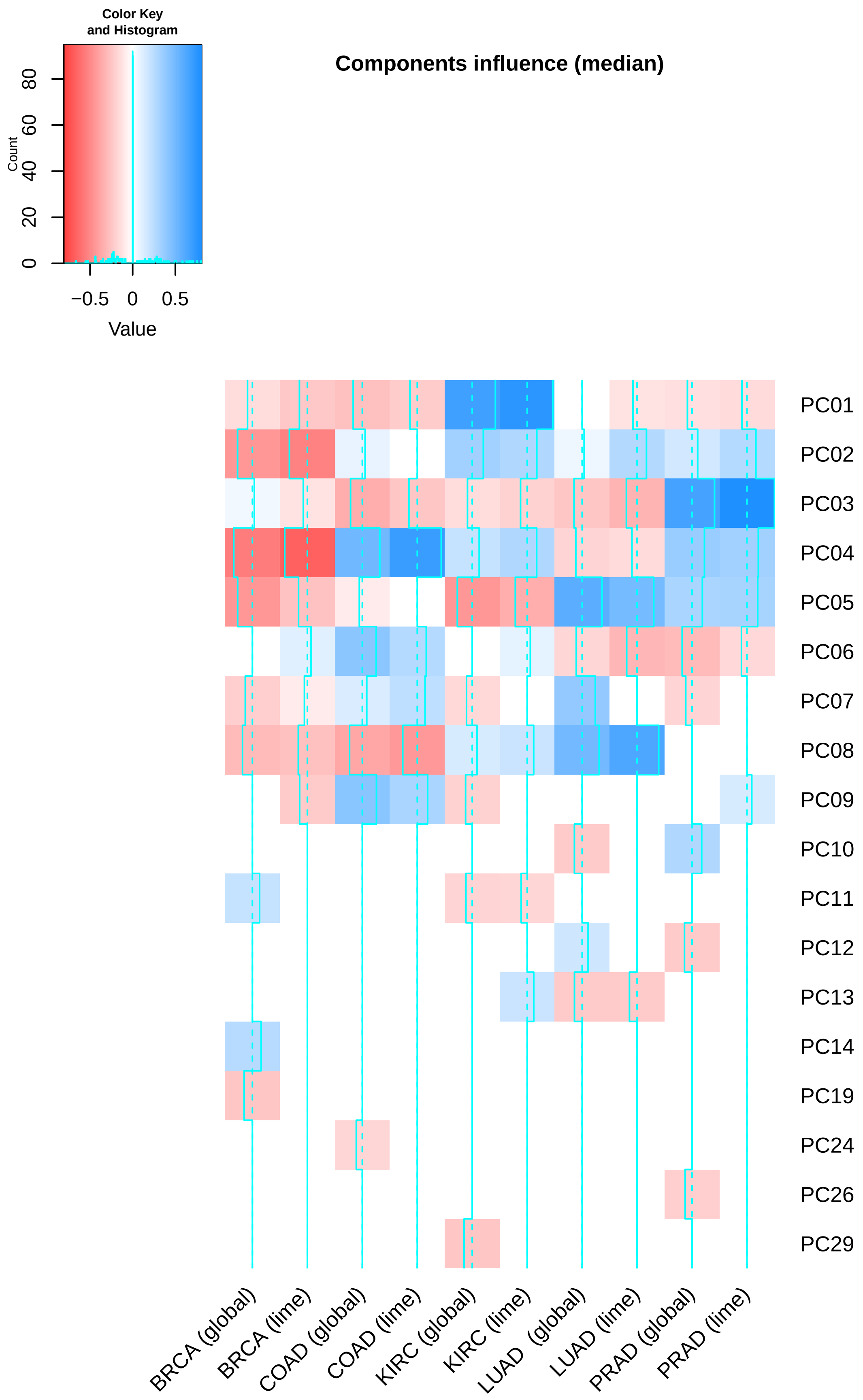

16 more influential components for LUAD classification
PC08
PC05
PC03
PC06
PC02
PC13
PC04
PC01
PC10
PC07
PC18
PC26
PC32
PC20
PC85
PC47
density
value

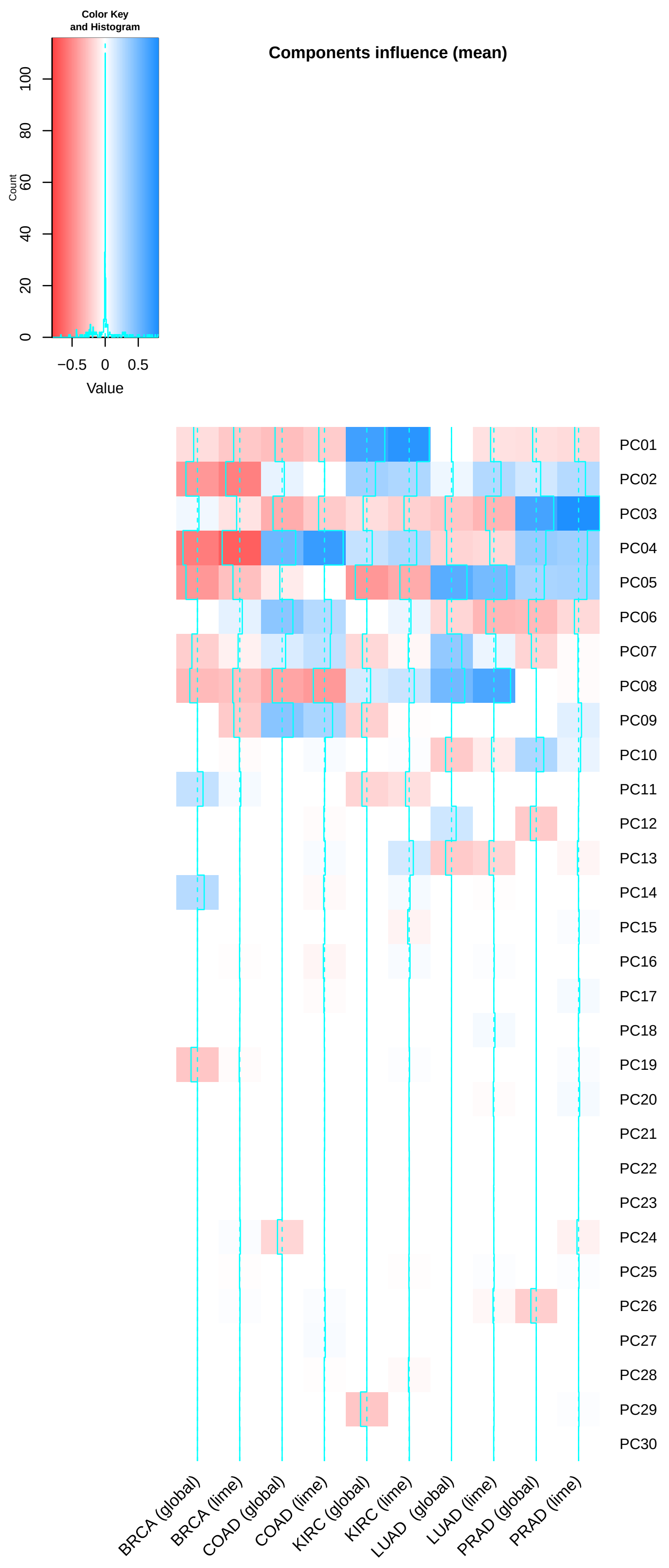


We can see that the more influential components are more or less symmetric, whereas the less influentual ones are bimodal, one of the modes lying on value 0 which represents the absence of influence for a subset of observations.

To avoid the components that only appear as influential for a few observations (and when they are included in the list of 10 more important components their value is very small) we use the median to compute the representing value of the component coefficient for the whole category. The median will ignore components that come up rarely and have small values by setting them to 0 in the summarize component coefficient.

We compute a single set of coefficients for each category using the median of the components coefficients from all the observations for both categories, and get a histogram to compare them with the surrogate lasso model coefficients:


Color Key
and Histogram
Count
0
20
40
60
80
−0.5
0
0.5
Value
Components influence (median)
PC01
PC02
PC03
PC04
PC05
PC06
PC07
PC08
PC09
PC10
PC11
PC12
PC13
PC14
PC19
PC24
PC26
PC29
BRCA (global)
BRCA (lime)
COAD (global)
COAD (lime)
KIRC (global)
KIRC (lime)
LUAD (global)
LUAD (lime)
PRAD (global)
PRAD (lime)

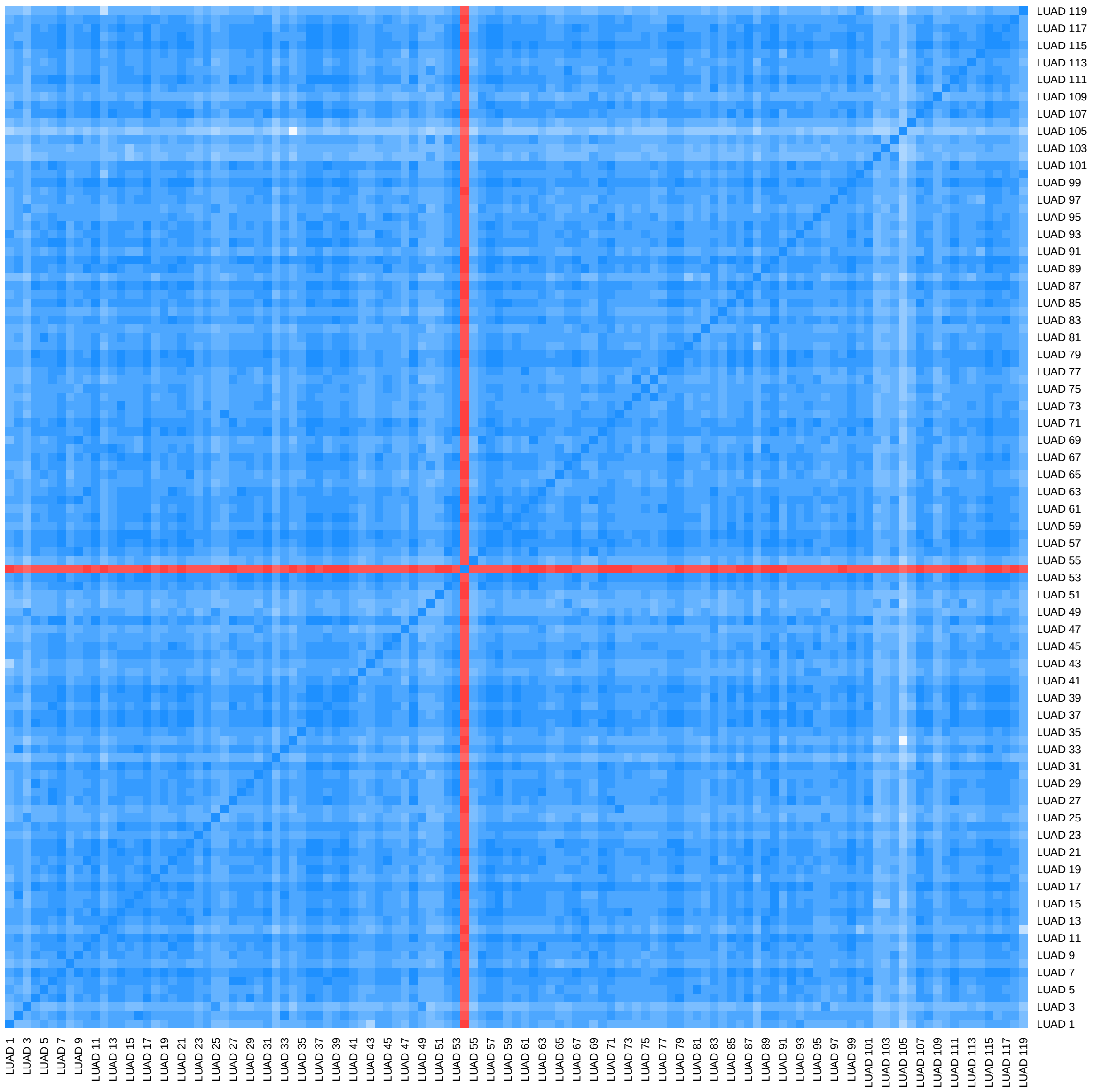

Color Key
and Histogram
Count
0
20
40
60
80
100
−0.5
0
0.5
Value
Components influence (mean)
PC01
PC02
PC03
PC04
PC05
PC06
PC07
PC08
PC09
PC10
PC11
PC12
PC13
PC14
PC15
PC16
PC17
PC18
PC19
PC20
PC21
PC22
PC23
PC24
PC25
PC26
PC27
PC28
PC29
PC30
BRCA (global)
BRCA (lime)
COAD (global)
COAD (lime)
KIRC (global)
KIRC (lime)
LUAD (global)
LUAD (lime)
PRAD (global)
PRAD (lime)

Disentangling the component coefficients of a particular observation to get the more influential individual genes:

```
LUAD_PCA_coords = x_train_pca$rotation[,as.matrix(lime_PCA_LUAD[lime_PCA_LUAD$case==1,"feature"])]
influential_components_weights = as.matrix(lime_PCA_LUAD[lime_PCA_LUAD$case==1,"feature_weight"])
gene_coeffs = LUAD_PCA_coords %*% influential_components_weights

influential_genes = names(sort(abs(gene_coeffs[,]), decreasing = TRUE)[1:100])
gene_coeffs[influential_genes,]
```

```
##    gene_15895    gene_15898    gene_11903    gene_15591    gene_15896
##  0.0002179539  0.0002122131  0.0002119146  0.0002076801  0.0002053202
##    gene_15894    gene_11352    gene_15161    gene_10674    gene_15899
##  0.0002019707  0.0002012240  0.0001954485  0.0001951199  0.0001948537
##     gene_9544    gene_13639     gene_3695     gene_9713    gene_11550
##  0.0001946533  0.0001936284  0.0001928243  0.0001891380  0.0001887668
##    gene_19648    gene_15900     gene_3964    gene_15577    gene_17394
##  0.0001885069  0.0001872786  0.0001864320  0.0001851801  0.0001832779
##      gene_638    gene_19542    gene_12224     gene_5651     gene_9001
##  0.0001802777  0.0001801109  0.0001770892  0.0001763087  0.0001747841
##     gene_7058    gene_19632    gene_15893     gene_2506     gene_7620
##  0.0001740950  0.0001728185  0.0001713295  0.0001696192  0.0001695081
##    gene_15897     gene_2319     gene_5391      gene_888     gene_2695
##  0.0001694308  0.0001686297  0.0001685750  0.0001683691  0.0001681127
##    gene_10218     gene_7998     gene_5157    gene_16283    gene_15633
##  0.0001666098  0.0001661951  0.0001657455  0.0001653013  0.0001652185
##     gene_8220    gene_12333     gene_3438    gene_18345    gene_13400
##  0.0001636932  0.0001633194  0.0001626617  0.0001621027  0.0001617970
##    gene_17770     gene_6998     gene_9209    gene_18275    gene_11391
##  0.0001616860  0.0001611069  0.0001610361 -0.0001610263  0.0001609100
##     gene_8221    gene_17393     gene_5737     gene_6160     gene_8431
##  0.0001607504  0.0001603219  0.0001601132  0.0001598720  0.0001592880
##     gene_1545     gene_4161     gene_8904    gene_15009    gene_15583
##  0.0001591133  0.0001587698  0.0001585634  0.0001580650  0.0001580189
##    gene_12870     gene_3836    gene_10353     gene_6411    gene_19279
##  0.0001579228  0.0001563204  0.0001560120  0.0001555533  0.0001546570
##     gene_5338      gene_445    gene_17174     gene_5998    gene_15681
## -0.0001542981  0.0001540522  0.0001539155  0.0001535374  0.0001533558
##      gene_423    gene_16239     gene_1340    gene_10008     gene_1846
##  0.0001531719  0.0001527249  0.0001517688  0.0001515839  0.0001515566
##     gene_9489     gene_3138    gene_13516      gene_598     gene_2818
## -0.0001506927  0.0001500486  0.0001490061  0.0001489683  0.0001484585
##     gene_1651     gene_4556     gene_1874     gene_2455     gene_6410
##  0.0001480061  0.0001478546  0.0001474712  0.0001471334  0.0001469981
##    gene_14104     gene_6978    gene_19608     gene_4422    gene_10523
##  0.0001465754  0.0001465431 -0.0001461111  0.0001459786  0.0001457639
##     gene_6985     gene_8178    gene_19525    gene_17026     gene_7442
##  0.0001456375  0.0001451669  0.0001449080  0.0001446813  0.0001445421
##     gene_4051    gene_18203    gene_11909     gene_2767    gene_16491
##  0.0001444192  0.0001442973  0.0001439638  0.0001435850 -0.0001434922
```

Correlation heatmap of interpretations for all the observations of category LUAD:


LUAD 119
LUAD 117
LUAD 115
LUAD 113
LUAD 111
LUAD 109
LUAD 107
LUAD 105
LUAD 103
LUAD 101
LUAD 99
LUAD 97
LUAD 95
LUAD 93
LUAD 91
LUAD 89
LUAD 87
LUAD 85
LUAD 83
LUAD 81
LUAD 79
LUAD 77
LUAD 75
LUAD 73
LUAD 71
LUAD 69
LUAD 67
LUAD 65
LUAD 63
LUAD 61
LUAD 59
LUAD 57
LUAD 55
LUAD 53
LUAD 51
LUAD 49
LUAD 47
LUAD 45
LUAD 43
LUAD 41
LUAD 39
LUAD 37
LUAD 35
LUAD 33
LUAD 31
LUAD 29
LUAD 27
LUAD 25
LUAD 23
LUAD 21
LUAD 19
LUAD 17
LUAD 15
LUAD 13
LUAD 11
LUAD 9
LUAD 7
LUAD 5
LUAD 3
LUAD 1