Marcel Spitzer

Feature Engineering and Visualization

last run 8 months ago · R notebook · 9998 views
using data from Mercedes-Benz Greener Manufacturing · Public

Report Code Data (1) Output (2) Comments (32) Log Versions (58) Forks (26) Fork Script

Tags data visualization pca

Report

Feature Engineering and Visualization

- Data Preparation
- Feature Engineering
  - One-Hot Encoding
  - Hierarchical Clustering
  - Principal Component Analysis (PCA)
  - Logistic PCA
  - Independent Component Analysis (ICA)
  - *NEW:* Multiple Correspondence Analysis (MCA)
- Backtransformation

Hi everybody,

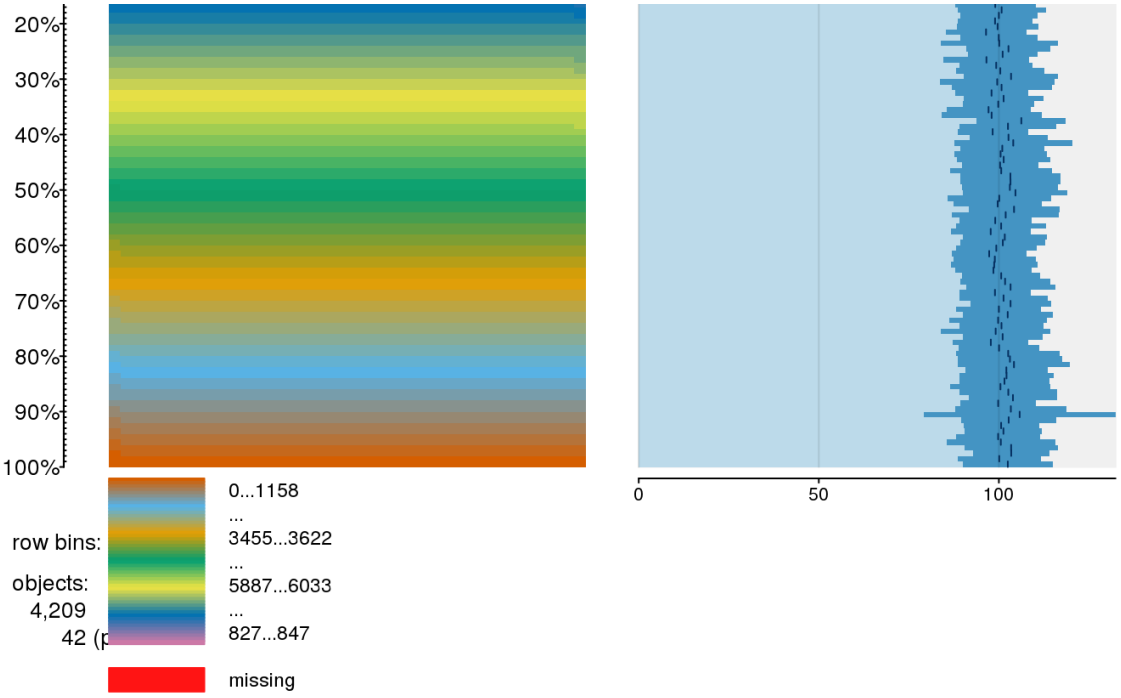
in this notebook, I would like to share with you some approaches for Feature Engineering and Visualization.

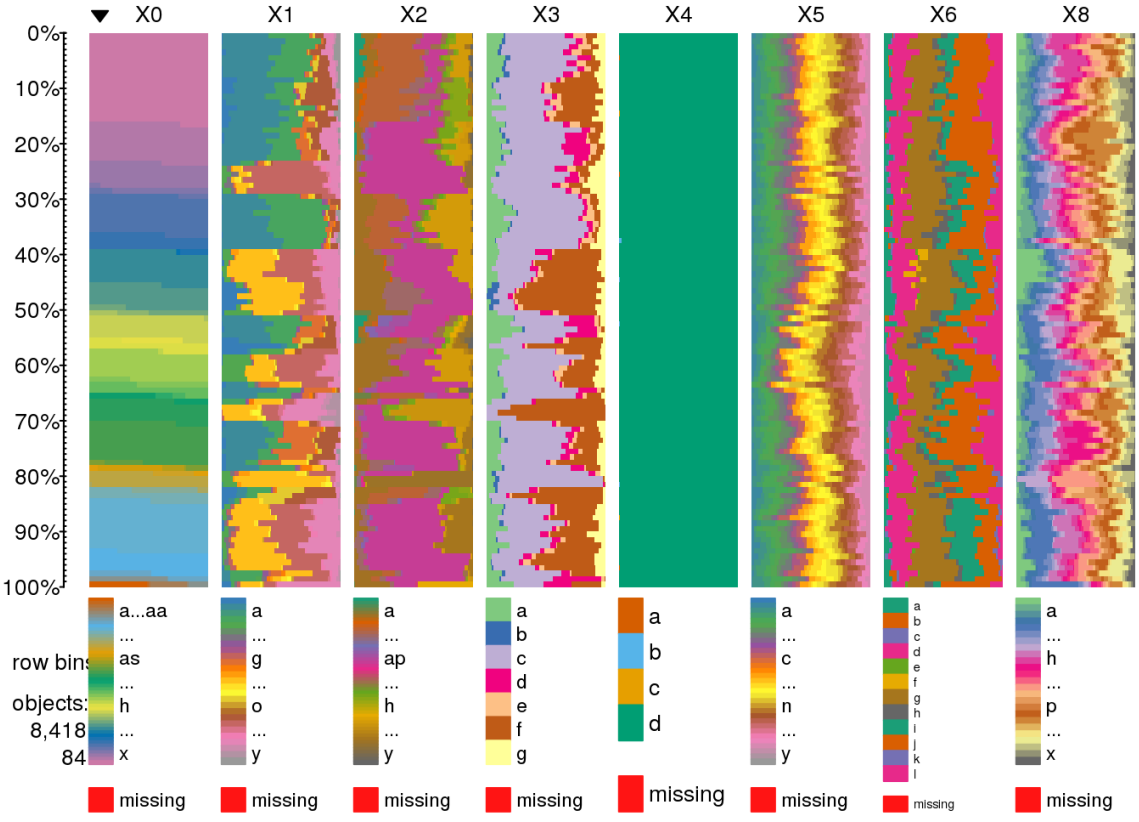
Data Preparation

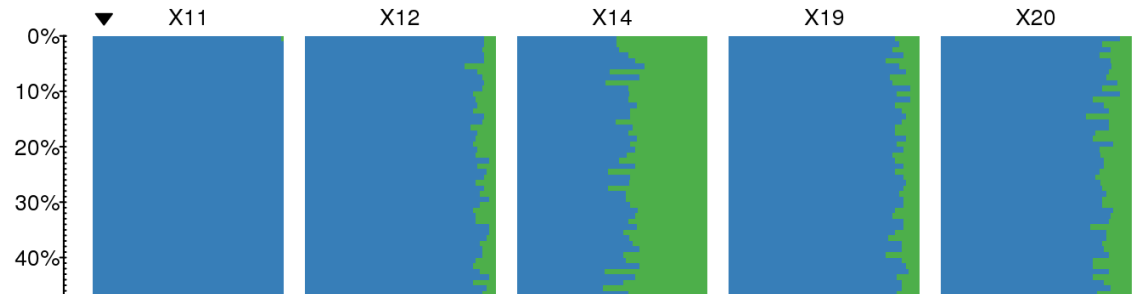
To apply every transformation on both the training and the test set, I separate the id and target columns and combine the feature columns.

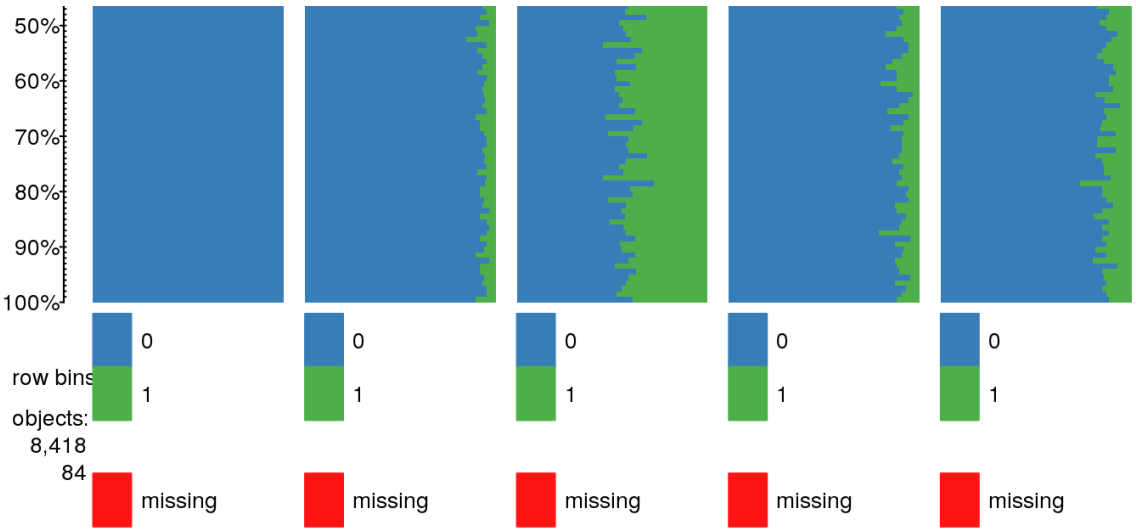
```
library(tidyverse)

train <- read_csv("../input/train.csv")
test <- read_csv("../input/test.csv")

# isolate id and target variables
train_id <- data.frame(ID=as.character(train$ID))
train_labels <- data.frame(y=train$y)
test_id <- data.frame(ID=as.character(test$ID))
train$ID <- NULL
train$y <- NULL
test$ID <- NULL

# combine features from train and test set
df_all <- rbind(train, test)

# split groups of categorical and binary features
categorical_vars = paste0("X", c(0,1,2,3,4,5,6,8))
categorical_df <- df_all %>% select(one_of(categorical_vars))
binary_df <- df_all %>% select(-one_of(categorical_vars))
```

Before applying any of the transformations, let's take a look at the raw features.


20%
30%
40%
50%
60%
70%
80%
90%
100%
0
50
100
row bins:
objects:
4,209
42 (p
0...1158
...
3455...3622
...
5887...6033
...
827...847
missing


X0
X1
X2
X3
X4
X5
X6
X8
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
row bins:
objects:
8,418
84
a...aa
as
h
x
a
g
o
y
ap
b
c
d
e
f
n
p
missing


X11
X12
X14
X19
X20
0%
10%
20%
30%
40%



We can see, that there are lots of binary features and some categorical which we are able transform into binary features via One-Hot encoding.

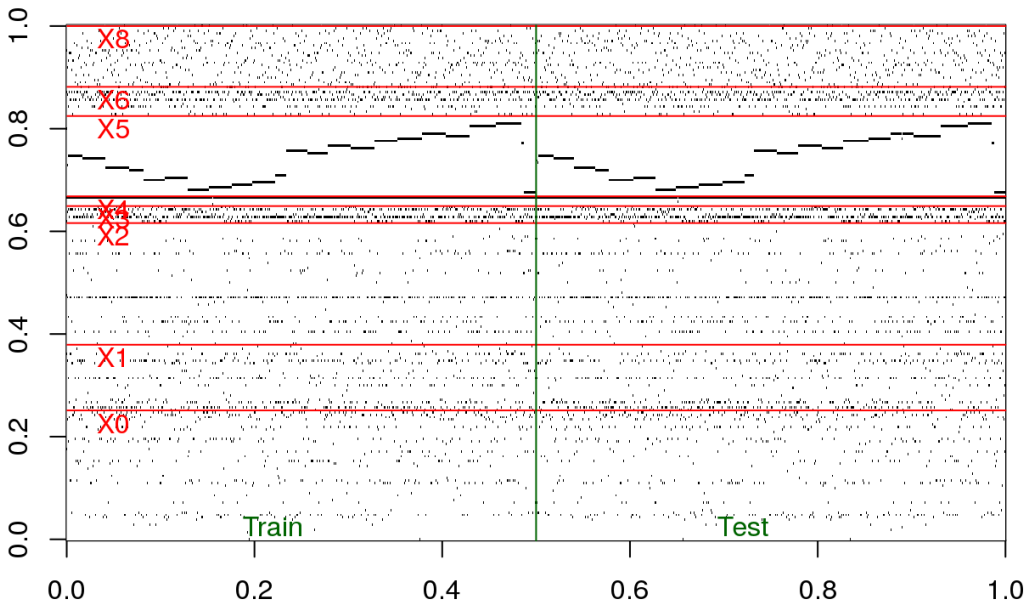
Feature Engineering

Now, let's apply some Feature Engineering techniques in order to compute some new features which are possibly better predictors. Every transformed column will be appended to the existing dataframe so that we don't lose any information.

One-Hot Encoding

The first technique which I want to apply is One-Hot encoding of the categorical variables. Since we don't know if there exists an implicit ordering of the categorical features, it is a reasonable way to create binary dummy variables.

```
library(caret)

# perform one-hot encoding
dmy <- dummyVars(~., data = categorical_df)
ohe_features <- data.frame(predict(dmy, newdata = categorical_df))

df_all <- cbind(df_all, ohe_features)
binary_df <- cbind(binary_df, ohe_features)

binary_df_train <- binary_df[1:nrow(train), ]
binary_df_test <- binary_df[(nrow(train)+1):nrow(binary_df),]

# visualize one-hot encoded features
image(as.matrix(ohe_features), col=c("white", "black"))
n_levels <- apply(categorical_df, 2, function(x){length(unique(x))})
n_levels <- n_levels/sum(n_levels)
abline(h=cumsum(n_levels), col="red")
text(0.05, cumsum(n_levels)-.025, names(n_levels), col="red")
abline(v=0.5, col="darkgreen")
text(0.22, 0.025, "Train", col="darkgreen")
text(0.72, 0.025, "Test", col="darkgreen")
```



In the pixel map above, we can see the one-hot encoded dummy variables. Each row represents a binary dummy variable and each column represents an observation. The red lines separate dummy variables of different categorical variables.

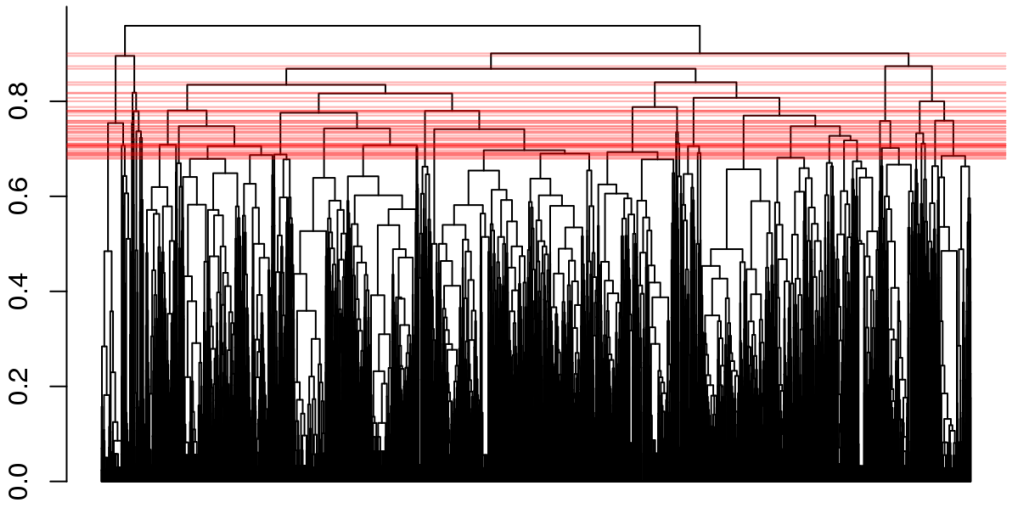
In this plot, one thing that is really outstanding is the pattern in X5. It could be an indicator that the observations were not randomly selected. Maybe X5 was a test parameter which was hold constant for several consecutive runs of the test. That would suggest a temporal ordering of the observations.

Hierarchical Clustering

In the next step, I would like to cluster the binary variables and use cluster indices of different clusterings as new features. To measure distance between binary vectors, I use Jaccard's distance.

```
library(proxy)

# compute distance matrix
jdist <- proxy::dist(binary_df, method = "Jaccard")

# perform hierarchical clustering
hc <- hclust(jdist)

# get all clusterings with 2 up to max_k clusters
max_k <- 50
clusters <- data.frame(sapply(2:max_k, function(k){ cutree(hc,k) }))
colnames(clusters) <- paste0("hc_group_", 2:max_k)

# add lines for each cut in the dendrogram
plot(hc, hang = -1, labels = FALSE, xlab = "", ann=FALSE)
cuts <- sort(hc$height, decreasing = TRUE)[2:max_k]
```

```
abline(h=cuts, col=alpha("red",0.3))
```



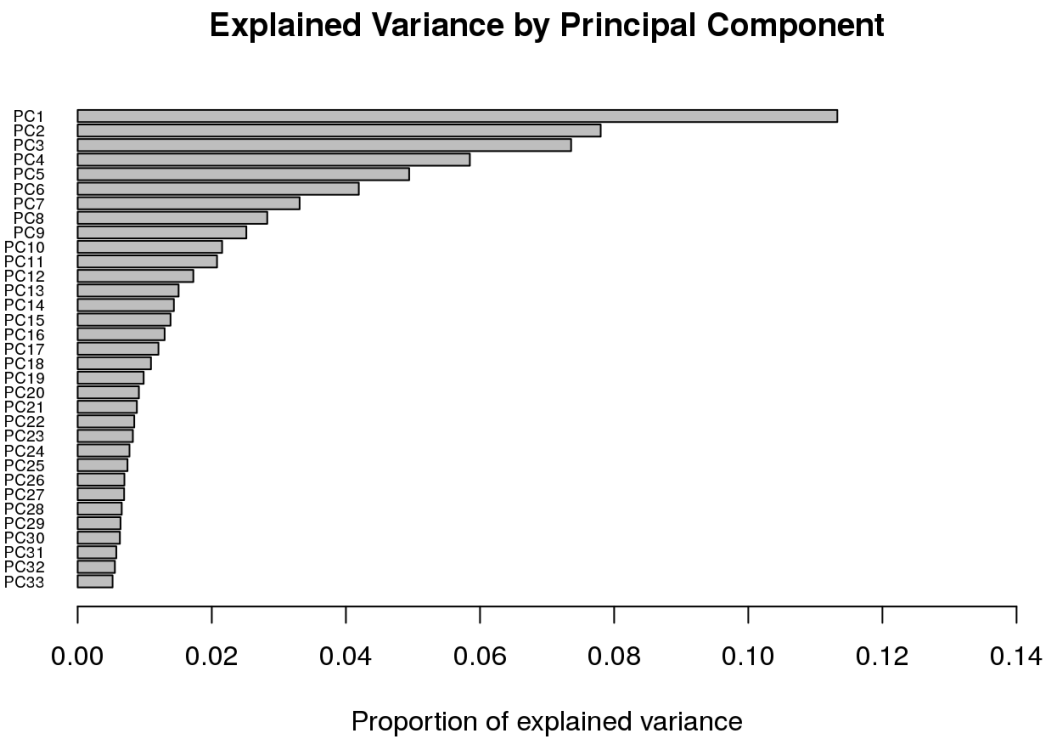
The red lines show the cuts we did by setting the number max_k of the maximum number of clusters. Each cut results in a clustering with the appropriate number of clusters (the intersections in the dendrogram). For each clustering, we add the cluster indices of the observations to our feature set.

If you want to read more about hierarchical clustering, please check out my latest kernel Analysis of clusters and outliers (https://www.kaggle.com/msp48731/analysis-of-clusters-and-outliers), where I described the approach in much more details and interpreted the results.

Principal Component Analysis (PCA)

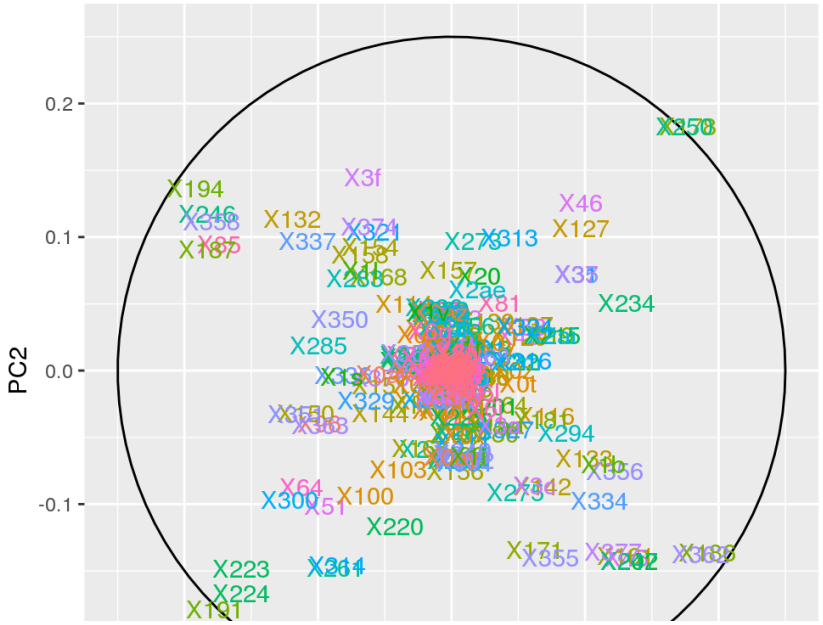
Next, I would like to perform a PCA on the binary features (including one-hot encoded). Notice that unlike in the previous vesions of this kernel I retain all of the principal components.

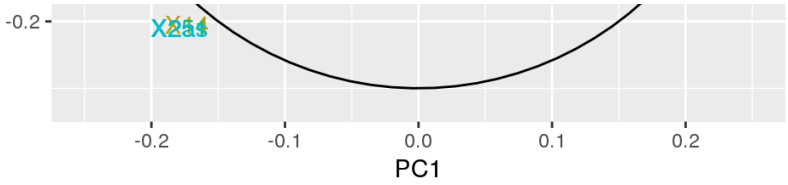
```
# perform pca
res_pca <- prcomp(binary_df_train)
pca_features <- predict(res_pca, newdata = binary_df)

# proportion of explained variance
importance_pca <- summary(res_pca)$importance
barplot(sort(importance_pca[2, importance_pca[2, ] > 0.005], decreasing = FALSE), horiz = TRUE, xlim=
c(0,0.14),
        las=1, cex.names=0.6, main="Explained Variance by Principal Component", xlab="Proportion of e
xplained variance")
```



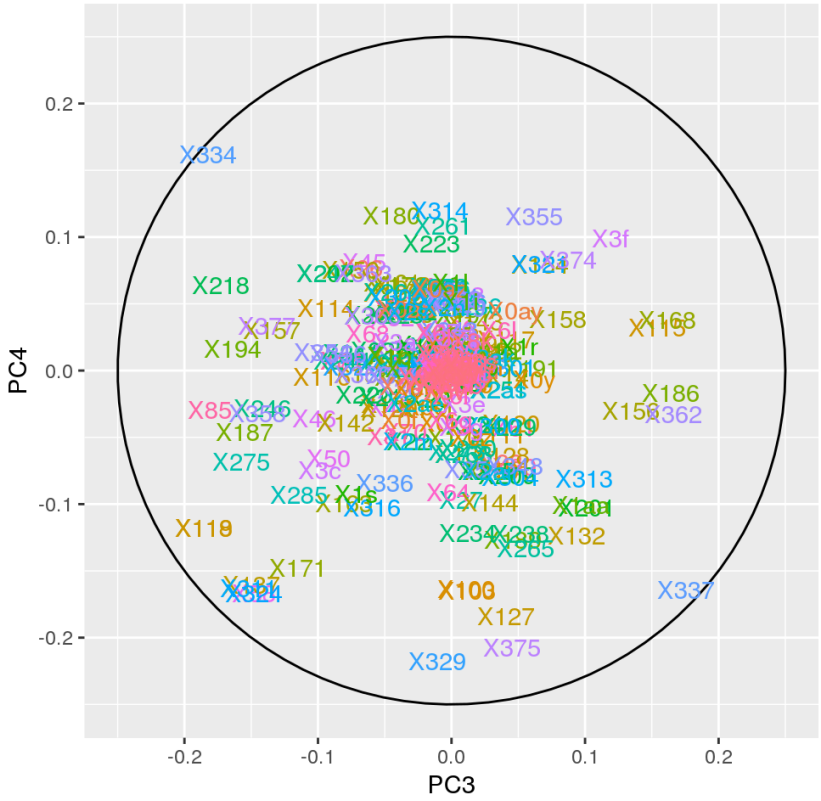
Next, let's look at the coefficients of the principal components

```
# visualize the impact of the original variables on principal components
theta <- seq(0, 2*pi, length.out = 100)
circle <- data.frame(x = cos(theta)/4, y = sin(theta)/4)

ggplot(circle,aes(x,y)) + geom_path() +
  geom_text(data=data.frame(res_pca$rotation, .names = row.names(res_pca$rotation)),
              mapping=aes(x = PC1, y = PC2, label = .names, colour = .names)) +
  coord_fixed(ratio=1) + labs(x = "PC1", y = "PC2") + theme(legend.position="none")
```

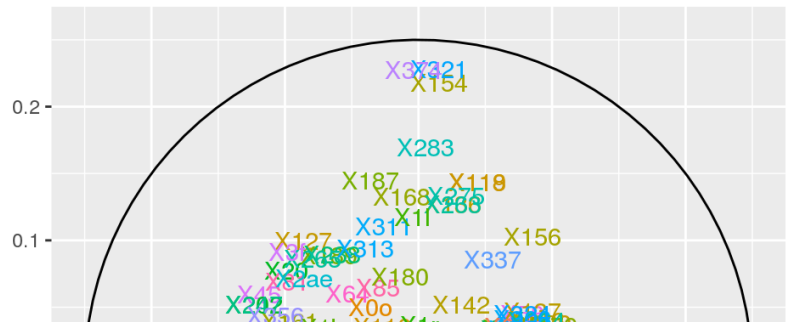


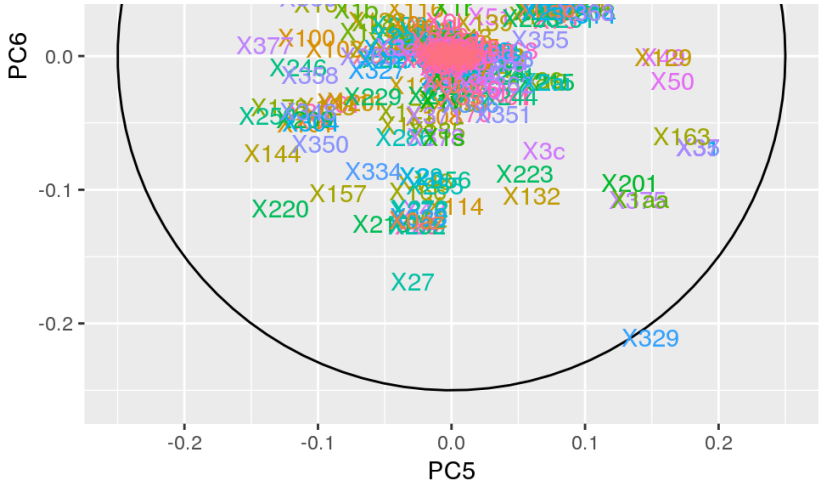



```
ggplot(circle,aes(x,y)) + geom_path() +
  geom_text(data=data.frame(res_pca$rotation, .names = row.names(res_pca$rotation)),
            mapping=aes(x = PC3, y = PC4, label = .names, colour = .names)) +
  coord_fixed(ratio=1) + labs(x = "PC3", y = "PC4") + theme(legend.position="none")
```




```
ggplot(circle,aes(x,y)) + geom_path() +
  geom_text(data=data.frame(res_pca$rotation, .names = row.names(res_pca$rotation)),
            mapping=aes(x = PC5, y = PC6, label = .names, colour = .names)) +
  coord_fixed(ratio=1) + labs(x = "PC5", y = "PC6") + theme(legend.position="none")
```



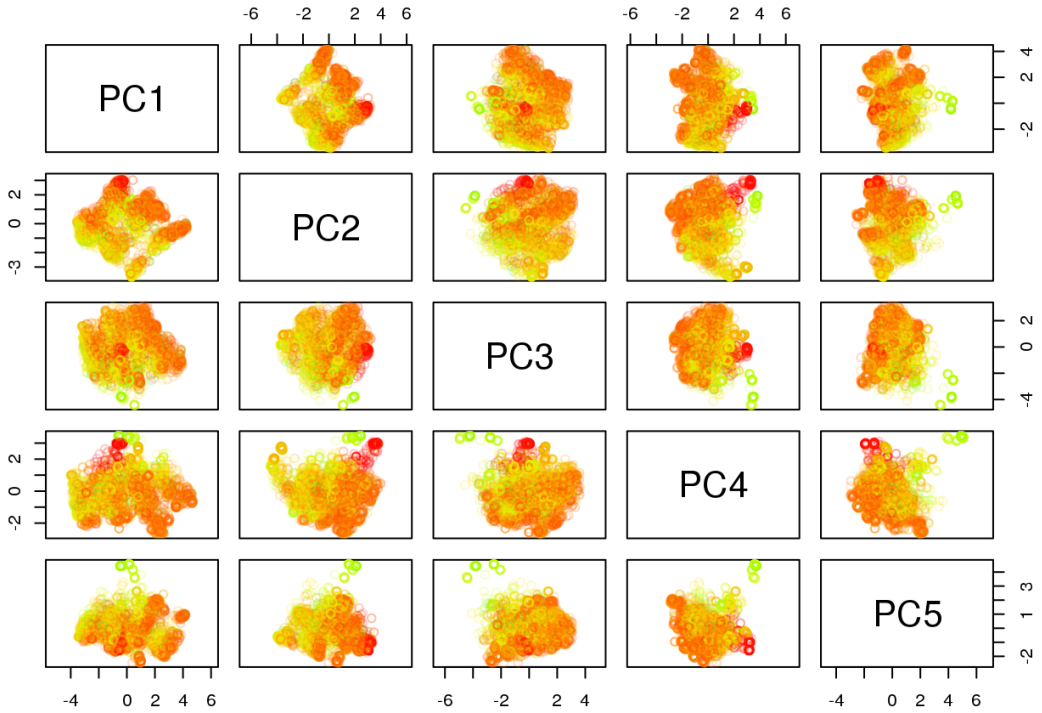


Update: I decreased the radius of the circle from 1.0 to 0.25

We see that for each of the plotted principal components, there are many features contributing a non-zero weight to its linear combination. None of the coefficients is significantly greater than 0.25.

Let's check if the principal components corresponds in any way with the target variable

```
breaks <- 20
pairs(res_pca$x[1:nrow(train), 1:5], col=alpha(rainbow(breaks)[as.numeric(cut(unlist(train_labels), b
reaks = breaks))], 0.2), asp=1)
```



We see some correspondence of the target variable with the principal components, especially for the lower values of y which are between 72 and 82 (the red dots). Some of the datapoints with target values in the range of 111 up to 120 could also be separated quite well (the lightgreen dots). The remaining points form big clusters around zero and cannot really be

distinguished.

Logistic PCA

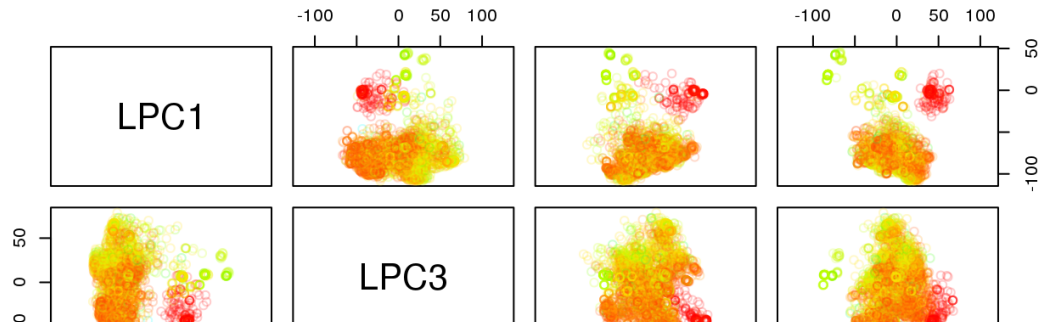
A more appropriate method for dimensionality reduction on binary data is called Logistic PCA. Unlike ordinary PCA, Logistic PCA maximizes the Bernoulli log- likelihood instead of the variance of linear combinations of the variables.

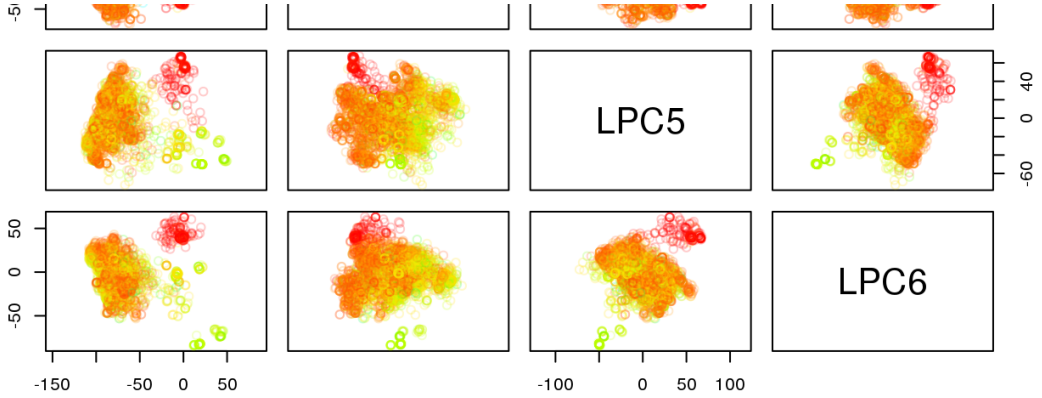
As an interesting fact, I would like to mention that the R-package was developed by Andrew J. Landgraf who recently won the March Machine Learning Mania Competition (http://blog.kaggle.com/2017/05/19/march-machine-learning-mania-1st-place-winners-interview-andrew-landgraf/) (thanks Matt Motoki (https://www.kaggle.com/mmotoki) for pointing this out in the comments). If you want to read more about Logistic PCA, check out his excellent paper "Dimensionality Reduction for Binary Data through the Projection of Natural Parameters" (https://arxiv.org/pdf/1510.06112.pdf) or the Introduction on CRAN (https://cran.r-project.org/web/packages/logisticPCA/vignettes/logisticPCA.html).

```
library(logisticPCA)
library(rARPACK)

# Find the appropriate values for k and m.
# > logsvd_model = logisticSVD(binary_df, k = 20)
# 8418 rows and 368 columns
# Rank 20 solution
# 95.6% of deviance explained
# 397 iterations to converge
#
# > logpca_cv = cv.lpca(binary_df, ks = 20, ms = 1:10)
#      m
#  k       1        2      3        4        5        6        7        8        9       10
#  20 400428 261586.6 185985 143663.3 118547.4 102668.9 92638.51 85579.33 80440.14 76707.54

k <- 20; m <- 12
logpca_model = logisticPCA(binary_df_train, k = k, m = m)
logpca_features <- predict(logpca_model, newdata=binary_df); colnames(logpca_features) <- paste0("LP
C", 1:k)
```

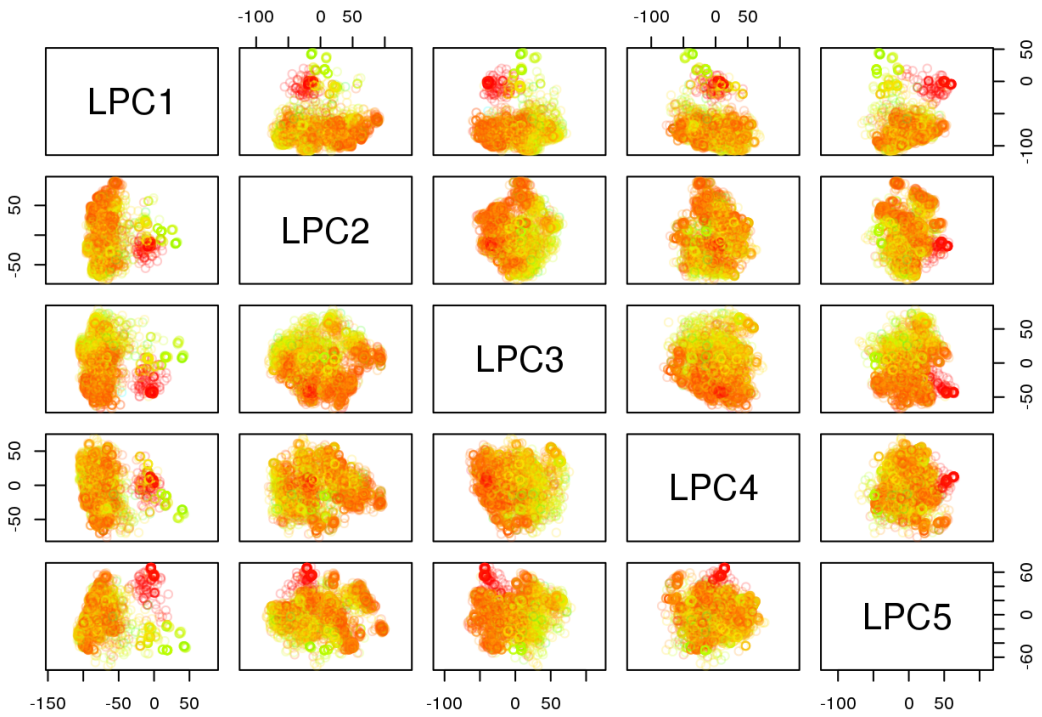
As before in the PCA section, let's check the correlation of the principal components with the target variable

```
breaks <- 20
pairs(logpca_features[1:nrow(train),c("LPC1", "LPC3", "LPC5", "LPC6")], col=alpha(rainbow(breaks)[as.
numeric(cut(unlist(train_labels), breaks = breaks))],0.2), asp=1)
```

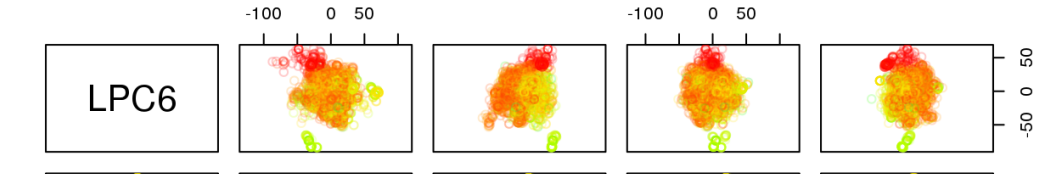


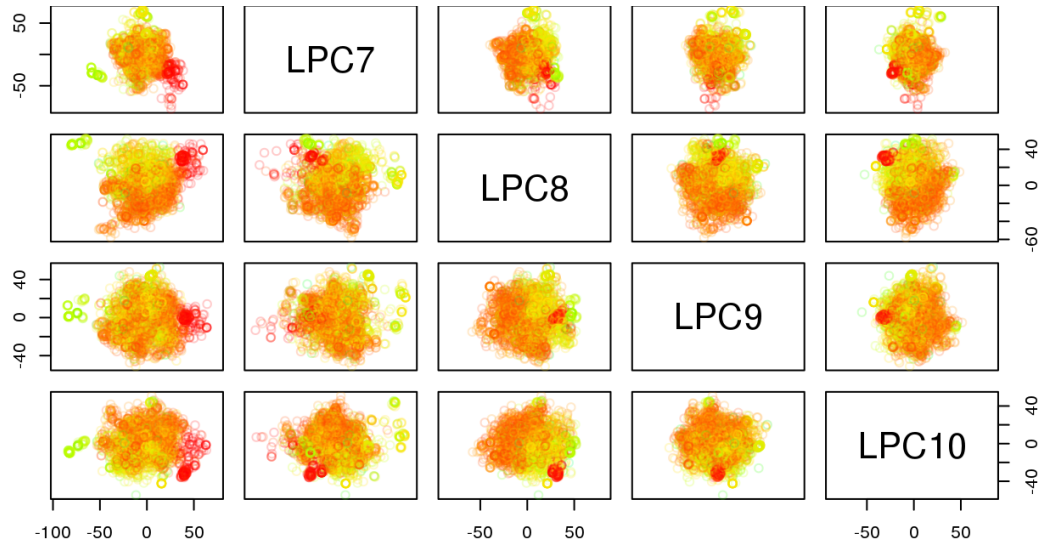



```
pairs(logpca_features[1:nrow(train),1:5], col=alpha(rainbow(breaks)[as.numeric(cut(unlist(train_label
s), breaks = breaks))],0.2), asp=1)
```




```
pairs(logpca_features[1:nrow(train),6:10], col=alpha(rainbow(breaks)[as.numeric(cut(unlist(train_labe
ls), breaks = breaks))],0.2), asp=1)
```





The separation looks much better than for standard PCA. You can see that datapoints with a small value of the target variable (the red dots) were separated very clearly. Even clusters of datapoints with y-values in the range of 111 up to 120 can be seen. The remaining datapoints form a big agglomeration centered around zero, as in the PCA approach.

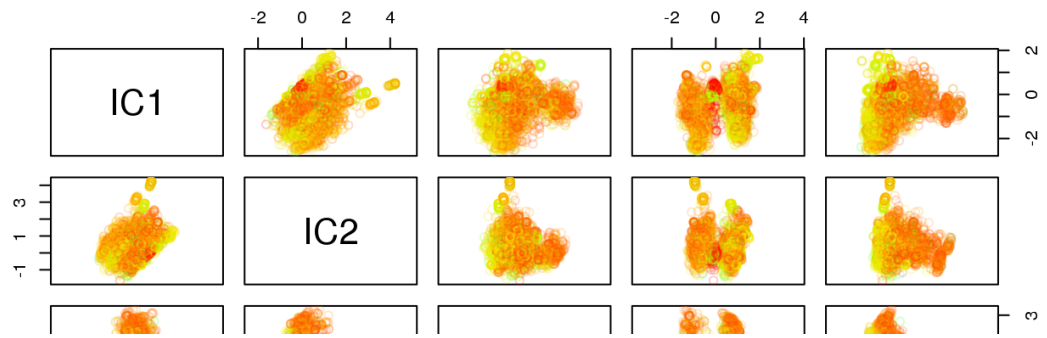
Independent Component Analysis (ICA)

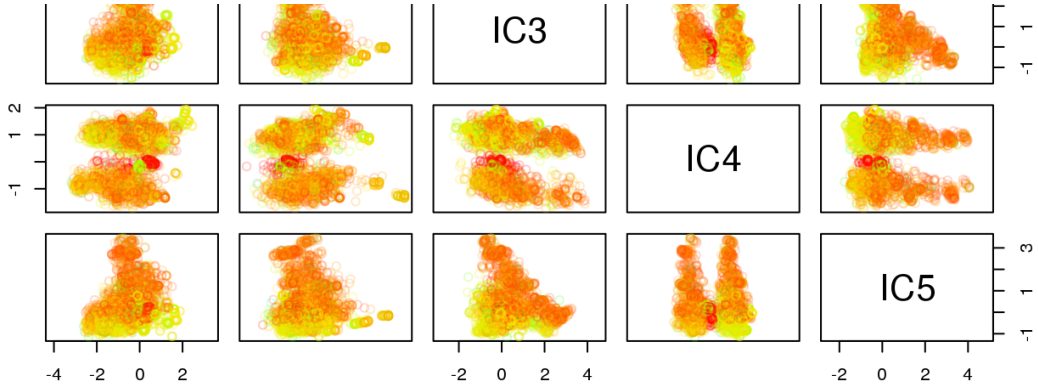
As a next step, I would like to determine some indepent components using ICA. In this approach, we just have to define the number of components we want to retain.

```
library(ica)

n_ica_features <- 12
ica_model <- icafast(binary_df, nc = n_ica_features, center=FALSE)
S <- as.matrix(binary_df)
Y <- tcrossprod(S, ica_model$Q)
ica_features <- Y %*% ica_model$R; colnames(ica_features) <- paste0("IC", 1:n_ica_features)
```

And again, let's check the correlation with the target variable

```
breaks <- 20
pairs(ica_features[1:nrow(train),1:5], col=alpha(rainbow(breaks)[as.numeric(cut(unlist(train_labels),
 breaks = breaks))],0.2), asp=1)
```





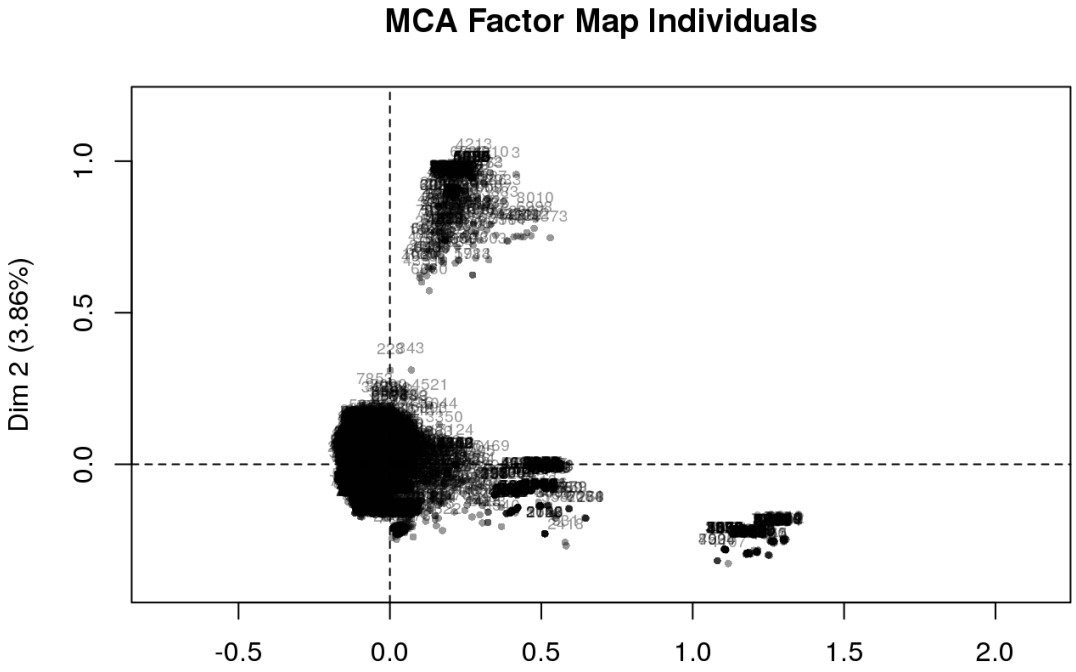
NEW: Multiple Correspondence Analysis (MCA)

Another technique which seems to be quite promising is called Multiple Correspondence Analysis which is an extension of Correspondence Analysis. It is used to analyze structures of several categorical variables and can be seen as a generalization of PCA.

```
library(FactoMineR)

n_comp <- 12
binary_factor_df <- data.frame(lapply(binary_df, as.factor))
binary_factor_df_train <- data.frame(lapply(binary_df_train, as.factor))
res_mca = MCA(binary_factor_df, ncp=n_comp, graph = FALSE)
```

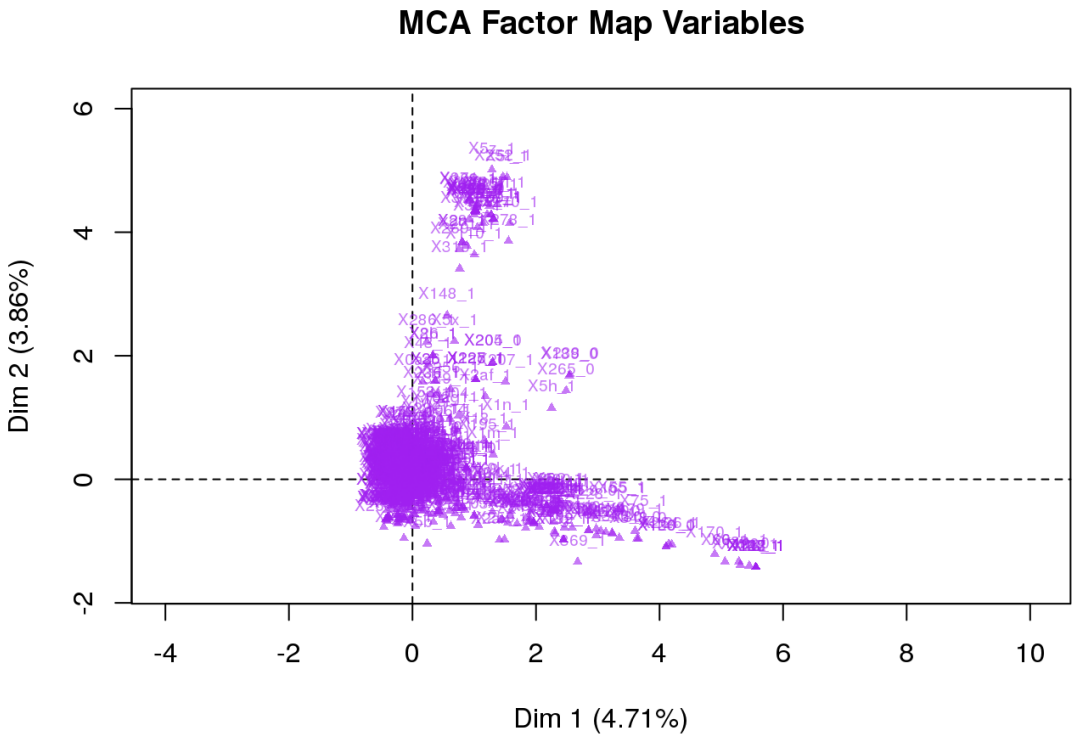
Let's take a look at the contributions of the observations and variables to the first two components:

```
plot.MCA(res_mca, invisible=c("var","quali.sup"), cex=0.6, col.ind=alpha("black", 0.4), label="ind",
title="MCA Factor Map Individuals")
```

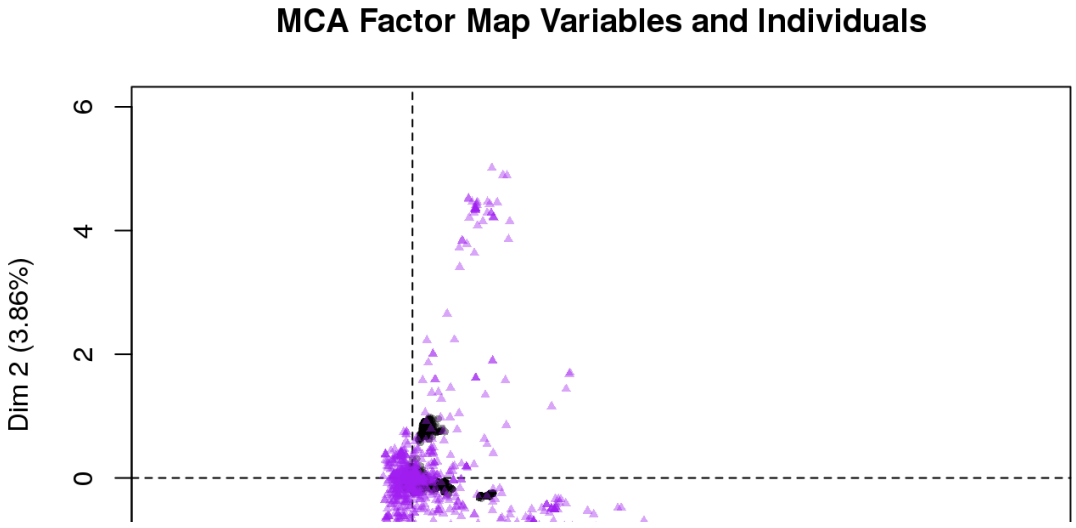


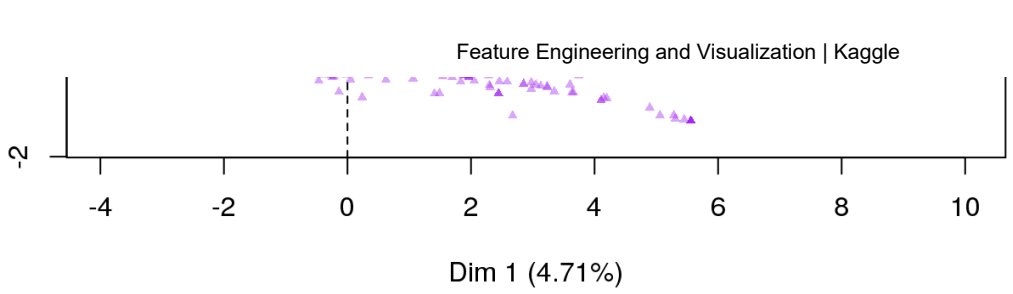
Dim 1 (4.71%)

```
plot.MCA(res_mca, invisible=c("ind","quali.sup"), cex=0.6, col.var=alpha("purple", 0.6), label="var",
 title="MCA Factor Map Variables")
```




```
plot.MCA(res_mca, invisible=c("quali.sup"), cex=0.6, col.var=alpha("purple", 0.4), col.ind=alpha("bla
ck", 0.4), label="none", title="MCA Factor Map Variables and Individuals")
```

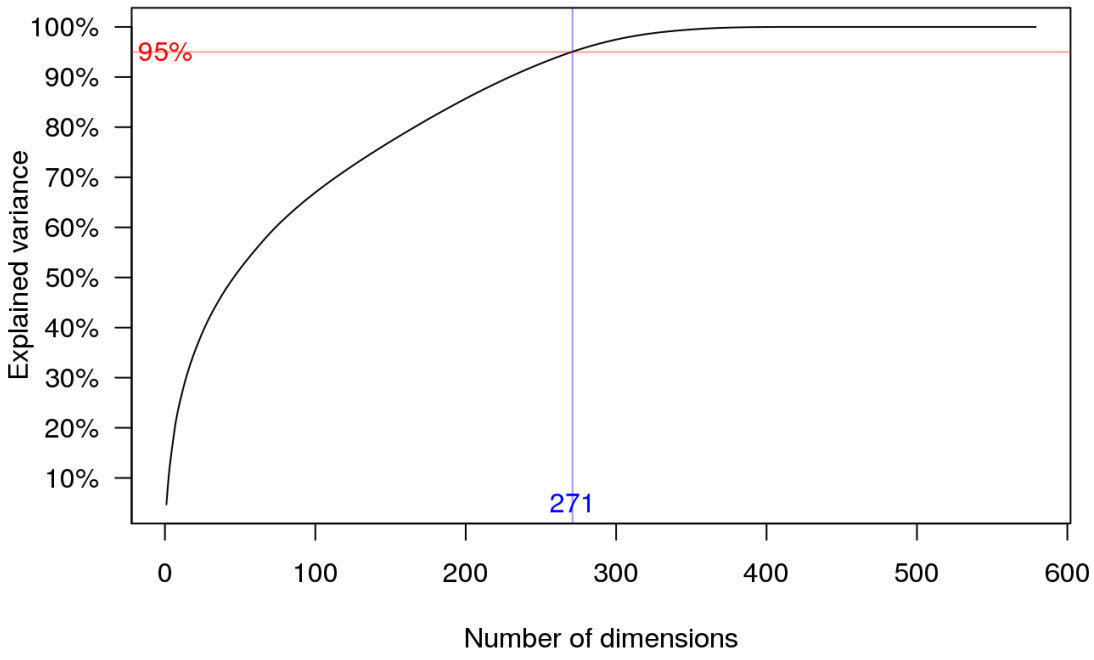




In the scatterplots above, we see the weights each observation (the individuals) and variable contributes to the first two components (dimensions) calculated by MCA. The first two dimensions capture 4.71 + 3.86 % of the variance of the target variable. In the scatterplot of the individuals we can see that there are three groups that are separated quite clearly. The big group around zero contains variables which don't contribute that much to the first two dimensions. The individuals of the upper group contribute a significant amount to the second dimension while the individuals of the lower group contribute a significant amount to the first dimension. A similar pattern can be seen in the plot of the variables, but other than in the previous plot we can't see some well separated clusters. Rather than that, the overall distribution looks more like an L-shape.

Let's see how many of the dimensions we have to retain to cover 95% of the variance of the target variable:

```
plot(res_mca$eig$`cumulative percentage of variance`, type="l", ylab="Explained variance", xlab="Numb
er of dimensions", yaxt="n")
axis(2, at=seq(0,100,by=10), labels = paste0(seq(0,100,by=10), "%"), las=1)
min_variance <- 95
abline(h=min_variance, col=alpha("red", 0.4)); text(0, min_variance, paste0(min_variance, "%"), col=
"red")
min_dim <- min(which(res_mca$eig$`cumulative percentage of variance`>min_variance))
abline(v=min_dim, col=alpha("blue", 0.4)); text(min_dim, 5, min_dim, col="blue")
```



So we have to retain approximately 271 dimensions to capture 95 % of the variance in y.

Let's extract the MCA features and take a look at the correlation with the target variable

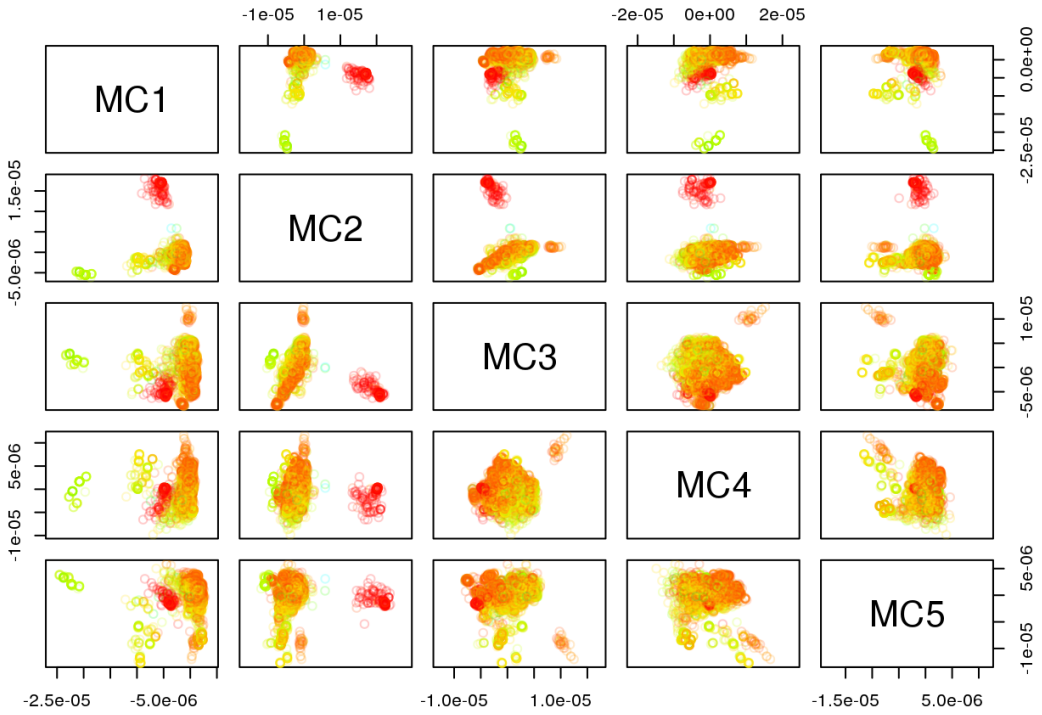
```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

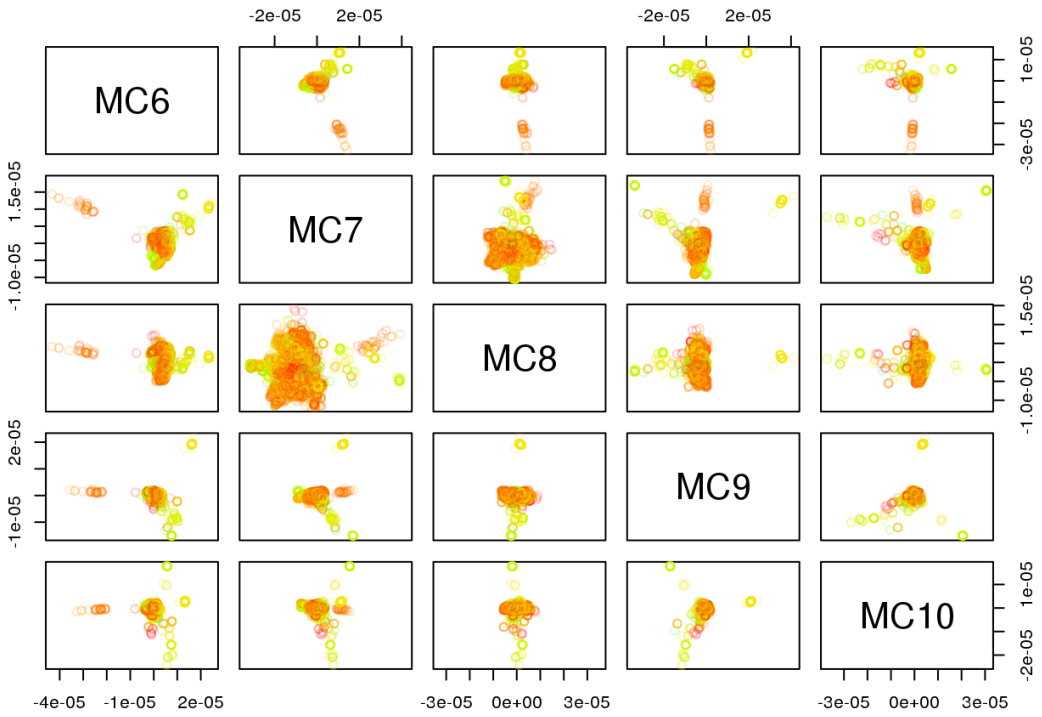
```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
res_mca = MASS::mca(binary_factor_df, nf=min_dim)
mca_features <- predict(res_mca, newdata = binary_factor_df); colnames(mca_features) <- paste0("MC",
1:min_dim)
```

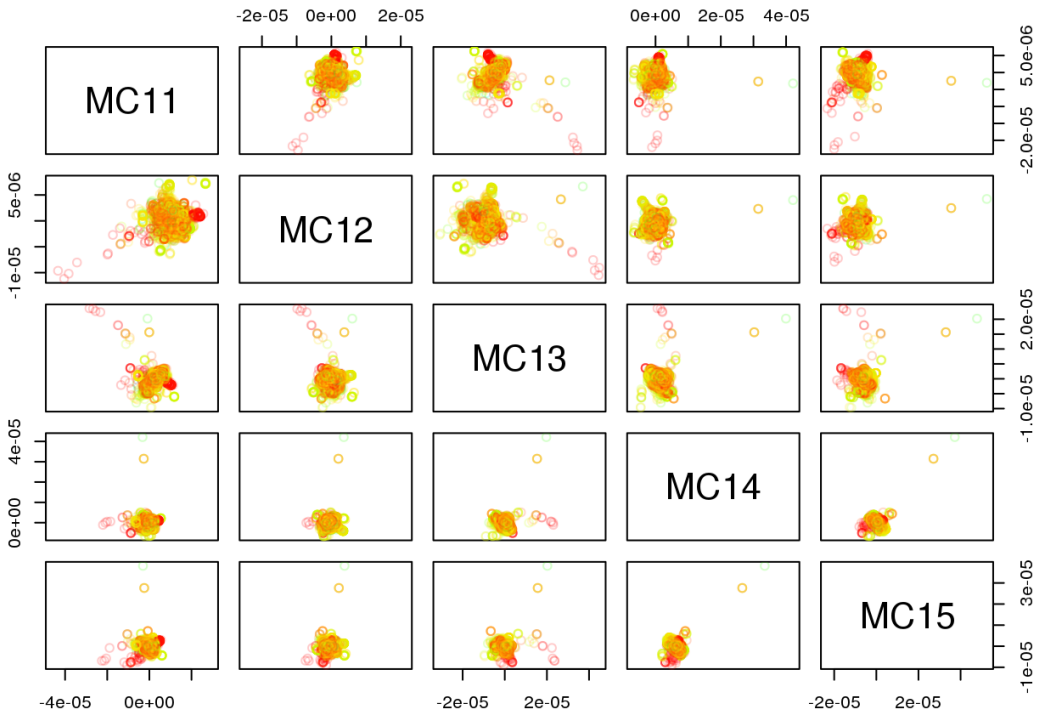
```
breaks <- 20
pairs(mca_features[1:nrow(train),1:5], col=alpha(rainbow(breaks)[as.numeric(cut(unlist(train_labels),
 breaks = breaks))],0.2), asp=1)
```




```
pairs(mca_features[1:nrow(train),6:10], col=alpha(rainbow(breaks)[as.numeric(cut(unlist(train_labels)
, breaks = breaks))],0.2), asp=1)
```




```
pairs(mca_features[1:nrow(train),11:15], col=alpha(rainbow(breaks)[as.numeric(cut(unlist(train_labels
), breaks = breaks))],0.2), asp=1)
```



As in the logistic PCA approach, the datapoints with low and mid range of y were separated quite well. Additionally, we see some more clusters of datapoints with a mid-range value of y (the orange dots).

Backtransformation

As a final step, let's combine all the results with id and target columns to get our enhanced training and test sets.

```
df_all <- cbind(df_all, clusters, pca_features, logpca_features, ica_features, mca_features)

ntrain <- nrow(train)
train <- df_all[1:ntrain, ]
test <- df_all[(ntrain+1):nrow(df_all), ]

train <- data.frame(ID = train_id, train, y = train_labels)
test <- data.frame(ID = test_id, test)
```

Did you find this Kernel useful?
Show your appreciation with an upvote

106

Comments (32)

All Comments | Sort by Hotness

Click here to enter a comment...

Matt Motoki • Posted on Version 43 • 9 months ago • Options • Reply

3

Thanks for posting this report; it's very good. The logistic PCA in particular stood out. I just thought I'd share an interesting side note about the author of the package: he recently won the March Machine Learning Mania competition. So congrats to him and thank you again for posting this.

Marcel Spitzer • Posted on Version 43 • 9 months ago • Options • Reply

1

Thank you very much for your feedback and the background information, Matt!

Heads or Tails • Posted on Latest Version • 8 months ago • Options • Reply

1

Congrats on your gold! You've definitely earned it with this great kernel. Keep up the good work!

Marcel Spitzer • Posted on Latest Version • 8 months ago • Options • Reply

0

Thank you very much :) I will definitely continue my work. It's a great experience.

Borja SP • Posted on Latest Version • 8 months ago • Options • Reply

1

amazing notebook! Very helpful to learn to apply PCA analysis. I have some questions: The new data I use to feed my model is composed now by the original dataset, the one-hot-encoded binary data generated, and the new PCA features. But models are not improving. As I understand, if PCA analysis serves to generate new set of features, not as large as the original one; why are you including all the features together?

Marcel Spitzer • Posted on Latest Version • 8 months ago • Options • Reply

2

Thank you very much for your feedback, Borja SP!

I made similar experience: the applied feature engineering techniques separate the datapoints quite well but the predictions - especially for xgboost models - don't improve significantly. There are several discussions about the predictive capability of the variables. I guess a huge problem is that it is pretty hard to explain the upper outliers based on the given data (see here or here for example).

The main reason for me to apply PCA and similar techniques was (1.) to transform the variables (think about it like a change in the perspective you are looking on the datapoints) and (2.) to reduce noise - both to better be able to explain the variance in y. The feature engineering techniques are unsupervised techniques, i.e. they don't take into account the values of the target variable. So we have no guarantee that there is a relationship between the target variable and the principal components. I retained all of the features because I didn't want to discard any viewpoint (according to (1.)) which is potentially important. Further, I focused on tree-based methods, which are able to implicitly perform feature selection. If the engineered variables are better predictors than the original ones, the tree construction algorithm will pick them.

moefasa • Posted on Version 44 • 9 months ago • Options • Reply

1

Hey Marcel, great kernel - I even bookmarked it for future projects :)

That said, for feature engineering, you may want to use the binary columns from your training data only. Otherwise you'll overfit. I tried it as you did and CV confirmed it. So I tried using just the training data, and then fitting an xgboost model to predict PCA1, PCA2, IC1, etc. to get the test features. I got my test rmse down to a comfortable level, and when I fit the whole model it did a lot better.

Edit: Actually, it turns out you can use the fit from the training data to transform the test data instead of fitting another model.

Marcel Spitzer • Posted on Version 44 • 9 months ago • Options • Reply

0

Hey @moefasa, thank you very much for your feedback! I'm glad to be in your bookmarks now :)

I'm somewhat confused about your point with overfitting. Since the target variable was not used in the transformation steps and we didn't build any regression model, how can the results of feature engineering be overfitted? Is it possible, that we get better predictive performance when reducing the amount of data because there is simply a lot of noise in it?

moefasa • Posted on Version 44 • 9 months ago • Options • Reply

2

I believe the response here explains it better than I can:
https://stats.stackexchange.com/questions/55718/pca-and-the-train-test-split

You can also see how it was done in this kernel. PCA was fit using the training data then used to transform the test data: https://www.kaggle.com/frednavruzov/baselines-to-start-with-lb-0-56

I've been taught that in order to optimize for generalization error, you're supposed to pretend you never had the test data to begin with. Although I've seen 'hacks' being done to get better scores (for example, adjusting priors).

Marcel Spitzer • Posted on Version 44 • 9 months ago • Options • Reply

0

Ok, this is a really interesting point which I wasn't aware of. Thank you very much for pointing me to the problem. As you said, the right way is to absolutely exclude the testset from the training phase in order to get generalizable models. To include it in PCA is somewhat cheaty.

Heads or Tails • Posted on Version 44 • 9 months ago • Options • Reply

1

I really like your plots and exploration methodology!

Is it possible to add a bit more text on your interpretation of the various plot, especially the hierarchical clustering, and what they tell us about the relations and structure within the data?

I hadn't noticed how similar the factor levels of X5 are between the train and test data. In fact, if you join them and sort by ID then the levels fit perfectly. What do you think that tells us?

Marcel Spitzer • Posted on Version 44 • 9 months ago • Options • Reply

1

Thank you very much for your feedback and the questions :-) Sure, I will add some interpretations soon.

My main intention to cluster the instances was to get 4-5 groups which relates to the target variable in somewhat different ways and to model each of the groups separately (as discussed in the other kernel). Another approach which I would like to try out is, to predict y based on cluster indices only. Maybe we are able to reduce some of the noise this way and get more stable predictions.

Until now, I wasn't able to get anything useful out of the pattern in X5. Maybe @PC Jimmmy has an explanation for it. It could indicate a temporal ordering of the observations.

Marcel Spitzer • Posted on Version 44 • 9 months ago • Options • Reply 0

since the compile time of this kernel is really long, I just added a new one with a more detailed description and interpretation of the hierarchical clustering. You can find it here. I hope that I didn't leave any question unanswered. Otherwise, please let me know ;-)

Jonathan Mallia • Posted on Version 35 • 9 months ago • Options • Reply 1

Hi Marcel, i got a question about the ICA section in this line: S <- scale(as.matrix(binary_df, center=FALSE, scale = FALSE)) First set scale to False the Scale again on the outer function? Is there a reason for that? Is scaling particularly required for ICA or it can be skipped?

Awesome stuff. Cheers

Marcel Spitzer • Posted on Version 35 • 9 months ago • Options • Reply 1

Hi Jonathan, thank you for your attention! You can center and/or scale the variables before doing ICA. If you do so, you have to apply the same transformations before calculating S. I experimented with this option and left the call of `scale` in the code, which is redundant in the current version (since I left the variables unchanged). I will remove it in the next version.

Bernardo Lares • Posted on Version 27 • 9 months ago • Options • Reply 0

Great! Loved this notebook. Would you be so kind to explain a bit the one-hot encoded features visualization, the Hierarchical Clustering on Binary Features plot, the PCAs circled ggplots, and why max_k=50? Like, what conclusions did you read there? Sorry for my newbieness but I'm really interested in understanding these :)

Marcel Spitzer • Posted on Version 30 • 9 months ago • Options • Reply 1

Hi Bernardo, thank you very much! I added some explanations and enhancements to the images. Hope that they are helpful. I will update this script regularly to answer all your questions.

Edit: I chose max_k to be 50 by looking at the cuts in the dendrogram. Of course, you can further increase it.

Jonathan Mallia • Posted on Version 30 • 9 months ago • Options • Reply 0

Hi!

Thank you for this very insightful.. I think there is a problem in this part as i couldn't get it to work..

barplot(sort(importance_pca[2, importance_pca[2,] > 0.005], decreasing = FALSE), horiz = TRUE, xlim=c(0,0.14), las=1, cex.names=0.6, main="Explained Variance by Principal Component", xlab="Proportion of explained variance")

Marcel Spitzer • Posted on Version 31 • 9 months ago • Options • Reply 0

Thank you, Jonathan :) Would you please post the error message?

Jonathan Mallia • Posted on Version 32 • 9 months ago • Options • Reply 0

My bad Marcel everything is fine :) i had a problem in my own script! Cheers

Adrianna Napiórk... • Posted on Version 35 • 9 months ago • Options • Reply 0

Hi :) It is a very nice report, thank you. Could you recommend some readings about PCA?

Marcel Spitzer • Posted on Version 35 • 9 months ago • Options • Reply 1

Thank you, Adrianna! There are many good intros to PCA out there. I recommend the explanation of Trevor Hastie and Rob Tibshirani in their excellent book "Introduction to Statistical Learning", p. 374ff. Additionally, I found this blogpost very helpful. If you want to read something about Logistic PCA, here is the paper.

Radu Stoicescu • Posted on Version 43 • 9 months ago • Options • Reply 2

An easy way to gain the intuition about what PCA does: http://setosa.io/ev/principal-component-analysis/

muzifft • Posted on Version 43 • 9 months ago • Options • Reply 0

Great work. Thank you!

muzifft • Posted on Version 43 • 9 months ago • Options • Reply 0

Great work. Thank you!

Chippy • Posted on Version 44 • 9 months ago • Options • Reply

0

Great notebook, reads really well and has some very interesting visuals. Thanks for sharing.

Marcel Spitzer • Posted on Version 44 • 9 months ago • Options • Reply

0

Thank you very much for your feedback, @Chippy! It's really motivating :)

Sheng Guo • Posted on Version 56 • 8 months ago • Options • Reply

0

Nice work! Help a lot, thank you.

Kai Wang • Posted on Latest Version • 8 months ago • Options • Reply

0

Great work. Thank you!

cmcmahon • Posted on Latest Version • 7 months ago • Options • Reply

0

Hi Marcel. Really found the notebook useful - Thank you.

I am having some performance issues running the logistic PCA. Particularly in comparison to the normal PCA. Does logistic PCA normally take much longer to run?

Marcel Spitzer • Posted on Latest Version • 7 months ago • Options • Reply

0

Thank you very much, cmcmahon!

Yes, logistic PCA takes more time for computation. The algorithm is more complex than for PCA.

Likelihood • Posted on Latest Version • 4 months ago • Options • Reply

0

Impressive ! Very useful!

Similar Kernels

Visualizing PCA With Leaf Dataset

Visualizing Classifier Boundaries Using Kernel PCA

Plant: PCA & T-SNE (W/ Image Scatter Plot)

Image Compression With PCA From Scratch (+Math)

Fruit Quiz - Mix Of PCA + Random Forest + SVM