Predicting Phishing with Cluster Analysis

Ryan Heslin

June 02, 2021

Today I use clustering to develop a model to classify phishing websites. Phishing sites masquerade as legitimate websites and try to trick visitors into revealing personal information such as credit card numbers.

The dataset, and a data dictionary, may be found at http://archive.ics.uci.edu/ml/datasets/Phishing+Websites#. It consists of several dozen features related to a sample of 5000 legitimate and 5000 phishing URLS. Most features are 2- or 3-level categories.

```
raw <- read_csv("../data/Phishing_Legitimate_full.csv")
```

```
-- Column specification --------------------------------------------------------
cols(
  .default = col_double()
)
i Use `spec()` for the full column specifications.
```

```
raw %>%
    select(1:6, last_col()) %>%
    head()
```

id	NumDots	SubdomainLevel	PathLevel	UrlLength	NumDash	CLASS_LABEL
1	3	1	5	72	0	1
2	3	1	3	144	0	1
3	3	1	2	58	0	1
4	3	1	6	79	1	1
5	3	0	4	46	0	1
6	3	1	1	42	1	1

```
sapply(raw[, -1], summary) %>%
    t() %>%
    as.data.frame() %>%
    rownames_to_column(var = "Variable")
```

Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
NumDots	1	2.000000	2.0000000	2.4451000	3.0000000	21
SubdomainLevel	0	0.000000	1.0000000	0.5868000	1.0000000	14
PathLevel	0	2.000000	3.0000000	3.3003000	4.0000000	18
UrlLength	12	48.000000	62.0000000	70.2641000	84.0000000	253
NumDash	0	0.000000	0.0000000	1.8180000	2.0000000	55

Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
NumDashInHostname	0	0.000000	0.0000000	0.1389000	0.0000000	9
AtSymbol	0	0.000000	0.0000000	0.0003000	0.0000000	1
TildeSymbol	0	0.000000	0.0000000	0.0131000	0.0000000	1
NumUnderscore	0	0.000000	0.0000000	0.3232000	0.0000000	18
NumPercent	0	0.000000	0.0000000	0.0738000	0.0000000	19
NumQueryComponents	0	0.000000	0.0000000	0.4586000	0.0000000	23
NumAmpersand	0	0.000000	0.0000000	0.2772000	0.0000000	22
NumHash	0	0.000000	0.0000000	0.0023000	0.0000000	1
NumNumericChars	0	0.000000	2.0000000	5.8103000	8.0000000	111
NoHttps	0	1.000000	1.0000000	0.9888000	1.0000000	1
RandomString	0	0.000000	1.0000000	0.5252000	1.0000000	1
IpAddress	0	0.000000	0.0000000	0.0172000	0.0000000	1
DomainInSubdomains	0	0.000000	0.0000000	0.0222000	0.0000000	1
DomainInPaths	0	0.000000	0.0000000	0.4289000	1.0000000	1
HttpsInHostname	0	0.000000	0.0000000	0.0000000	0.0000000	0
HostnameLength	4	14.000000	18.0000000	18.8243000	22.0000000	137
PathLength	0	18.000000	30.0000000	35.5649000	48.0000000	161
QueryLength	0	0.000000	0.0000000	8.6065000	0.0000000	188
DoubleSlashInPath	0	0.000000	0.0000000	0.0009000	0.0000000	1
NumSensitiveWords	0	0.000000	0.0000000	0.1093000	0.0000000	3
EmbeddedBrandName	0	0.000000	0.0000000	0.0571000	0.0000000	1
PctExtHyperlinks	0	0.000000	0.0714286	0.2413342	0.3158744	1
PctExtResourceUrls	0	0.030303	0.2475113	0.3929323	0.7857143	1
ExtFavicon	0	0.000000	0.0000000	0.1672000	0.0000000	1
InsecureForms	0	1.000000	1.0000000	0.8440000	1.0000000	1
RelativeFormAction	0	0.000000	0.0000000	0.2487000	0.0000000	1
ExtFormAction	0	0.000000	0.0000000	0.1018000	0.0000000	1
AbnormalFormAction	0	0.000000	0.0000000	0.0576000	0.0000000	1
PctNullSelfRedirectHyperlinks	0	0.000000	0.0000000	0.1361355	0.0476190	1
FrequentDomainNameMismatch	0	0.000000	0.0000000	0.2153000	0.0000000	1
FakeLinkInStatusBar	0	0.000000	0.0000000	0.0055000	0.0000000	1
RightClickDisabled	0	0.000000	0.0000000	0.0140000	0.0000000	1
PopUpWindow	0	0.000000	0.0000000	0.0049000	0.0000000	1
SubmitInfoToEmail	0	0.000000	0.0000000	0.1288000	0.0000000	1
IframeOrFrame	0	0.000000	0.0000000	0.3396000	1.0000000	1
MissingTitle	0	0.000000	0.0000000	0.0322000	0.0000000	1
ImagesOnlyInForm	0	0.000000	0.0000000	0.0304000	0.0000000	1
SubdomainLevelRT	-1	1.000000	1.0000000	0.9566000	1.0000000	1
UrlLengthRT	-1	-1.000000	0.0000000	0.0202000	1.0000000	1
PctExtResourceUrlsRT	-1	-1.000000	1.0000000	0.3533000	1.0000000	1
AbnormalExtFormActionR	-1	1.000000	1.0000000	0.7932000	1.0000000	1
ExtMetaScriptLinkRT	-1	0.000000	0.0000000	0.1734000	1.0000000	1
PctExtNullSelfRedirectHyperlinksRT	-1	-1.000000	1.0000000	0.3141000	1.0000000	1
CLASS_LABEL	0	0.000000	0.5000000	0.5000000	1.0000000	1

All classes seem well dispersed. I convert the indicator variables to factors and drop one that's all zeroes. Then I arrange the factors by difference in proportions. Some features are exceptionally rare.

```
is_one_hot <- function(x) {
    identical(sort(unique(x)), c(0, 1))
}
# drop useless column
cleaned <- raw %>%
    select(where(~n_distinct(.x) > 1)) %>%
    mutate(across(where(~is_one_hot(.x)), ~factor(.x, levels = c(0, 1), labels = c("no", "yes"))), acros
        c(-1, 0, 1))), ~factor(.x, labels = c("phish", "suspicious", "legitimate"))))

facs <- cleaned %>%
    select(where(~is.factor(.x) && nlevels(.x) == 2)) %>%
    map(~fct_count(.x, prop = TRUE)) %>%
    bind_rows(.id = "Variable")

facs %>%
    group_by(Variable) %>%
    summarize(Range = abs(diff(p))) %>%
    arrange(-Range)
```

Variable	Range
AtSymbol	0.9994
DoubleSlashInPath	0.9982
NumHash	0.9954
PopUpWindow	0.9902
FakeLinkInStatusBar	0.9890
NoHttps	0.9776
TildeSymbol	0.9738
RightClickDisabled	0.9720
IpAddress	0.9656
DomainInSubdomains	0.9556
ImagesOnlyInForm	0.9392
MissingTitle	0.9356
EmbeddedBrandName	0.8858
AbnormalFormAction	0.8848
ExtFormAction	0.7964
SubmitInfoToEmail	0.7424
InsecureForms	0.6880
ExtFavicon	0.6656
FrequentDomainNameMismatch	0.5694
RelativeFormAction	0.5026
IframeOrFrame	0.3208
DomainInPaths	0.1422
RandomString	0.0504
CLASS_LABEL	0.0000

A few components capture almost all the variation, though treating dummy variables as quantitative for the purposes of PCA is a crude approach.

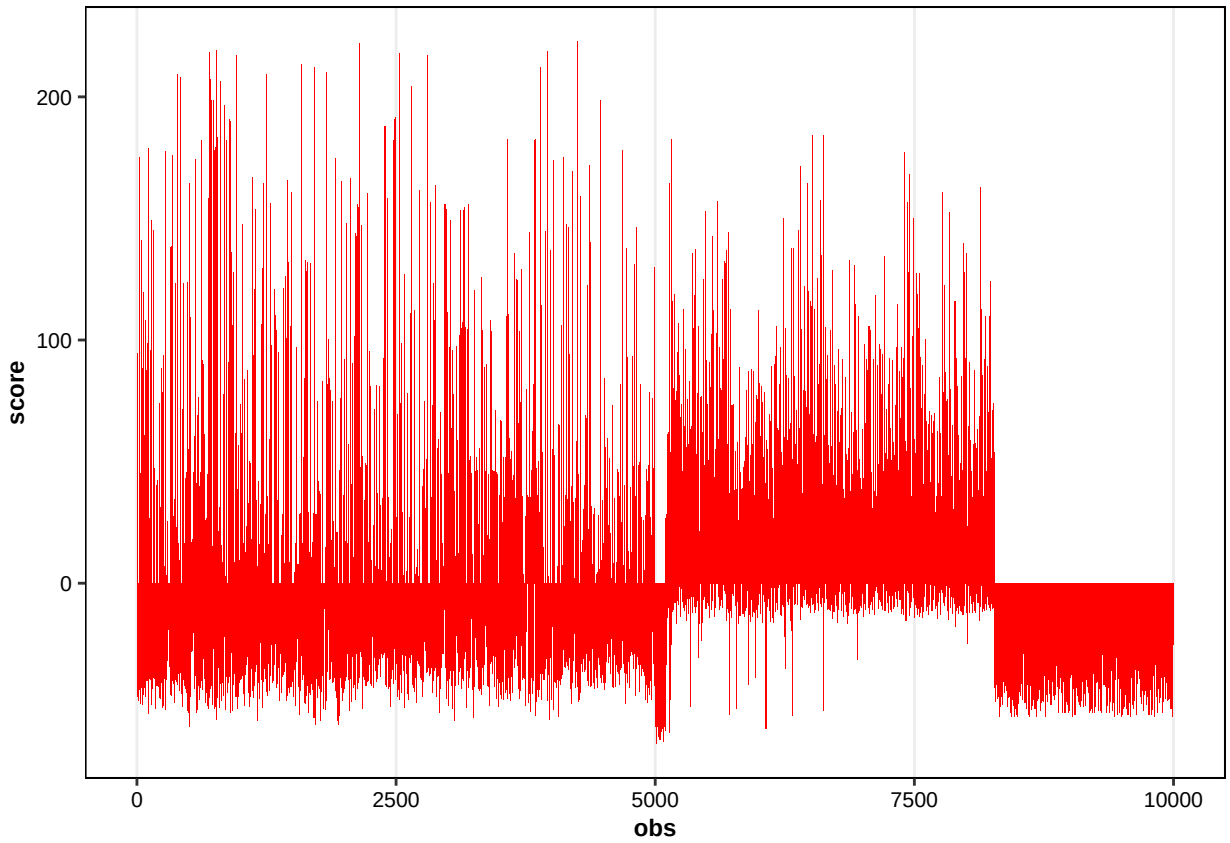
PC 1 scores fall after observation 8000. That suggests the ordering of observations is nonrandom, but sampling test and training sets will fix that.

Only a few components are necessary.

```
pca <- raw %>%
    select(-c("id", "HttpsInHostname", "CLASS_LABEL")) %>%
    princomp(scale. = TRUE)

summary(pca)
```

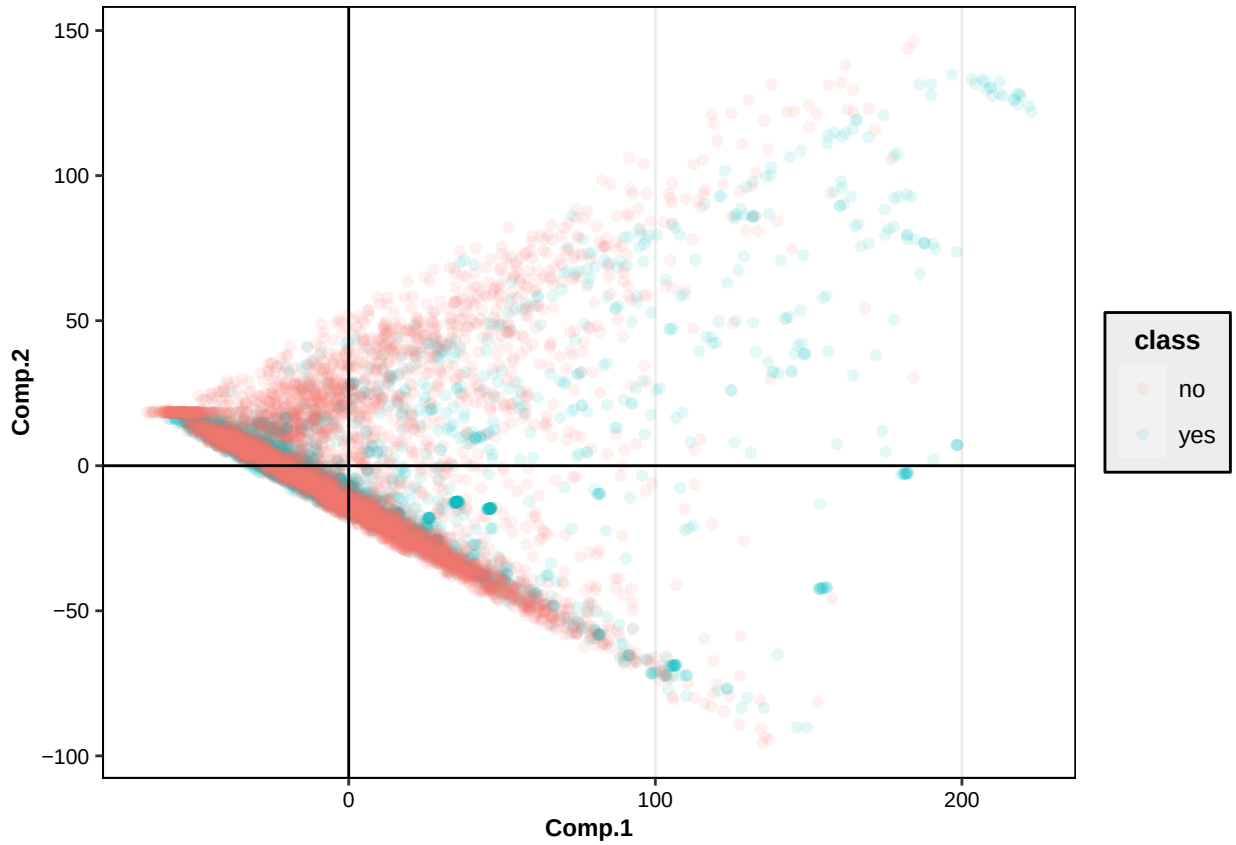
```
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4     Comp.5       Comp.6
Standard deviation     40.8228357 25.8530682 9.35711417 7.30471930 2.58330554 1.3185323158
Proportion of Variance  0.6683754  0.2680639 0.03511543 0.02140036 0.00267649 0.0006972615
Cumulative Proportion   0.6683754  0.9364393 0.97155468 0.99295504 0.99563153 0.9963287937
                             Comp.7       Comp.8       Comp.9     Comp.10      Comp.11      Comp.12
Standard deviation     1.2590231377 1.1577282348 0.9879394443 0.897648825 0.8164001076 0.6301016021
Proportion of Variance 0.0006357429 0.0005375605 0.0003914486 0.000323167 0.0002673131 0.0001592338
Cumulative Proportion  0.9969645366 0.9975020971 0.9978935457 0.998216713 0.9984840258 0.9986432596
                            Comp.13      Comp.14      Comp.15       Comp.16       Comp.17
Standard deviation     0.6099185180 0.5891884888 0.5530388461 0.48254073470 0.46301465342
Proportion of Variance 0.0001491962 0.0001392267 0.0001226663 0.00009338606 0.00008598121
Cumulative Proportion  0.9987924558 0.9989316825 0.9990543489 0.99914773493 0.99923371615
                            Comp.18       Comp.19       Comp.20       Comp.21       Comp.22
Standard deviation     0.45381694424 0.43901846596 0.42926575610 0.41257877550 0.40631000024
Proportion of Variance 0.00008259914 0.00007730003 0.00007390377 0.00006826968 0.00006621084
Cumulative Proportion  0.99931631529 0.99939361532 0.99946751909 0.99953578877 0.99960199962
                            Comp.23       Comp.24       Comp.25       Comp.26       Comp.27
Standard deviation     0.35272963836 0.35166653835 0.33800124770 0.31990405820 0.31089126648
Proportion of Variance 0.00004989971 0.00004959937 0.00004581954 0.00004104437 0.00003876423
Cumulative Proportion  0.99965189932 0.99970149870 0.99974731824 0.99978836261 0.99982712684
                            Comp.28       Comp.29       Comp.30       Comp.31       Comp.32
Standard deviation     0.29822234898 0.25531050044 0.24532461306 0.20692808211 0.17716595402
Proportion of Variance 0.00003566929 0.00002614276 0.00002413773 0.00001717327 0.00001258852
Cumulative Proportion  0.99986279613 0.99988893889 0.99991307661 0.99993024988 0.99994283840
                            Comp.33       Comp.34        Comp.35        Comp.36        Comp.37
Standard deviation     0.1684776810 0.16366028478 0.142520914704 0.118714197126 0.105082489585
Proportion of Variance 0.0000113841 0.00001074238 0.000008146505 0.000005652225 0.000004428685
Cumulative Proportion  0.9999542225 0.99996496489 0.999973111396 0.999978763621 0.999983192306
                            Comp.38        Comp.39        Comp.40        Comp.41        Comp.42
Standard deviation     0.095027885907 0.092367673512 0.083054540733 0.073968772445 0.068819825702
Proportion of Variance 0.000003621731 0.000003421796 0.000002766565 0.000002194375 0.000001899508
Cumulative Proportion  0.999986814037 0.999990235833 0.999993002398 0.999995196774 0.999997096282
                            Comp.43        Comp.44        Comp.45         Comp.46         Comp.47
Standard deviation     0.054923970933 0.052835164773 0.029803045436 0.0172785608379 0.0156544370312
Proportion of Variance 0.000001209868 0.000001119593 0.000000356234 0.0000001197372 0.0000000982854
Cumulative Proportion  0.999998306150 0.999999425743 0.999999781977 0.9999999017146 1.0000000000000
```

```
tibble(obs = 1:nrow(raw), score = pca$scores[, 1]) %>%
    ggplot(aes(x = obs, y = score)) + geom_col(fill = "red")
```

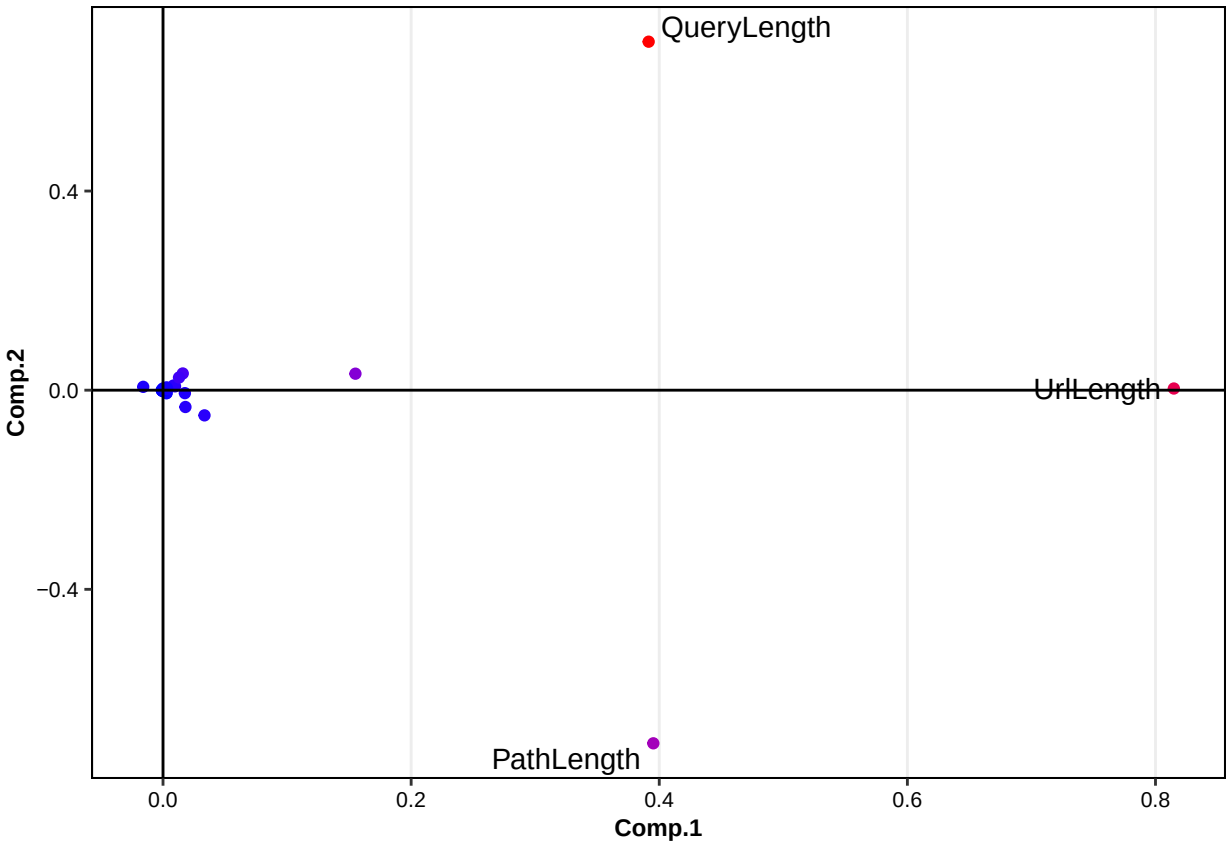

200
100
0
score
0
2500
5000
7500
10000
obs

Only a few loadings are high for both of the first two components.

```
as_tibble(pca$scores[, 1:2]) %>%
    mutate(class = cleaned$CLASS_LABEL) %>%
    ggplot(aes(x = Comp.1, y = Comp.2, color = class)) + geom_jitter(alpha = 0.1) + geom_vline(xinterce
    geom_hline(yintercept = 0)
```

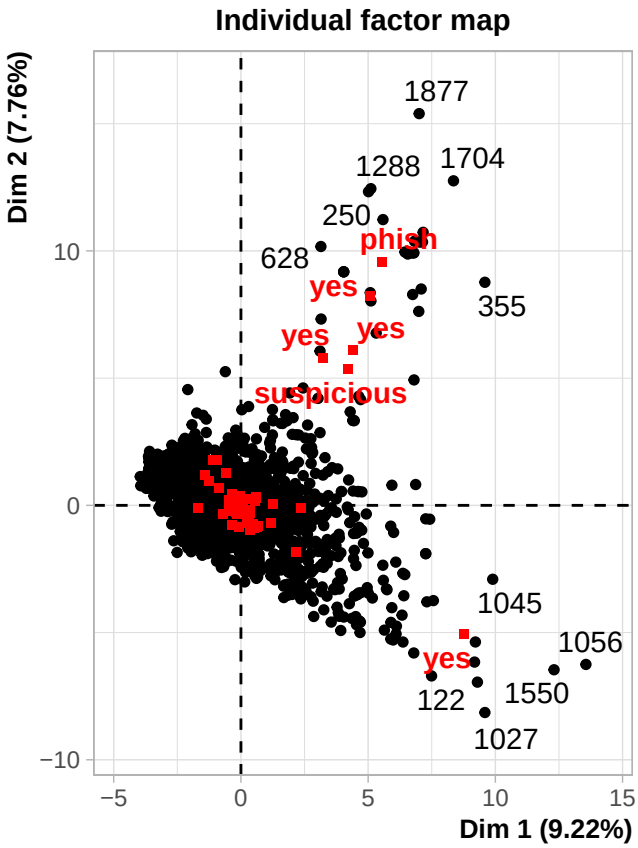


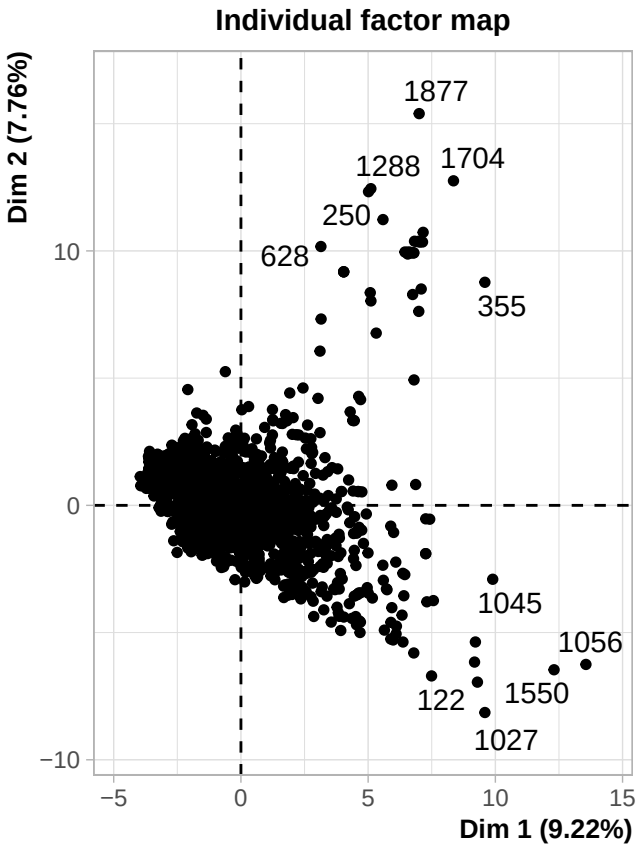

```
as_tibble(pca$loadings[, 1:2], rownames = "variable") %>%
    ggplot(aes(x = Comp.1, y = Comp.2, label = variable)) + geom_point(aes(color = abs(Comp.1 + Comp.2)
    scale_color_gradient(low = "blue", high = "red") + ggrepel::geom_text_repel() + theme(legend.positi
    geom_vline(xintercept = 0) + geom_hline(yintercept = 0)
```

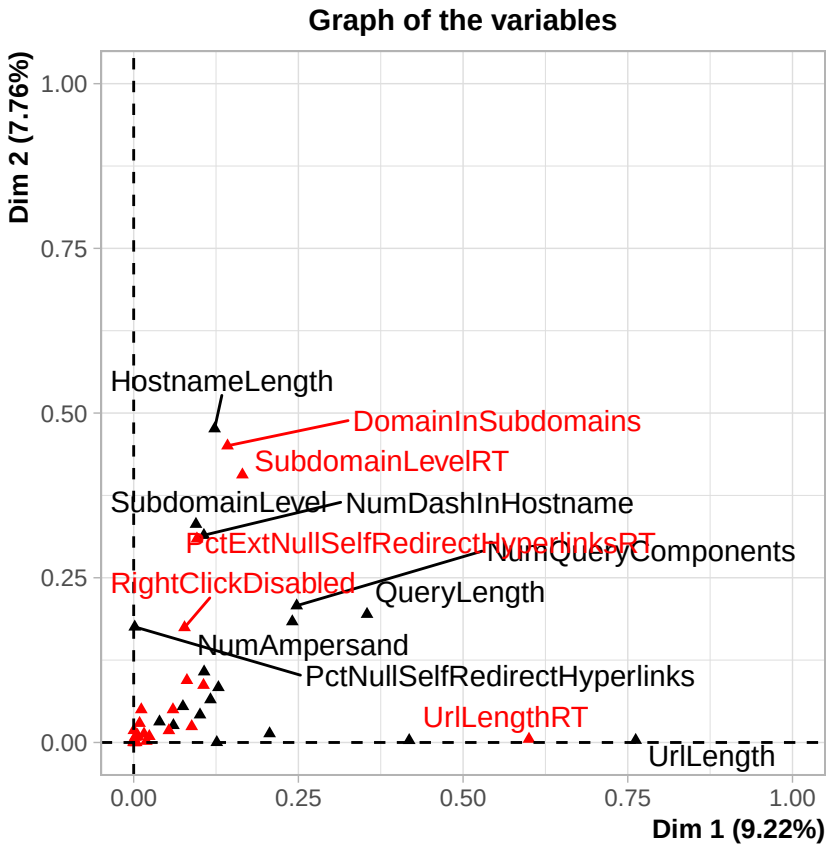


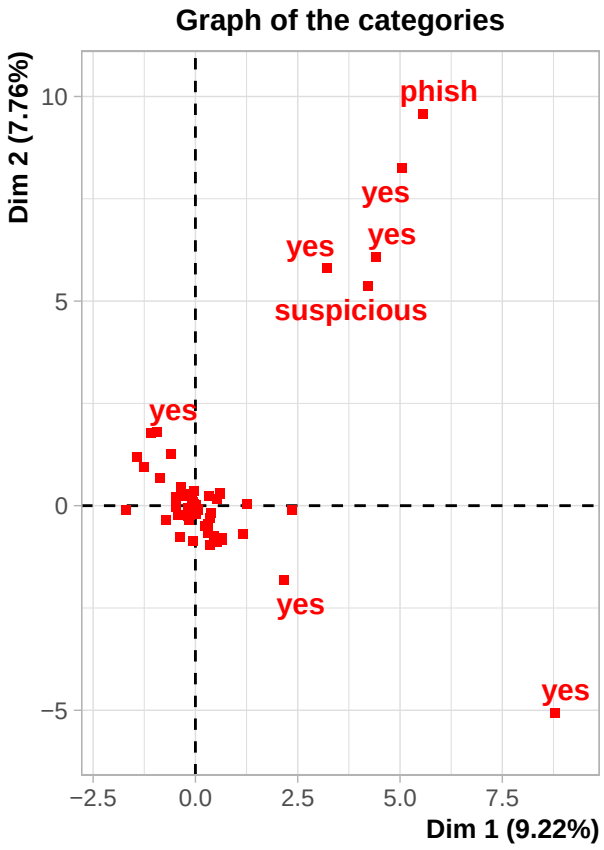
An FAMD analysis, which combines PCA on quantitative variables with MCA on factors, fails to extract much variance in the first few factors. We need to reduce the number of variables.

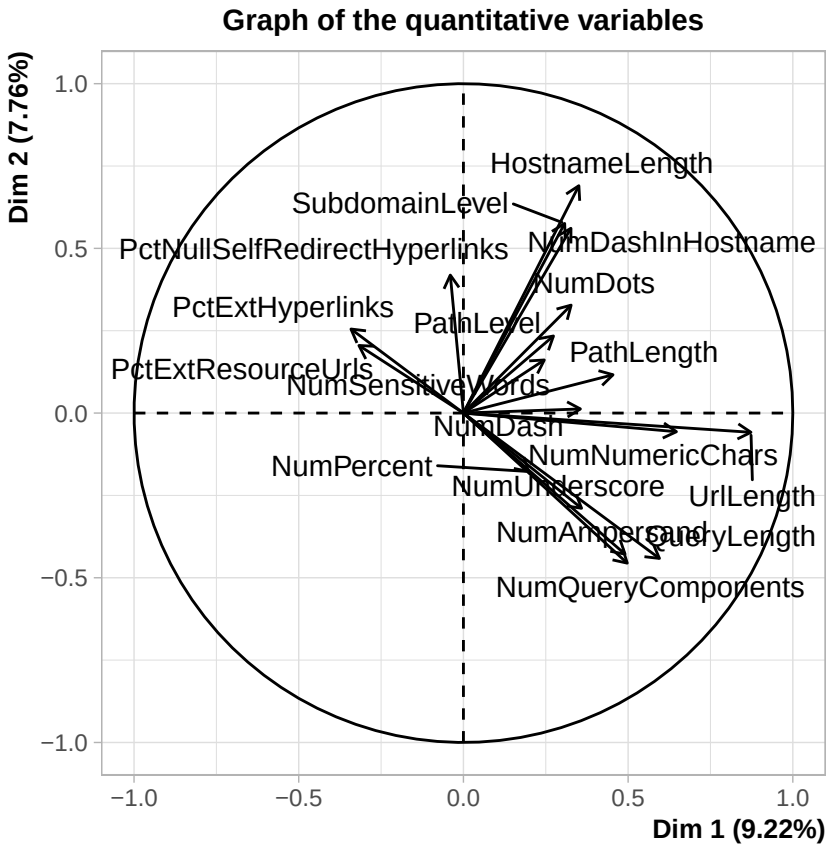
```
library(FactoMineR)
library(factoextra)
set.seed(555)

train_i <- sample(1:nrow(cleaned), nrow(cleaned)/5, replace = FALSE)
famd <- FAMD(mutate(cleaned[train_i, -c(1, ncol(cleaned))], across(where(is.numeric), scale)))
```


Individual factor map
Dim 2 (7.76%)
Dim 1 (9.22%)
1877
1288
1704
250
phish
628
yes
355
yes
yes
suspicious
1045
1056
yes
122
1550
1027
10
0
−10
−5
0
5
10
15

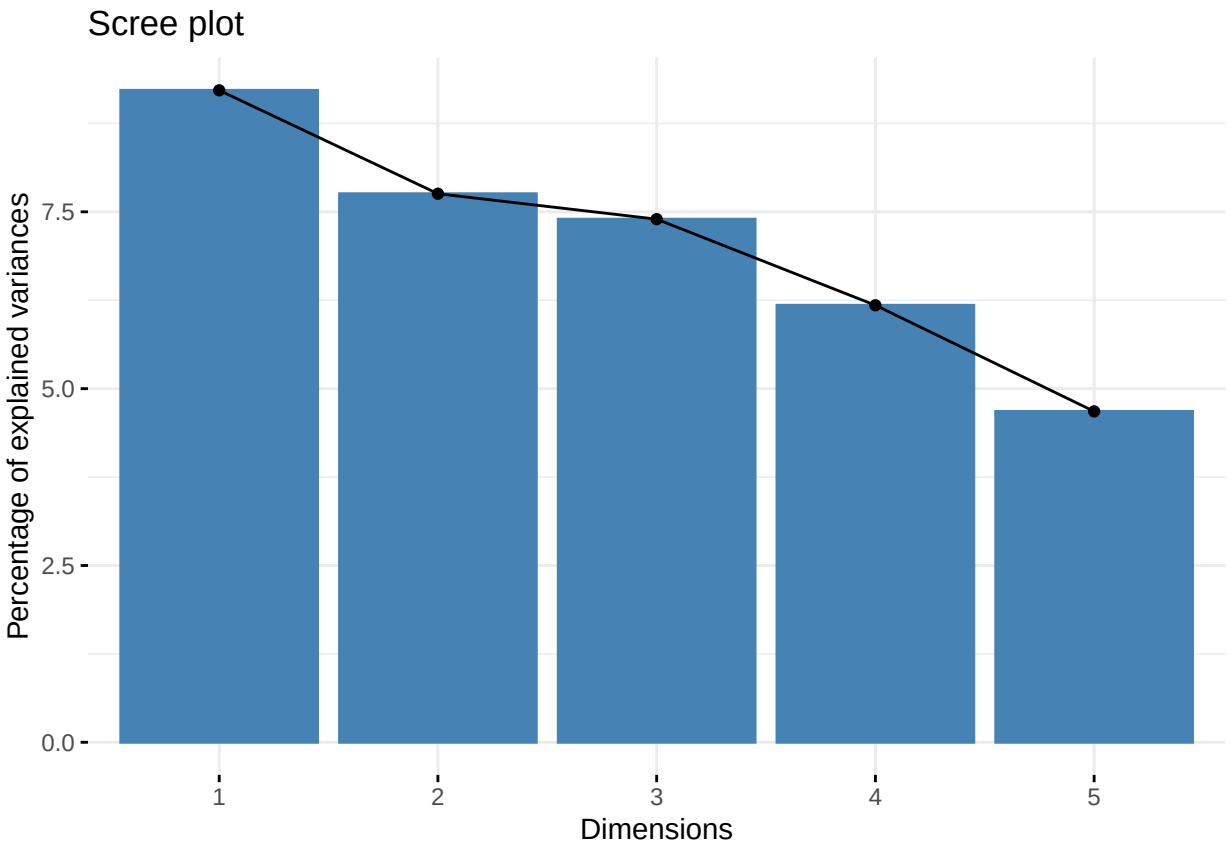

Individual factor map
Dim 2 (7.76%)
Dim 1 (9.22%)
1877
1704
1288
250
628
355
1045
1056
1550
122
1027
10
0
−10
−5
0
5
10
15


Graph of the variables
Dim 2 (7.76%)
1.00
0.75
0.50
0.25
0.00
HostnameLength
DomainInSubdomains
SubdomainLevelRT
SubdomainLevel
NumDashInHostname
PctExtNullSelfRedirectHyperlinksRT
NumQueryComponents
RightClickDisabled
QueryLength
NumAmpersand
PctNullSelfRedirectHyperlinks
UrlLengthRT
UrlLength
0.00
0.25
0.50
0.75
1.00
Dim 1 (9.22%)


Graph of the categories
Dim 2 (7.76%)
Dim 1 (9.22%)
10
5
0
−5
−2.5
0.0
2.5
5.0
7.5
phish
yes
yes
yes
suspicious
yes
yes
yes

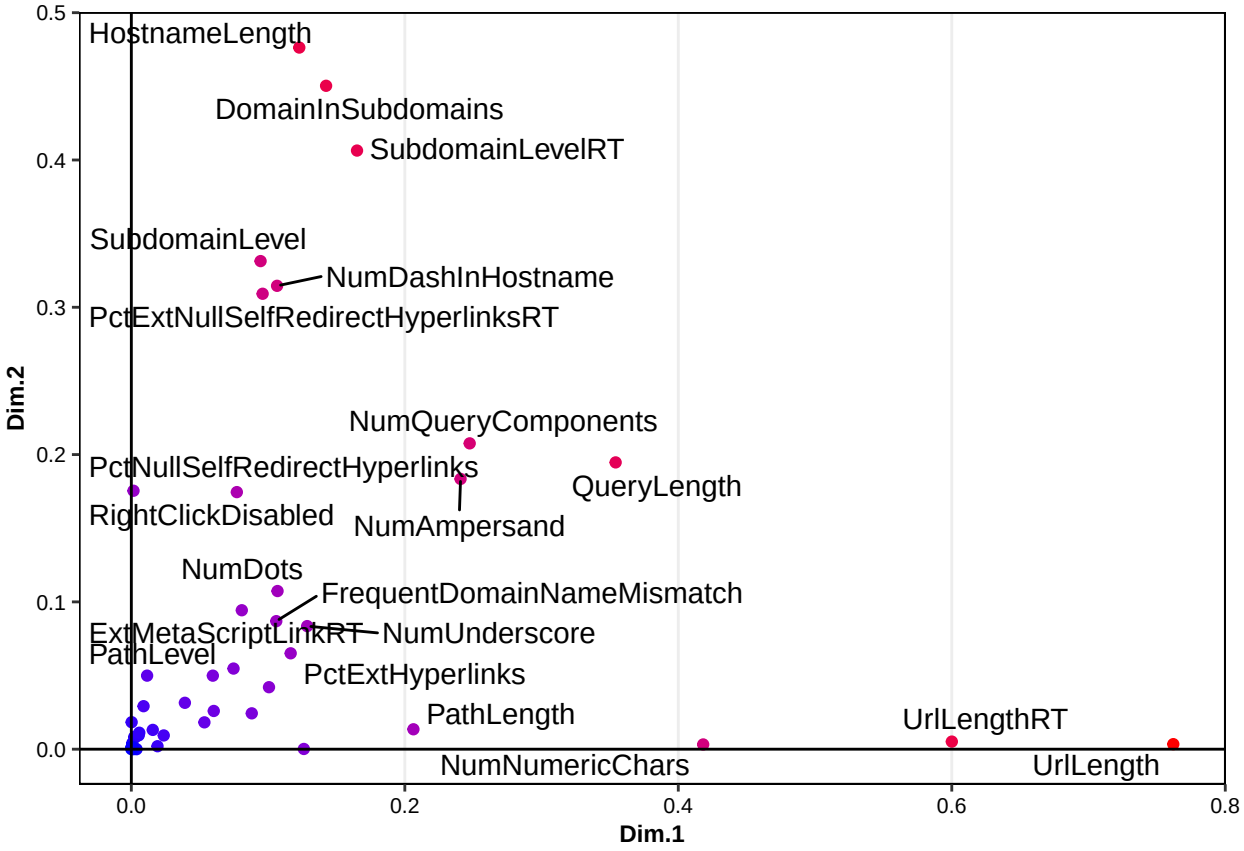



```
fviz_screeplot(famd)
```


Scree plot
Percentage of explained variances
7.5
5.0
2.5
0.0
1
2
3
4
5
Dimensions

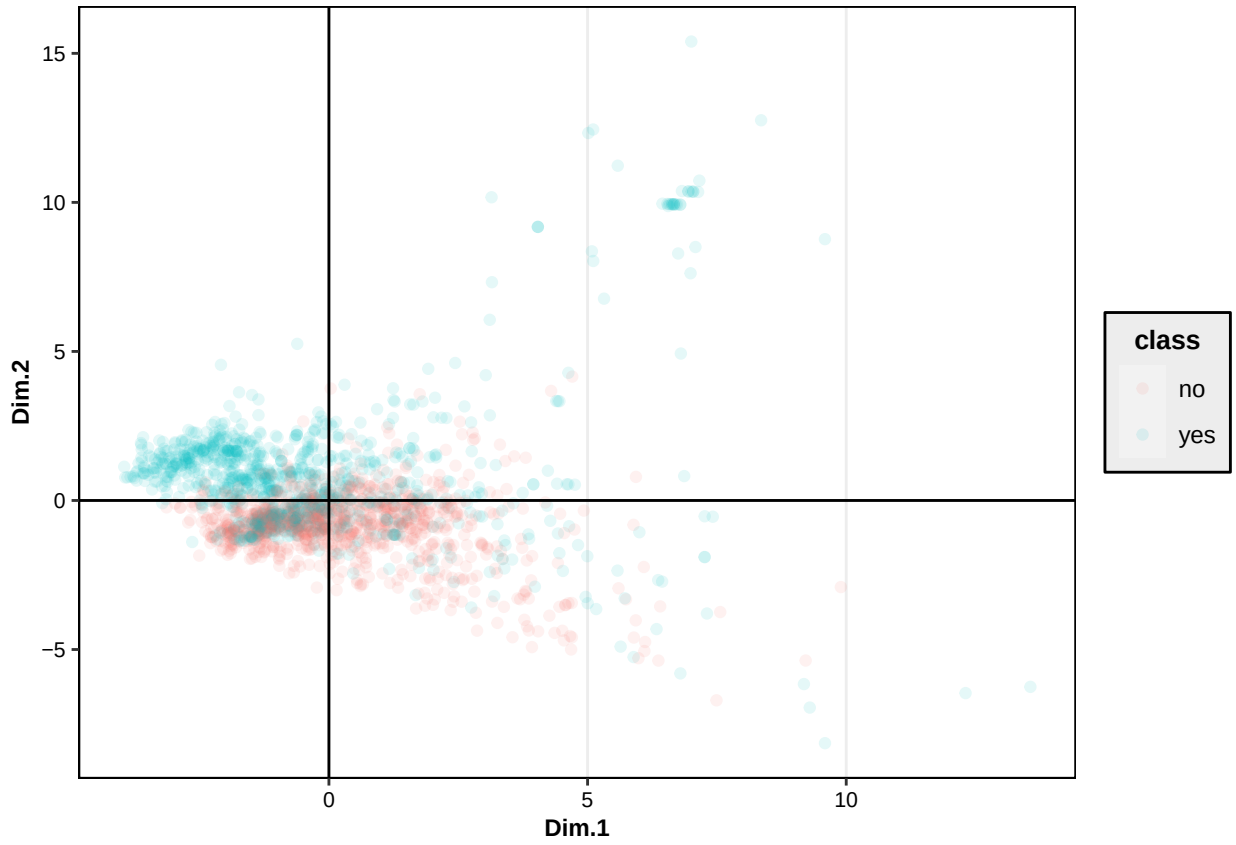
It's hard to discern a pattern by comparing factor scores for the variables.

```
as_tibble(famd$var$coord, rownames = "variable") %>%
    ggplot(aes(x = Dim.1, y = Dim.2)) + geom_point(aes(color = abs(Dim.1 + Dim.2))) + scale_color_gradi
    high = "red") + ggrepel::geom_text_repel(aes(label = variable)) + theme(legend.position = "none") +
    geom_vline(xintercept = 0) + geom_hline(yintercept = 0)
```



Still, it seems phishing sites score a little higher on factor 2 than legitimate sites.

```
as_tibble(famd$ind$coord[, 1:2]) %>%
    mutate(class = cleaned$CLASS_LABEL[train_i]) %>%
    ggplot(aes(x = Dim.1, y = Dim.2, color = class)) + geom_jitter(alpha = 0.1) + geom_vline(xintercept
    geom_hline(yintercept = 0)
```



We seem to have narrowed down a set of URL features associated with phishing.

```
famd$var$cos2 %>%
    as_tibble(rownames = "variable") %>%
    select(1:3) %>%
    arrange(-(Dim.1 + Dim.2)) %>%
    head()
```

variable	Dim.1	Dim.2
UrlLength	0.5806250	0.0000117
HostnameLength	0.0150895	0.2268860
DomainInSubdomains	0.0202876	0.2028342
UrlLengthRT	0.1800203	0.0000132
NumNumericChars	0.1748913	0.0000103
QueryLength	0.1254446	0.0379059

```r
as_tibble(famd$var$contrib[, 1:2], rownames = "variable") %>%
    arrange(-pmax(Dim.1, Dim.2)) %>%
    head()
```

variable	Dim.1	Dim.2
UrlLength	15.597155	0.0830441
UrlLengthRT	12.282118	0.1250746
HostnameLength	2.514406	11.5876430
DomainInSubdomains	2.915502	10.9562496
SubdomainLevelRT	3.378355	9.8863100
NumNumericChars	8.560154	0.0779480

Correspondence Analysis

I make sure to read up on the CA algorithm before continuing. If I read the Wikipedia article right, it goes like this:

1. Divide row and column sums of 2-way contingency table by total sums.
2. Create diagonal matrix from inverse of square roots of these vectors.

$$W_m = \text{diag}(1/(\sqrt{w_m})$$
$$W_n = \text{diag}(1/\sqrt{w_n})$$

3. Divide contingency table by sum of cells.
4. Compute standardized residuals:

$$S = W_M(P - w_m w_n)W_n$$

where P is the data matrix. This is the difference of the contingency table and the outer product of the marginal sums, with each cell scaled by its column and row sum. That outer product is actually just the expected frequencies matrix from the chi-squared test.

Then singular value decomposition is applied

$$S = U\Sigma V^*$$

The sum of the singular values in Σ is also the total sum of squares.

As with PCA, each singular vector of the projection accounts for a share of total variance, higher being better. Factor scores for the rows are derived by:

$$F_m = W_m U\Sigma$$

And for the columns:

$$F_n = W_n V\Sigma$$

The singular vectors of the dimension are scaled by the inverse square roots of its sums. This means distances in principal coordinates equal chi-square distances.

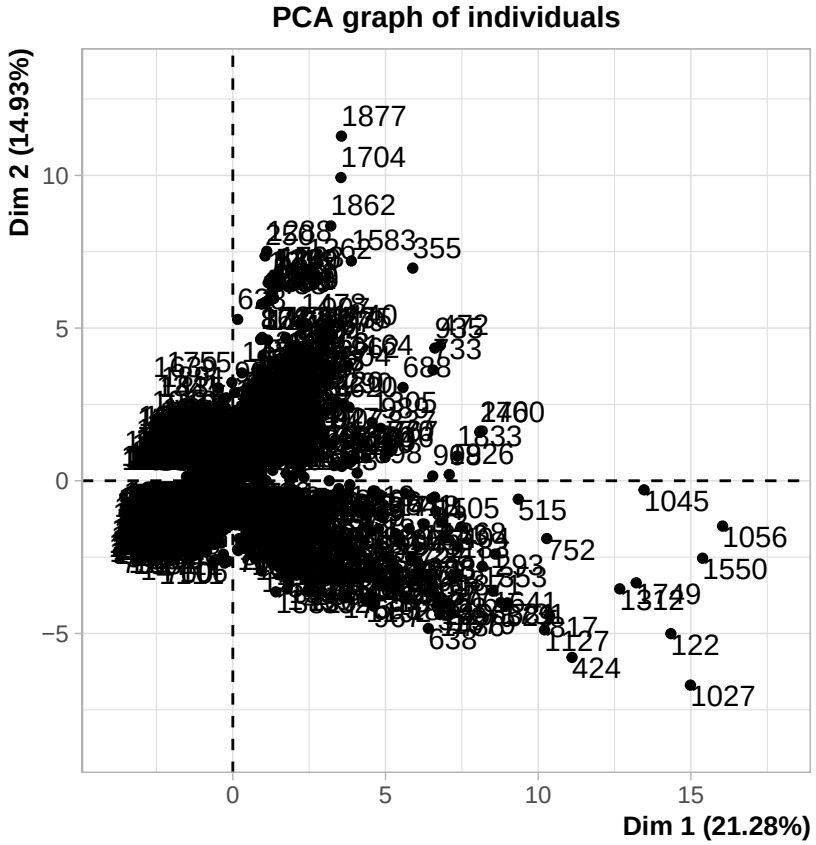
In a nutshell, CA uses SVD of the expected frequencies of the contingency table, while PCA uses eigendecomposition of the covariance or correlation matrix.

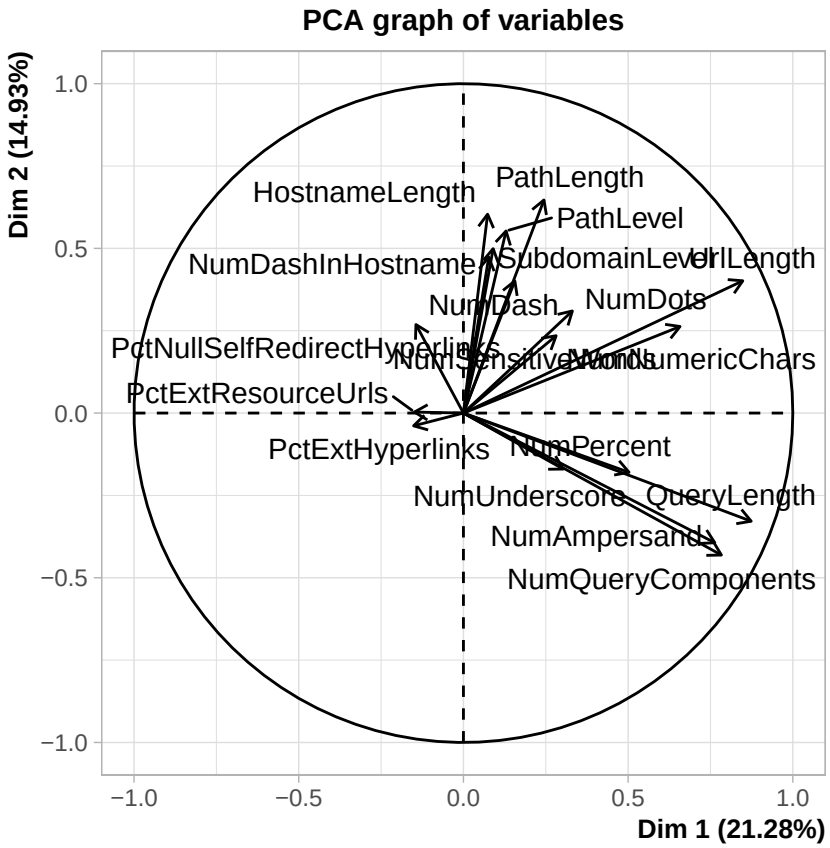
Multiple correspondence analysis conducts CA on an indicator matrix representing the level of each categorical variable to which each observation belongs

Cluster Analysis

A PCA on only the numeric variables captures more than 97% of variance in three components. That is very good.

```
pca2 <- select(cleaned[train_i, -1], where(~!is.factor(.x))) %>%
    PCA(ncp = 3)
```





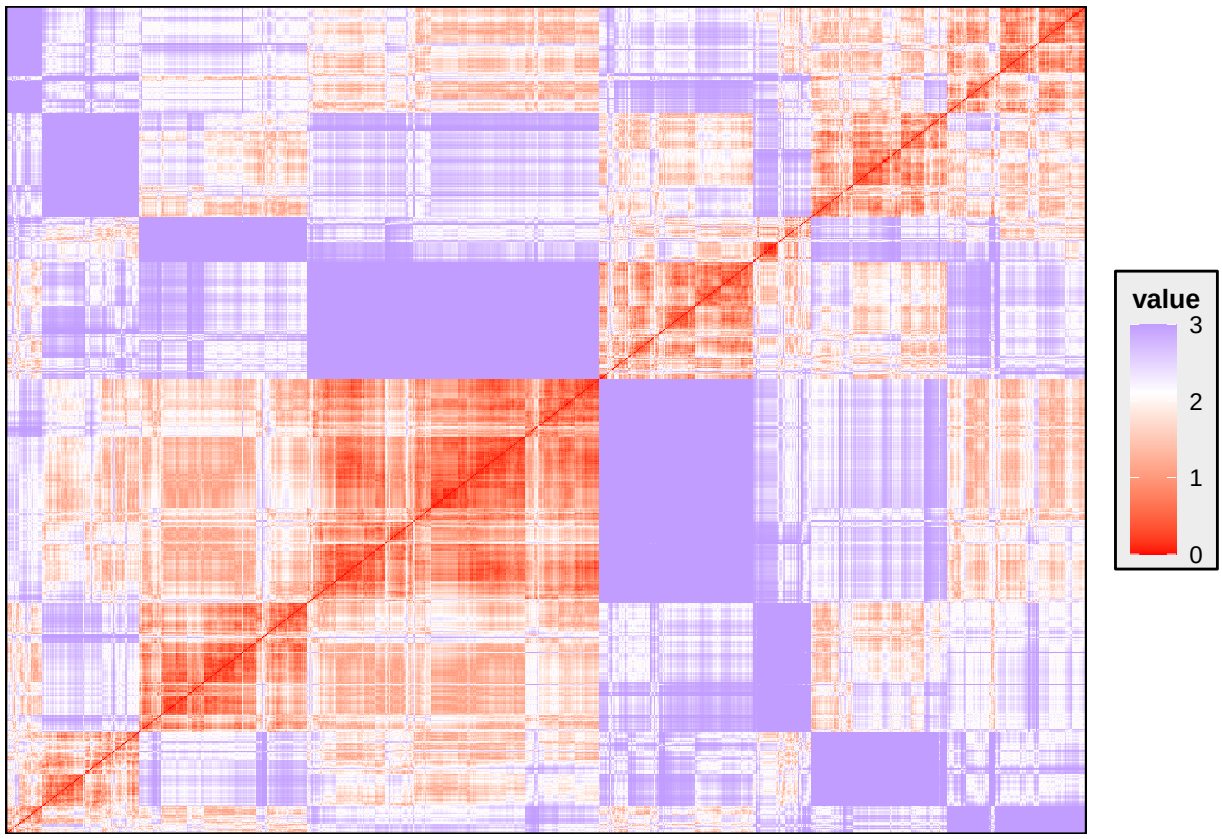

```
head(pca2$var$coord)
```

```
                     Dim.1     Dim.2      Dim.3
NumDots          0.33075537 0.3098955  0.4695739
SubdomainLevel   0.09025700 0.4990444  0.6104764
PathLevel        0.12942506 0.5528001 -0.3638644
UrlLength        0.84810488 0.4015114 -0.2620845
NumDash          0.15606872 0.4011265 -0.5382057
NumDashInHostname 0.07689383 0.4736822  0.3965387
```

```
distance <- pca2$ind$coord %>%
    get_dist(stand = TRUE, method = "canberra")
```

A distance visualization for the distances matrix of the components.

```
fviz_dist(distance, show_labels = FALSE)
```

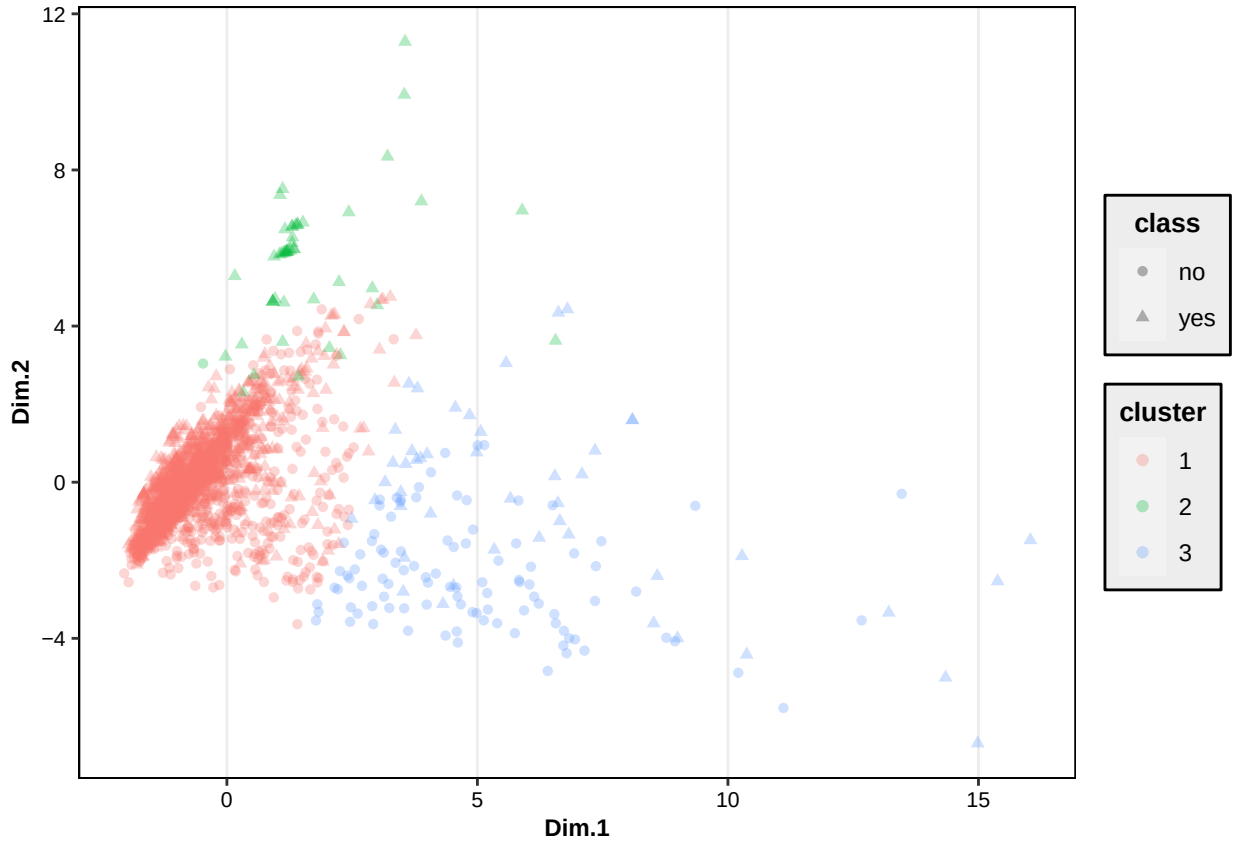


We have far too many numeric variables as is, so I take the first two principal component scores and use them to classify the observations hierarchically. I can use these clusterings to suggest a good number of clsuters for a `kmeans` clustering. Some information on the method.

```
hcpc <- HCPC(pca2, method = "complete", metric = "manhattan", nb.clust = 0L, cluster.CA = "rows", graph
```

The HCPC suggests 3 clusters, but cluster 2 has low class homogeneity.

```
as_tibble(hcpc$call$X, rownames = "obs") %>%
    arrange(as.numeric(obs)) %>%
    mutate(class = cleaned$CLASS_LABEL[train_i]) %>%
    ggplot(aes(x = Dim.1, y = Dim.2)) + geom_jitter(aes(color = as.factor(clust), shape = class), alpha
    labs(color = "cluster")
```



We see many of the same variables are most distinctive within each cluster.

```
hcpc$desc.var$quanti %>%
    map(as_tibble, rownames = "variable") %>%
    bind_rows(.id = "cluster") %>%
    group_by(cluster) %>%
    arrange(abs(-`Mean in category` - `Overall mean`))
```

cluster	variable	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
1	NumPercent	-10.411187	0.0238491	0.0735000	0.2331088	0.6450564	0.0000000
3	PctNullSelfRedirectHyperlinks	5.583157	0.0002537	0.1427912	0.0020848	0.3214969	0.0000000
1	NumSensitiveWords	-10.835390	0.0748752	0.1040000	0.2949866	0.3635712	0.0000000
1	NumDashInHostname	-15.007499	0.0915141	0.1575000	0.3176279	0.5947216	0.0000000

cluster	variable	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
1	NumAmpersand	-28.195573	0.0488075	0.2535000	0.2661374	0.9819561	0.0000000
3	PctExtHyperlinks	-3.629163	0.1325295	0.2289273	0.2029211	0.3344939	0.0002843
2	PctExtHyperlinks	-1.976870	0.1365656	0.2289273	0.2755774	0.3344939	0.0480563
3	NumSensitiveWords	6.058379	0.2789116	0.1040000	0.5567436	0.3635712	0.0000000
1	PctExtHyperlinks	4.213868	0.2393481	0.2289273	0.3428486	0.3344939	0.0000251
1	NumUnderscore	-15.854095	0.1996672	0.3250000	0.7979869	1.0692871	0.0000000
2	PctExtResourceUrls	-4.077454	0.1636098	0.3815678	0.2924415	0.3826994	0.0000455
1	NumQueryComponents	-28.751292	0.1719357	0.4510000	0.5301241	1.3128591	0.0000000
3	PctExtResourceUrls	-4.403884	0.2477338	0.3815678	0.3043887	0.3826994	0.0000106
2	PctNullSelfRedirectHyperlinks	9.311273	0.5609217	0.1427912	0.4580427	0.3214969	0.0000000
2	NumSensitiveWords	10.554786	0.6400000	0.1040000	0.9112629	0.3635712	0.0000000
1	PctExtResourceUrls	5.992861	0.3985237	0.3815678	0.3866139	0.3826994	0.0000000
3	NumPercent	12.376769	0.7074830	0.0735000	2.1355777	0.6450564	0.0000000
3	SubdomainLevel	-2.701545	0.4421769	0.6125000	0.5609668	0.7939419	0.0069018
1	SubdomainLevel	-10.709281	0.5496395	0.6125000	0.5351262	0.7939419	0.0000000
3	NumUnderscore	19.165592	1.9523810	0.3250000	2.2049213	1.0692871	0.0000000
2	NumDashInHostname	31.570096	2.7800000	0.1575000	1.8032193	0.5947216	0.0000000
3	NumAmpersand	33.041060	2.8299320	0.2535000	2.2508476	0.9819561	0.0000000
2	SubdomainLevel	24.955886	3.3800000	0.6125000	2.4648732	0.7939419	0.0000000
3	NumQueryComponents	33.650503	3.9591837	0.4510000	2.5684574	1.3128591	0.0000000
1	NumDots	-11.736876	2.3183583	2.4330000	1.0268882	1.3211779	0.0000000
2	NumDash	3.550638	3.3000000	1.7810000	2.0904545	3.0628482	0.0003843
3	NumDots	5.404420	3.0000000	2.4330000	2.4467110	1.3211779	0.0000001
3	PathLevel	-2.361768	2.8775510	3.2195000	1.9653390	1.8232717	0.0181880
2	PathLevel	2.279423	3.8000000	3.2195000	1.5362291	1.8232717	0.0226419
2	NumDots	13.368477	4.9000000	2.4330000	2.5865034	1.3211779	0.0000000
1	NumNumericChars	-18.221649	4.6283971	5.8865000	7.3900529	9.3389838	0.0000000
1	QueryLength	-32.186746	2.7487521	8.3250000	8.8284062	23.4334456	0.0000000
2	NumNumericChars	6.357880	14.1800000	5.8865000	8.7651355	9.3389838	0.0000000
3	NumNumericChars	17.003809	18.4965986	5.8865000	16.9947467	9.3389838	0.0000000
3	HostnameLength	-3.314770	16.7619048	18.9930000	5.1220921	8.4760221	0.0009172
1	HostnameLength	-13.120265	18.1708264	18.9930000	5.8338706	8.4760221	0.0000000
3	PathLength	-3.403390	28.2244898	34.8080000	23.1751086	24.3597647	0.0006656
1	PathLength	2.187098	35.2018857	34.8080000	24.3245813	24.3597647	0.0287353
2	HostnameLength	30.582611	55.2000000	18.9930000	14.8108069	8.4760221	0.0000000
3	QueryLength	37.172264	77.4965986	8.3250000	35.1531874	23.4334456	0.0000000
1	UrlLength	-25.430338	63.2695507	69.3625000	24.8641902	32.4075777	0.0000000
2	UrlLength	8.619597	108.3800000	69.3625000	29.6532561	32.4075777	0.0000000
3	UrlLength	23.882390	130.8231293	69.3625000	40.7029559	32.4075777	0.0000000

I apply clustering analysis. Two clusters are well differentiated, but one is almost evenly split.

```
table_clusters <- function(df, clust_vec, class_var) {
    class_var <- substitute(class_var)
    df %>%
        mutate(cluster = as.factor(clust_vec)) %>%
        group_by(cluster) %>%
        group_modify(~fct_count(.x[[as.character(class_var)]], prop = TRUE))
}
clust <- kmeans(famd$ind$coord, nstart = 5, centers = 3)

table_clusters(cleaned[train_i, ], clust$cluster, CLASS_LABEL)
```

cluster	f	n	p
1	no	44	0.1073171
1	yes	366	0.8926829
2	no	524	0.5464025
2	yes	435	0.4535975
3	no	438	0.6941363
3	yes	193	0.3058637

Using the principal components for clustering does pretty terribly, however.

```
pc_clust <- kmeans(pca2$ind$coord, nstart = 5, centers = 3)
table_clusters(cleaned[train_i, ], pc_clust$cluster, CLASS_LABEL)
```

cluster	f	n	p
1	no	583	0.4376877
1	yes	749	0.5623123
2	no	93	0.7265625
2	yes	35	0.2734375
3	no	330	0.6111111
3	yes	210	0.3888889

Prediction

Following this approach, I conduct k-means clustering for two clusters, split the dataset by cluster, then apply a cross-validated lasso regression to each subset to predict whether the link is legitimate.

This requires me to create the model matrix to represent categorical variables, which makes interpretation more difficult but does not impact accuracy.

```
library(glmnet)

train <- cleaned[train_i, ]
test <- cleaned[-train_i, ]

train_cv <- train %>%
    split(kmeans(famd$ind$coord, nstart = 5, centers = 2)$cluster) %>%
```

```
map(~select(.x, where(~!is.factor(.x) || n_distinct(.x) > 1), -id)) %>%
map(~list(data = .x, mat = model.matrix(~., .x[, -ncol(.x)]))) %>%
map(~cv.glmnet(.x$mat, .x$data$CLASS_LABEL, type.measure = "deviance", family = "binomial"))
```

In a binomial model , a one-unit increase in β_i increases the odds by a factor of e^{β_i}.

Cluster 1 is predominantly phishing, cluster 2 predominantly not. In each case the most prominent coefficient is that for the presence of an IP address.

```
train_cv %>%
    map(~broom::tidy(.x$glmnet.fit) %>%
        group_by(term) %>%
        filter(lambda == .x$lambda.min) %>%
        slice_max(order_by = abs(estimate), n = 1) %>%
        ungroup() %>%
        arrange(-abs(estimate)) %>%
        head() %>%
        knitr::kable())
```

```
$`1`
```

term	step	estimate	lambda	dev.ratio
InsecureFormsyes	53	3.708025	0.0022067	0.8317968
NoHttpsyes	53	-3.462006	0.0022067	0.8317968
PctExtNullSelfRedirectHyperlinksRTsuspicious	53	-3.412880	0.0022067	0.8317968
FrequentDomainNameMismatchyes	53	3.271340	0.0022067	0.8317968
MissingTitleyes	53	2.798801	0.0022067	0.8317968
PctExtNullSelfRedirectHyperlinksRTlegitimate	53	-2.793139	0.0022067	0.8317968

```
$`2`
```

term	step	estimate	lambda	dev.ratio
NoHttpsyes	44	-4.507026	0.0038437	0.8289878
FrequentDomainNameMismatchyes	44	3.317067	0.0038437	0.8289878
IpAddressyes	44	2.919854	0.0038437	0.8289878
PctExtNullSelfRedirectHyperlinksRTlegitimate	44	-2.562613	0.0038437	0.8289878
NumHashyes	44	2.219502	0.0038437	0.8289878
SubmitInfoToEmailyes	44	-2.186465	0.0038437	0.8289878

```
FAMD_test <- FAMD(mutate(test, across(where(is.numeric), scale)), sup.var = 50, graph = FALSE)
test_clust <- kmeans(FAMD_test$ind$coord, centers = 2, nstart = 5)$cluster

test_dat <- test %>%
    split(test_clust) %>%
    map(~select(.x, where(~!is.factor(.x) || n_distinct(.x) > 1), -id)) %>%
    map(~list(data = .x, mat = model.matrix(~., .x[, -ncol(.x)])))
```

Finally, I fit the model to the test data.

```
test_fit <- map2(train_cv, test_dat, ~predict(.x, newx = .y$mat, type = "class", s = .x$lambda.min))
```

Here's the confusion matrix. Overall accuracy is about 90%. Since the class split is even, I'd say this model worked pretty well.

```
CM <- table(test$CLASS_LABEL, unsplit(test_fit, test_clust))
knitr::kable(CM)
```

	no	yes
no	3713	281
yes	607	3399

```
knitr::kable(CM/colSums(CM))
```

	no	yes
no	0.8594907	0.0650463
yes	0.1649457	0.9236413