

Phishing: jak się nie dać “złapać”

By Vitalii Morskyi & Julia Makarska

Zjawisko Phishingu zachodzi już od wielu lat. Jednak między innymi ostatni rok pokazał nam jak ważne jest bezpieczeństwo w internecie. Od roku świat się zatrzymał i przeniósł wszystko do Internetu. Z uwagi na ten fakt podjęcie tematu Phishingu uznaliśmy za bardzo na miejscu. Chcemy pokazać jak łatwo można dać się okraść. Przedstawiony przez nas projekt obejmuje tylko niewielki kawałek tej metody oszustwa, jednak uznaliśmy, że temat jest ciekawy. Phishing jest to atak oparty na wiadomościach e-mail lub SMS. Przestępcy internetowi próbują Cię oszukać i wymusić na Tobie działania zgodne z ich oczekiwaniami.

Importowanie danych, pakietów R i modułów

```
library("stringi")
library("stringr")
library("lattice")
library("ggplot2")
library("ggExtra")
library("hrbrthemes")
library("rgl")
library("tidyverse")
library("GGally")

source("modules/split-url.r")
source("modules/url-ambiguity.r")
source("modules/url-lengths.r")
source("modules/url-special-symbol-count.r")

# Mendeley Data : Dataset of Malicious and Benign Webpages
dfm <- read.csv("data/Webpages_Classification_10k.csv", row.names = "X")

# Aalto University : PhishStorm - phishing / legitimate URL dataset
dfp <- read.csv("data/PhishStorm_urlset_96k.csv")

head(dfm, n = 1L)

##                               url url_len      ip_add geo_loc tld   who_is https
## 0 http://www.pureevents.com/       26 94.65.190.24  Greece com complete yes
##   js_len js_obf_len
## 0      30          0
##
## 0 cancer whitenigger niggor phuking stroke trots honkers chink slave blackout bomd vietcong shitface
##   label
## 0  good
head(dfp)
```

```

##
## 1 nobell.it/70ffb52d079109dca5664cce6f317373782/login.SkyPe.com/en/cgi-bin/verification/login/70ffb5
## 2
## 3
## 4
## 5
## 6
##   ranking mld_res mld.ps_res card_rem ratio_Rrem ratio_Arem jaccard_RR
## 1 10000000    1      0     18  107.6111  107.27778      0
## 2 10000000    0      0     11  150.6364  152.27273      0
## 3 10000000    0      0     14   73.5000  72.64286      0
## 4 10000000    0      0      6  562.0000  590.66667      0
## 5 10000000    0      0      8   29.0000  24.12500      0
## 6 10000000    0      0      2  223.5000  234.00000      0
##   jaccard_RA jaccard_AR jaccard_AA jaccard_ARrd jaccard_ARrem label
## 1          0          0          0        0.8    0.795729      1
## 2          0          0          0        0.0    0.768577      1
## 3          0          0          0        0.0    0.726582      1
## 4          0          0          0        0.0    0.859640      1
## 5          0          0          0        0.0    0.748971      1
## 6          0          0          0        0.0    0.852227      1

```

Wstępne przygotowanie danych.

Przeprowadzono wstępna edycję datasetów, ponieważ na pierwszy rzut widać, że: - wejściowe ramki danych zawierają zbędne kolumny, których nie uwzględniamy w naszej analizie - mają różne nazwy najważniejszych dla naszej analizy kolumn - *domain* i *url* - otagowanie domen w zbiorze danych "PhishTank" nie jest zbyt zrozumiałe

```
dfm[c("content", "url_len", "ip_add")] <- NULL
head(dfm)
```

```

##
## 0
## 1
## 2
## 3 http://www.naughtycelebrity.com/sites/jennifer-love-hewitt-pictures/jennifer-love-hewitt-sexy.htm
## 4
## 5
##   geo_loc tld      who_is https js_len js_obf_len label
## 0      Greece com    complete    yes   30.0      0.000  good
## 1 United Kingdom de incomplete   no  815.4    399.546  bad
## 2  United States com incomplete   yes  176.0      0.000  good
## 3       Japan com incomplete   no  342.0      0.000  bad
## 4  United States com complete    yes  175.0      0.000  good
## 5       China com incomplete   no  767.7    429.912  bad

```

W ramce danych `dfm` pozostawiamy następujące kolumny: - url - Adres URL strony internetowej. - geo_loc - Lokalizacja geograficzna, w której jest hostowana strona internetowa. - tld - Domena najwyższego poziomu strony internetowej. - who_is - Czy informacje o domenie WHO IS są konkurencyjne, czy nie. - https - Czy witryna korzysta z https czy http. - js_len - Długość kodu JavaScript na stronie. - js_obf_len - Długość zaciemnionego kodu JavaScript. - label - Etykieta klasy dla niegroźnej lub złośliwej strony internetowej.

```

colnames(df1) [1] <- "url"
df1[c("card_rem", "ratio_Rrem", "ratio_Arem",
      "jaccard_RR", "jaccard_RA", "jaccard_AR",
      "jaccard_AA", "jaccard_ARrd", "jaccard_ARrem")] <- NULL
df1$label <- factor(df1$label)
df1$mld_res <- factor(df1$mld_res)
df1$mld.ps_res <- factor(df1$mld.ps_res)
levels(df1$label) <- c("good", "bad")
levels(df1$mld_res) <- c("no", "yes")
levels(df1$mld.ps_res) <- c("no", "yes")
head(df1)

##
## 1 nobell.it/70ffb52d079109dca5664cce6f317373782/login.SkyPe.com/en/cgi-bin/verification/login/70ffb5
## 2
## 3
## 4
## 5
## 6
##   ranking mld_res mld.ps_res label
## 1 10000000    yes      no   bad
## 2 10000000    no       no   bad
## 3 10000000    no       no   bad
## 4 10000000    no       no   bad
## 5 10000000    no       no   bad
## 6 10000000    no       no   bad

```

W ramce danych df1 pozostawiamy następujące kolumny: - url - Adres URL strony internetowej. - ranking - Ranking mld.ps według rankingu witryny Alexa. - mld_res - Czy są wyniki wyszukiwania czy nie dla mld w adresie URL. - mld.ps_res - Czy są wyniki wyszukiwania czy nie dla mld.ps w adresie URL. - label - Etykieta klasy dla niegroźnej lub złośliwej strony internetowej.

Obliczanie badanych cech.

Najpierw dzielimy link na następujące części: | *split_url()* oraz *clean_split_url()* - protocol (*schemat*) - domain name (*nazwa serwera*) - path (*ścieżka do pliku*) - query (*zapytanie*) - fragment (*fragment*)

```

split_res <- clean_split_url(df1$url)
split_res[20:29,]

```

```

##      url
##  [1,] "asladconcentration.com/papluk1/webscrcmd=_home-customer&nav=1/"
##  [2,] "www.regaranch.info/grafika/file/2012/atualizacao/www.itaú.com.br/"
##  [3,] "optimistic-pessimism.com/aoluserupdatealert.info.htm"
##  [4,] "mercadolivre.com.br.premiosfidelidade2012.com.br/confirmar/"
##  [5,] "www.everythinggoingon.net/~gpeveryt/home/Email/"
##  [6,] "mercadolivre.com.br.premiosfidelidade2012.com.br/"
##  [7,] "www.revitolcream.org/wp-content/plugins/all-in-one-seo-pack/rex/secure-code17/security/"
##  [8,] "jameshowardmusic.com/wp-content/themes/widescreen/includes/cache/bbnew/bb.php"
##  [9,] "xini.eu/00Qe"
## [10,] "myxxxxcollection.com/v1/js/555klisdr/bpd.com.do/do/l.popular.php"
##      protocol host
##  [1,] NA      "asladconcentration.com"

```

```

## [2,] NA      "www.regaranch.info"
## [3,] NA      "optimistic-pessimism.com"
## [4,] NA      "mercadolivre.com.br.premiosfidelidade2012.com.br"
## [5,] NA      "www.everythinggoingon.net"
## [6,] NA      "mercadolivre.com.br.premiosfidelidade2012.com.br"
## [7,] NA      "www.revitolcream.org"
## [8,] NA      "jameshowardmusic.com"
## [9,] NA      "xini.eu"
## [10,] NA     "myxxxcollection.com"
##       path
## [1,] "/paplkuk1/websrcmd=_home-customer&nav=1/"
## [2,] "/grafika/file/2012/atualizacao/www.itaú.com.br/"
## [3,] "/aoluserupdatealert.info.htm"
## [4,] "/confirmar/"
## [5,] "/~gpeveryt/home/Email/"
## [6,] "/"
## [7,] "/wp-content/plugins/all-in-one-seo-pack/rex/secure-code17/security/"
## [8,] "/wp-content/themes/widescreen/includes/cache/bbnew/bb.php"
## [9,] "/00Qe"
## [10,] "/v1/js/555klisdr/bpd.com.do/do/l.popular.php"
##       query fragment
## [1,] NA      NA
## [2,] NA      NA
## [3,] NA      NA
## [4,] NA      NA
## [5,] NA      NA
## [6,] NA      NA
## [7,] NA      NA
## [8,] NA      NA
## [9,] NA      NA
## [10,] NA     NA

```

Bierzemy pod uwagę następujące cechy każdej z wymienionych powyżej części: - długość | `url_lengths()` - następujące stosunki długości: | `url_lengths()` - Domain Name divided by URL (Nazwa domeny w stosunku do adresu URL) - Path divided by URL (Ścieżka w stosunku do adresu URL) - Argument divided by URL (Argument w stosunku do adresu URL) - Path divided by Domain Name (Ścieżka podzielona według nazwy domeny) - Argument divided by Domain Name (Argument w stosunku do nazwy domeny) - Argument divided by Path (Argument w stosunku do ścieżki) - ciąg znaków postaci litera-cyfra-litera | `letter_digit_letter()` - ciąg znaków postaci cyfra-litera-cyfra | `digit_letter_digit()` - połączenie dwóch poprzednich ciągów | `combined_url_ambiguity()` - liczba liter | `lett_dig_symb_count()` - liczba cyfr | `lett_dig_symb_count()` - liczba znaków interpunkcyjnych | `lett_dig_symb_count()`

```

lengths_res <- url_lengths(split_res)
lengths_res[20:29,]

```

	url_l	protocol_l	host_l	path_l	query_l	fragment_l	host_by_url	path_by_url
[1,]	63	NA	22	41	NA	NA	0.3492063	0.65079365
[2,]	65	NA	18	47	NA	NA	0.2769231	0.72307692
[3,]	52	NA	24	28	NA	NA	0.4615385	0.53846154
[4,]	59	NA	48	11	NA	NA	0.8135593	0.18644068
[5,]	47	NA	25	22	NA	NA	0.5319149	0.46808511
[6,]	49	NA	48	1	NA	NA	0.9795918	0.02040816
[7,]	87	NA	20	67	NA	NA	0.2298851	0.77011494
[8,]	77	NA	20	57	NA	NA	0.2597403	0.74025974

```

## [9,]    12      NA     7     5      NA      NA  0.5833333  0.41666667
## [10,]    63      NA    19    44      NA      NA  0.3015873  0.69841270
##   query_by_url path_by_host query_by_host query_by_path
## [1,]           NA  1.86363636      NA      NA
## [2,]           NA 2.61111111      NA      NA
## [3,]           NA 1.16666667      NA      NA
## [4,]           NA 0.22916667      NA      NA
## [5,]           NA 0.88000000      NA      NA
## [6,]           NA 0.02083333      NA      NA
## [7,]           NA 3.35000000      NA      NA
## [8,]           NA 2.85000000      NA      NA
## [9,]           NA 0.71428571      NA      NA
## [10,]          NA 2.31578947      NA      NA

ldl_res <- letter_digit_letter(split_res)
dld_res <- digit_letter_digit(split_res)
xyx_res <- combined_url_ambiguity(split_res)
cbind(ldl_res, dld_res, xyx_res)[20:29,]

##   ldl_url ldl_protocol ldl_host ldl_path ldl_query ldl_fragment dld_url
## [1,]    0        NA     0     0      NA      NA      NA    0
## [2,]    0        NA     0     0      NA      NA      NA    0
## [3,]    0        NA     0     0      NA      NA      NA    0
## [4,]    0        NA     0     0      NA      NA      NA    0
## [5,]    0        NA     0     0      NA      NA      NA    0
## [6,]    0        NA     0     0      NA      NA      NA    0
## [7,]    0        NA     0     0      NA      NA      NA    0
## [8,]    0        NA     0     0      NA      NA      NA    0
## [9,]    0        NA     0     0      NA      NA      NA    0
## [10,]   0        NA     0     0      NA      NA      NA    0
##   dld_protocol dld_host dld_path dld_query dld_fragment xyx_url
## [1,]      NA     0     0      NA      NA      NA    0
## [2,]      NA     0     0      NA      NA      NA    0
## [3,]      NA     0     0      NA      NA      NA    0
## [4,]      NA     0     0      NA      NA      NA    0
## [5,]      NA     0     0      NA      NA      NA    0
## [6,]      NA     0     0      NA      NA      NA    0
## [7,]      NA     0     0      NA      NA      NA    0
## [8,]      NA     0     0      NA      NA      NA    0
## [9,]      NA     0     0      NA      NA      NA    0
## [10,]     NA     0     0      NA      NA      NA    0
##   xyx_protocol xyx_host xyx_path xyx_query xyx_fragment
## [1,]      NA     0     0      NA      NA
## [2,]      NA     0     0      NA      NA
## [3,]      NA     0     0      NA      NA
## [4,]      NA     0     0      NA      NA
## [5,]      NA     0     0      NA      NA
## [6,]      NA     0     0      NA      NA
## [7,]      NA     0     0      NA      NA
## [8,]      NA     0     0      NA      NA
## [9,]      NA     0     0      NA      NA
## [10,]     NA     0     0      NA      NA

ldsc_res <- lett_dig_symb_count(split_res)
ldsc_res[20:29,]

```

```

##      lett_url lett_protocol lett_host lett_path lett_query lett_fragment
## [1,]      52          NA        21       31          NA          NA
## [2,]      50          NA        16       34          NA          NA
## [3,]      47          NA        22       25          NA          NA
## [4,]      48          NA        39        9          NA          NA
## [5,]      40          NA        23       17          NA          NA
## [6,]      39          NA        39        0          NA          NA
## [7,]      70          NA        18       52          NA          NA
## [8,]      67          NA        19       48          NA          NA
## [9,]       8          NA         6        2          NA          NA
## [10,]     48          NA        18       30          NA          NA
##      dig_url dig_protocol dig_host dig_path dig_query dig_fragment symb_url
## [1,]      2          NA         0        2          NA          NA         7
## [2,]      4          NA         0        4          NA          NA        11
## [3,]      0          NA         0        0          NA          NA         5
## [4,]      4          NA         4        0          NA          NA         7
## [5,]      0          NA         0        0          NA          NA         6
## [6,]      4          NA         4        0          NA          NA         6
## [7,]      2          NA         0        2          NA          NA        15
## [8,]      0          NA         0        0          NA          NA        10
## [9,]      2          NA         0        2          NA          NA         2
## [10,]     4          NA         0        4          NA          NA        11
##      symb_protocol symb_host symb_path symb_query symb_fragment
## [1,]          NA         1        6          NA          NA
## [2,]          NA         2        9          NA          NA
## [3,]          NA         2        3          NA          NA
## [4,]          NA         5        2          NA          NA
## [5,]          NA         2        4          NA          NA
## [6,]          NA         5        1          NA          NA
## [7,]          NA         2       13          NA          NA
## [8,]          NA         1        9          NA          NA
## [9,]          NA         1        1          NA          NA
## [10,]         NA         1       10          NA          NA

```

Powtarzamy analogicznie, obliczenia dla drugiego zbioru danych, ale tym razem już nie będziemy wyświetlać wyników każdego obliczenia.

```

split_res_2 <- clean_split_url(dfm$url)
lengths_res_2 <- url_lengths(split_res_2)
ldl_res_2 <- letter_digit_letter(split_res_2)
dld_res_2 <- digit_letter_digit(split_res_2)
xyx_res_2 <- combined_url_ambiguity(split_res_2)
ldsc_res_2 <- lett_dig_symb_count(split_res_2)

```

Zebrańie wszystkich parametrów do jednej ramki danych.

Najpierw tworzymy jedną macierz z wynikami wszystkich obliczeń, którą potem konwertujemy w ramkę danych.

```

params_df <- as.data.frame(cbind(
  lengths_res,
  ldl_res,

```

```

    dld_res,
    xyx_res,
    ldsc_res
))
params_df_2 <- as.data.frame(cbind(
  lengths_res_2,
  ldl_res_2,
  dld_res_2,
  xyx_res_2,
  ldsc_res_2
))
dim(params_df)

## [1] 95911     48

```

Otrzymano dość dużą ramkę danych, więc przydałoby się ją troszkę zmniejszyć. Po przeanalizowaniu jej struktury zauważono, że niektóre kolumny posiadają 0 poziomów, czyli nie przechowują żadnej informacji, więc można je usunąć. Zatem można połączyć wejściową ramkę danych z otrzymaną. Dokonano analogicznych działań dla obu zbiorów danych.

```

str(params_df)

## 'data.frame': 95911 obs. of 48 variables:
## $ url_l      : num  225 81 177 60 116 36 61 60 19 193 ...
## $ protocol_l : num NA NA NA NA NA NA NA NA NA ...
## $ host_l     : num  9 15 16 18 19 25 28 19 14 19 ...
## $ path_l     : num 125 66 161 42 60 11 33 41 5 81 ...
## $ query_l    : num 90 NA NA NA 36 NA NA NA 92 ...
## $ fragment_l : num NA NA NA NA NA NA NA NA NA ...
## $ host_by_url: num 0.04 0.1852 0.0904 0.3 0.1638 ...
## $ path_by_url: num 0.556 0.815 0.91 0.7 0.517 ...
## $ query_by_url: num 0.4 NA NA NA 0.31 ...
## $ path_by_host: num 13.89 4.4 10.06 2.33 3.16 ...
## $ query_by_host: num 10 NA NA NA 1.89 ...
## $ query_by_path: num 0.72 NA NA NA 0.6 ...
## $ ldl_url     : num 4 0 4 0 0 0 0 0 0 10 ...
## $ ldl_protocol: num NA NA NA NA NA NA NA NA NA ...
## $ ldl_host    : num 0 0 0 0 0 0 0 0 0 0 ...
## $ ldl_path    : num 4 0 4 0 0 0 0 0 0 0 ...
## $ ldl_query   : num 0 NA NA NA 0 NA NA NA NA 10 ...
## $ ldl_fragment: num NA NA NA NA NA NA NA NA NA ...
## $ dld_url     : num 4 0 5 0 0 0 0 0 0 11 ...
## $ dld_protocol: num NA NA NA NA NA NA NA NA NA ...
## $ dld_host    : num 0 0 0 0 0 0 0 0 0 0 ...
## $ dld_path    : num 4 0 5 0 0 0 0 0 0 0 ...
## $ dld_query   : num 0 NA NA NA 0 NA NA NA NA 11 ...
## $ dld_fragment: num NA NA NA NA NA NA NA NA NA ...
## $ xyx_url     : num 8 0 8 0 0 0 0 0 0 13 ...
## $ xyx_protocol: num NA NA NA NA NA NA NA NA NA ...
## $ xyx_host    : num 0 0 0 0 0 0 0 0 0 0 ...
## $ xyx_path    : num 8 0 8 0 0 0 0 0 0 0 ...
## $ xyx_query   : num 0 NA NA NA 0 NA NA NA NA 13 ...

```

```

## $ xyx_fragment : num NA ...
## $ lett_url      : num 135 65 111 52 82 32 47 45 13 132 ...
## $ lett_protocol: num NA NA NA NA NA NA NA NA NA ...
## $ lett_host     : num 8 13 15 16 18 23 27 18 9 18 ...
## $ lett_path     : num 68 52 96 36 52 9 20 27 4 62 ...
## $ lett_query    : num 59 NA NA NA 12 NA NA NA NA 52 ...
## $ lett_fragment: num NA NA NA NA NA NA NA NA NA ...
## $ dig_url       : num 58 1 47 0 21 0 7 4 4 35 ...
## $ dig_protocol  : num NA NA NA NA NA NA NA NA NA ...
## $ dig_host      : num 0 0 0 0 0 0 0 0 4 0 ...
## $ dig_path      : num 45 1 47 0 0 0 7 4 0 1 ...
## $ dig_query     : num 13 NA NA NA 21 NA NA NA NA 34 ...
## $ dig_fragment  : num NA NA NA NA NA NA NA NA NA ...
## $ symb_url      : num 28 13 19 8 13 4 7 11 2 23 ...
## $ symb_protocol: num NA NA NA NA NA NA NA NA NA ...
## $ symb_host     : num 1 2 1 2 1 2 1 1 1 ...
## $ symb_path     : num 12 11 18 6 8 2 6 10 1 17 ...
## $ symb_query    : num 14 NA NA NA 3 NA NA NA NA 4 ...
## $ symb_fragment: num NA NA NA NA NA NA NA NA NA ...

params_df[, c("protocol", "protocol_1", "ldl_protocol",
             "dld_protocol", "xyx_protocol", "lett_protocol",
             "dig_protocol", "symb_protocol")] <- NULL

params_df_2[, c("protocol", "protocol_1", "ldl_protocol",
                "dld_protocol", "xyx_protocol", "lett_protocol",
                "dig_protocol", "symb_protocol")] <- NULL

fdfp <- cbind(dfp, params_df)
fdfm <- cbind(dfm, params_df_2)

cat("First DataFrame dimensions (dfdp): \n", dim(fdfp),
    "\nSecond DataFrame dimensions (dfdm):\n", dim(fdfm), "\n")

## First DataFrame dimensions (dfdp):
##  95911 46
## Second DataFrame dimensions (dfdm):
##  10000 49

```

Wizualizacja wyników analizy.

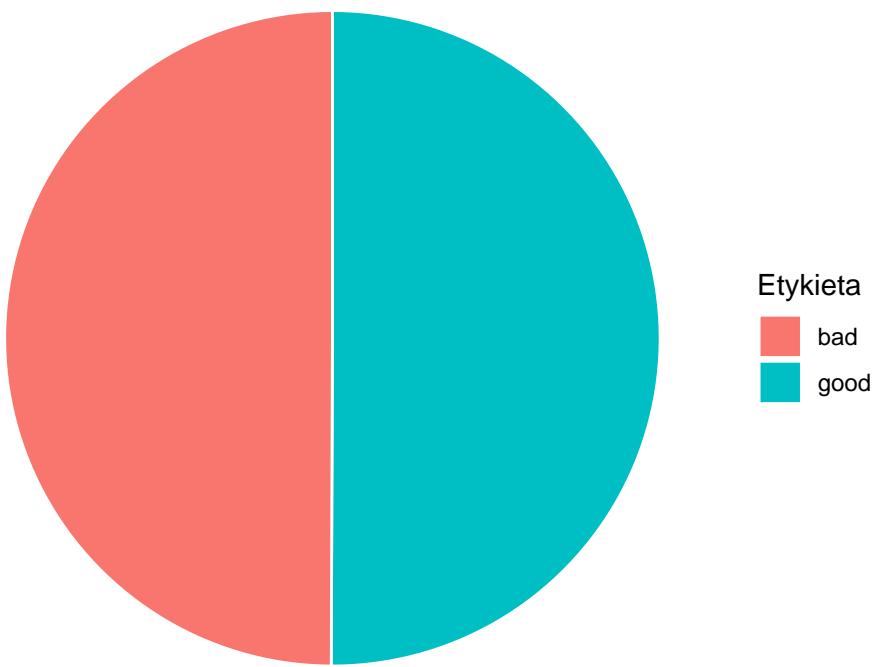
```
fdfm[fdfm$label == "bad", "color"] <- '#6dd38c'
fdfm[fdfm$label == "good", "color"] <- '#f3aca7'

fdfp[fdfp$label == "bad", "color"] <- '#6dd38c'
fdfp[fdfp$label == "good", "color"] <- '#f3aca7'

# ggsave(
#   "images/plot_1.png",
#   plot = last_plot(),
#   device = "png",
#   path = NULL,
#   scale = 1.2,
#   # width = 300,
#   # height = 300,
#   # units = "mm",
#   dpi = 400,
#   bg = "transparent"
# )

data_distribution_df <- data.frame(counts = c(sum(fdfp$label == "good"), sum(fdfp$label == "bad")),
                                      labels = c("good", "bad"))
ggplot(data_distribution_df, aes(x = "", y = counts, fill = labels)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar("y", start = 0) +
  theme_void() + ggtitle("Porównanie bezpieczeństwa domen w zbiorze danych \"PhishStorm\".") +
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12)) +
  labs(fill = "Etykieta", color = "Etykieta")
```

Porównanie bezpieczeństwa domen w zbiorze danych "PhishStorm".

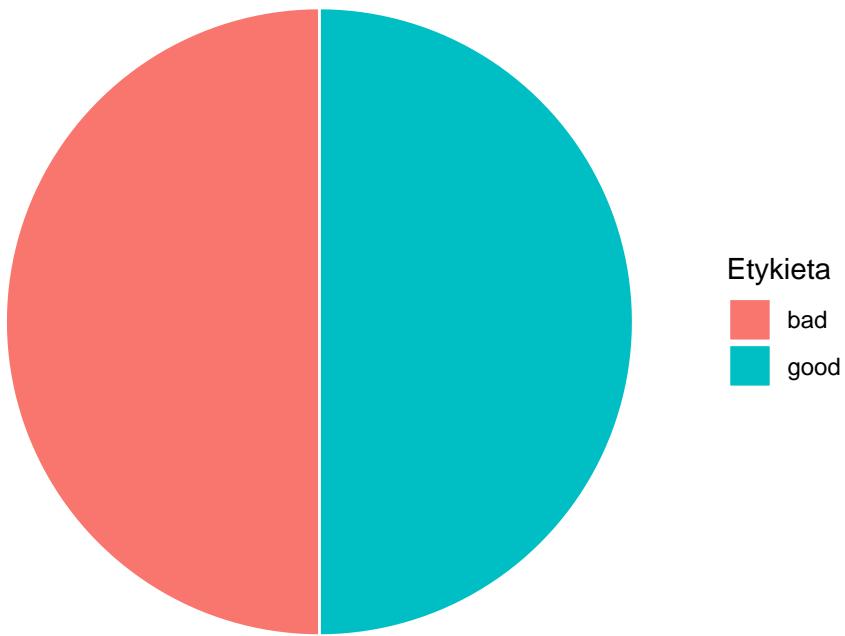


```

data_distribution_df2 <- data.frame(counts = c(sum(fdfm$label == "good"), sum(fdfm$label == "bad")),
                                     labels = c("good", "bad"))
ggplot(data_distribution_df2, aes(x = "", y = counts, fill = labels)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar("y", start = 0) +
  theme_void() +
  ggtitle("Porównanie bezpieczeństwa domen w poprzednio przygotowanym\nzbiorze danych \"Mendeley Data\"")
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12)) +
  labs(fill = "Etykieta", color = "Etykieta")

```

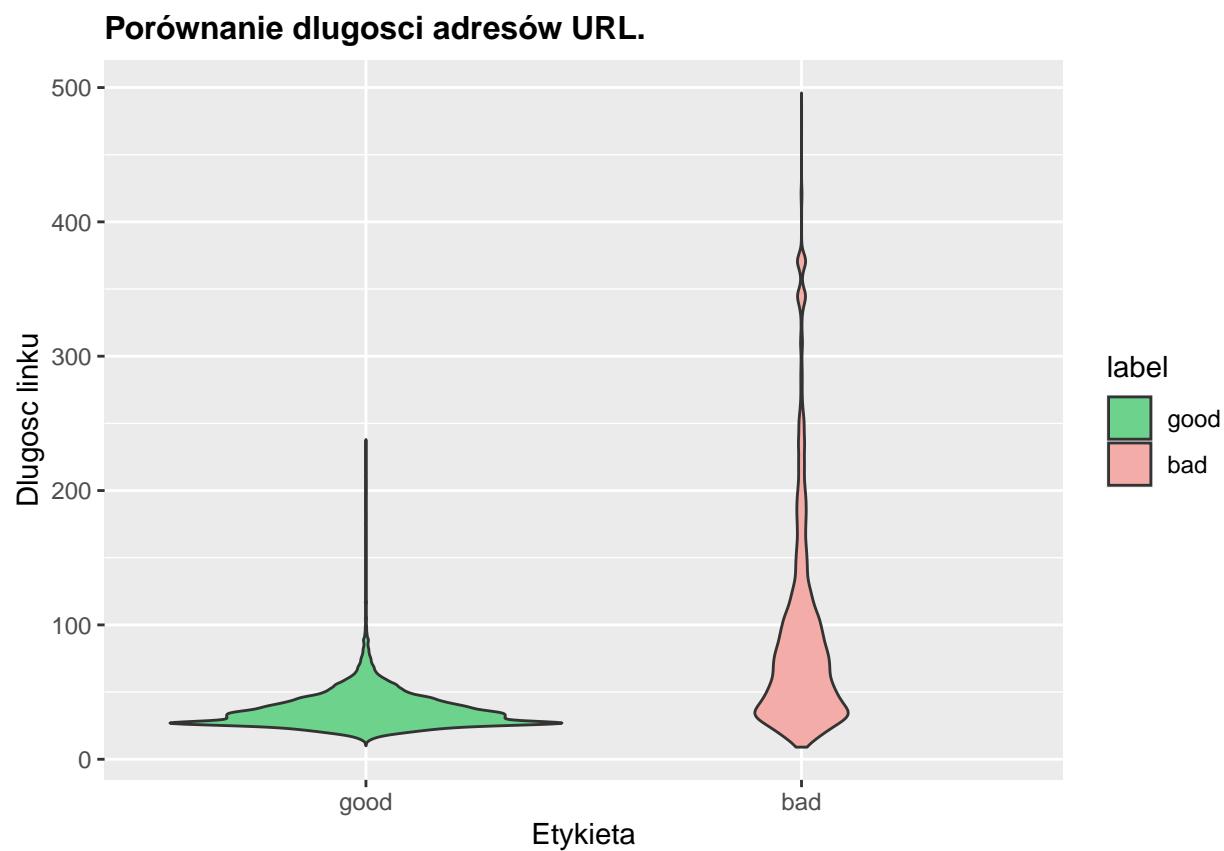
**Porównanie bezpieczeństwa domen w poprzednio przygotowanym
zbiorze danych "Mendeley Data".**



```

ggplot(fdःfp[fdःfp$url_l < 500, ], aes(x = label, y = url_l, group = label, fill = label)) +
  geom_violin() +
  ylab("Długość linku") +
  xlab("Etykieta") +
  ggtitle("Porównanie długości adresów URL.") +
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12),
        # panel.background = element_rect(fill = "#f0bc5e", colour = "black")
        # rect = element_rect(fill = "#d8d7c4")
  ) +
  scale_fill_manual(values = c("#6dd38c", "#f3aca7")) +
  labs(color = "Etykieta")

```

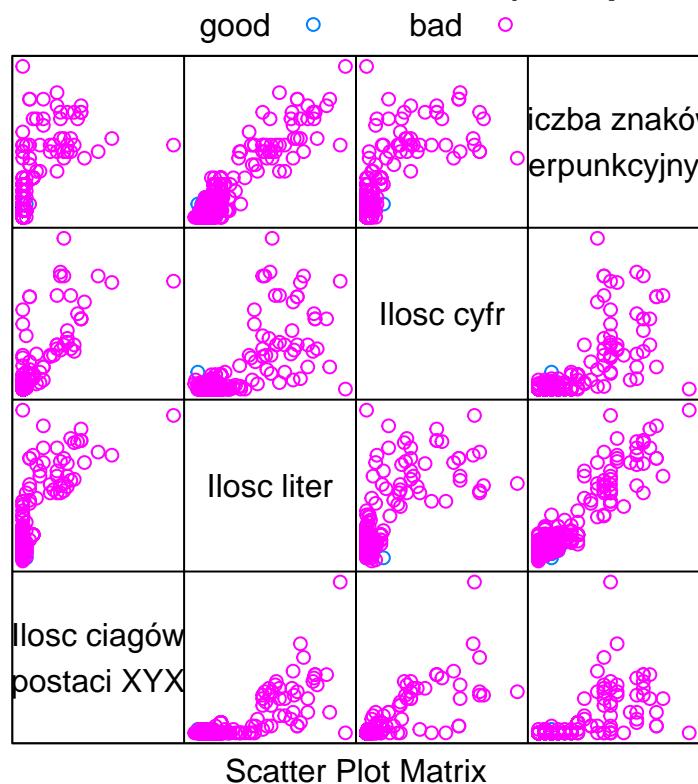


```

splom(~data.frame(xyx_host, lett_host, dig_host, symb_host),
      data = fdsp[sample(nrow(fdsp), 1000),],
      pch = 1,
      main = "Rozkład symboli w hoscie adresu URL (host part of the URL).",
      groups = label,
#      xlab = c("A", "B", "C", "D"),
#      xlab = "", # czymś takim można usunąć ten napis "Scatter Plot Matrix"
#      ylab = c("A", "B", "C", "D"),
      pscales = 0,
      auto.key = list(columns = 2),
      varnames = c("Ilość ciągów\npostaci XYX", "Ilość liter",
                  "Ilość cyfr", "Liczba znaków\nninterpunkcyjnych")
)

```

Rozkład symboli w hoscie adresu URL (host part of the URL).



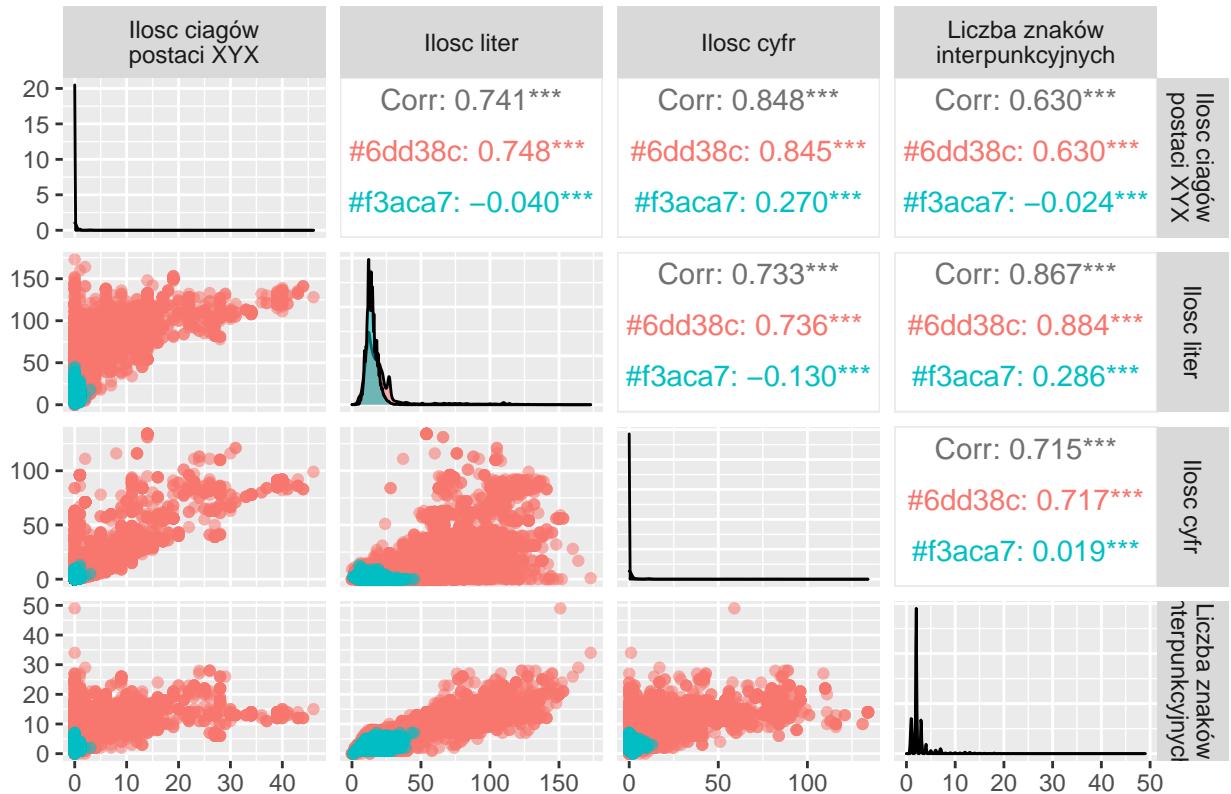
```

ggpairs(fdfp[fdfp$url_l < 500 & sample(nrow(fdfp), 500), ],
        aes(color = color,
            alpha = .5),
        columns = c("xyx_host", "lett_host", "dig_host", "symb_host"),
        columnLabels = c("Ilość ciągów\npostaci XYX",
                        "Ilość liter",
                        "Ilość cyfr",
                        "Liczba znaków\ninterpunkcyjnych")) +
  ggtitle("Rozkład symboli w hoscie adresu URL.") +
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12)
        #panel.background = element_rect(fill = "#f0bc5e", colour = "black"),
        #rect = element_rect(fill = "#d8d7c4"))
  ) +
  labs(color = "Etykieta")

```

Warning in fdfp\$url_l < 500 & sample(nrow(fdfp), 500): d^zugoœæ d^zuszego obiektu
 ## nie jest wielokrotnoœci¹ d^zugoœci krótszego obiektu

Rozkład symboli w hoscie adresu URL.

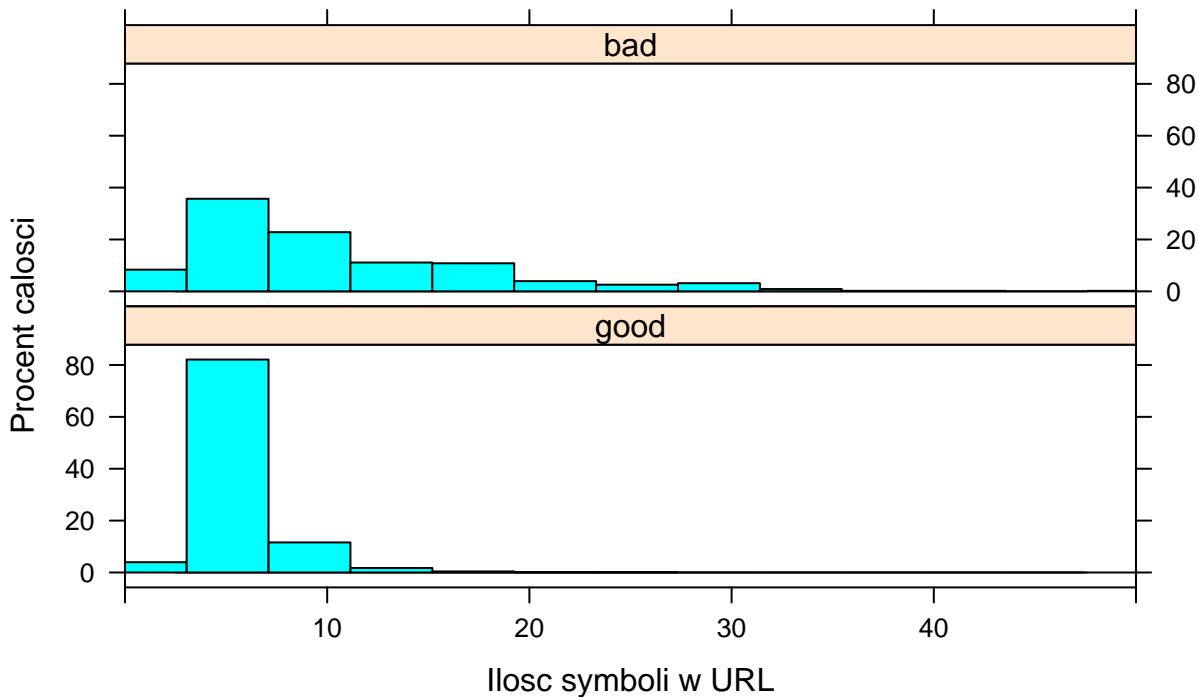


```

histogram(~ symb_url | label ,
  data = fdfp[sample(nrow(fdfp), 2000),],
  main = "Porównanie liczba znaków interpunkcyjnych\nw dobrych i złych domenach",
  xlab = "Ilość symboli w URL",
  ylab = "Procent całości",
  layout = c(1, 2),
  nint = 20,
  xlim = c(0, 50)
)

```

Porównanie liczba znaków interpunkcyjnych w dobrych i złych domenach

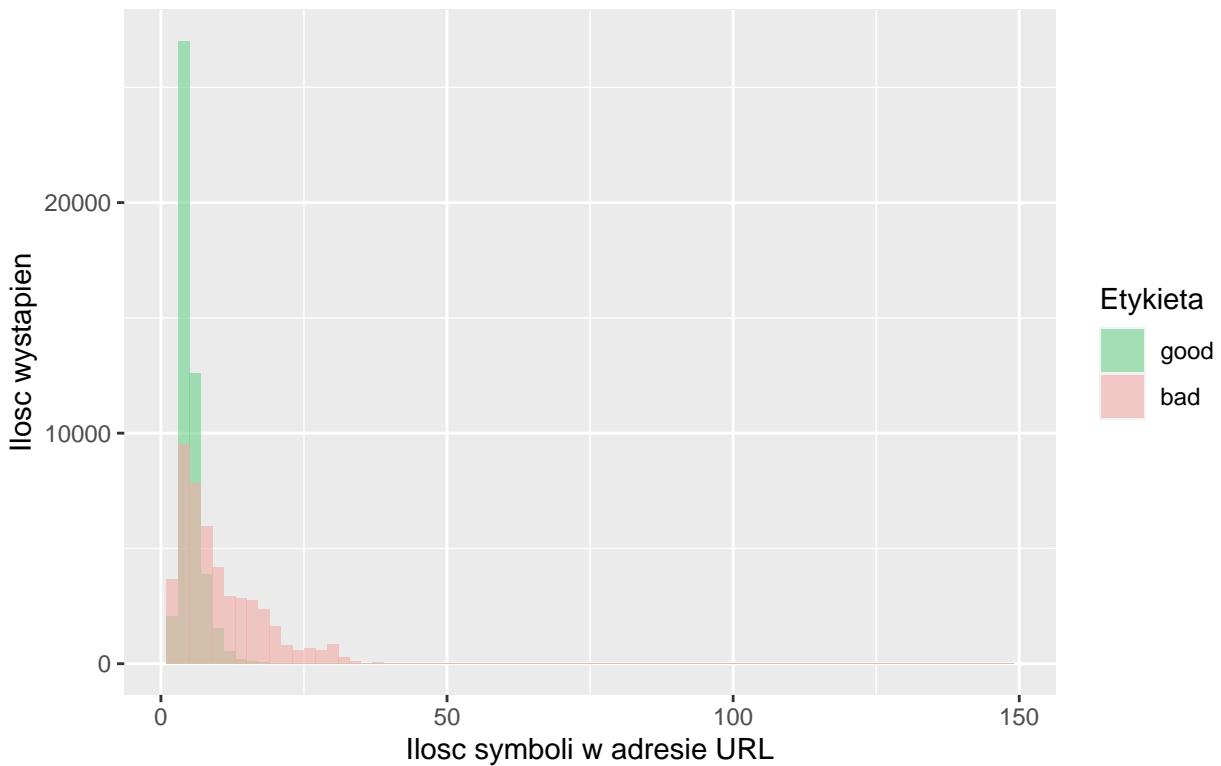


```

ggplot(fdःfp[fdःfp$url_l < 500, ], aes(x = symb_url, fill = label)) +
  geom_histogram(binwidth = 2, alpha = 0.6, position = 'identity') +
  ylab("Ilość wystąpień") +
  xlab("Ilość symboli w adresie URL") +
  ggtitle("Porównanie ilości znaków interpunkcyjnych\nw dobrych i złych domenach.") +
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12)
        # panel.background = element_rect(fill = "#f0bc5e", colour = "black"),
        # rect = element_rect(fill = "#d8d7c4")
    ) +
  scale_fill_manual(values = c("#6dd38c", "#f3aca7")) +
  labs(fill = "Etykieta", color = "Etykieta")

```

Porównanie ilości znaków interpunkcyjnych w dobrych i złych domenach.



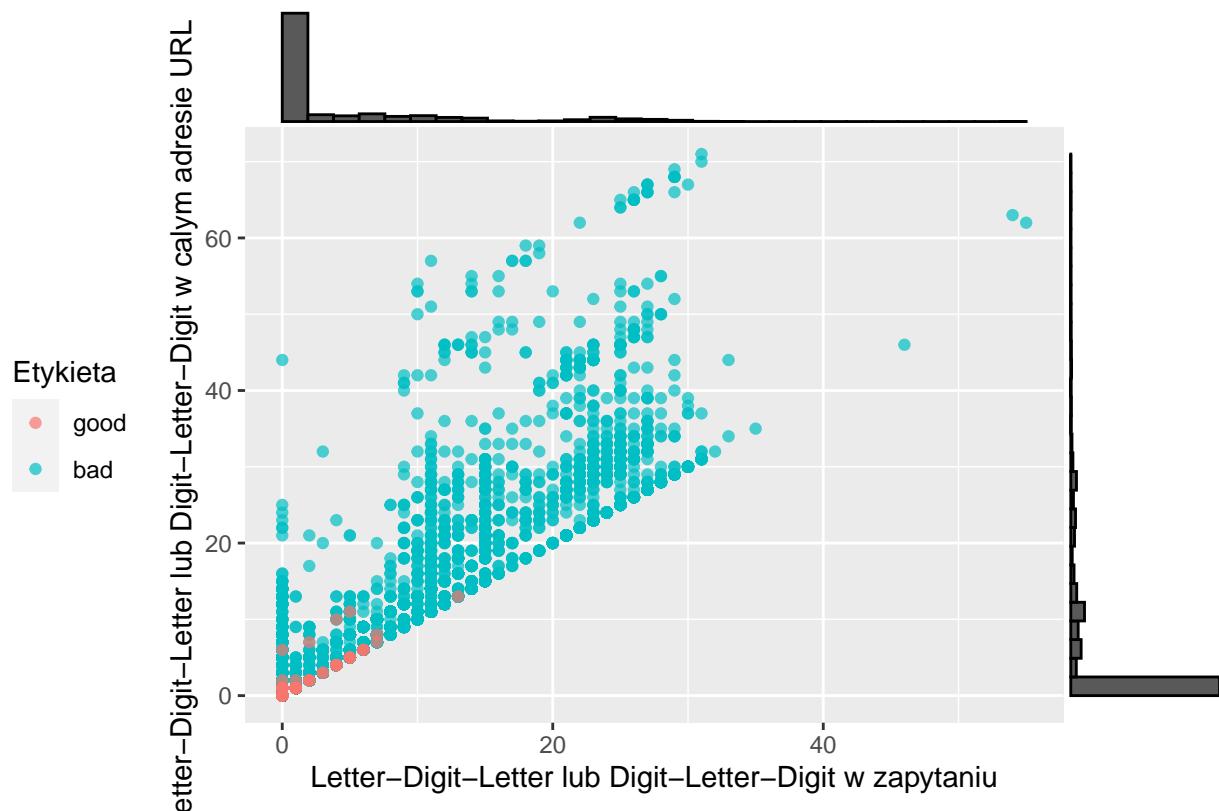
```

p <- ggplot(fdःfp[fdःfp$url_l < 1000, ], aes(x = xyx_query, y = xyx_url, color = label)) +
  geom_point(alpha = .7, na.rm = TRUE) +
  theme(legend.position = "left") +
  ylab("Letter-Digit-Letter lub Digit-Letter-Digit w całym adresie URL") +
  xlab("Letter-Digit-Letter lub Digit-Letter-Digit w zapytaniu") +
  ggtitle("Porównanie podejrzanych ciągów w adresach URL.") +
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12)) +
  labs(color = "Etykieta")

ggMarginal(p, type = "histogram")

```

Porównanie podejrzanych ciągów w adresach URL.

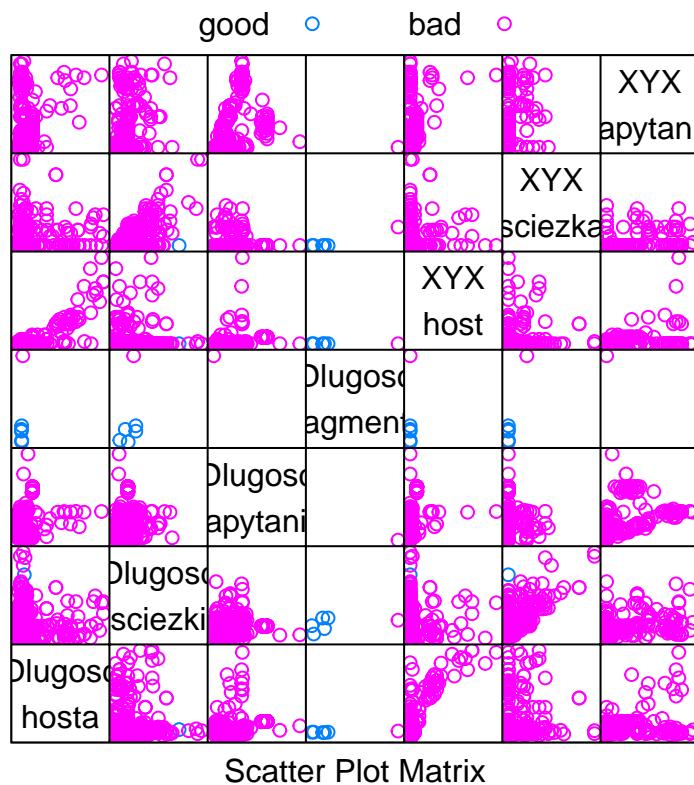


```

splom(~data.frame(host_1, path_1, query_1, fragment_1, xyx_host, xyx_path, xyx_query),
      data = fdःp[sample(nrow(fdःp), 2000),],
      pch = 1,
      main = "Rozkład znaków w domenach.",
      groups = label,
      pscales = 0,
      auto.key = list(columns = 2),
      varnames = c("Długość\nhosta", "Długość\nścieżki", "Długość\nzapytania",
                  "Długość\nfragmentu", "XYX\nhost", "XYX\nścieżka",
                  "XYX\nzapytanie")
)

```

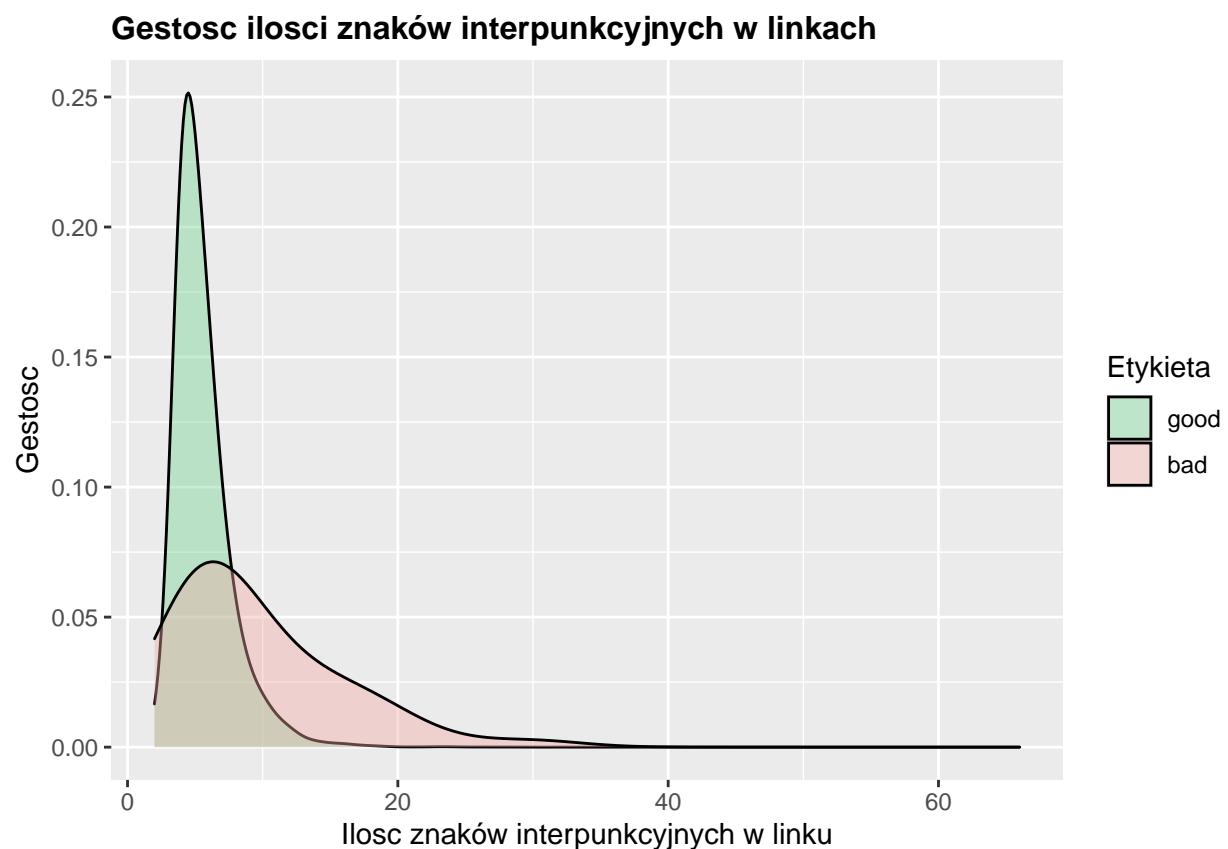
Rozkład znaków w domenach.



```

ggplot(data = fdfp[fdfp$url_l < 200, ], aes(x = symb_url, group = label, fill = label)) +
  geom_density(adjust = 5, alpha = .4) +
  ylab("Gęstość") +
  xlab("Ilość znaków interpunkcyjnych w linku") +
  ggtitle("Gęstość ilości znaków interpunkcyjnych w linkach") +
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12),
        #panel.background = element_rect(fill = "#f0bc5e", colour = "black"),
        #rect = element_rect(fill = "#d8d7c4")
  ) +
  scale_fill_manual(values = c("#6dd38c", "#f3aca7")) +
  labs(fill = "Etykieta", color = "Etykieta")

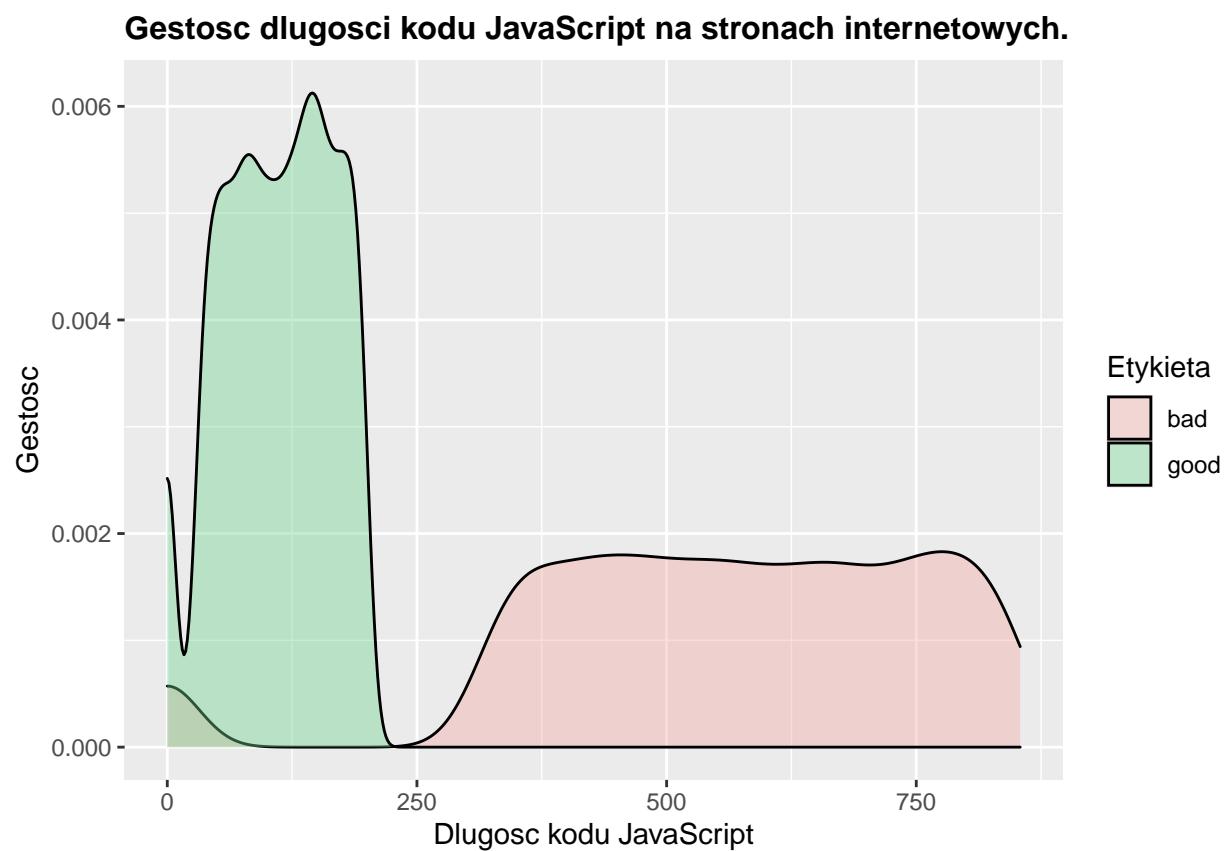
```



```

ggplot(data = fdfm, aes(x = js_len, group = label, fill = label)) +
  geom_density(adjust = 1, alpha = .4) +
  ylab("Gęstość") +
  xlab("Długość kodu JavaScript") +
  ggtitle("Gęstość długości kodu JavaScript na stronach internetowych.") +
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12),
        #panel.background = element_rect(fill = "#f0bc5e", colour = "black"),
        #rect = element_rect(fill = "#d8d7c4")
  ) +
  scale_fill_manual(values = c("#f3aca7", "#6dd38c")) +
  labs(fill = "Etykieta", color = "Etykieta")

```

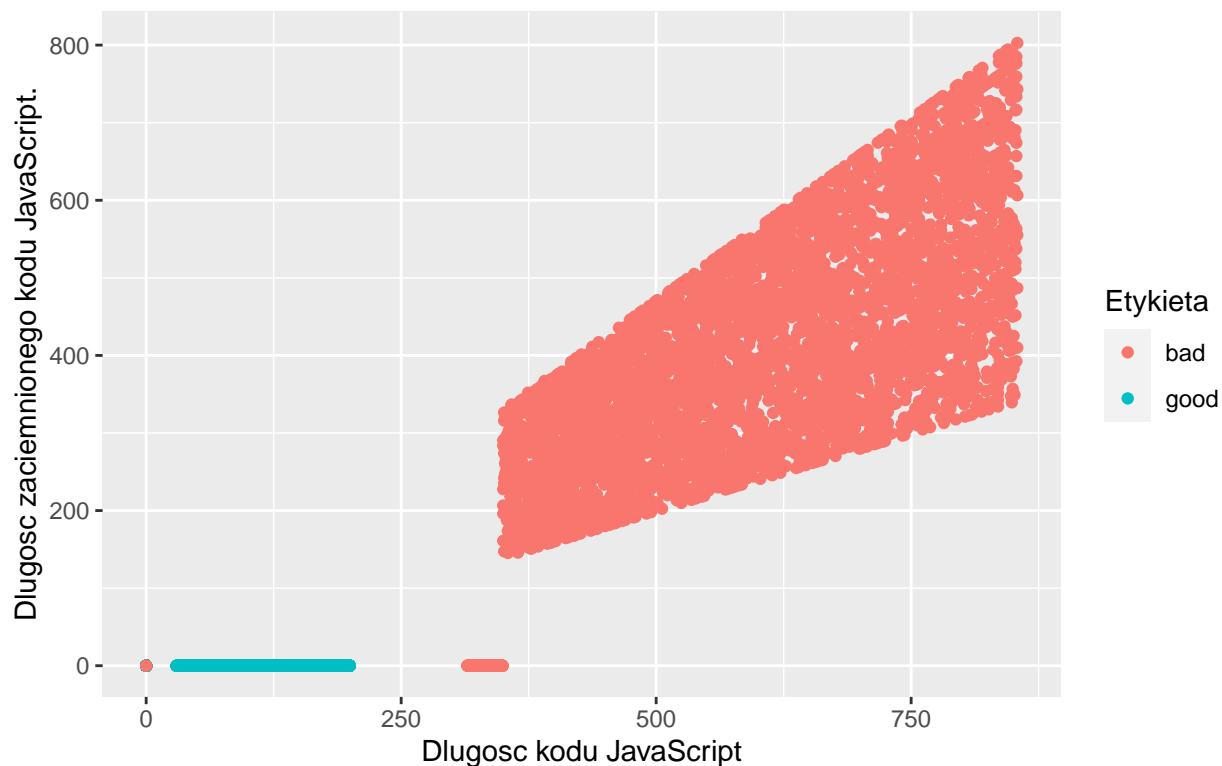


```

ggplot(data = fdfm, aes(x = js_len, y = js_obf_len, color = label) ) +
  geom_point() +
  ylab("Długość zaciemnionego kodu JavaScript.") +
  xlab("Długość kodu JavaScript") +
  ggtitle("Związek pomiędzy długością kodu JavaScript\\na bezpieczeństwem domen") +
  theme(plot.title = element_text(family = "",
                                    face = 'bold',
                                    colour = 'black',
                                    size = 12)
        #panel.background = element_rect(fill = "#f0bc5e", colour = "black"),
        #rect = element_rect(fill = "#d8d7c4")
    ) +
  scale_fill_manual(values = c("#6dd38c", "#f3aca7")) +
  labs(fill = "Etykieta", color = "Etykieta")

```

**Związek pomiędzy długoscia kodu JavaScript
a bezpiecznoscia domen**



Jakieś wnioski przydałoby się dodać