DataAnalysisLab2

Joy Chowdhury

February 3, 2017

R

Listing 3-0

Listing 3-0 demonstrates how the working directory can be set using R and how a vector of package names can be compared against installed packages to determine new packages to be installed, saving bandwidth and time. The packages are used to create graphs and other visuals.

```
setwd("~/Class Notes and Assignments/SRT411/SRT411-DataAnalysisLab-2/")
pkg <- c("ggplot2", "scales", "maptools", "sp", "maps", "grid", "car")
new.pkg <- pkg[!(pkg %in% installed.packages())]
if (length(new.pkg)) {
  install.packages(new.pkg)
}
```

Listing 3-2

Listing 3-2 uses an if statement to check whether a database exists, and then decides if it will use the download.file function to download a reputation database from the datadrivensecurity website to be saved in a subrepository of the working directory called data, as a file called reputation.data.

```
avURL <- "http://datadrivensecurity.info/book/ch03/data/reputation.data"
avRep <- "data/reputation.data"
if (file.access(avRep)) {
  download.file(avURL, avRep)
}
```

Listing 3-4

Listing 3-4 converts the # separated database into a dataframe using R, providing headers to the generated columns with the function colnames() and a vector of header strings.

```
av <- read.csv(avRep, sep = "#", header = FALSE)
colnames(av) <- c("IP", "Reliability", "Risk", "Type", "Country", "Locale", "Coord", "X")
str(av)
```

```
## 'data.frame':    258626 obs. of  8 variables:
##  $ IP         : Factor w/ 258626 levels "1.0.232.167",..: 154069 154065 154066 171110 64223 197880 1
##  $ Reliability: int  4 4 4 6 4 4 4 4 4 6 ...
##  $ Risk       : int  2 2 2 3 5 2 2 2 2 3 ...
##  $ Type       : Factor w/ 34 levels "APT;Malware Domain",..: 25 25 25 31 25 25 25 25 25 31 ...
##  $ Country    : Factor w/ 153 levels "","A1","A2","AE",..: 34 34 34 143 141 143 34 34 34 1 ...
##  $ Locale     : Factor w/ 2573 levels "","Aachen","Aarhus",..: 2506 2506 2506 1 1374 2342 2506 2506 2
##  $ Coord      : Factor w/ 3140 levels "-0.139500007033,98.1859970093",..: 489 489 489 1426 2676 1384
##  $ X          : Factor w/ 34 levels "1;6","11","11;12",..: 2 2 2 8 2 2 2 2 2 8 ...
```

```r
head(av)
```

```
##                IP Reliability Risk          Type Country     Locale
## 1 222.76.212.189           4    2 Scanning Host      CN     Xiamen
## 2 222.76.212.185           4    2 Scanning Host      CN     Xiamen
## 3 222.76.212.186           4    2 Scanning Host      CN     Xiamen
## 4    5.34.246.67           6    3      Spamming      US
## 5  178.94.97.176           4    5 Scanning Host      UA     Merefa
## 6    66.2.49.232           4    2 Scanning Host      US Union City
##                          Coord  X
## 1   24.4797992706,118.08190155 11
## 2   24.4797992706,118.08190155 11
## 3   24.4797992706,118.08190155 11
## 4                  38.0,-97.0 12
## 5  49.8230018616,36.0507011414 11
## 6 37.5962982178,-122.065696716 11
```

Listing 3-7

Listing 3-7 displays the 5 number summary developed by Tukey. It is used to determine the range(min and max), and the first and third percentiles, along with the median and mean, of each specified column..

```r
summary(av$Reliability)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.798   4.000  10.000
```

```r
summary(av$Risk)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.221   2.000   7.000
```

Listing 3-9

Listing 3-9 demonstrates how the table() function in R can count values of quantitative variables for a column in a dataframe. Essentially, categorical data is aggregated and the count of each unique data is displayed. It also shows the difference betwen table() and summary(). Summary organizes the malware qualitative data by aggregating each unique string and counting the number of times they appear

```r
table(av$Reliability)
```

```
##
##      1      2      3      4      5      6      7      8      9     10
##   5612 149117  10892  87040      7   4758    297     21    686    196
```

```r
table(av$Risk)
```

```
##
##      1      2      3      4      5      6      7
##     39 213852  33719   9588   1328     90     10
```

```r
summary(av$Type, maxsum=10)
```

```
##                Scanning Host                Malware Domain
##                       234180                          9274
##                   Malware IP                Malicious Host
```

```
##                          6470                            3770
##                      Spamming                             C&C
##                          3487                             610
## Scanning Host;Malicious Host    Malware Domain;Malware IP
##                           215                             173
## Malicious Host;Scanning Host                         (Other)
##                           163                             284
```

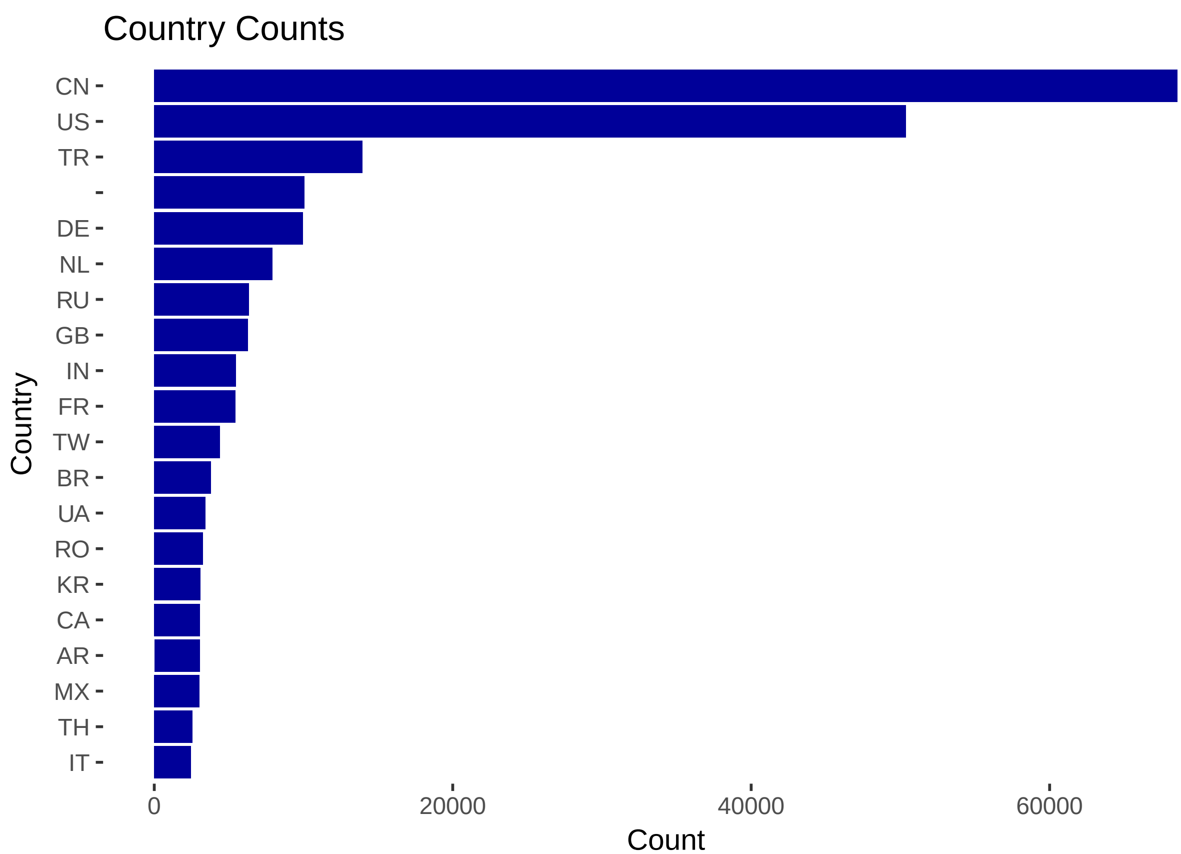
```r
summary(av$Country, maxsum=10)
```

```
##      CN      US      TR              DE      NL      RU      GB      IN
##   68583   50387   13958   10055    9953    7931    6346    6293    5480
## (Other)
##   79640
```

Listing 3-11

Listing 3-11 demonstrates the capabilities of ggplot2 library by creating a bar graph of the Country statistics in the dataset.

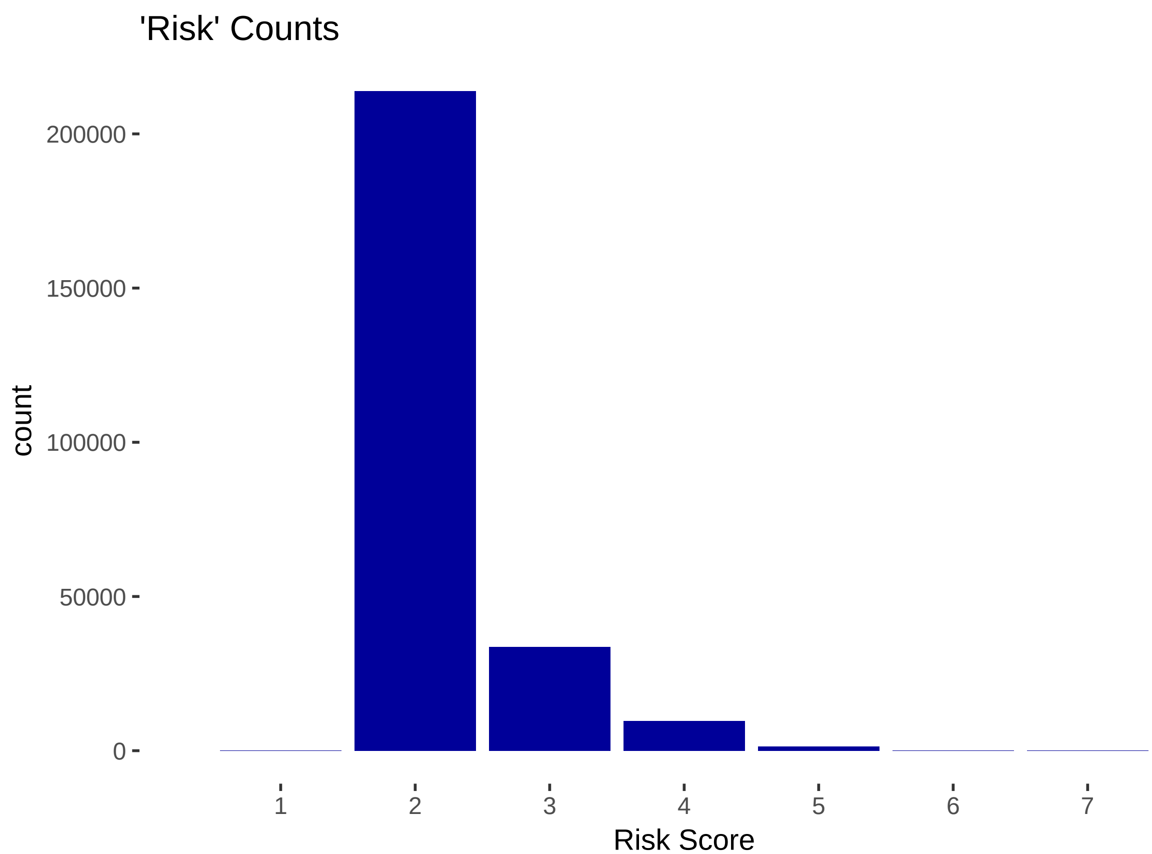
```r
library(ggplot2)
country.top20 <- names(summary(av$Country))[1:20]
gg <- ggplot(data=subset(av, Country %in% country.top20),
             aes(x=reorder(Country, Country, length)))
gg <- gg + geom_bar(fill="#000099")
gg <- gg + labs(title="Country Counts",
                x="Country",
                y="Count")
gg <- gg + coord_flip()
gg <- gg + theme(panel.grid=element_blank(),
                 panel.background=element_blank())
print(gg)
```



Listing 3-12

Shows how the ggplot2 library can be used to create a bar graph of the number of each type of Categorical data in the Risk factor.

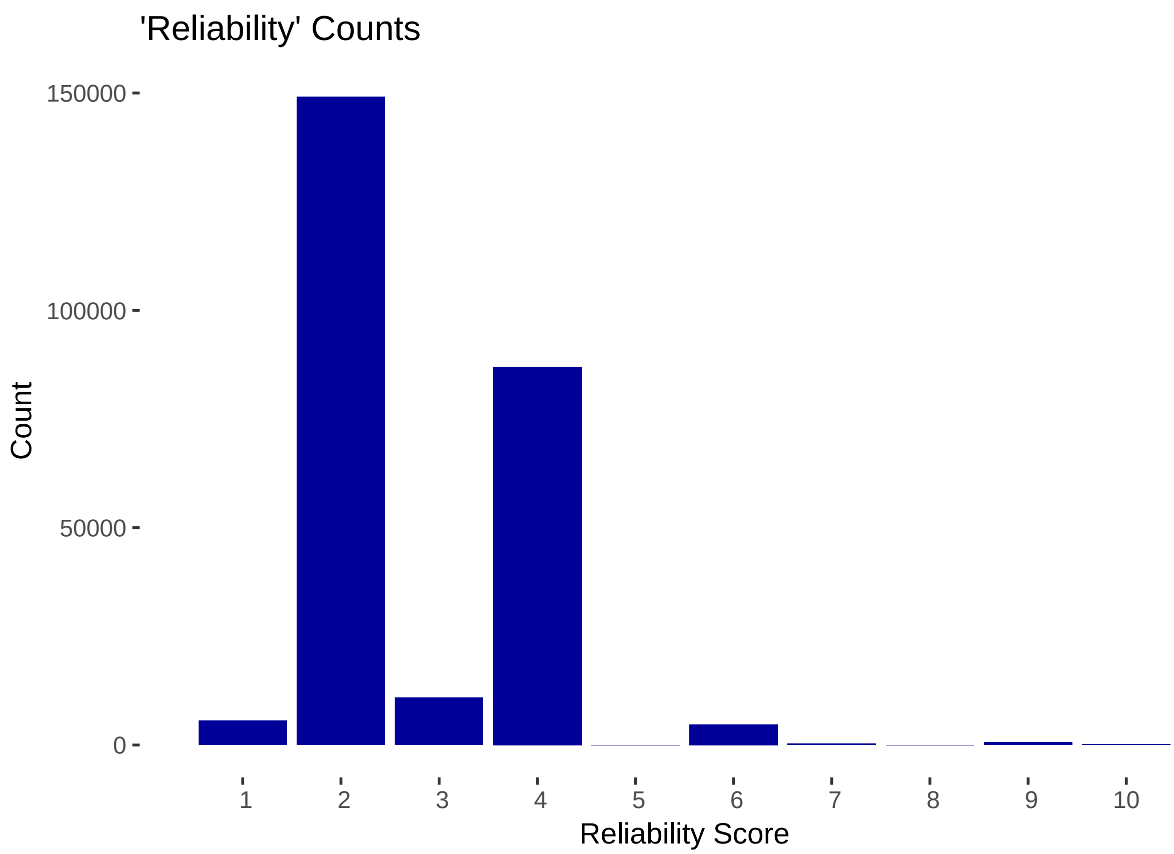
```
gg <- ggplot(data=av, aes(x=Risk))
gg <- gg + geom_bar(fill="#000099")
gg <- gg + scale_x_discrete(limits=seq(max(av$Risk)))
gg <- gg + labs(title="'Risk' Counts", x="Risk Score", y="count")
gg <- gg + theme(panel.grid=element_blank(), panel.background=element_blank())
print(gg)
```



Listing 3-13

Shows how the ggplot2 library can be used to create a bar graph of the number of each type of Categorical data in the Reliability factor.

```
gg <- ggplot(data=av, aes(x=Reliability))
gg <- gg + geom_bar(fill="#000099")
gg <- gg + scale_x_discrete(limits=seq(max(av$Reliability)))
gg <- gg + labs(title="'Reliability' Counts",
                x="Reliability Score",
                y="Count")
gg <- gg + theme(panel.grid=element_blank(),
                 panel.background=element_blank())
print(gg)
```



Listing 3-17

TO look at the percentage of total malicious nodes contributed by the first 10 countries in the list, we divide each value by the number of rows in the dataframe.

```
country10 <- summary(av$Country, maxsum=10)
country10.perc10 <- country10/nrow(av)
print(country10.perc10)
```

```
##         CN         US         TR                       DE         NL
## 0.26518215 0.19482573 0.05396983 0.03887854 0.03848414 0.03066590
##         RU         GB         IN    (Other)
## 0.02453736 0.02433243 0.02118890 0.30793501
```

Listing 3-19

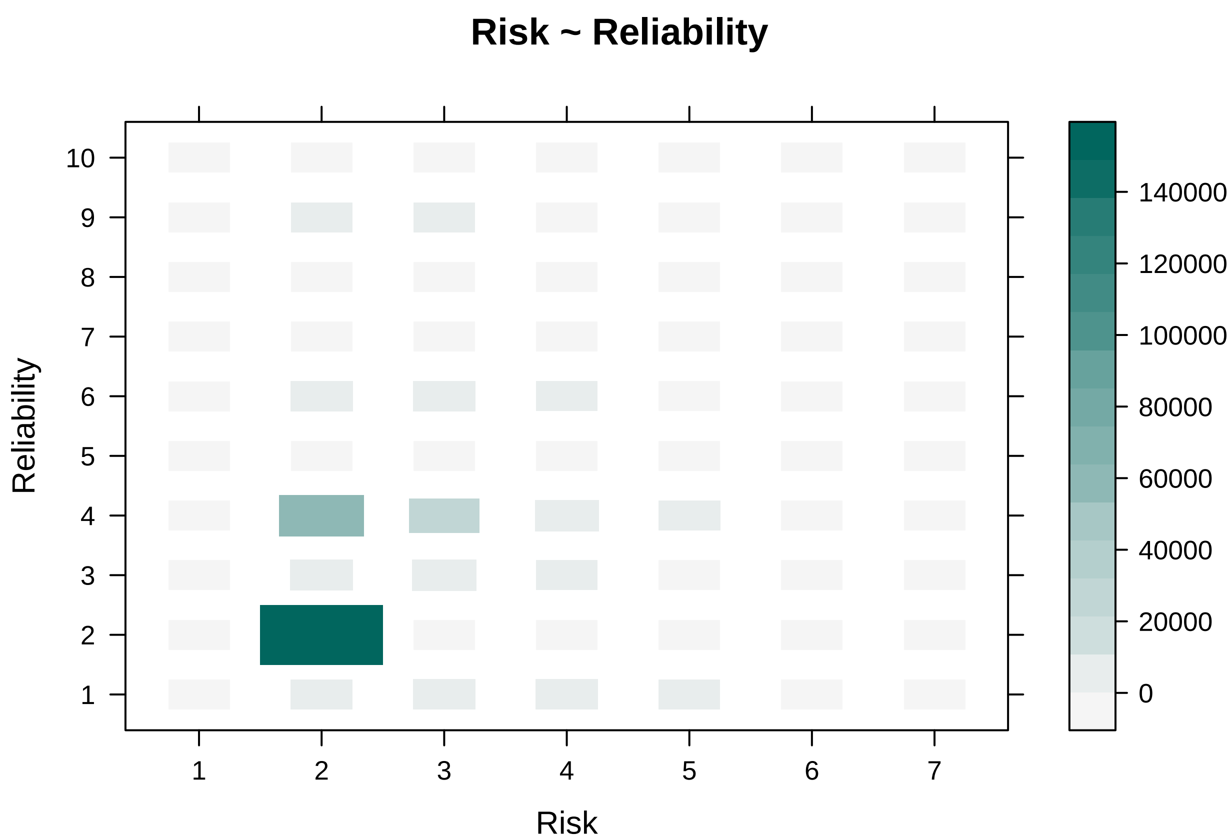
A contingency table, which is a tabular view of the relationships between two variables, is used to determine which nodes to pay attention to when doing data-driven security analysis. The xtabs is used to generate a matrix which represents quantity using size and colour. This shows around where in the relationship the values in the dataset bias are concentrated.

```
rr.tab <- xtabs(~Risk+Reliability, data=av)
ftable(rr.tab)
```

```
##      Reliability      1      2      3      4      5      6      7      8      9     10
## Risk
```

```
## 1                    0      0     16     7      0     8     8      0      0      0
## 2                  804 149114   3670 57653      4  2084    85     11    345     82
## 3                 2225      3   6668 22168      2  2151   156      7    260     79
## 4                 2129      0    481  6447      0   404    43      2     58     24
## 5                  432      0     55   700      1   103     5      1     20     11
## 6                   19      0      2    60      0     8     0      0      1      0
## 7                    3      0      0     5      0     0     0      0      2      0
```

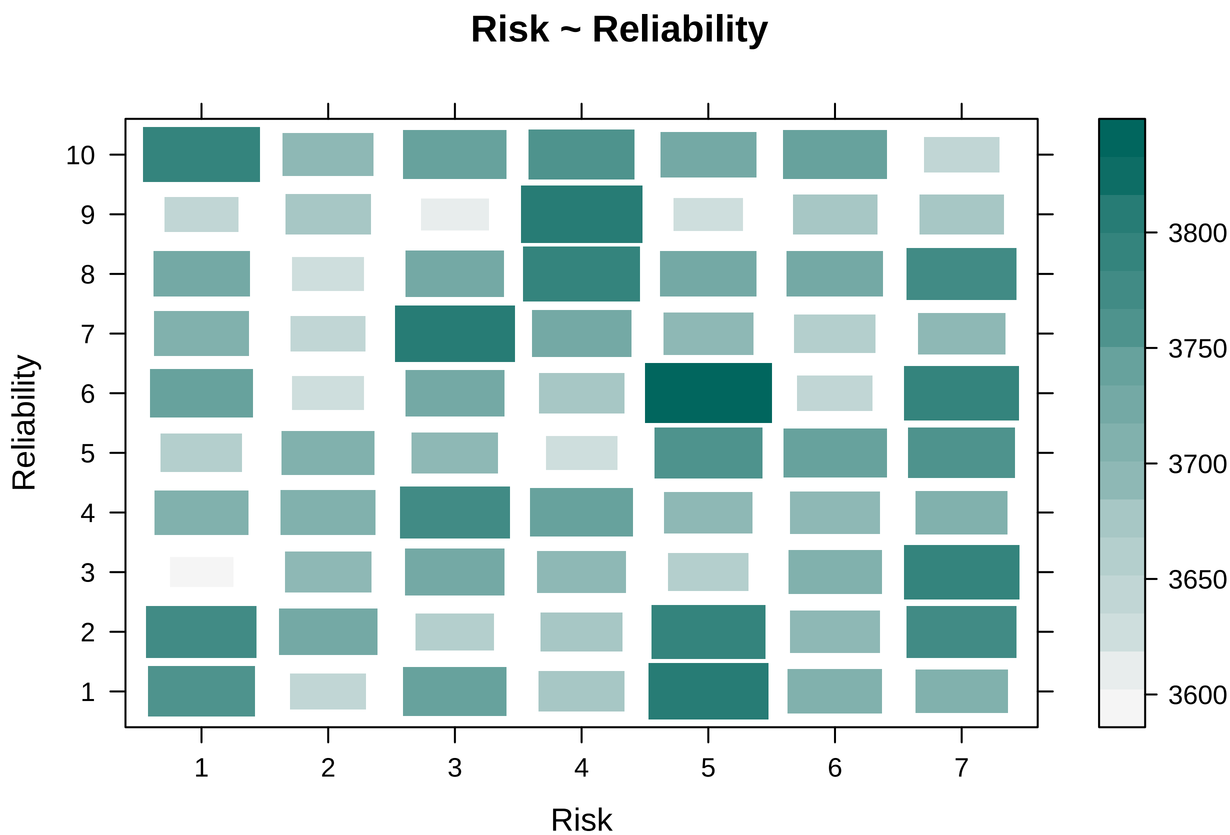
```r
library(lattice)
rr.df <- data.frame(table(av$Risk, av$Reliability))
colnames(rr.df) <- c("Risk", "Reliability", "Freq")
levelplot(Freq~Risk*Reliability,
          data=rr.df, main="Risk ~ Reliability",
          ylab="Reliability",
          xlab="Risk",
          shrink=c(0.5,1),
          col.regions=colorRampPalette(c("#F5F5F5", "#01665E"))(20))
```



Listing 3-21

Produces a matrix representing quantity with size and colour (levelplot) using random samples generated using random samples from the Risk (1:10) and Reliability (1:7) category (realization of the random process). The randomness implies that it is unbiased, however, the process of selecting random samples may introduce its own bias, so multiple runs of the sample() function should be executed. This visual is used to evaluate whether the real world data is due to chance or if there is meaning to the data.

```r
set.seed(1492)
rel <- sample(1:7, 260000, replace=T)
rsk <- sample(1:10, 260000, replace=T)
tmp.df <- data.frame(table(factor(rsk), factor(rel)))
colnames(tmp.df) <- c("Risk", "Reliability", "Freq")
levelplot(Freq~Reliability*Risk,
          data=tmp.df,
          main="Risk ~ Reliability",
          ylab="Reliability",
          xlab="Risk",
          shrink=c(0.5, 1),
          col.regions=colorRampPalette(c("#F5F5F5", "#01665E"))(20))
```

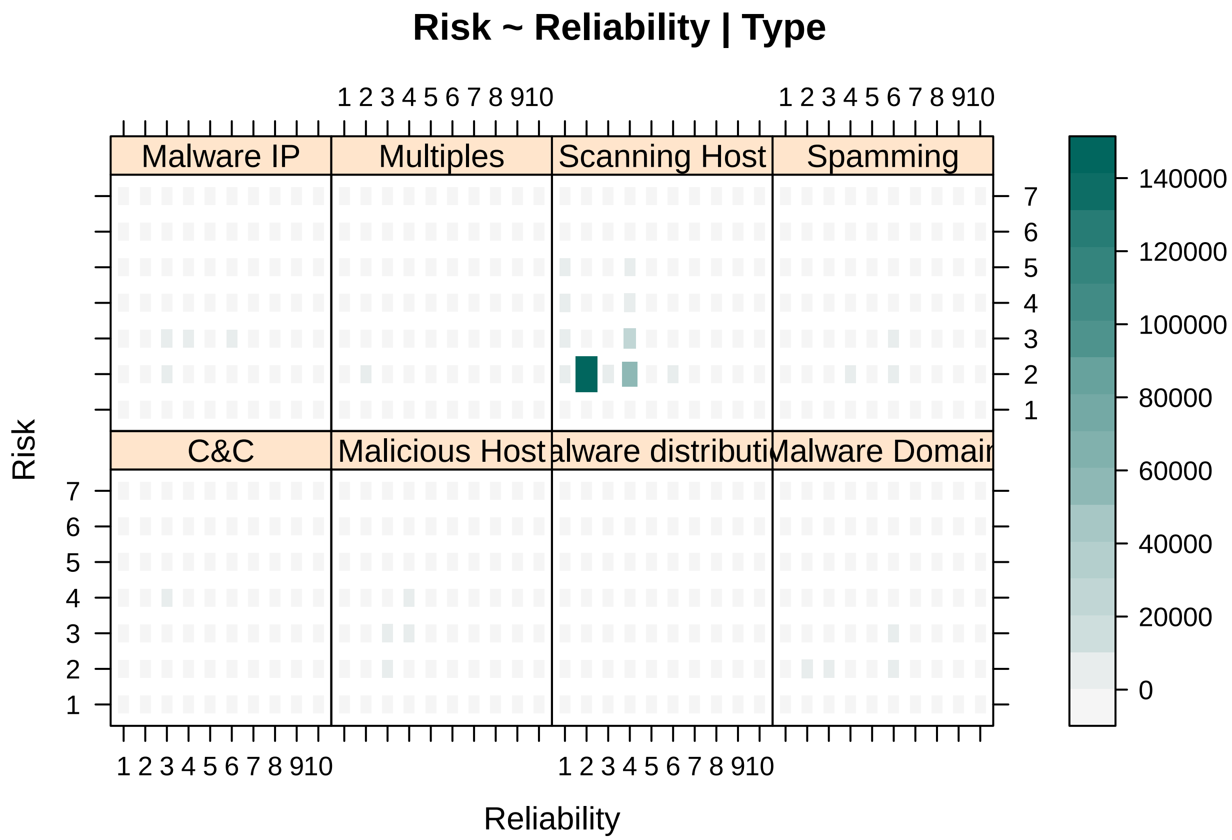


Listing 3-22

Compares each type of host to their Risk-Reliability measurement by creating a three-way contingency table. Since Type can also be multiple types, the values are parsed so that those with the ';' character, indicating multiple types, is given their own category: "Multiples".

```r
av$simpletype <- as.character(av$Type)
av$simpletype[grep(';', av$simpletype)] <- "Multiples"
av$simpletype <- factor(av$simpletype)
rrt.df <- data.frame(table(av$Risk, av$Reliability, av$simpletype))
colnames(rrt.df) <- c("Risk", "Reliability", "simpletype", "Freq")
levelplot(Freq ~ Reliability * Risk | simpletype,
```

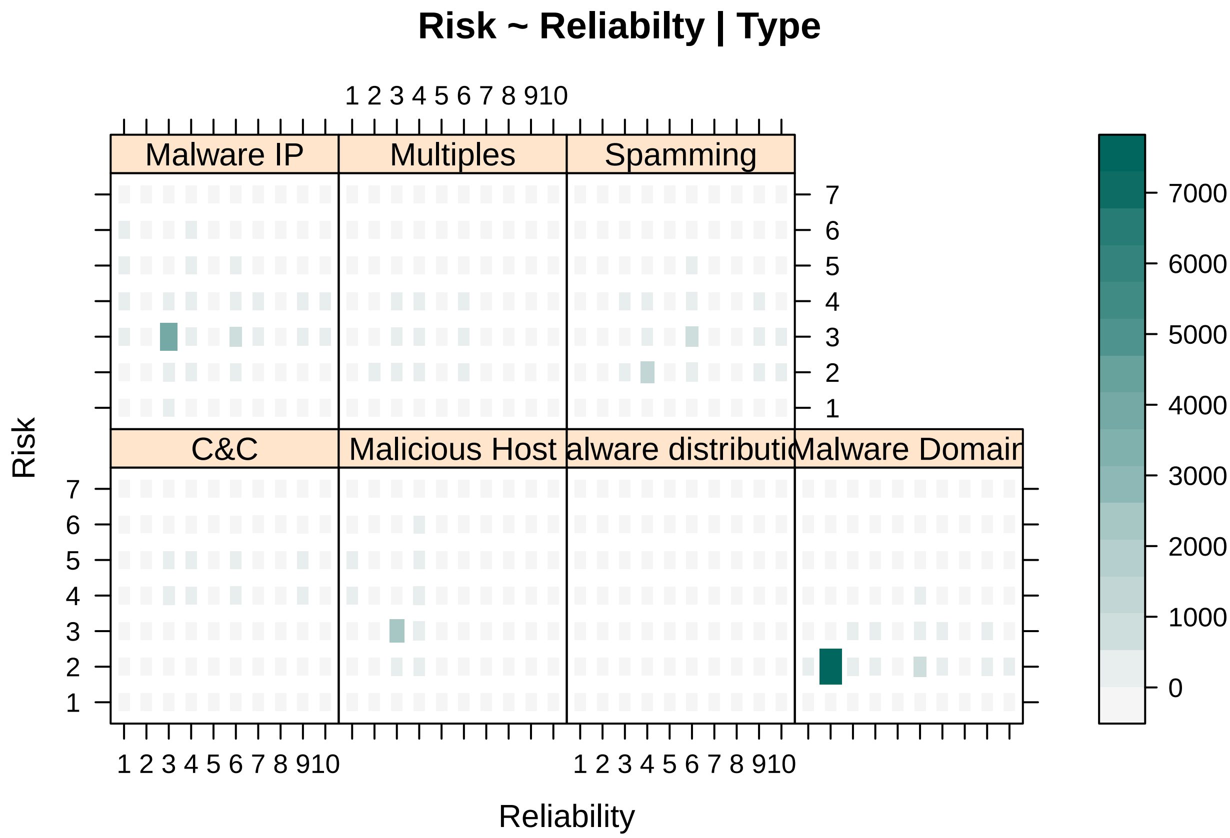
```
          data=rrt.df,
          main="Risk ~ Reliability | Type",
          ylab="Risk",
          xlab="Reliability",
          shrink=c(0.5, 1),
          col.regions=colorRampPalette(c("#F5F5F5", "#01665E"))(20))
```



Listing 3-24

Omits the Scanning Host category from the three-way contingency table because the majority of entries are in that category and are generally low risk and reliability.

```
rrt.df <- subset(rrt.df, simpletype != "Scanning Host")
levelplot(Freq ~ Reliability*Risk|simpletype,
          data =rrt.df,
          main="Risk ~ Reliabilty | Type",
          ylab = "Risk",
          xlab = "Reliability", shrink = c(0.5, 1),
          col.regions = colorRampPalette(c("#F5F5F5","#01665E"))(20))
```



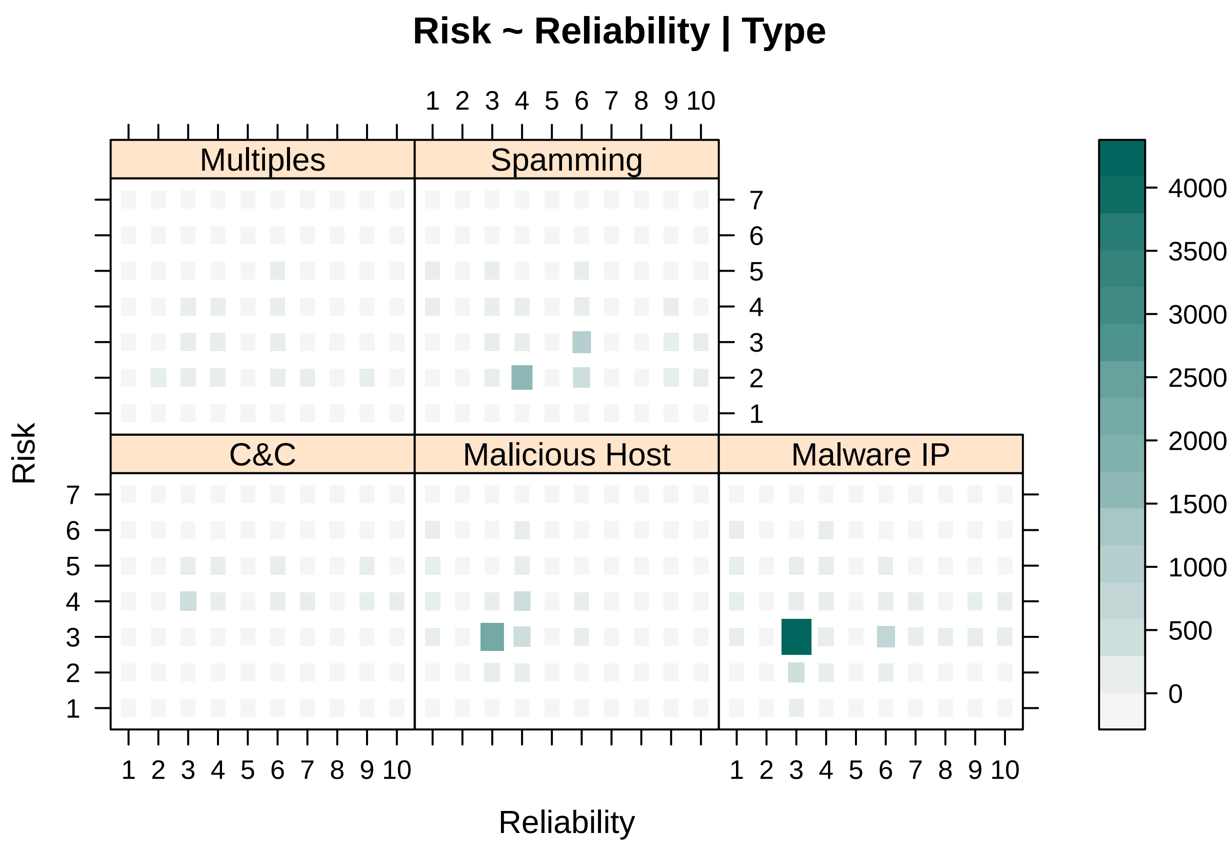
Listing 3-26

Filters out Malware Domain from the three-way contingency graph since the majority is a risk and reliability around 2 and 3. Also filters out Malware distribution since it does not seem to contribute any risk.

```
rrt.df <- subset(rrt.df, !(simpletype %in% c("Malware distribution", "Malware Domain")))
sprintf("Count: %d; Percent: %2.1f%%",
        sum(rrt.df$Freq),
        100*sum(rrt.df$Freq)/nrow(av))
```

```
## [1] "Count: 15171; Percent: 5.9%"
```

```
levelplot(Freq ~ Reliability * Risk | simpletype, data=rrt.df,
          main="Risk ~ Reliability | Type", ylab = "Risk",
          xlab="Reliability",
          shrink=c(0.5, 1),
          col.regions=colorRampPalette(c("#F5F5F5", "#01665E"))(20))
```


Risk ~ Reliability | Type
Multiples
Spamming
C&C
Malicious Host
Malware IP
Risk
Reliability
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7
0
500
1000
1500
2000
2500
3000
3500
4000

SRT411DataAnalysisLab2-Python

March 8, 2017

1 Python

1.1 Listing 3-1

Listing 3-1 demonstrates how the os library can be imported to a python script to use functions like chdir and path.expanduser to set the working directory. In python, libraries for graphics are pandas and numpy.

```
In [1]: %matplotlib inline
        import os
        os.chdir(os.path.expanduser("~") + "/Documents/Class Notes and Assignments/
```

1.2 Listing 3-3

Listing 3-3 shows how the os and urllib library can be imported into a python code to retrieve the database and save it in a similar fashion, if it does not already exist in the data repository.

```
In [2]: import urllib
        import os.path

        avURL = "http://datadrivensecurity.info/book/ch03/data/reputation.data"
        avRep = "data/reputation.data"
        if not os.path.isfile(avRep) :
            urllib.urlretrieve(avURL, filename = avRep)
```

1.3 Listing 3-5

Listing 3-5 uses the pandas library to convert the # separated values into a data frame.

```
In [3]: import pandas as pd
        import sys
        av = pd.read_csv(avRep,sep="#")

        av.columns = ["IP","Reliability","Risk","Type","Country","Locale","Coord","
        print(av)
```

```
                    IP  Reliability  Risk            Type Country  \
0        222.76.212.185            4     2   Scanning Host      CN
```

1	222.76.212.186	4	2	Scanning Host	CN
2	5.34.246.67	6	3	Spamming	US
3	178.94.97.176	4	5	Scanning Host	UA
4	66.2.49.232	4	2	Scanning Host	US
5	222.76.212.173	4	2	Scanning Host	CN
6	222.76.212.172	4	2	Scanning Host	CN
7	222.76.212.171	4	2	Scanning Host	CN
8	174.142.46.19	6	3	Spamming	NaN
9	66.2.49.244	4	2	Scanning Host	US
10	62.75.130.16	4	2	Scanning Host	DE
11	62.75.130.17	4	2	Scanning Host	DE
12	62.75.130.18	4	2	Scanning Host	DE
13	62.75.130.19	4	2	Scanning Host	DE
14	112.216.121.87	4	3	Scanning Host	KR
15	112.216.121.78	4	3	Scanning Host	KR
16	112.216.121.77	4	3	Scanning Host	KR
17	112.216.121.75	4	3	Scanning Host	KR
18	112.216.121.74	4	3	Scanning Host	KR
19	222.45.58.249	4	2	Scanning Host	CN
20	222.45.58.244	4	2	Scanning Host	CN
21	120.31.136.119	4	2	Scanning Host	CN
22	201.57.0.248	4	2	Scanning Host	BR
23	218.65.30.37	4	4	Scanning Host	CN
24	218.65.30.38	4	3	Scanning Host	CN
25	178.94.97.59	4	5	Scanning Host	UA
26	84.241.180.134	6	3	Malware IP	NL
27	62.75.130.12	4	2	Scanning Host	DE
28	62.75.130.13	4	2	Scanning Host	DE
29	62.75.130.14	4	2	Scanning Host	DE
...	...	...	...	...	...
258595	78.27.127.220	4	2	Scanning Host	FI
258596	78.27.127.210	4	2	Scanning Host	FI
258597	78.188.27.29	1	2	Scanning Host	TR
258598	223.4.10.45	6	2	Malware Domain	CN
258599	221.6.207.4	4	3	Scanning Host	CN
258600	78.27.127.51	4	2	Scanning Host	FI
258601	78.27.127.57	4	2	Scanning Host	FI
258602	78.188.27.26	1	2	Scanning Host	TR
258603	78.188.27.27	1	2	Scanning Host	TR
258604	60.168.158.231	4	4	Scanning Host	CN
258605	78.188.27.28	1	2	Scanning Host	TR
258606	180.215.161.174	4	4	Scanning Host	IN
258607	78.27.127.211	4	2	Scanning Host	FI
258608	190.229.178.34	4	3	Scanning Host	AR
258609	190.229.178.37	4	3	Scanning Host	AR
258610	190.229.178.155	4	3	Scanning Host	AR
258611	78.27.127.48	4	2	Scanning Host	FI
258612	23.83.79.89	9	2	Malware Domain	NaN

258613	188.190.124.120	6	3	Malware Domain	UA
258614	78.27.127.50	4	2	Scanning Host	FI
258615	78.27.127.47	4	2	Scanning Host	FI
258616	75.98.171.83	4	2	Spamming	US
258617	114.112.189.27	4	2	Scanning Host	CN
258618	114.112.189.139	4	2	Scanning Host	CN
258619	173.208.220.245	9	2	Spamming	US
258620	179.244.194.219	4	2	Spamming	BR
258621	216.99.159.166	4	2	Scanning Host	US
258622	216.99.159.169	3	2	Scanning Host	US
258623	216.99.159.176	3	2	Scanning Host	US
258624	216.99.159.117	3	3	Scanning Host	US

	Locale	Coord	x
0	Xiamen	24.4797992706,118.08190155	11
1	Xiamen	24.4797992706,118.08190155	11
2	NaN	38.0,-97.0	12
3	Merefa	49.8230018616,36.0507011414	11
4	Union City	37.5962982178,-122.065696716	11
5	Xiamen	24.4797992706,118.08190155	11
6	Xiamen	24.4797992706,118.08190155	11
7	Xiamen	24.4797992706,118.08190155	11
8	NaN	24.4797992706,118.08190155	12
9	Union City	37.5962982178,-122.065696716	11
10	NaN	51.0,9.0	11
11	NaN	51.0,9.0	11
12	NaN	51.0,9.0	11
13	NaN	51.0,9.0	11
14	NaN	37.0,127.5	11
15	NaN	37.0,127.5	11
16	NaN	37.0,127.5	11
17	NaN	37.0,127.5	11
18	NaN	37.0,127.5	11
19	Nanjing	32.0616989136,118.777801514	11
20	Nanjing	32.0616989136,118.777801514	11
21	Foshan	23.0268001556,113.131500244	11
22	NaN	-10.0,-55.0	11
23	Nanchang	28.5499992371,115.933296204	11
24	Nanchang	28.5499992371,115.933296204	11
25	Merefa	49.8230018616,36.0507011414	11
26	NaN	52.5,5.75	7
27	NaN	51.0,9.0	11
28	NaN	51.0,9.0	11
29	NaN	51.0,9.0	11
...	...	...	..
258595	Helsinki	60.1755981445,24.9342002869	11
258596	Helsinki	60.1755981445,24.9342002869	11
258597	Istanbul	41.0186004639,28.9647006989	11

```
258598       Beijing    39.9289016724,116.388298035   6
258599       Nanjing    32.0616989136,118.777801514  11
258600      Helsinki    60.1755981445,24.9342002869  11
258601      Helsinki    60.1755981445,24.9342002869  11
258602      Istanbul    41.0186004639,28.9647006989  11
258603      Istanbul    41.0186004639,28.9647006989  11
258604         Hefei     31.863899231,117.280799866  11
258605      Istanbul    41.0186004639,28.9647006989  11
258606           NaN                      20.0,77.0  11
258607      Helsinki    60.1755981445,24.9342002869  11
258608       Tucuman  -26.8241004944,-65.2226028442  11
258609       Tucuman  -26.8241004944,-65.2226028442  11
258610       Tucuman  -26.8241004944,-65.2226028442  11
258611      Helsinki    60.1755981445,24.9342002869  11
258612           NaN    60.1755981445,24.9342002869   6
258613       Kharkov    49.9808006287,36.2527008057   6
258614      Helsinki    60.1755981445,24.9342002869  11
258615      Helsinki    60.1755981445,24.9342002869  11
258616     Ann Arbor   42.2775993347,-83.7408981323  12
258617       Beijing    39.9289016724,116.388298035  11
258618       Beijing    39.9289016724,116.388298035  11
258619   Kansas City   39.1068000793,-94.5660018921  12
258620           NaN                     -10.0,-55.0  12
258621        Walnut   34.0115013123,-117.853500366  11
258622        Walnut   34.0115013123,-117.853500366  11
258623        Walnut   34.0115013123,-117.853500366  11
258624        Walnut   34.0115013123,-117.853500366  11

[258625 rows x 8 columns]
```

```
In [4]: av.head().to_csv(sys.stdout)
```

```
,IP,Reliability,Risk,Type,Country,Locale,Coord,x
0,222.76.212.185,4,2,Scanning Host,CN,Xiamen,"24.4797992706,118.08190155",11
1,222.76.212.186,4,2,Scanning Host,CN,Xiamen,"24.4797992706,118.08190155",11
2,5.34.246.67,6,3,Spamming,US,,"38.0,-97.0",12
3,178.94.97.176,4,5,Scanning Host,UA,Merefa,"49.8230018616,36.0507011414",11
4,66.2.49.232,4,2,Scanning Host,US,Union City,"37.5962982178,-122.065696716",11
```

1.4 Listing 3-6

Listing 3-6 demonstrates how the dataframe can be displayed in a more aesthetic HTML format by importing HTML from the IPython.display library.

```
In [5]: from IPython.display import HTML
        HTML(av.head().to_html())
```

```
Out[5]: <IPython.core.display.HTML object>
```

1.5 Listing 3-8

The describe function is the python version of the 5 number summary from R. It outputs the medians of the first, second, and third quartiles, as well as the mean, min, max, and standard deviation of the data.

```
In [6]: av['Reliability'].describe()

Out[6]: count    258625.000000
        mean          2.798036
        std           1.130419
        min           1.000000
        25%           2.000000
        50%           2.000000
        75%           4.000000
        max          10.000000
        Name: Reliability, dtype: float64
```

```
In [7]: av['Risk'].describe()

Out[7]: count    258625.000000
        mean          2.221363
        std           0.531572
        min           1.000000
        25%           2.000000
        50%           2.000000
        75%           2.000000
        max           7.000000
        Name: Risk, dtype: float64
```

1.6 Listing 3-10

The count of each categorical value is aggregated and reorganized in contextual order e.g. factor level where 2 implies that it is a greater ranking than one, but not necessarily in quantity.

```
In [8]: def factor_col(col) :
            factor = pd.Categorical(col)
            return pd.value_counts(factor,sort=True).reindex(factor.categories.toli
            #return pd.value_counts(factor, sort=True).reindex(factor.select_dtypes
        print factor_col(av['Reliability'])

1       5612
2     149117
3      10892
4      87039
5          7
6       4758
7        297
8         21
```

```
9         686
10        196
dtype: int64
```

```
In [9]: print factor_col(av['Risk'])
```

```
1         39
2     213851
3      33719
4       9588
5       1328
6         90
7         10
dtype: int64
```

```
In [10]: print factor_col(av['Type']).head(n=10)
```

```
APT;Malware Domain                     1
C&C                                  610
C&C;Malware Domain                    31
C&C;Malware IP                        20
C&C;Scanning Host                      7
Malicious Host                      3770
Malicious Host;Malware Domain          4
Malicious Host;Malware IP              2
Malicious Host;Scanning Host         163
Malware Domain                      9274
dtype: int64
```

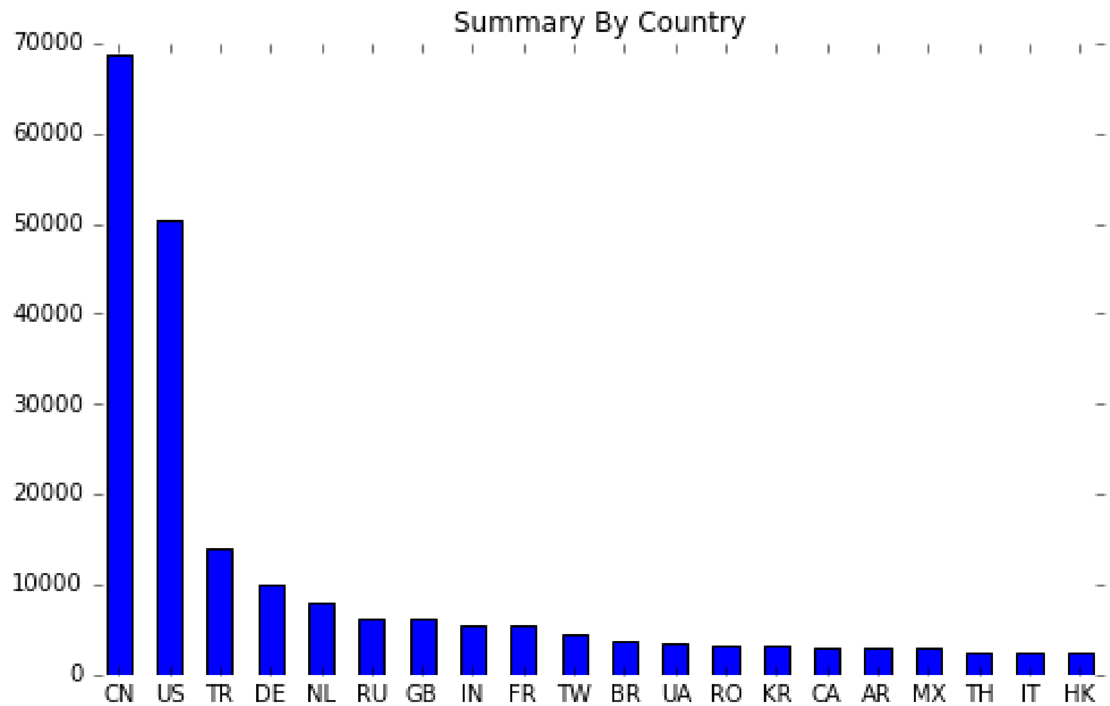
```
In [11]: print factor_col(av['Country']).head(n=10)
```

```
A1      267
A2        2
AE     1827
AL        4
AM        6
AN        3
AO      256
AR     3046
AT       51
AU      155
dtype: int64
```

1.7 Listing 3-14

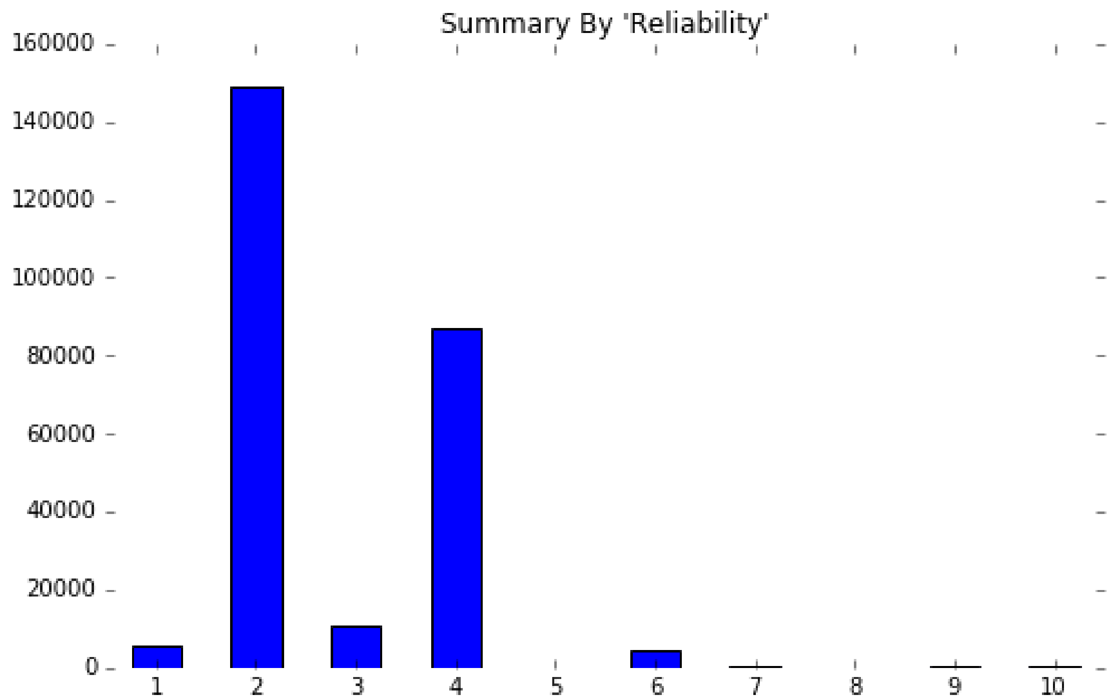
It is possible to plot the count of 20 countries as a bar graph, in descending order, using the matplotlib.pyplot library. We use this do determine which contry accounts for the most malicious nodes. We further investigate the dataframe by graphing the Reliability and Risk categorical value counts to get an overview of the characteristics of the majority of the nodes.

```
In [12]: import matplotlib.pyplot as plt
         country_ct = pd.value_counts(av['Country'])
         plt.axes(frameon=0)
         country_ct[:20].plot(kind='bar', rot=0, title="Summary By Country", figsiz
```



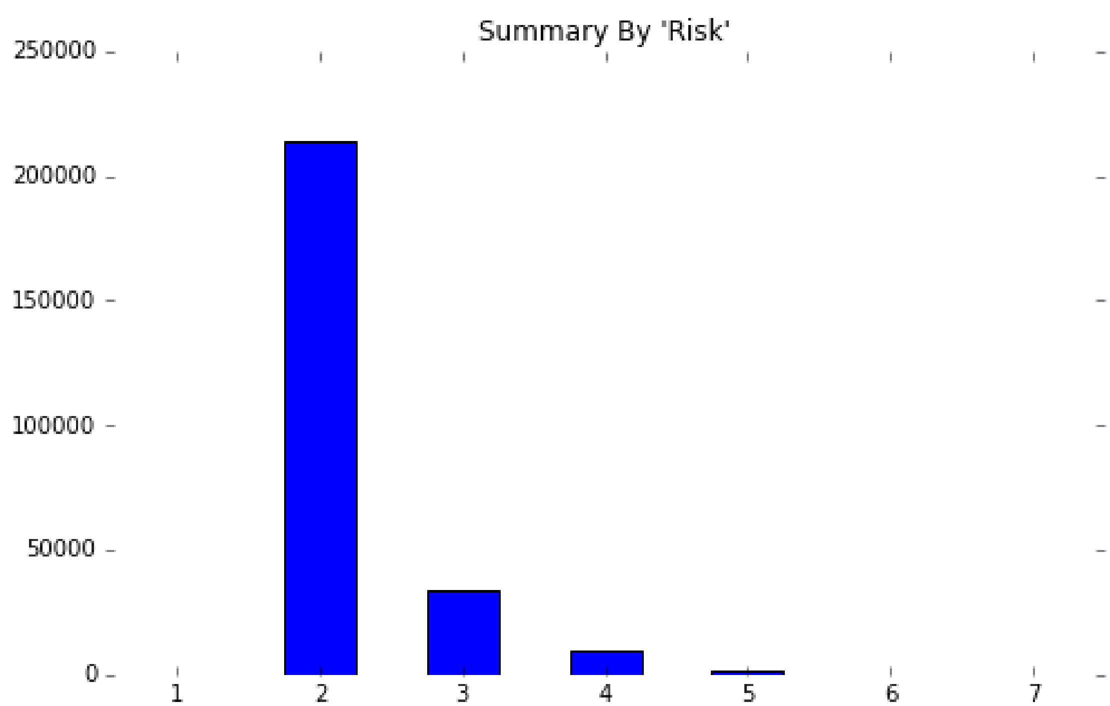
1.8 Listing 3-15

```
In [13]: plt.axes(frameon=0)
         factor_col(av['Reliability']).plot(kind='bar', rot=0, title="Summary By 'F
```



1.9 Listed 3-16

```
In [14]: plt.axes(frameon=0)
         factor_col(av['Risk']).plot(kind='bar', rot=0, title="Summary By 'Risk'",
```



1.10 Listing 3-18

The contribution of malicious nodes by countries can be reflected as a percentage output as well by dividing the value count of each country with the length of the factor.

```
In [15]: top10 = pd.value_counts(av['Country'])[0:9]
         top10.astype(float) / len(av['Country'])
```

```
Out[15]: CN     0.265179
         US     0.194826
         TR     0.053970
         DE     0.038484
         NL     0.030666
         RU     0.024537
         GB     0.024333
         IN     0.021189
         FR     0.021069
         Name: Country, dtype: float64
```

1.11 Listing 3-20

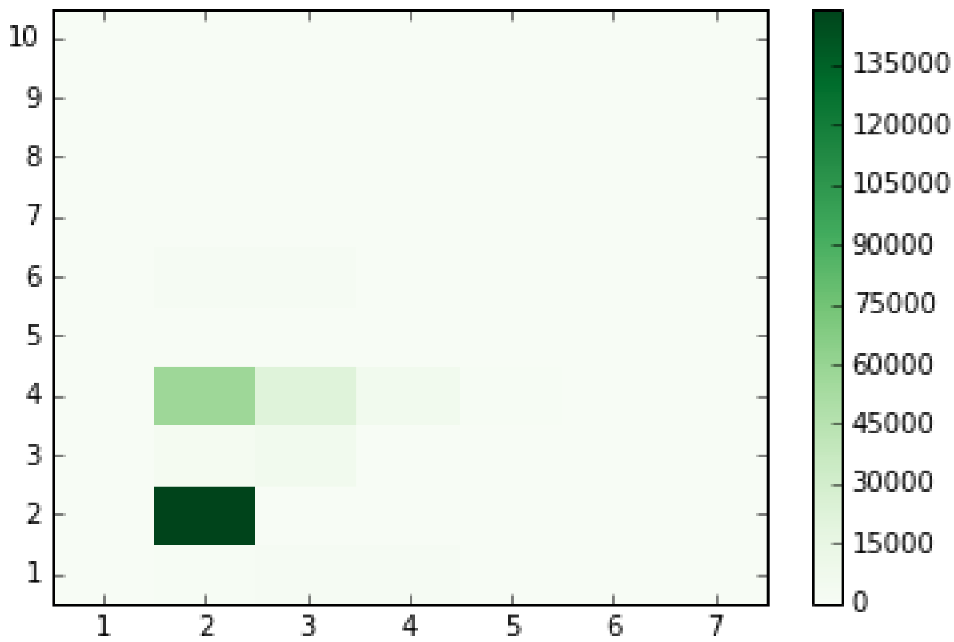
A numerical output of the count of nodes for each permutation of Risk~Reliability combination is made using the cm function from the pandas library. Then, using numpy and cmap from mplotlib, you can produce a heatmap of the permutations to get a visual of where the majority of the data is concentrated.

```
In [16]: from matplotlib import cm
         from numpy import arange
         pd.crosstab(av['Risk'], av['Reliability'])
```

```
Out[16]: Reliability     1       2     3      4  5     6    7   8    9  10
         Risk
         1               0       0    16      7  0     8    8   0    0   0
         2             804  149114  3670  57652  4  2084   85  11  345  82
         3            2225       3  6668  22168  2  2151  156   7  260  79
         4            2129       0   481   6447  0   404   43   2   58  24
         5             432       0    55    700  1   103    5   1   20  11
         6              19       0     2     60  0     8    0   0    1   0
         7               3       0     0      5  0     0    0   0    2   0
```

```
In [17]: xtab = pd.crosstab(av['Reliability'], av['Risk'])
         plt.pcolor(xtab,cmap=cm.Greens)
         plt.yticks(arange(0.5, len(xtab.index), 1),xtab.index)
         plt.xticks(arange(0.5, len(xtab.columns), 1),xtab.columns)
         plt.colorbar()
```

```
Out[17]: <matplotlib.colorbar.Colorbar at 0x92af320>
```



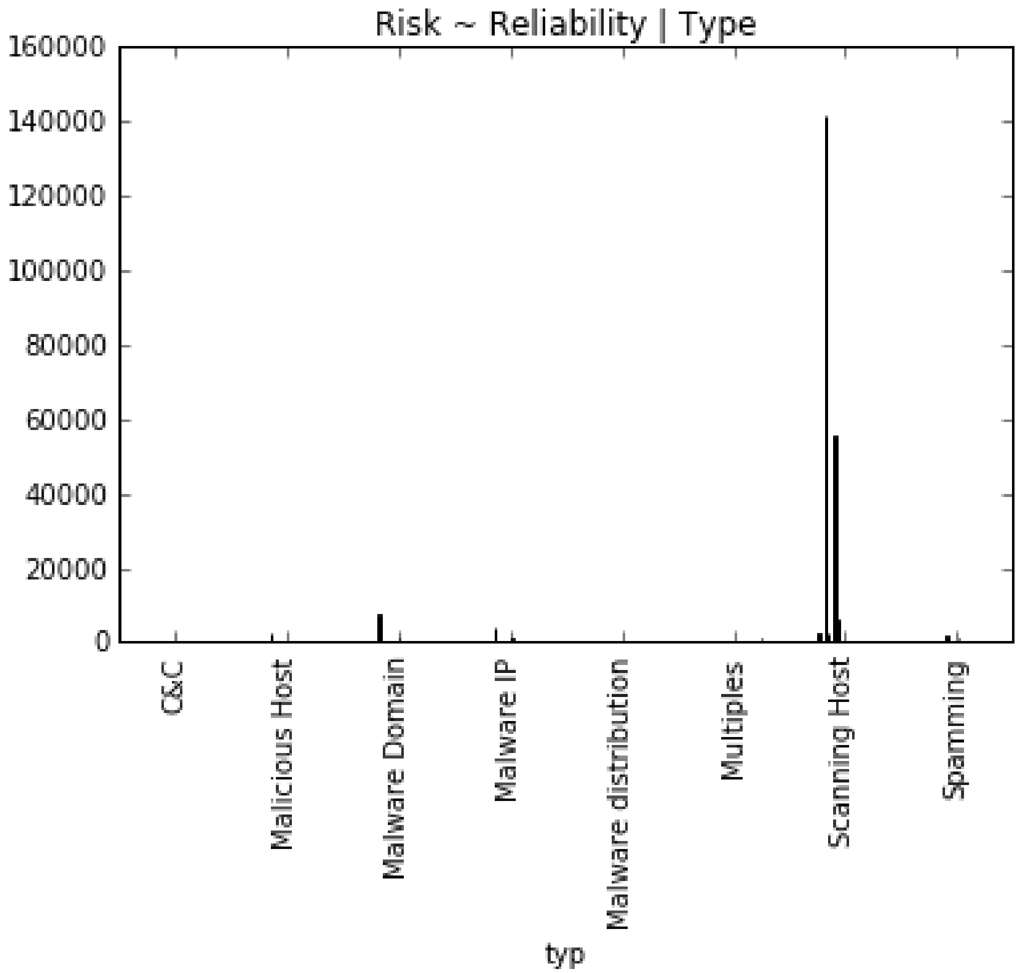
1.12 Listing 3-23

A three-way contingency table, relating Risk, Reliablility, and malicious node type and outputting it as a simple bargraph.

```
In [18]: av['newtype'] = av['Type']
         av[av['newtype'].str.contains(";")] = "Multiples"
         typ = av['newtype']
         rel = av['Reliability']
         rsk = av['Risk']
         xtab = pd.crosstab(typ, [ rel, rsk ], rownames=['typ'], colnames=['rel', '
         print xtab.to_string()
```

```
rel                       1                                     2       3
rsk                       2     3     4     5    6  7           2  3    1     2     3    4
typ
C&C                       0     0     1     2    1  0           0  0    0     0     0  313
Malicious Host            0     6    51    41    8  1           0  0    1   206  2250    7
Malware Domain           12     1     0     0    0  0        7309  0    2   246    55    2
Malware IP                0    23    11    15   10  2           0  3   12   415  4091   71
Malware distribution      0     0     0     0    0  0           0  0    0     0     1    0
Multiples                 0     0     0     0    0  0           0  0    0     0     0    0
Scanning Host           790  2189  2056   366    0  0      141543  0    1  2685   159   35
Spamming                  1     2     9     7    0  0           1  0    0    22     9   17
```

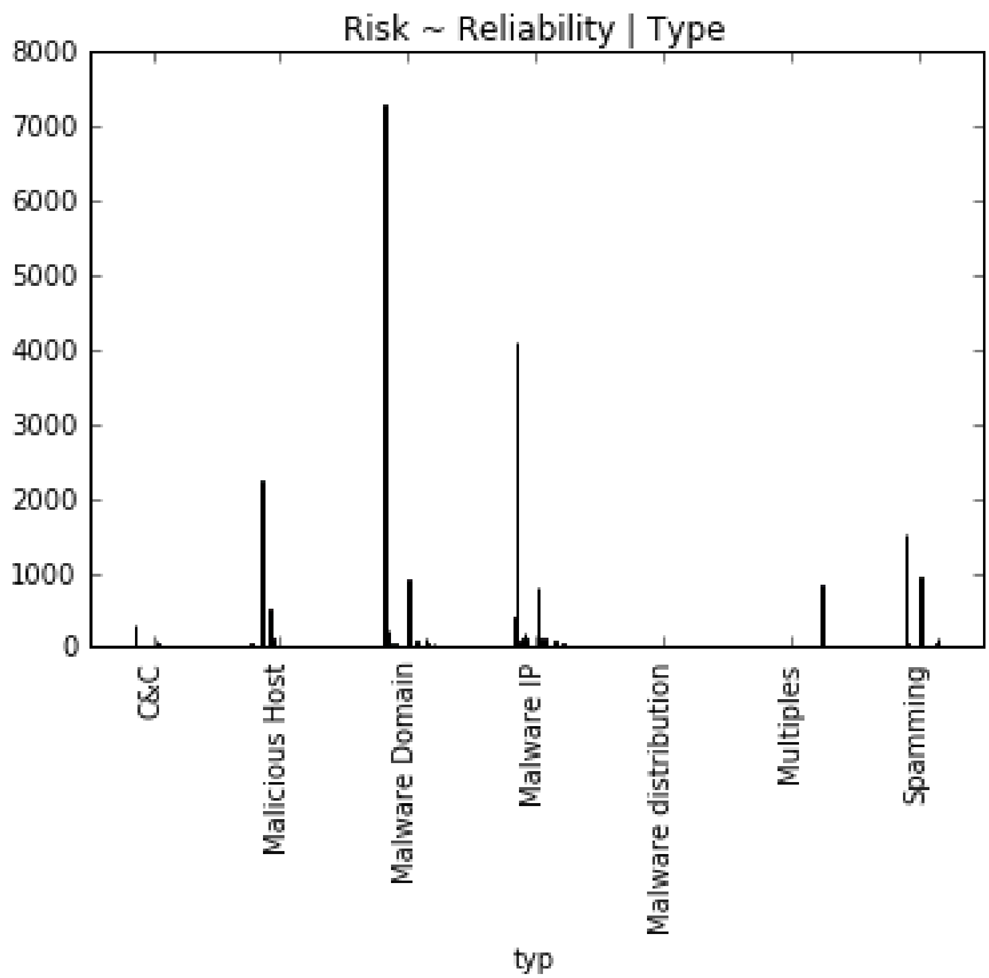
```
In [19]: xtab.plot(kind='bar', legend=False, title="Risk ~ Reliability | Type").gri
```



1.13 Listing 3-25

The Scanning Host category is omitted since the majority of the nodes are concentrated there but have a negligable Risk~Reliability permutation.

```
In [20]: rrt_df = av[av['newtype'] !="Scanning Host"]
         typ = rrt_df['newtype']
         rel = rrt_df['Reliability']
         rsk = rrt_df['Risk']
         xtab = pd.crosstab(typ, [ rel, rsk ], rownames=['typ'], colnames=['rel', '
         xtab.plot(kind='bar', legend=False, title="Risk ~ Reliability | Type").gri
```



1.14 Listing 3-27

The Malware distribution and Malware Domain are omitted since Malware Domain has no nodes, and Malware distribution has negligeable Risk~Reliability permutation. A count of the nodes in the modified data shows that it accounts for 5.9% of all nodes.

```
In [21]: rrt_df = rrt_df[rrt_df['newtype'] != "Malware distribution" ]
         rrt_df = rrt_df[rrt_df['newtype'] != "Malware Domain" ]
         typ = rrt_df['newtype']
         rel = rrt_df['Reliability']
         rsk = rrt_df['Risk']
         xtab = pd.crosstab(typ, [ rel, rsk ], rownames=['typ'], colnames=['rel', '
         print "Count: %d; Percent: %2.1f%%" % (len(rrt_df), (float(len(rrt_df)) /

Count: 15171; Percent: 5.9%
```

```
In [22]: xtab.plot(kind='bar', legend=False, title="Risk ~ Reliability | Type").gri
```