Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

NULLABOR: AN R PACKAGE FOR VISUAL STATISTICAL INFERENCE

Niladri Roy Chowdhury
Novartis Pharmaceuticals

Hadley Wickham
Rstudio

Dianne Cook
Monash University

Abstract

Statistical graphics play an important role in statistical analysis, model checking and data mining, but they have not been used for statistical inference. Buja, Cook, Hofmann, Lawrence, Lee, Swayne, and Wickham (2009) introduced protocols to allow inference to statistical graphics, which is also described in Wickham, Cook, Hofmann, and Buja (2010). Later Hofmann, Majumder, and Cook (2013) describes a comparative study between classical inference methods and visual inference methods in a linear modeling setting. The recent developments in visual inference demands an open source platform which can be used to implement the visual methods. The R package **nullabor** provides methods of generating lineups automatically for different null generating methods and also construction of diagnostic plots to measure the quality of a lineup. In this paper we discuss the different tools in **nullabor** which helps to use visual inference easily.

Keywords: visual inference, lineup, null generating mechanisms.

1. Introduction

Recent developments suggest that statistical graphics can be used to perform statistical inference. Buja *et al.* (2009) introduced protocols to allow inference to statistical graphics, which were later validated by Hofmann *et al.* (2013). The recent developments require a software package which would make the implementation of visual inference easier and automatic. Buja *et al.* (2009) describes two protocols which are known as the lineup protocol and the Rorschach protocol, which is also described in Wickham *et al.* (2010). In the lineup protocol, the plot of the actual data is randomly placed in a set of null plots, These null plots are generated by a mechanism which is consistent with the null hypothesis. A human judge is shown the lineup and asked to identify the plot which is the most different or the plot which has the strongest structure. If the human judge can identify the plot of the actual data, we reject the null hypothesis. On the other hand, in the Rorschach protocol, all the plots are generated

from the null distribution assuming that the null hypothesis is true. This protocol helps us to calibrate our eyes to the natural variability of the plots and hence reduce our sensitivity to structures which appear purely due to randomness. This package provides functions to implement these two protocols in different scenarios.

The second part of the package introduces distance metrics. Distance metrics are used to measure distances between the plots in a lineup. The null plots in a lineup may affect the decision of the human subjects since they base their decision on a finite number of null plots. Hence it is essential to measure the quality of the lineups by calculating the distances between the plot of the actual data and the null plots and the null plots among themselves. To compare several lineups, the distances obtained from a certain lineup are compared to the empirical distribution of the distances.

This chapter is organized in the following way. Section 2 describes the different null generating mechanisms which are used to obtain the null plots in the lineup. The two protocols are described in Section 3. Section 4 discusses five different distance metrics. These are targeted for different varieties of data and plot types. Calculating different statistics from a lineup are described in the next section. Section 6 describes the procedure to select the optimal number of bins for binned distance. Finally Section 7 gives the method of obtaining and plotting the empirical distribution of the distance metrics.

2. Null generating mechanisms

The null plots in a lineup is obtained by a method which is consistent with the null hypothesis. Assuming that the null hypothesis is true, the null datasets are obtained, often, from the actual dataset. This process of generating the null datasets is called the null generating mechanism. There may be a variety of null generating mechanisms but the package provides three different methods.

2.1. Generate null data with a specific distribution

The `"null_dist"` function takes as input a variable name of the data and a particular distribution. This variable in the data is substituted by random generations of the particular distribution. The different distributions include beta, cauchy, chi-squared, exponential, f, gamma, geometric, log-normal, lognormal, logistic, negative binomial, normal, poisson, t and weibull. A list of parameters of distribution can also be provided as input. In case it is not provided, `"fitdistr"` is used to estimate the parameters from the given data. The function `"null_dist"` returns a function that given the data generates a null data set.

```
> head(null_dist("mpg", dist = "normal")(mtcars))
```

```
                      mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         29.536386   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     16.897772   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        16.395593   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive     9.644472   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 11.456224   8  360 175 3.15 3.440 17.02  0  0    3    2
```

```
Valiant           25.417741   6  225 105 2.76 3.460 20.22  1  0    3    1
```

2.2. Generate null data by permuting a variable

The `"null_permute"` function takes as input a variable name of the data. This variable is permuted to obtain the null dataset. The function `"null_dist"` returns a function that given the data generates a null data set.

```
> head(null_permute("mpg")(mtcars))
```

```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4        19.7   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag    21.4   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive   30.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 21.0  8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant          15.8   6  225 105 2.76 3.460 20.22  1  0    3    1
```

2.3. Generate null data with null residuals from a model

The function `"null_lm"` takes as input a model specification formula as defined by `"lm"` and method for generating null residuals from the model. The three built in methods are 'rotate', 'pboot' and 'boot' defined by `"resid_rotate"`, `"resid_pboot"` and `"resid_boot"` respectively. The function returns a function which given the data generates a null dataset.

```
> head(null_lm(wt~mpg, method = 'rotate')(mtcars))
```

```
                  mpg cyl disp  hp drat       wt  qsec vs am gear carb
Mazda RX4        21.0   6  160 110 3.90 3.049169 16.46  0  1    4    4
Mazda RX4 Wag    21.0   6  160 110 3.90 2.634875 17.02  0  1    4    4
Datsun 710       22.8   4  108  93 3.85 2.917568 18.61  1  1    4    1
Hornet 4 Drive   21.4   6  258 110 3.08 3.126533 19.44  1  0    3    1
Hornet Sportabout 18.7  8  360 175 3.15 4.281755 17.02  0  0    3    2
Valiant          18.1   6  225 105 2.76 3.368690 20.22  1  0    3    1
                      .resid  .fitted
Mazda RX4        -0.03998504 3.089154
Mazda RX4 Wag    -0.45427860 3.089154
Datsun 710        0.08196630 2.835602
Hornet 4 Drive    0.09372421 3.032809
Hornet Sportabout 0.86861876 3.413136
Valiant          -0.12896315 3.497653
```

3. Protocols

Buja *et al.* (2009) introduces two protocols which bridges the gulf between traditional statistical inference and exploratory data analysis. The two protocols can be implemented in the following way:

3.1. The lineup protocol

In this protocol, the plot of the real data is randomly embedded amongst a set of null plots. The matrix of plots is known as a lineup. The null plots are generated by a method consistent with the null hypothesis. The lineup is shown to an observer. If the observer can pick the real data as different from the others, this puts weight on the statistical significance of the structure in the plot. The `"lineup"` function returns a set of generated null datasets and the real data embedded randomly among these null datasets. The method of null generation should be provided in the lineup function for the null datasets to be generated automatically along with the real dataset. The users also have the option of generating the null datasets themselves and providing them in the `"lineup"` function. The position of the real dataset can be left missing and the function picks the position at random. The function then returns the position as an encrypted code. The encrypted code is copied and pasted on the R console to obtain the true position of the plot.

```
> head(lineup(null_permute("mpg"), mtcars), 20)


decrypt("jriG PHhH D5 QZXDhDZ5 Jf")
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb .sample
1  15.8   6 160.0 110 3.90 2.620 16.46  0  1    4    4       1
2  16.4   6 160.0 110 3.90 2.875 17.02  0  1    4    4       1
3  21.4   4 108.0  93 3.85 2.320 18.61  1  1    4    1       1
4  30.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1       1
5  17.8   8 360.0 175 3.15 3.440 17.02  0  0    3    2       1
6  18.7   6 225.0 105 2.76 3.460 20.22  1  0    3    1       1
7  19.2   8 360.0 245 3.21 3.570 15.84  0  0    3    4       1
8  10.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2       1
9  21.0   4 140.8  95 3.92 3.150 22.90  1  0    4    2       1
10 27.3   6 167.6 123 3.92 3.440 18.30  1  0    4    4       1
11 15.5   6 167.6 123 3.92 3.440 18.90  1  0    4    4       1
12 15.2   8 275.8 180 3.07 4.070 17.40  0  0    3    3       1
13 18.1   8 275.8 180 3.07 3.730 17.60  0  0    3    3       1
14 21.0   8 275.8 180 3.07 3.780 18.00  0  0    3    3       1
15 15.2   8 472.0 205 2.93 5.250 17.98  0  0    3    4       1
16 22.8   8 460.0 215 3.00 5.424 17.82  0  0    3    4       1
17 14.3   8 440.0 230 3.23 5.345 17.42  0  0    3    4       1
18 21.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1       1
19 10.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2       1
20 21.5   4  71.1  65 4.22 1.835 19.90  1  1    4    1       1
```

The lineup data can be then used to generate the lineup using **ggplot2**. The lineup is shown to one or more observers who are asked to identify the plot which is different. If the observer

can identify the plot of the real data correctly, we reject the null hypothesis and conclude that the plot of the real data has stronger structure than the null plots.

```
> qplot(mpg, wt, data = lineup(null_permute("mpg"), mtcars)) + facet_wrap(~ .sample)

decrypt("jriG PHhH D5 QZXDhDZ5 l")
```

Figure 1 is an example of a lineup plot where the plot of the actual data is embedded into nineteen null plots. The null plots are obtained by permuting the variable "mpg" while keeping the other variables fixed. The position of the actual plot is not known to the user. It can be obtained by copying the output in the R console. Hence the user can also act as a human judge.

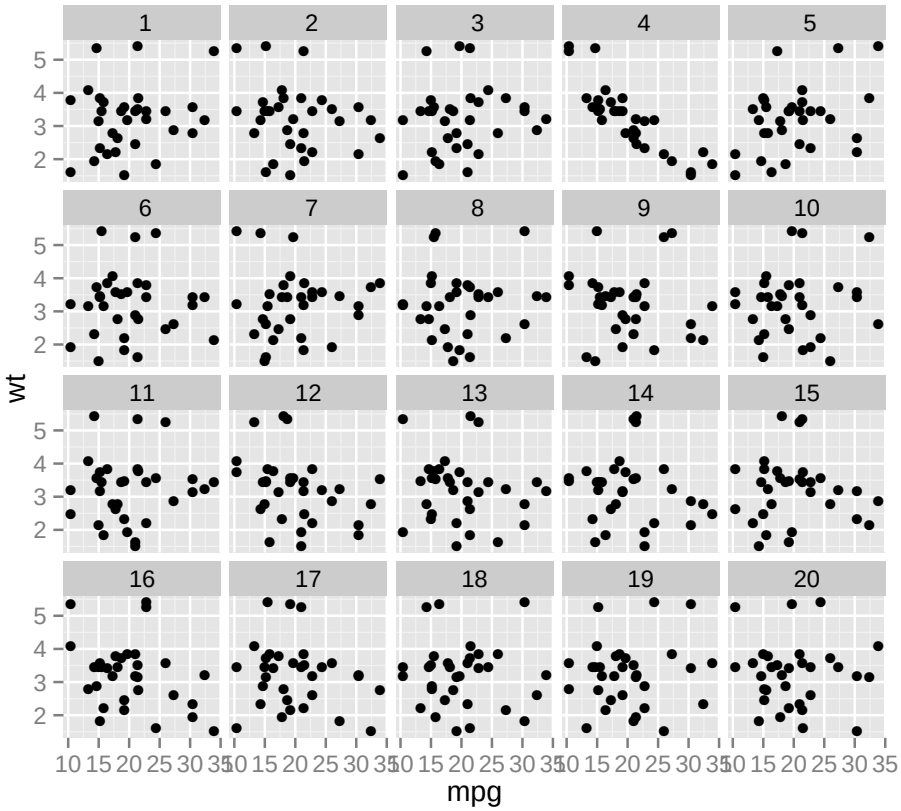

Figure 1: A typical lineup. The plot of the actual data is embedded in a set of null plots which are generated assuming that the null hypothesis is true. The variable "mpg" is permuted 19 times to obtain the 19 null plots. Can you identify the plot which has the steepest slope? The true position of the plot can be obtain by copying and pasting "decrypt(...)" from the output into R console.

3.2. The Rorschach protocol

The Rorschach protocol is used to calibrate the eyes for variation due to sampling. The plots generated corresponds to the null datasets, data that is consistent with a null hypothesis. The `"rorschach"` function returns a set of null plots which are shown to observers to calibrate their eyes with variation. Like the `"lineup"` function, the null generating mechanism should be provided as an input along with a real dataset. A probability can also be given as input which dictates the chance of including the true data with null data.

```
> head(rorschach(null_permute("mpg"), mtcars, n = 8, p = 0), 20)
```

```
   .n  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1   1 17.3   6 160.0 110 3.90 2.620 16.46  0  1    4    4
2   1 32.4   6 160.0 110 3.90 2.875 17.02  0  1    4    4
3   1 21.0   4 108.0  93 3.85 2.320 18.61  1  1    4    1
4   1 21.0   6 258.0 110 3.08 3.215 19.44  1  0    3    1
5   1 15.5   8 360.0 175 3.15 3.440 17.02  0  0    3    2
6   1 30.4   6 225.0 105 2.76 3.460 20.22  1  0    3    1
7   1 14.7   8 360.0 245 3.21 3.570 15.84  0  0    3    4
8   1 22.8   4 146.7  62 3.69 3.190 20.00  1  0    4    2
9   1 15.0   4 140.8  95 3.92 3.150 22.90  1  0    4    2
10  1 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
11  1 22.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
12  1 18.7   8 275.8 180 3.07 4.070 17.40  0  0    3    3
13  1 26.0   8 275.8 180 3.07 3.730 17.60  0  0    3    3
14  1 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
15  1 21.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
16  1 24.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
17  1 21.5   8 440.0 230 3.23 5.345 17.42  0  0    3    4
18  1 19.7   4  78.7  66 4.08 2.200 19.47  1  1    4    1
19  1 15.2   4  75.7  52 4.93 1.615 18.52  1  1    4    2
20  1 21.4   4  71.1  65 4.22 1.835 19.90  1  1    4    1
```

```
> qplot(mpg, wt, data = rorschach(null_permute("mpg"), mtcars, n = 9, p = 0))
+ facet_wrap(~ .n)
```

The Rorschach protocol helps us to calibrate our vision to the natural variability in plots in which the data is generated from scenarios consistent with the null hypothesis. Figure 2 is drawn using the rorschach data using package **ggplot2**. Can we identify any interesting pattern in the plots? Is there a plot among these which has a stronger structure than others?

4. Distance metrics

There are five different distance metrics in **nullabor** package, named `"bin_dist"`, `"box_dist"`, `"reg_dist"`, `"sep_dist"` and `"uni_dist"`. The different distance metrics are constructed so that they can identify the different properties of the data. `"uni_dist"` works for univariate data while the others works for all types of bivariate data. Binned distance is a generic distance which can be used in any situations while the other distance metrics are constructed

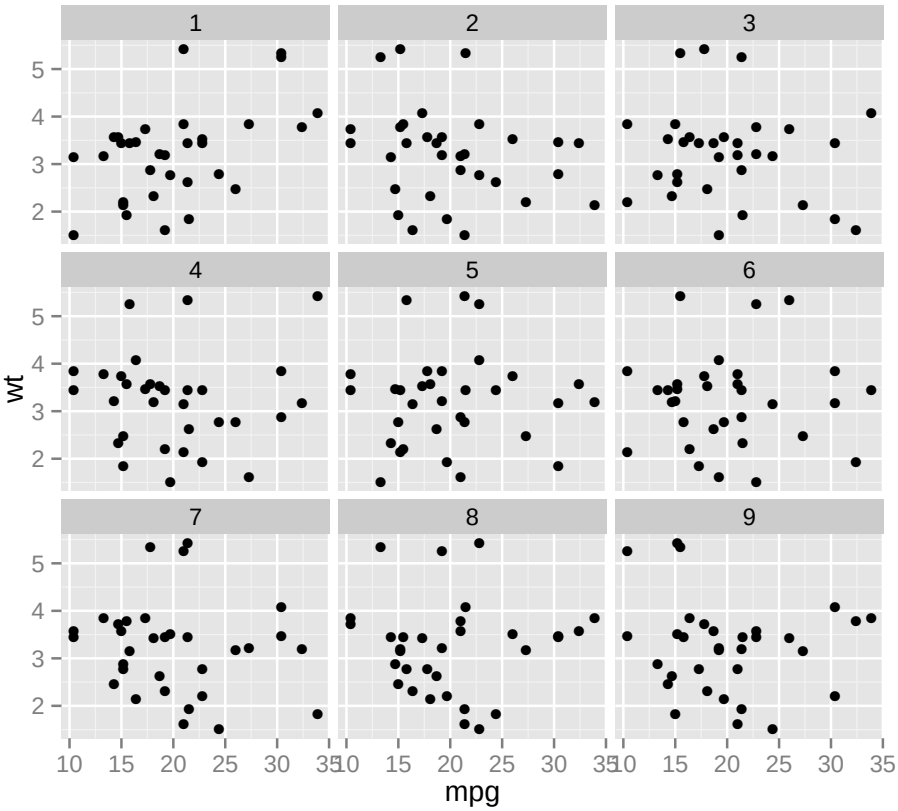

Figure 2: Illustration of the Rorschach protocol. These nine scatterplots are obtained by permuting the variable "mpg" while keeping the variable "wt" fixed. Does it look like the strength of the relationship is same in all the plots? Is there a plot which has more interesting pattern than others?

so that they can identify the effect of graphical elements in a plot like an overlaid regression line or presence of defined clusters. To calculate some of the metrics, additional informations like a class variable or the number of bins should be provided.

4.1. Distance for univariate data

`"uni_dist"` is a distance metric which calculates the euclidean distance between the first four central moments of two univariate data. A typical usage would be when one needs to calculate the distance between the two histograms drawn from two datasets.

```
> uni_dist(X = rnorm(100), PX = rpois(100, 2))

[1] 2.980522
```

4.2. Distance based on regression parameters

`"reg_dist"` is a distance metric which calculates the euclidean distance between the regression parameters of a model fitted to one plot and that of another plot. It is advisable to use this distance in situations where a regression line is overlaid on a scatterplot.

```
> X <- data.frame(X1 = mtcars$wt, X2 = mtcars$mpg)
> PX <- data.frame(X1 = sample(mtcars$wt), X2 = mtcars$mpg)
> reg_dist(X, PX)

[1] 132.9617
```

4.3. Distance based on boxplots

`"box_dist"` is a distance metric which works for side-by-side boxplots with two levels. The first quartile, median and the third quartile are calculated for each box and the absolute distances of these are calculated for the two boxes. `"box_dist"` calculates the euclidean distance between these absolute distances for the two plots. The boxplot distance should be used in situations where a side-by-side boxplot is used to compare the distribution of a variable at two different levels.

```
> X <- data.frame(X1 = as.factor(mtcars$am), X2 = mtcars$mpg)
> PX <- data.frame(X1 = as.factor(sample(mtcars$am)), X2 = mtcars$mpg)
> box_dist(X, PX)

[1] 12.94846
```

4.4. Distance based on separation

`"sep_dist"` is a distance metric based on the separation between clusters. The separation between clusters is defined by the minimum distances of a point in the cluster to a point in

another cluster. The separation between the clusters for a given dataset is calculated. An euclidean distance is calculated between the separation for the given dataset and another dataset. The number of clusters in the dataset should be provided. If not, the hierarchical clustering method is used to obtain the clusters.

```
> X <- data.frame(X1 = mtcars$wt, X2 = mtcars$mpg,
cl = as.numeric(as.factor(mtcars$cyl)))
> PX <- data.frame(X1 = sample(mtcars$wt), X2 = mtcars$mpg,
cl = as.numeric(as.factor(mtcars$cyl)))
> sep_dist(X, PX, clustering = TRUE)

[1] 0.2552364

> sep_dist(X, PX, clustering = FALSE, nclust = 3)

[1] 0.04571194
```

4.5. Binned Distance

`"bin_dist"` is a generic distance which works for any situation for any dataset. For a given bivariate dataset, X and Y variables are divided into p and q bins respectively to obtain pq cells. The number of points falling in each cell are counted for a given dataset. `"bin_dist"` between two datasets calculates the euclidean distance between the cell counts of these two data. The values of p and q should be provided as arguments.

```
> X <- data.frame(X1 = mtcars$wt, X2 = mtcars$mpg)
> PX <- data.frame(X1 = sample(mtcars$wt), X2 = mtcars$mpg)
> bin_dist(X,PX, lineup.dat = NULL, X.bin = 5, Y.bin = 5)

[1] 7.745967
```

5. Calculation of mean distances and difference

If the plot of the actual data can be easily identified in a lineup, then the plot should have stronger structures than the null plots. The distance metrics can be used to calculate the distance between the plot of the actual data and the null plots. The distances between the null plots can also be calculated.

5.1. Calculating the mean distances for the plots in the lineup

The distances between the true plot and all the null plots are calculated and the mean of these distances is calculated. Similarly, for each null plot, the distances between the null plots are calculated and averaged to obtain the mean distance for each null plot. `"calc_mean_dist"` calculates the mean distance corresponding to each plot in the lineup. If the mean distance of the true plot is larger than the mean distances of all the null plots, the lineup is considered

easy. If one of the null plots has a larger mean distance than the true plot, the lineup is considered difficult.

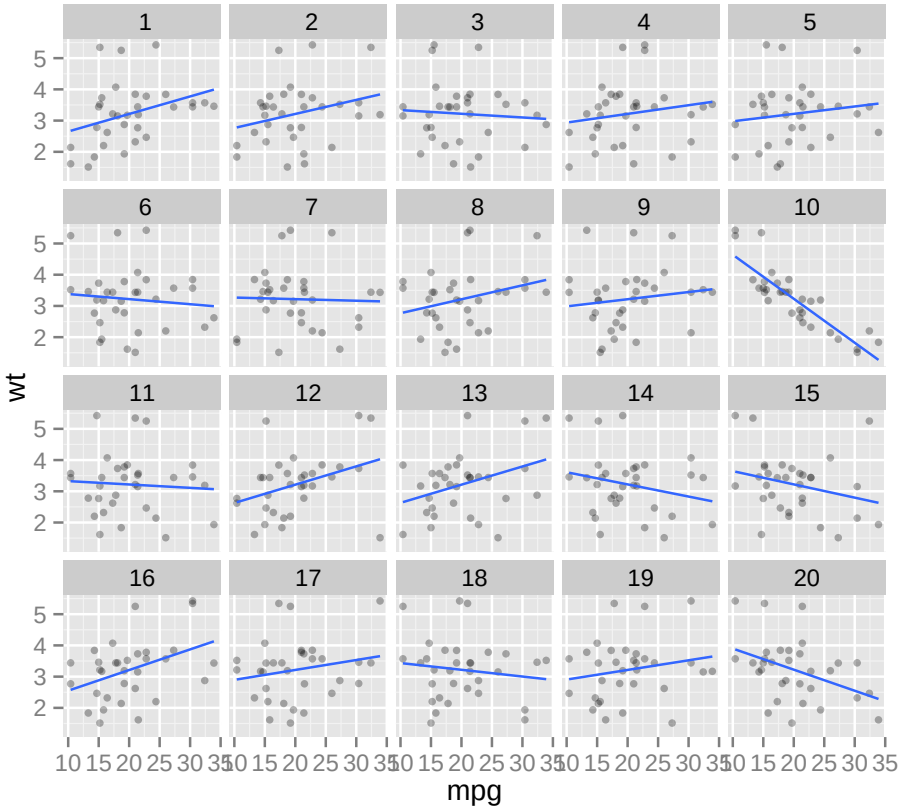

Figure 3: Lineup plot of size $m = 20$ of scatterplots between "mpg" and "wt" with a regression line overlaid. The plot of the actual data is randomly placed among the set of null plots. The position of the plot of the actual data is 10.

```
> lineup.dat <- lineup(null_permute('mpg'), mtcars, pos = 10)
> calc_mean_dist(lineup.dat, var = c('mpg', 'wt'), met = 'reg_dist', pos = 10)
Source: local data frame [20 x 2]

   plotno  mean.dist
1       1  1.4105848
2       2  1.0551943
3       3  0.8912037
4       4  0.7146171
5       5  0.6708689
6       6  0.9966672
7       7  0.7674444
8       8  1.0456351
9       9  0.6642878
```

```
10     10 10.1982236
11     11  0.8703412
12     12  1.5049771
13     13  1.4878584
14     14  1.7874604
15     15  1.9487011
16     16  1.8179699
17     17  0.7709551
18     18  1.1412758
19     19  0.7568216
20     20  3.4081131
```

5.2. Calculating difference measure for lineups

The mean distances for each plot in the lineup are obtained using `"calc_mean_dist"`.`"calc_diff"` calculates the difference between the mean distance for the true plot and the maximum mean distance for the null plots.

```
> calc_diff(lineup.dat, var = c('mpg', 'wt'), met = 'reg_dist', dist.arg = NULL,
pos = 10)

[1] 6.79011
```

6. Selection of bins

Binned distance is highly affected by the choice of the number of bins. The number of bins is provided by the user and this can be subjective. This motivates to design a way to select the optimum number of bins to be used. `"opt_diff"` finds the optimal number of bins in both x and y direction which should be used to calculate the binned distance. The binned distance is calculated for each combination of provided choices of number of bins in x and y direction and finds the difference using `"calc_diff"` for each combination. The combination for which the difference is maximum should be used.

```
> opt.diff <- opt_bin_diff(lineup.dat, var = c('mpg', 'wt'), 2, 10, 2, 10, pos = 10,
plot = TRUE)
> head(opt.diff$dat)

Source: local data frame [6 x 3]
Groups: xbins

  xbins ybins       Diff
1     2     2 -0.1033782
2     2     3 -0.7769468
3     2     4 -0.4836467
4     2     5 -0.4836467
```

```
5     2     6 -0.4956530
6     2     7 -0.4836467

> opt.diff$p
```

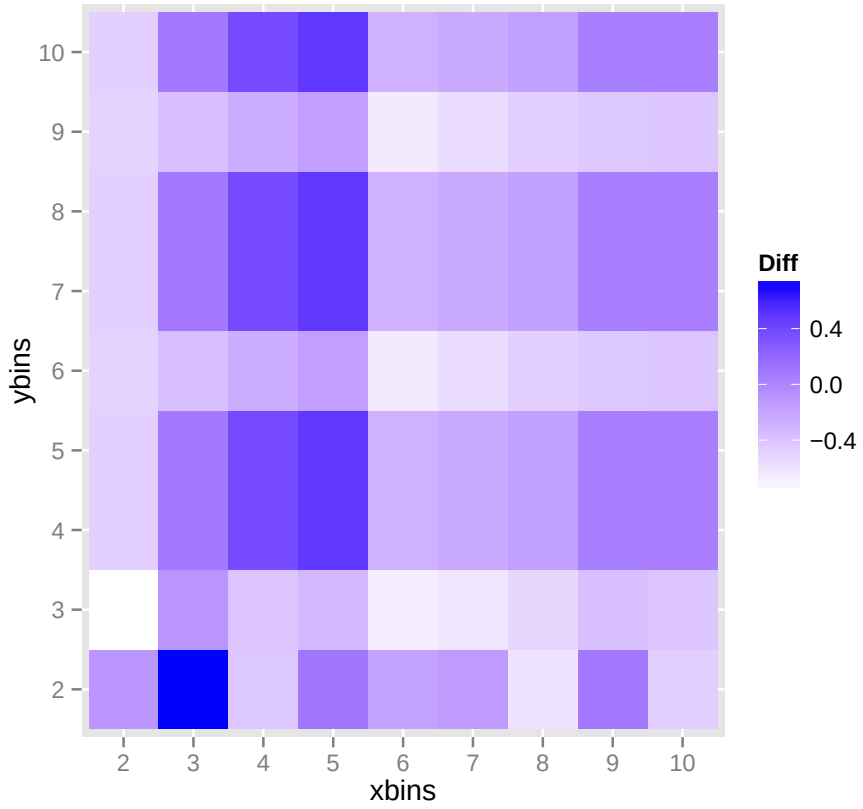

Figure 4: Illustration of the optimum bin selection procedure. The obtained lineup data from subsection 5.1 is used to generate difference between the true plot and the maximum of the null plots using a binned distance for all combinations of number of bins on both x and y axes. The number of bins on x and y axes are plotted on x and y respectively with the colors showing the differences. The darker color represents larger difference.

7. Distribution of distance metrics

Comparing several lineups require a common platform. For a given lineup, the distance for the plot of the actual data can be compared to the empirical distribution of all the null plots. An extreme value of this distance indicates that the plot of the actual data is not consistent with the null hypothesis. Hence such a lineup would be considered "easy". If the same plot of actual data is used with another set of null plots in another lineup, there may be a null plot which is more extreme than the actual plot. Hence such a lineup would be considered

"difficult". The following functions are used to obtain the empirical distribution of the distance metric and plot the distribution using **ggplot2**.

7.1. Empirical distribution of distance metrics

`"distmet"` function provides the empirical distribution of the distance metrics based on the mean distance of the true plot and the mean distance from the null plots. The lineup data, the null generating mechanism and the choice of the distance metric has to be provided. Users have the flexibility of using their distance metrics. The position of the true plot in the lineup has to be provided as well. If the distance metrics require additional arguments, those have to be provided as well.

The distribution of the regression based distance is obtained as following. The output calculates the distances for the plots in the lineup and also the difference between the plot of the actual data and the maximum of the null plots. It also provides the null plots which are closest to the plot of the actual data. The position of the actual data plot and the values for the null distributions are also given to be used for the plotting function.

```
> dist.vals1 <- distmet(lineup.dat, var = c('mpg', 'wt'),'reg_dist', null_permute('mpg'),
pos = 10,  repl = 1000, dist.arg = NULL)
> head(dist.vals1$lineup)

Source: local data frame [6 x 2]

  plotno mean.dist
1      1 1.4105848
2      2 1.0551943
3      3 0.8912037
4      4 0.7146171
5      5 0.6708689
6      6 0.9966672

> dist.vals1$diff

[1] 6.79011

> head(dist.vals1$closest)

[1] 20 15 16 14 12

> head(dist.vals1$null_values)

[1] 1.0564997 0.4444821 0.3020866 0.3106895 0.8703144 0.9713040

> dist.vals$pos
[1] 10
```

7.2. Plotting the empirical distribution of the distance metric

`"distplot"` functions plots the empirical distribution of the distance metric, given the output of `"distmet"` function. The distribution is shown in grey along the distance for the true plot in orange and the distances for the null plots in black.

The empirical distribution of the regression based distance metric is plotted in the following way. This plot can be matched with Figure 3. Plot 10 is the plot of the actual data which can easily be identified from the lineup and it can also be seen in Figure 5. Plot 20 also has a moderate negative relationship which is also evident.

```
> distplot(dist.vals1)
```

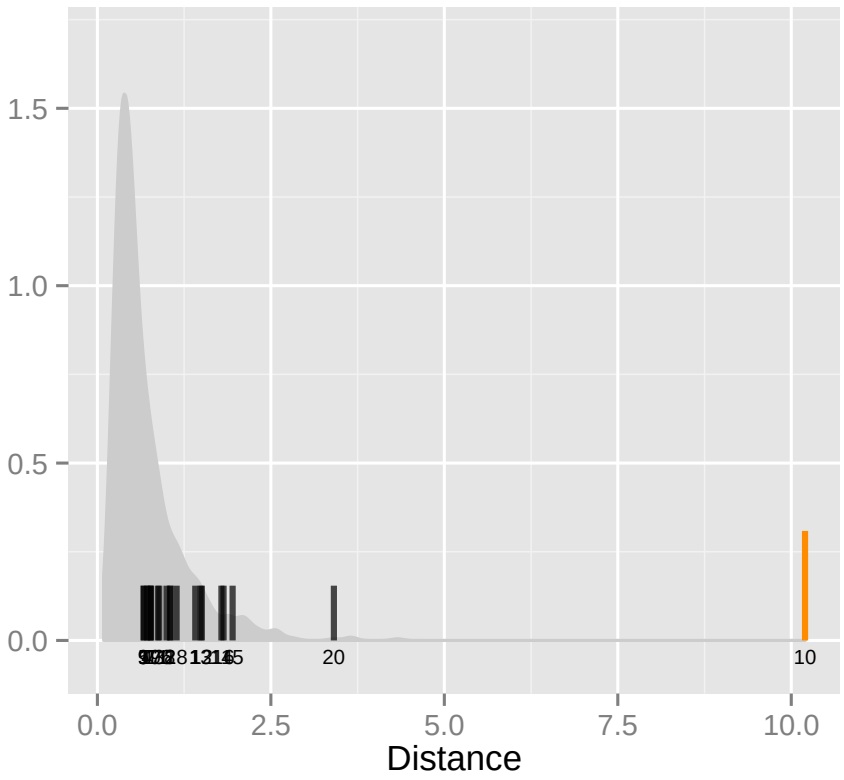

Figure 5: Distribution of the **regression based distance** with the distance for the null plots of the lineup represented in black and the distance for the plot of the actual data represented by orange. The distance for the actual plot is much higher than the null plots and the actual plot can be easily identified as seen in Figure 3.

The empirical distribution of the binned distance metric is plotted. The number of bins used is 6 on both axes. The bins are selected randomly in this case which had a huge effect on the binned distance. Figure 6 shows that the plot of the actual data is unidentifiable which is incorrect.

```
> dist.vals2 <- distmet(lineup.dat, var = c('mpg', 'wt'), 'bin_dist', null_permute('mpg'),
pos = 10, repl = 1000, dist.arg = list(lineup.dat = lineup.dat, X.bin = 6, Y.bin = 6))
> distplot(dist.vals2)
```

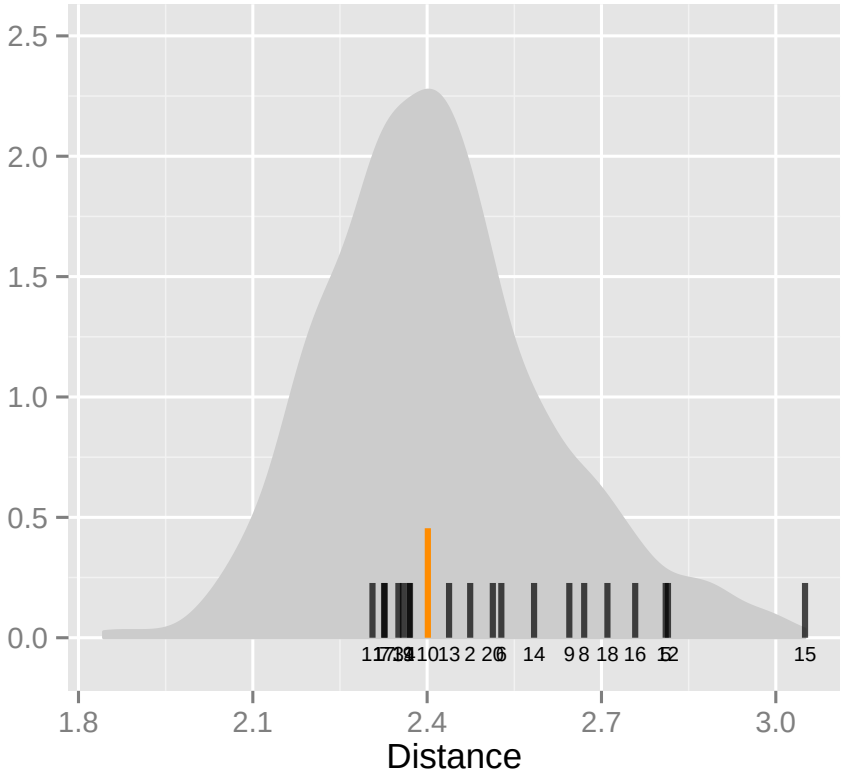

Figure 6: Distribution of the **binned distance** with the distance for the null plots of the lineup represented in black and the distance for the plot of the actual data represented by orange. The number of bins used is 6 on both axes. The distance for the actual plot falls within the distances for the null plots. This shows that the lineup is difficult in the sense that the actual cannot be easily identified. Hence the selection of the number of bins is important for binned distance.

The empirical distribution of the binned distance metric is plotted. The number of bins used is 3 on x-axis and 2 on y- axis. The bins are selected using the optimum bin selection procedure. Figure 7 shows that the plot of the actual data can easily be identifiable.

```
> dist.vals3 <- distmet(lineup.dat, var = c('mpg', 'wt'), 'bin_dist', null_permute('mpg'),
pos = 10, repl = 1000, dist.arg = list(lineup.dat = lineup.dat, X.bin = 3, Y.bin = 2))
> distplot(dist.vals3)
```

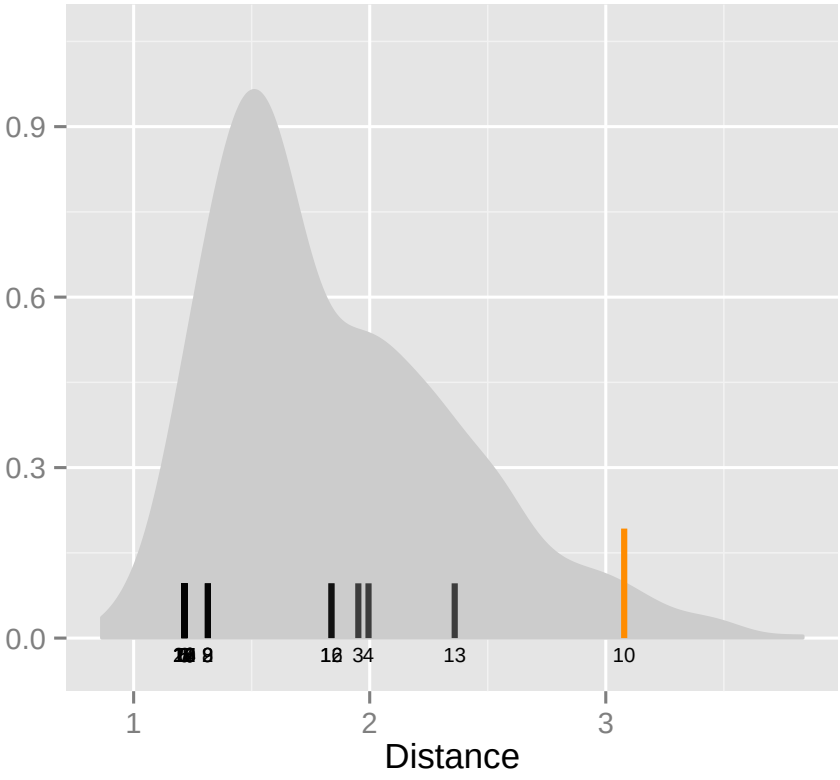

Figure 7: Distribution of the **binned distance** with the distance for the null plots of the lineup represented in black and the distance for the plot of the actual data represented by orange. The number of bins used is 3 on x-axis and 2 on y-axis. The number of bins are selected by the optimal bin selection method using Figure 4. The distance for the actual plot is larger than the distances for the null plots which matches the lineup.

References

Buja A, Cook D, Hofmann H, Lawrence M, Lee E, Swayne D, Wickham H (2009). "Statistical Inference for Exploratory Data Analysis and Model Diagnostics." *Royal Society Philosophical Transactions A*, **367**(1906), 4361–4383.

Hofmann H, Majumder M, Cook D (2013). *Experiments for Visual Inference*. http://www.public.iastate.edu/~hofmann/experiments.html, Accessed: 2015-08-31.

Wickham H, Cook D, Hofmann H, Buja A (2010). "Graphical Inference for Infovis." *IEEE Transactions on Visualization and Computer Graphics*, **16**.

Affiliation:

Niladri Roy Chowdhury
Novartis Pharmaceuticals
East Hanover, New Jersey, USA
E-mail: niladri.ia@gmail.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd