GenerateFakeClassificationData

June 26, 2019

- The Python package rpy2 has an RMagic module that facilitates using R inside a Jupyter notebook running Python

```
In [1]: # Load the RMagic extension into Jupyter Notebook
        %load_ext rpy2.ipython
```

Any Jupyter notebook cell that contains R code must be prefaced by the %%R "magic" statement.

```
In [2]: %%R
        library(tidyverse)
        library(GGally)
```

```
R[write to console]: Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang

R[write to console]: Registered S3 method overwritten by 'rvest':
  method            from
  read_xml.response xml2

R[write to console]:  Attaching packages  tidyverse 1.2.1

R[write to console]:  ggplot2 3.1.1      purrr   0.3.2
 tibble  2.1.1      dplyr   0.8.1
 tidyr   0.8.3      stringr 1.4.0
 readr   1.3.1      forcats 0.4.0

R[write to console]:  Conflicts  tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()

R[write to console]: Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
R[write to console]:
Attaching package: GGally


R[write to console]: The following object is masked from package:dplyr:

    nasa
```

1 Make funky data

- We'll use the Python package Scikit-learn
- The module sklearn.datasets has a function called make_classification()
- Creates clusters of points
- Points are normally distributed about vertices in N-dimensional hypercube
  - N is the number of informative dimensions
  - Cube side length = 2 * class_sep

```
In [3]: from sklearn.datasets import make_classification, make_moons, make_circles
        import pandas as pd
```

```
In [4]: Xgauss, Y = make_classification(
            n_samples=200,
            n_features=2,
            n_informative=2,
            n_redundant=0,
            n_repeated=0,
            n_classes=2,
            n_clusters_per_class=1,
            weights=None,
            flip_y=0.00,
            class_sep=0.6,
            hypercube=True,
            shift=0.0,
            scale=1.0,
            shuffle=False,
            random_state=None,)
```

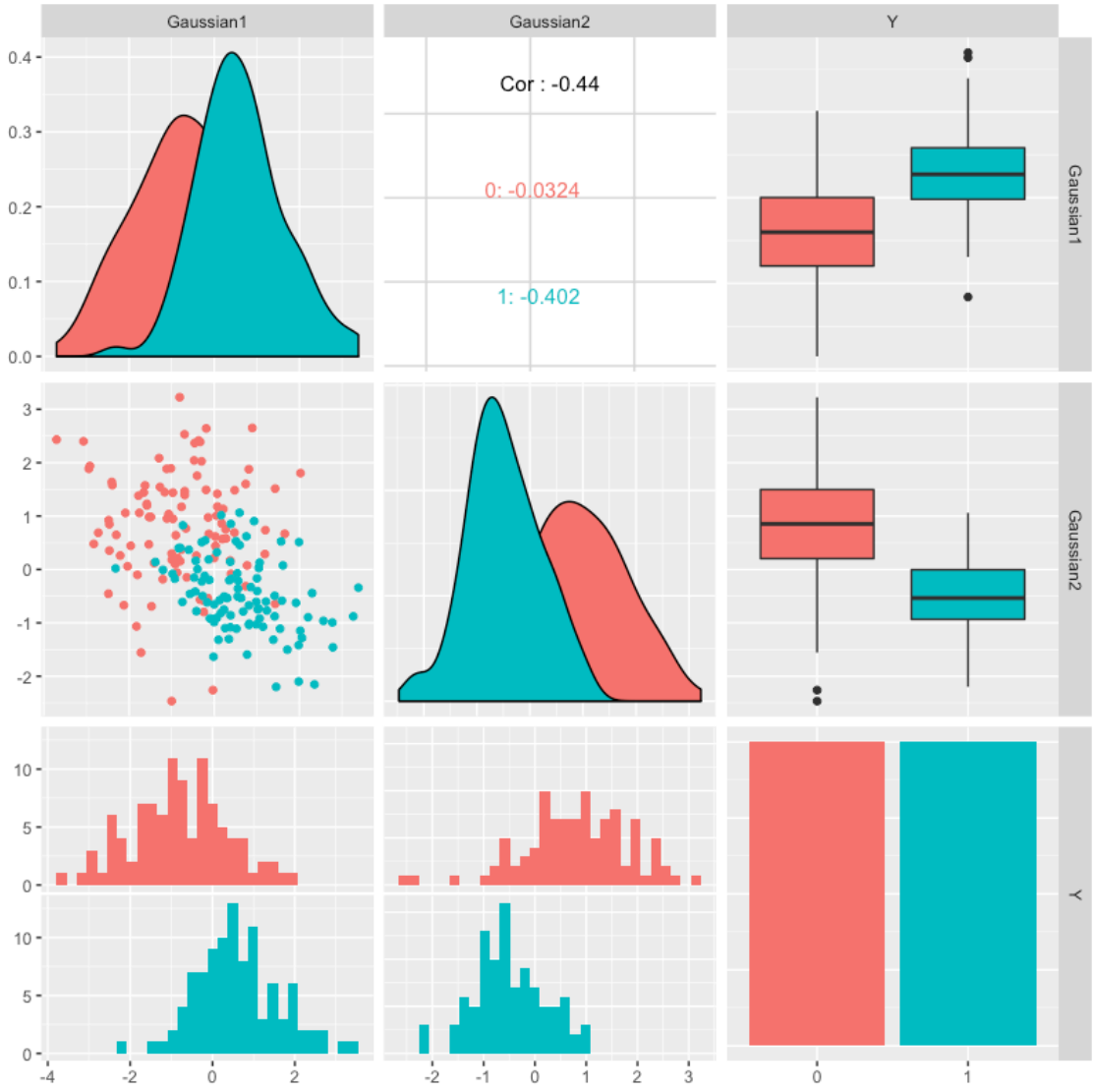
1.1 Create a Python DataFrame

```
In [5]: data1 = pd.DataFrame( Xgauss )
        data1.columns = ['Gaussian1', 'Gaussian2']
        # add intercepts
        #data1['Gaussian1'] += 10
        #data1['Gaussian2'] -= 10
        data1[ 'Y' ] = Y
```

1.2 Pass the data to R and make a scatterplot matrix

```
In [6]: %%R -i data1 -w 8 -h 8 -u in -r 100
        data1$Y <- factor( data1$Y )
        data1 %>% ggpairs( aes( color=Y ) )
```

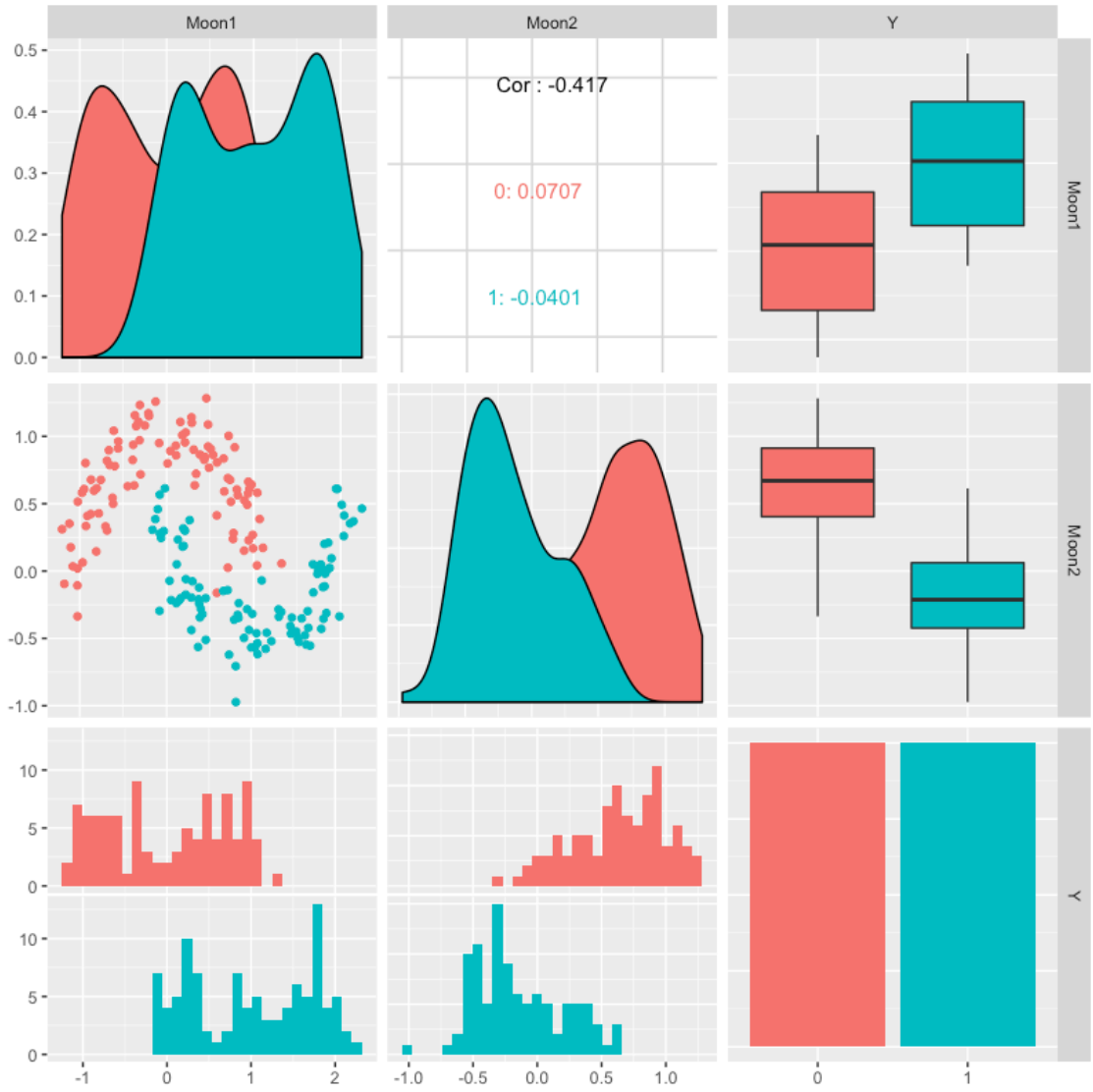
```
R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
In [7]: data1.drop( columns='Y', inplace=True )
```

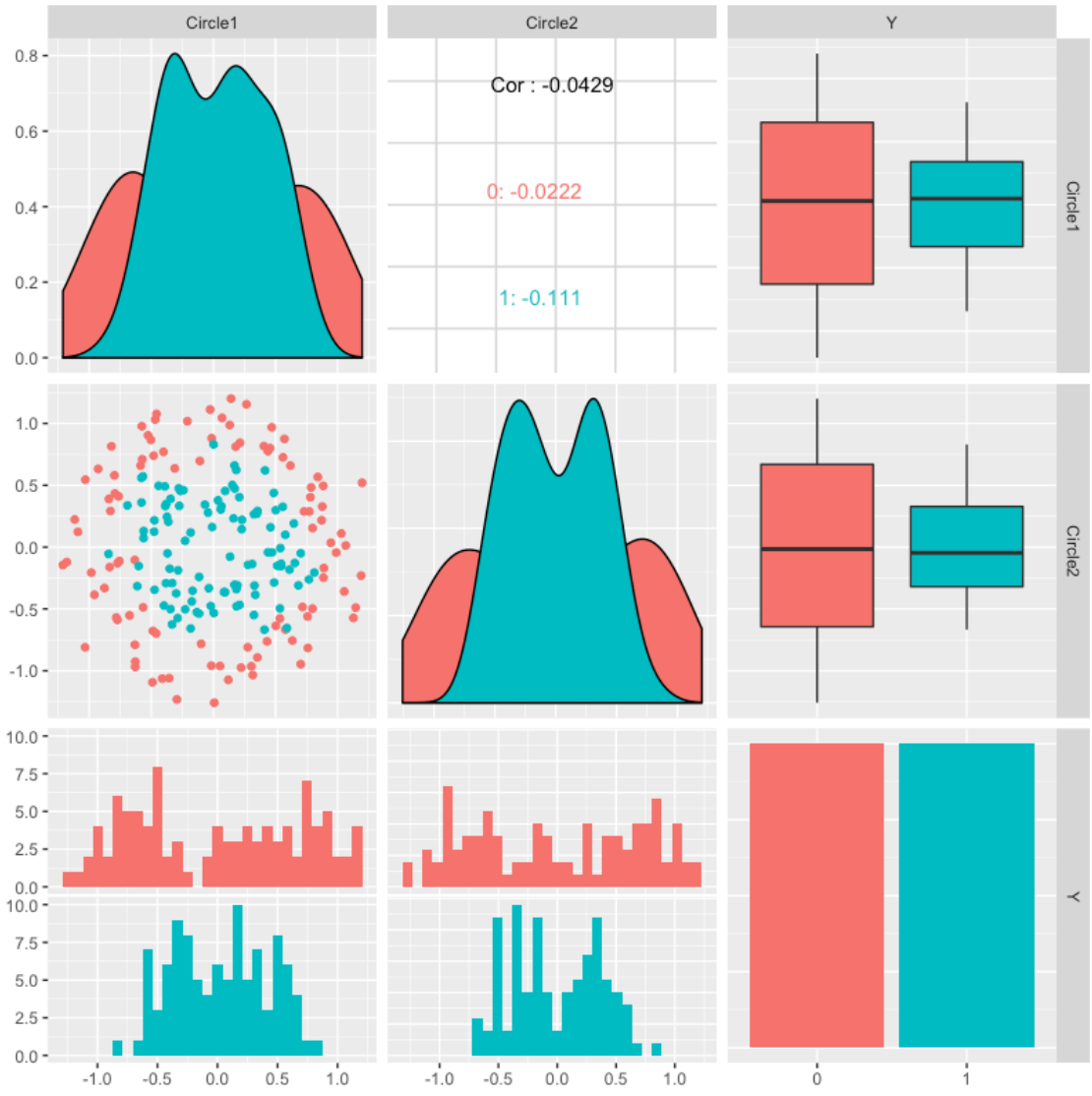
```
In [8]: Xmoon, Y = make_moons( n_samples=200, shuffle=False, noise=0.15, random_state=None)

In [9]: data2 = pd.DataFrame( Xmoon )
        data2.columns = ['Moon1', 'Moon2']
        data2[ 'Y' ] = Y

In [10]: %%R -i data2 -w 8 -h 8 -u in -r 100
         data2$Y <- factor( data2$Y )
         data2 %>% ggpairs( aes( color=Y ) )
```

```
R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

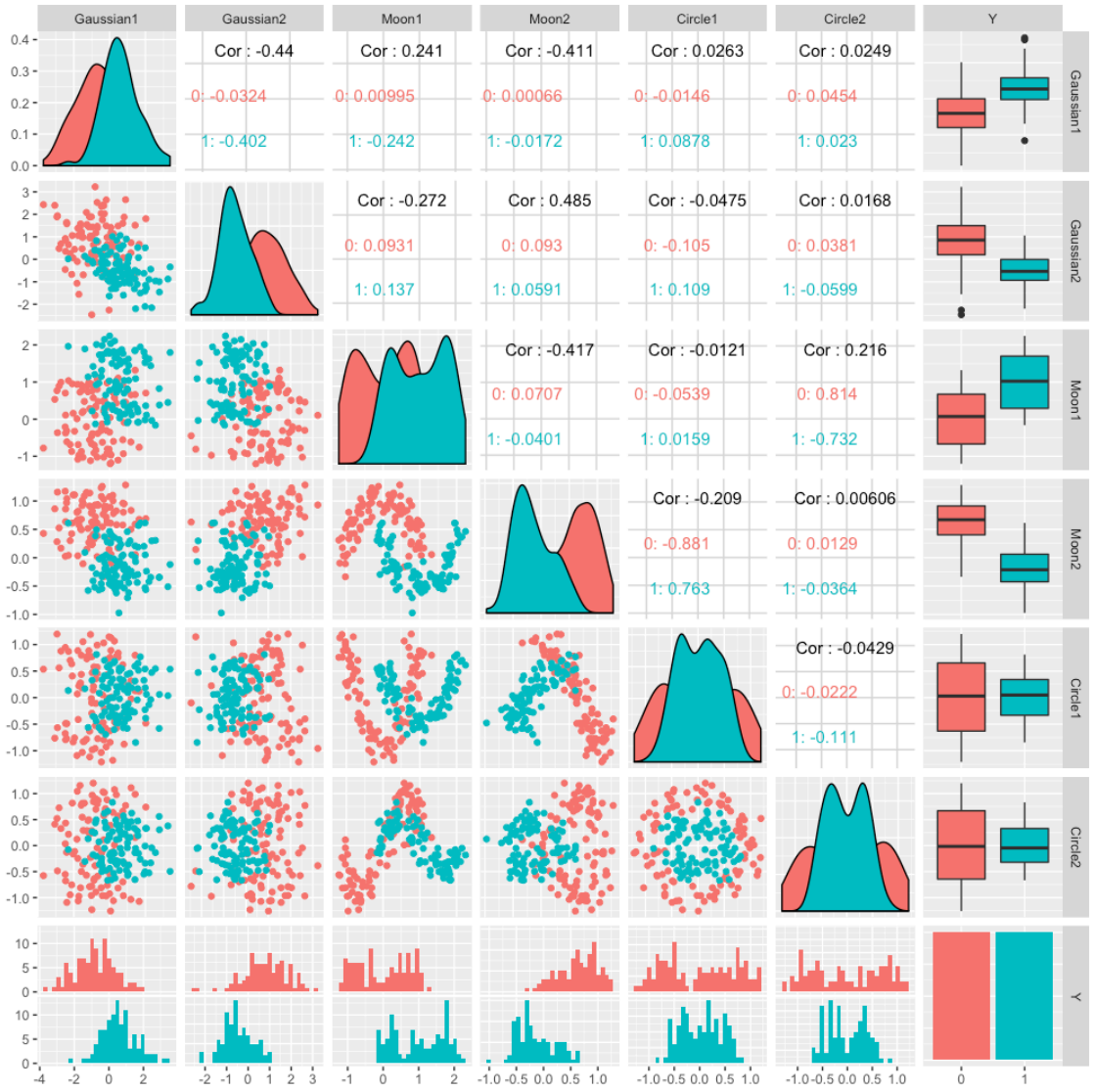


```
In [11]: data2.drop( columns='Y', inplace=True )

In [12]: Xcircle, Y = make_circles( n_samples=200, shuffle=False, noise=0.15, random_state=None

In [13]: data3 = pd.DataFrame( Xcircle )
         data3.columns = ['Circle1', 'Circle2']
         data3[ 'Y' ] = Y

In [14]: %%R -i data3 -w 8 -h 8 -u in -r 100
         data3$Y <- factor( data3$Y )
         data3 %>% ggpairs( aes( color=Y ) )
```

```
R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
In [15]: data = pd.concat( (data1, data2, data3), axis=1)

In [16]: data.shape

Out[16]: (200, 7)

In [17]: data.columns

Out[17]: Index(['Gaussian1', 'Gaussian2', 'Moon1', 'Moon2', 'Circle1', 'Circle2', 'Y'], dtype=

In [18]: %%R -i data -w 10 -h 10 -u in -r 100
         data$Y <- factor( data$Y )
         data %>% ggpairs( aes( color=Y ) )

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
In [19]: data.sample(frac=1).to_csv( 'funkydata.csv', index=False )
```

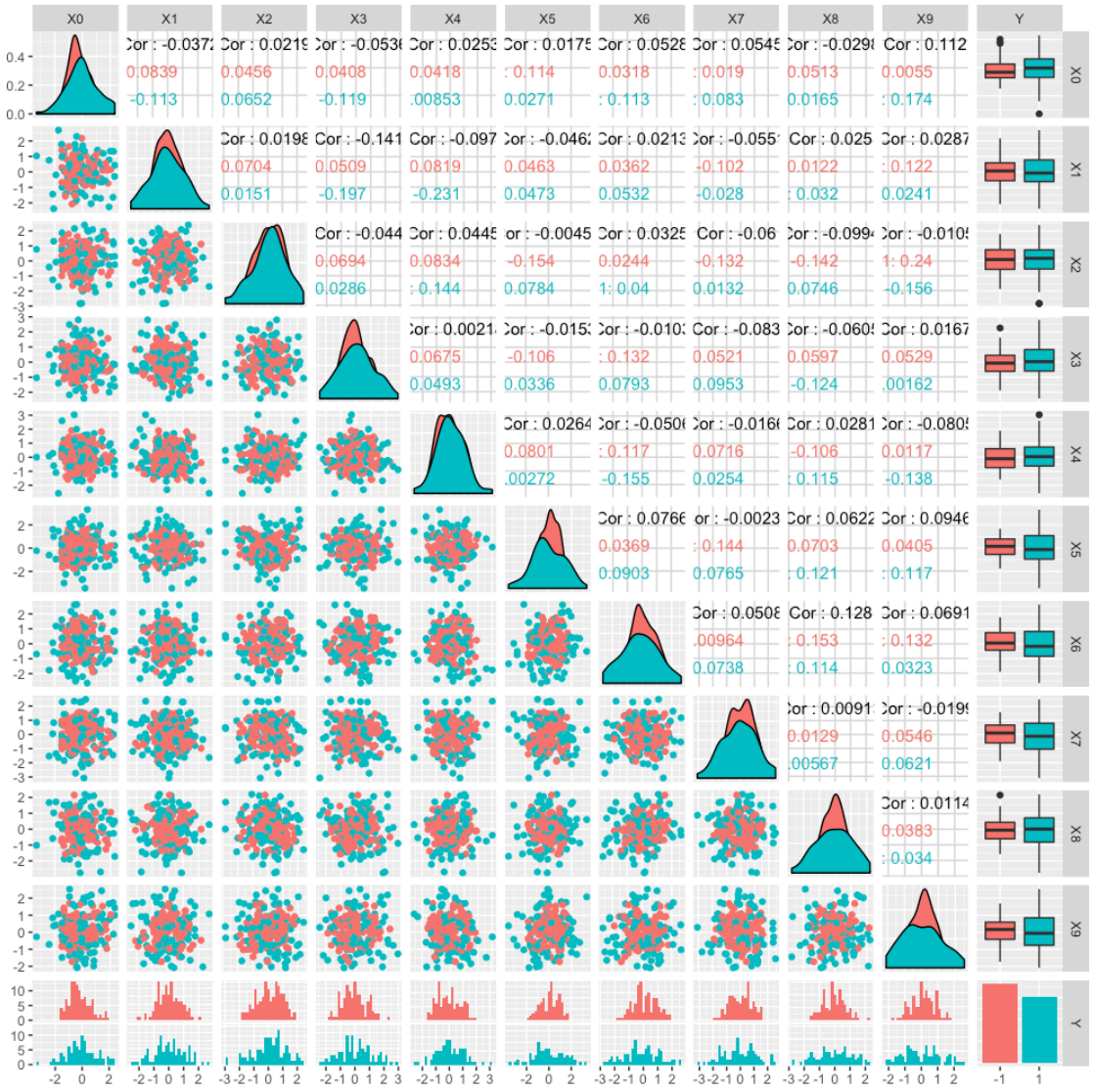
2 Make Unequal variance data

```
In [20]: from sklearn.datasets import make_hastie_10_2

In [21]: Xhastie, y = make_hastie_10_2( 200 )

In [22]: Xhastie.shape

Out[22]: (200, 10)

In [23]: data4 = pd.DataFrame( Xhastie )
```

```
In [24]: data4['Y'] = y
```

```
In [25]: %%R -i data4 -w 10 -h 10 -u in -r 100
         data4$Y <- factor( data4$Y )
         data4 %>% ggpairs( aes( color=Y ) )
```

```
R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
In [26]: data4.sample(frac=1).to_csv( 'unequal_variance_data.csv', index=False )
```