IP-2019.1 data statistics

Save this "**2019.1.statistics.ipynb file**" to Google Drive and open it with Colab.

```
import pylab
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns, numpy as np
from google.colab import files

# to save and download figures of paper
global SAVE_FIGS_PAPER
SAVE_FIGS_PAPER = True
if SAVE_FIGS_PAPER:
  DPI_resolution = 200
  width_resolution = 2400
  height_resolution = 1600
  !pip install plotly>=4.0.0
  !wget https://github.com/plotly/orca/releases/download/v1.2.1/orca-1.2.1-x86_64
  !chmod +x /usr/local/bin/orca
  !apt-get install xvfb libgtk2.0-0 libgconf-2-4
```

```
--2020-12-25 21:24:00--  https://github.com/plotly/orca/releases/download/v1.2.1
Resolving github.com (github.com)... 140.82.121.3
Connecting to github.com (github.com)|140.82.121.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/990372
--2020-12-25 21:24:01--  https://github-production-release-asset-2e65be.s3.amazo
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-produc
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-pr
HTTP request sent, awaiting response... 200 OK
Length: 51607939 (49M) [application/octet-stream]
Saving to: '/usr/local/bin/orca'

/usr/local/bin/orca 100%[===================>]  49.22M  21.1MB/s    in 2.3s

2020-12-25 21:24:03 (21.1 MB/s) - '/usr/local/bin/orca' saved [51607939/51607939

Reading package lists... Done
```

Salvo com sucesso

```
Done
ewest version (2.24.32-1ubuntu1).
libgconf-2-4 is already the newest version (3.2.6-4ubuntu1).
xvfb is already the newest version (2:1.19.6-1ubuntu4.8).
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.
```

PE deliveries from ALL students

```
file1 = 'http://vision.ufabc.edu.br/MCTest/public/CAE2021/2019.1.PE.csv'
df_PE = pd.read_csv(file1, decimal=".", sep=',')
df_PE['PE'] = df_PE['PE']*10
df_PE.sort_values(by=['Class'], inplace=True)
df_PE.sample(5)
```

	Class	ProfT	ProfP	PE	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8	PE9	PE10	P
874	32	prof11	prof17	0.0	0	0	0	0	0	0	0	0	0	0	
130	4	prof01	prof20	0.0	0	0	0	0	0	0	0	0	0	0	
840	30	prof11	prof02	0.0	0	0	0	0	0	0	0	0	0	0	
906	33	prof11	prof27	0.0	0	0	0	0	0	0	0	0	0	0	
1041	37	prof11	prof27	0.0	0	0	0	0	0	0	0	0	0	0	

```
limiar = 0  # WE CONSIDER ONLY MEDIAN LESSONS IN EP > 0

median = df_PE.groupby(['Class'])['PE'].median()
v1 = [str(i) for i in list(median[median>limiar].index.values)]
dfVPL_PE = df_PE.query('Class == ' + str(v1))
print('VPL-PE:' + str(v1))

v2 = [str(i) for i in list(median[median<=limiar].index.values)]
v3 = v2[-2:] # v3 in IP-BL
v2 = v2[:-2] # remove IP-BL
dfVPL_NOT_PE = df_PE.query('Class == ' + str(v2))
print('VPL-NOT-PE:' + str(v2))

dfVPL_NOT_BL = df_PE.query('Class == ' + str(v3))
print('VPL-BL:' + str(v3))
```

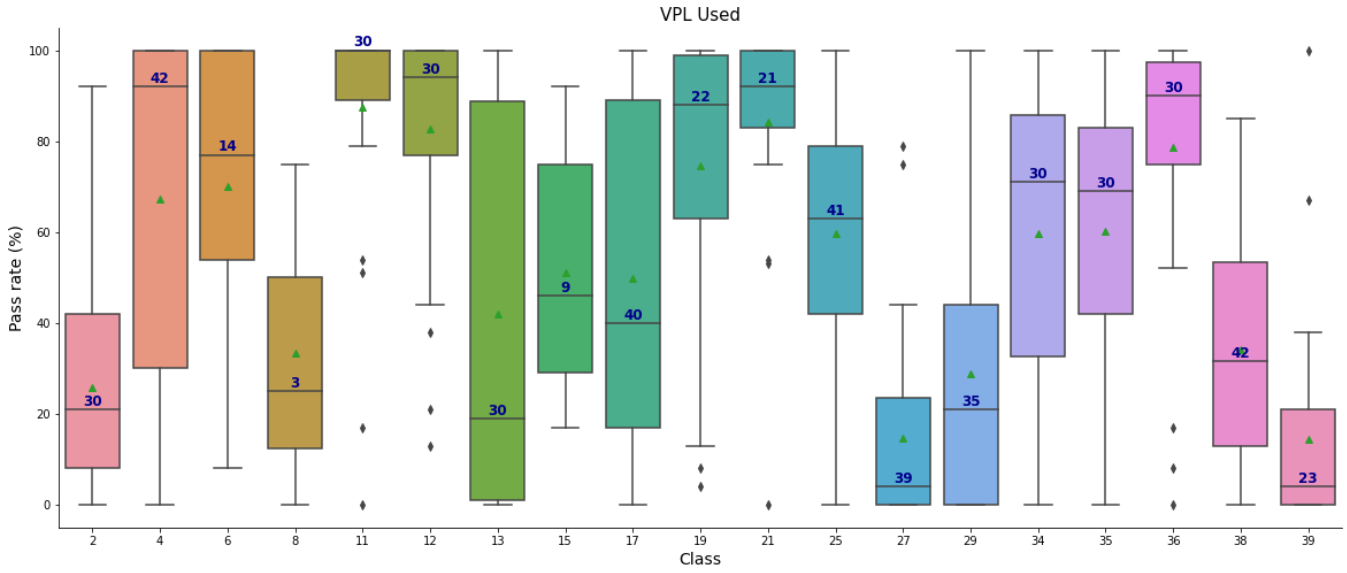
```
VPL-PE:['2', '4', '6', '8', '11', '12', '13', '15', '17', '19', '21', '25', '27'
VPL-NOT-PE:['1', '3', '5', '7', '9', '10', '14', '16', '18', '20', '22', '23', '
VPL-BL:['45', '46']
```

```
def draw_VPL_Used(dfVPL, title):
    plt.rcParams['figure.figsize'] = (20.0, 8.0)

    ax = sns.boxplot(x="Class", y="PE", data=dfVPL, showmeans=True)

    # Calculate number of obs per group & median to position labels
    medians = dfVPL.groupby(['Class'])['PE'].median().values
    nobs = dfVPL.groupby(['Class'])['PE'].count()
    nobs = [str(x) for x in nobs.tolist()]

    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
```

Salvo com sucesso

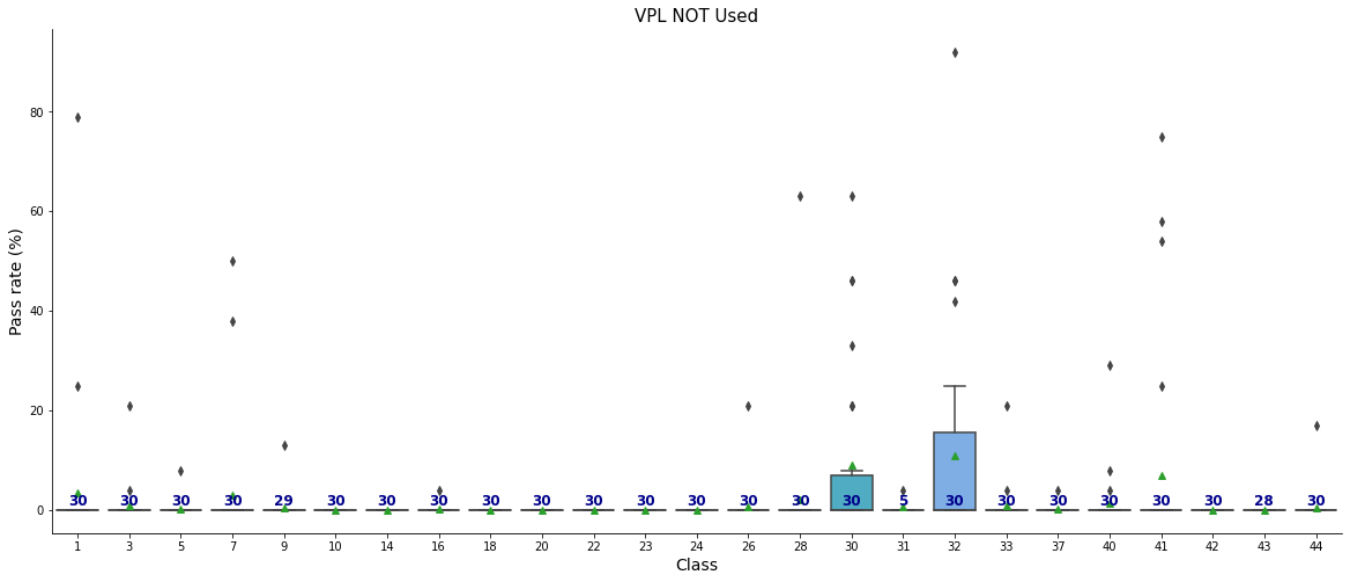
```
    ax.spines['right'].set_visible(False)

    plt.title(title, size='15')

    pos = range(len(nobs))
    for tick,label in zip(pos,ax.get_xticklabels()):
      ax.text(pos[tick], medians[tick] + 0.9, nobs[tick], horizontalalignment='cent

    ax.set_xlabel("Class", fontsize=14)
    ax.set_ylabel("Pass rate (%)", fontsize=14)
    ax.plot()
```

```
title = 'VPL Used'
draw_VPL_Used(dfVPL_PE,title)
```

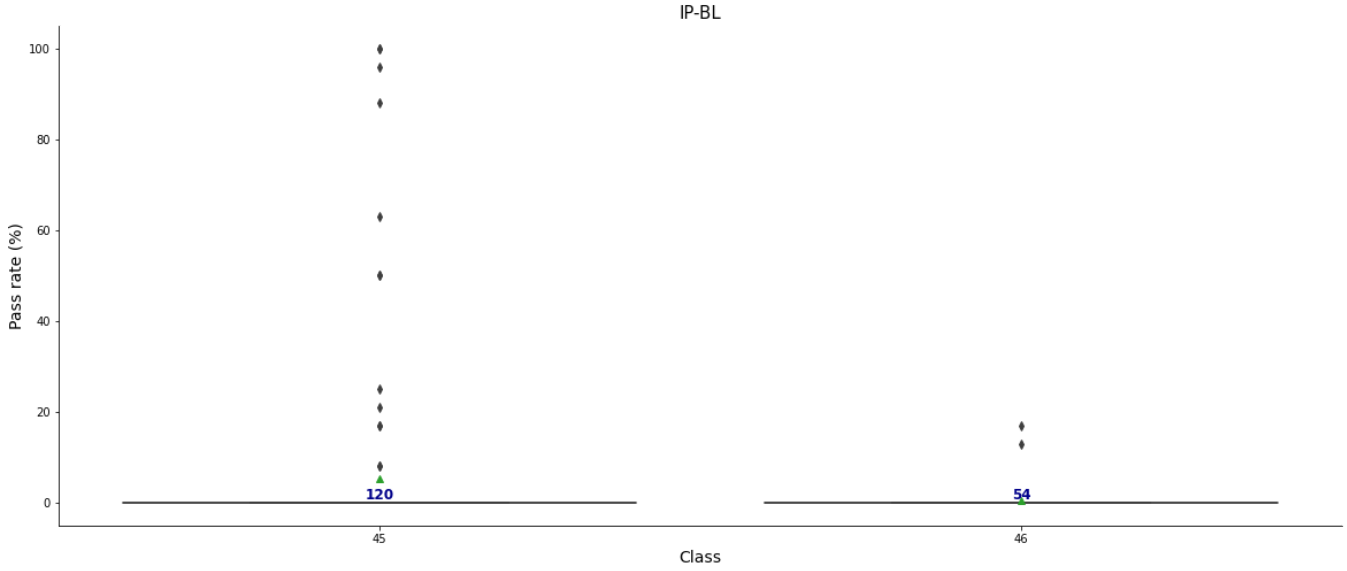


Salvo com sucesso

```
itle)
```




```
1   title = 'IP-BL'
2   draw_VPL_Used(dfVPL_NOT_BL,title)
```



Salvo com sucesso

In the Boxplot graph, the line in the middle of each rectangle is the median, in red is the number of students who taught some PE.

Cris Sato's explanation of the boxplot:

The box chart presents several useful information:

- The green line inside the box is the median
- The bottom line of the box is the first quartile (Q1), that is, the 25th percentile.
- The top line of the box is the third quartile (Q3), that is, the 75th percentile.
- the lower dash in black is the lowest value in the data above $Q1 - 1.5IQR$, where $IQR = Q3 - Q1$.
- the upper dash in black is the highest value in the data below $Q3 + IQ$ 1.5

The IQR value is called ** interquartile range **.

The points below the bottom line or above the top line are called ** outliers **.

Final grades for ALL 46 classes (academic system)

```
file2 = 'http://vision.ufabc.edu.br/MCTest/public/CAE2021/2019.1.grade.csv'
df_grade = pd.read_csv(file2, decimal=".")
df_grade.sort_values(by=['Adopted','Class'], inplace=True)
df_grade['Dropout'] = df_grade['Dropout'].fillna(0)
df_grade['New'] = df_grade['New'].fillna(0)

col_pass_rate = 'Pass rate (%)'
df_grade.rename(columns={"Approved": col_pass_rate}, inplace=True)

df_grade[col_pass_rate] = df_grade[col_pass_rate]*100

for i in v1:
  df_grade.loc[df_grade['Class'] == int(i), 'Adopted'] = 'VPL Used'

for i in v2:
  df_grade.loc[df_grade['Class'] == int(i), 'Adopted'] = 'VPL NOT Used'

df_grade.head()
```

Salvo com sucesso

	Class	A	B	C	D	F	O	Total	Dropout	New	Total-Moodle	Total-diff	Pass rate (%)	PE	D/
0	1	4	5	2	2	6	9	28	2.0	1.0	31	2	46.428571	NaN	[
2	3	3	5	2	3	9	3	25	0.0	0.0	31	6	52.000000	NaN	[

We include a column with Total-Moodle, including the total number of students enrolled in Moodle

The academic system (PROGRAD) does not offer the exact number of dropouts. We block students and new students in the second week (Adjustment). Thus, abandonment is the official + total difference - Dif2 - Adjustment

8	9	2	7	9	3	5	2	28	0.0	0.0	29	1	75.000000	NaN	[

```
stud_moodle = np.array(df_PE.groupby(['Class'])['PE'].count())

df_grade.sort_values(by=['Class'], inplace=True)
df_grade['Total-Moodle2'] = stud_moodle  # what's in moodle - dropout + new
df_grade.sort_values(by=['Adopted','Class'], inplace=True)

df_grade['Total-Diff2'] = df_grade['Total-Moodle2'] - df_grade['Total'] # diff Mo

df_grade['Dropout'] += df_grade['Total-Moodle2'] - df_grade['Total'] +  df_grade[
df_grade['Dropout (%)'] = df_grade['Dropout'] / df_grade['Total'] * 100

df_grade_sort = df_grade.sort_values(by=['Adopted', 'Class'])
dfVPL = df_grade_sort.query('Adopted == "VPL Used" and Class != 45 and Class != 4
dfVPL_BL = df_grade_sort.query('Class == 45 or Class == 46')
dfVPLNOT = df_grade_sort.query('Adopted == "VPL NOT Used"')

print("Total-Moodle2 %d " % df_grade['Total-Moodle2'].sum())
print("Total-Moodle (new+dropout): %d " % df_grade['Total-Moodle'].sum())
print("Total PROGRAD: %d " % df_grade['Total'].sum())
print("Dropout: %d " % df_grade['Dropout'].sum())
print("New: %d " % df_grade['New'].sum())
_grade['Total-diff'].sum())
f_grade['Total-Diff2'].sum())
df_grade.sample(5)
```

Salvo com sucesso

```
Total-Moodle2 1437
Total-Moodle (new+dropout): 1466
Total PROGRAD: 1343
Dropout: 148
New: 27
Total-diff: 123
Total-diff2: 94
```

	Class	A	B	C	D	F	O	Total	Dropout	New	Total-Moodle	Total-diff	Pass rate (%)	PE	I
42	43	1	3	2	3	15	3	27	7.0	3.0	31	4	33.333333	NaN	
17	18	2	2	4	3	9	4	24	9.0	1.0	31	6	45.833333	NaN	
23	24	1	2	3	2	18	0	26	6.0	1.0	31	5	30.769231	NaN	
35	36	5	15	6	2	2	0	30	0.0	0.0	30	0	93.333333	7.88	

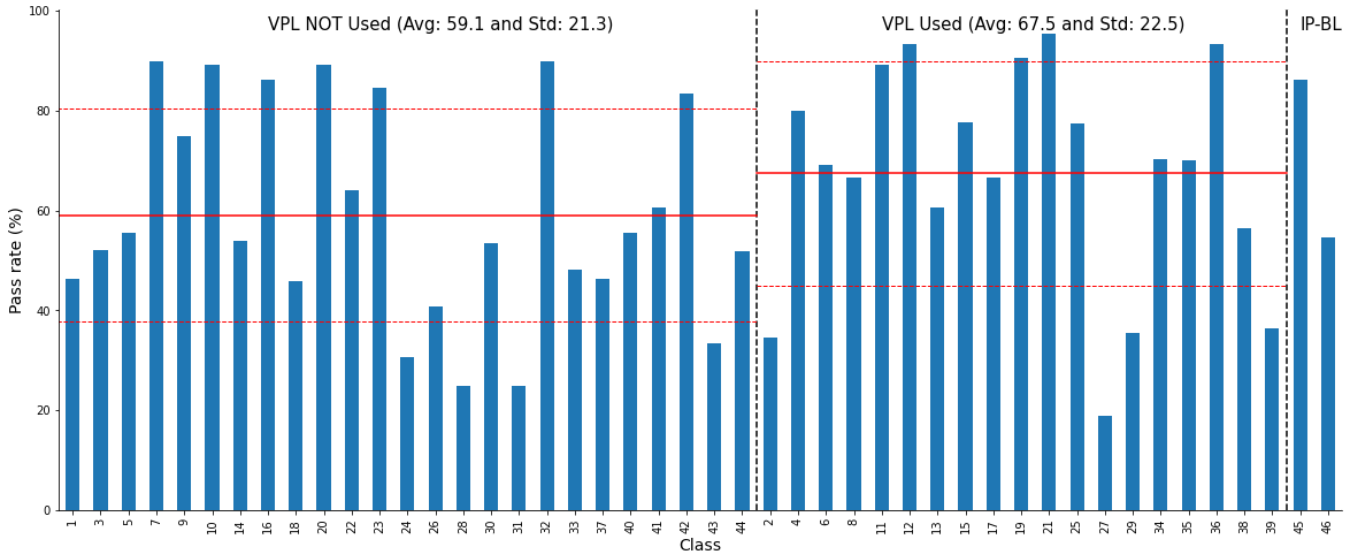
```
def drawGraphs(df_grade, dfVPL, dfVPLNOT, col, file):

  xy = df_grade[['Class',col]]
  nobs = df_grade[col].values

  # removo valores negativos???
  if col == 'Dropout (%)':
    xy[col][xy[col] < 0] = 0

  ax = xy.plot(kind='bar',x='Class',y=col)
  ax.get_legend().remove()

  x = 24.5
  x1 = 12
  x2 = 36.3
  x3 = 43.5

  plt.axvline(x, color="k", linestyle="--");
  plt.axvline(x2, color="k", linestyle="--");

  NOTusedMean = dfVPLNOT[col].mean()
  y1 = NOTusedMean
  plt.plot([-1, x], [y1, y1], color='red', linewidth=1.5, linestyle="-")
  #ax.annotate('Avg: %.1f'% y1, xy=(x1, y1+3), size='15', color='red')

  NOTusedSTD = dfVPLNOT[col].std()
  y1 = NOTusedMean+NOTusedSTD
```

Salvo com sucesso

```
    plt.plot([-1, x], [y1, y1], color='red', linewidth=1, linestyle="--")
    #ax.annotate('+Std: %.1f'% y1, xy=(x1, y1+3), size='12', color='red')

    y1 = NOTusedMean-NOTusedSTD
    plt.plot([-1, x], [y1, y1], color='red', linewidth=1, linestyle="--")
    #ax.annotate('-Std: %.1f'% y1, xy=(x1, y1+4), size='12', color='red')

    #### Used

    usedMean = dfVPL[col].mean()
    y1 = usedMean
    plt.plot([x, x3], [y1, y1], color='red', linewidth=1.5, linestyle="-")
    #ax.annotate('Avg: %.1f'% y1, xy=(x2, y1+3), size='15', color='red')

    usedSTD = dfVPL[col].std()
    y1 = usedMean+usedSTD
    plt.plot([x, x3], [y1, y1], color='red', linewidth=1, linestyle="--")
    #ax.annotate('+Std: %.1f'% y1, xy=(x2, y1+3), size='12', color='red')

    y1 = usedMean-usedSTD
    plt.plot([x, x3], [y1, y1], color='red', linewidth=1, linestyle="--")
    #ax.annotate('-Std: %.1f'% y1, xy=(x2, y1+3), size='12', color='red')

    y0 = nobs.max() + 0.6

    ax.annotate('VPL NOT Used (Avg: %.1f and Std: %.1f)' % (NOTusedMean, NOTusedSTI
    ax.annotate('VPL Used (Avg: %.1f and Std: %.1f)' % (usedMean, usedSTD), xy=(29,
    ax.annotate('IP-BL', xy=(44, y0), size='15')
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    #ax.grid(linewidth=.5)

    ax.set_xlabel("Class", fontsize=14)
    ax.set_ylabel(col, fontsize=14)
    ax.plot()

    if SAVE_FIGS_PAPER:
      plt.savefig(file, dpi=DPI_resolution)
      files.download(file)
    print('\n\n')
```

Salvo com sucesso

```
VPLNOT,col_pass_rate, 'fig02new.tif')
```



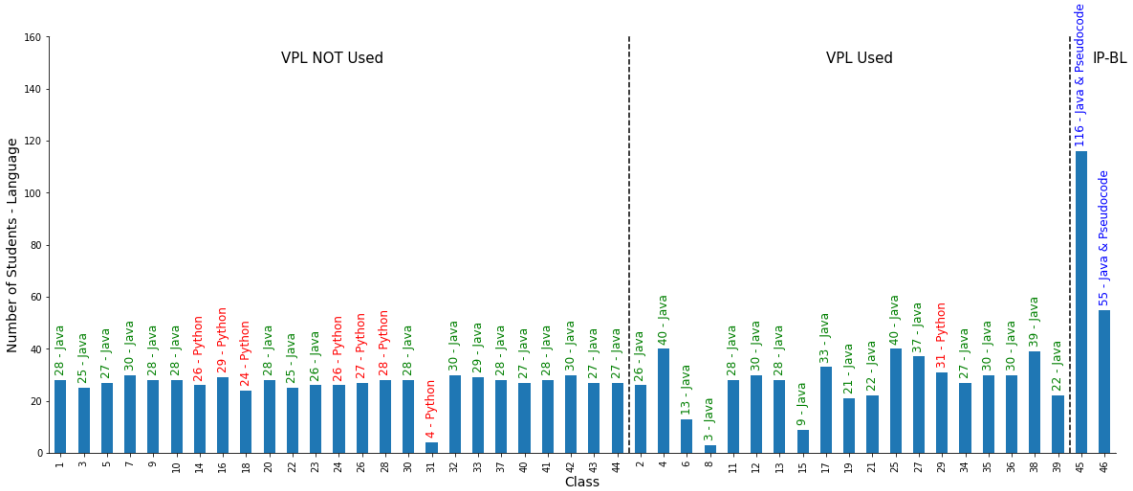

```
1  drawGraphs(df_grade,dfVPL,dfVPLNOT,'Dropout (%)', 'fig06new.tif')
```

Salvo com sucesso

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab

/usr/local/lib/python3.6/dist-packages/pandas/core/series.py:1021: SettingWithCo
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
```

```
xyz = df_grade_sort[['Class','Total','Language']]
ax = xyz.plot(kind='bar',x='Class',y='Total')
ax.set_ylim([0,160])
ax.get_legend().remove()

x = 24.5
x1 = 12
x2 = 36.3
x3 = 43.5

pos = range(len(xyz['Language']))
for tick,label in zip(pos,ax.get_xticklabels()):
  y0 = xyz['Total'].loc[int(label.get_text())-1]
  lang0 = xyz['Language'].loc[int(label.get_text())-1]
  lang = str(y0) + ' - ' + lang0
  # [ 'normal' | 'bold' | 'heavy' | 'light' | 'ultrabold' | 'ultralight']
  #if int(y)<60:
  if lang0 == 'Java':
    ax.text(pos[tick], y0 + 3, lang, horizontalalignment='center', size='12', col
  elif lang0 == 'Java & Pseudocode':
    ax.text(pos[tick], y0 + 3, lang, horizontalalignment='center', size='12', col
  else:
    ax.text(pos[tick], y0 + 3, lang, horizontalalignment='center', size='12', col


plt.axvline(x, color="k", linestyle="--");
plt.axvline(x3, color="k", linestyle="--");

ax.annotate('VPL NOT Used', xy=(9.5, 150), size='15')
ax.annotate('VPL Used', xy=(33, 150), size='15')
ax.annotate('IP-BL', xy=(44.5, 150), size='15')

                                                (False)
                                                le(False)

ax.set_xlabel("Class", fontsize=14)
ax.set_ylabel("Number of Students - Language", fontsize=14)

if SAVE_FIGS_PAPER:
  plt.savefig('fig05new.tif', dpi=DPI_resolution)
  files.download('fig05new.tif')
```

Salvo com sucesso




```
1  mean = dfVPL.groupby(['Class'])['PE'].mean()
2  nobs = dfVPL.groupby(['Class'])['PE'].count()
```

```
1  col,lin = df_grade.shape
2  x = []
3  for i in range(0,col):
4    v = df_grade[['Class',col_pass_rate]].loc[i]
5    try:
6      x.append([i+1,v[1],round(mean[v[0]],1),nobs[v[0]]])
7    except:
8      pass
```

Salvo com sucesso

```
array([[ 2.      , 34.615385,  2.6     ,  1.      ],
       [ 4.      , 80.      ,  6.8     ,  1.      ],
       [ 6.      , 69.230769,  7.      ,  1.      ],
       [ 8.      , 66.666667,  3.3     ,  1.      ],
       [11.      , 89.285714,  8.8     ,  1.      ],
       [12.      , 93.333333,  8.3     ,  1.      ],
       [13.      , 60.714286,  4.2     ,  1.      ],
       [15.      , 77.777778,  5.2     ,  1.      ],
       [17.      , 66.666667,  5.      ,  1.      ],
```

```
       [19.      , 90.47619 ,  7.1     ,  1.      ],
       [21.      , 95.454545,  8.4     ,  1.      ],
       [25.      , 77.5     ,  6.      ,  1.      ],
       [27.      , 18.918919,  1.5     ,  1.      ],
       [29.      , 35.483871,  2.9     ,  1.      ],
       [34.      , 70.37037 ,  6.      ,  1.      ],
       [35.      , 70.      ,  6.      ,  1.      ],
       [36.      , 93.333333,  7.9     ,  1.      ],
       [38.      , 56.410256,  3.4     ,  1.      ],
       [39.      , 36.363636,  1.4     ,  1.      ]])
```

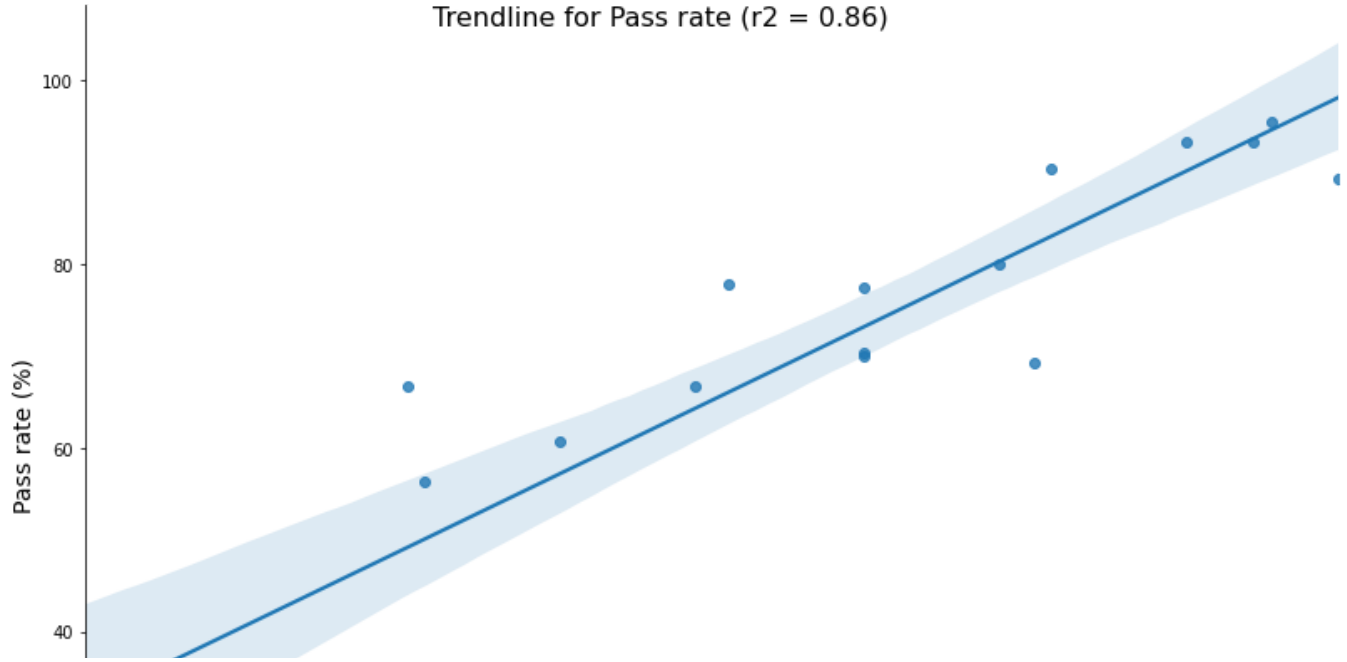
```
data = {'Class': x[:,0].astype(int),
        'Pass rate': x[:,1],
        'PE': x[:,2],
        'N' : x[:,3].astype(int)}

grade_vpl = pd.DataFrame(data)
```

```
correl = grade_vpl['Pass rate'].corr(grade_vpl['PE'])
r2 = correl**2
print(correl, r2)
```

```
0.9293857826418399 0.8637579329767853
```

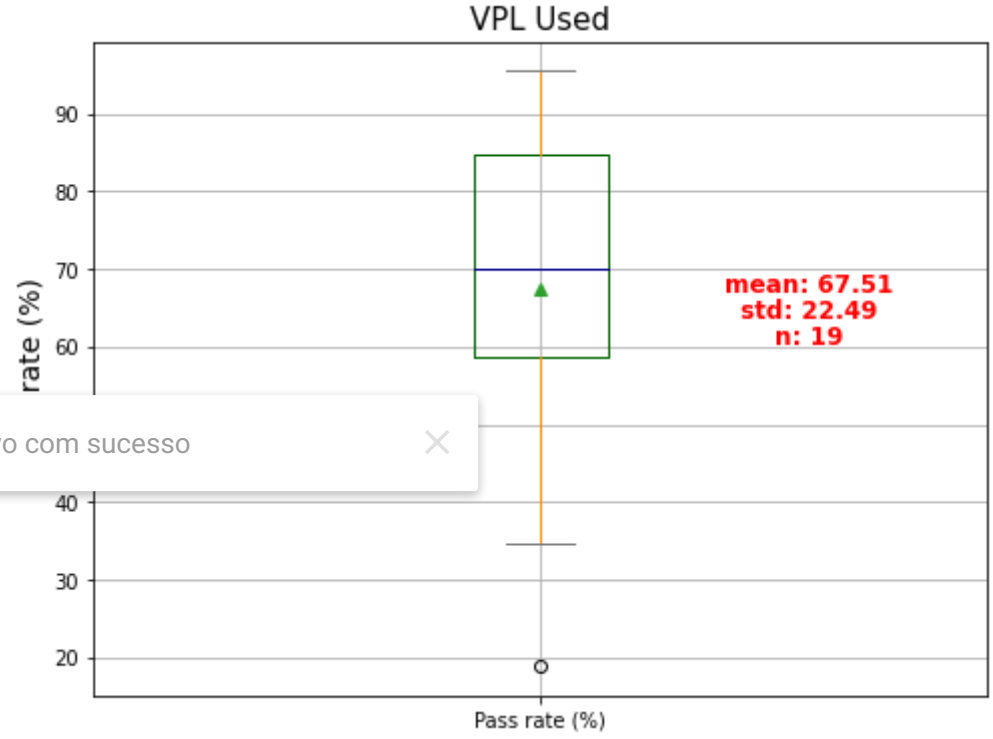
```
import seaborn as sns
x=sns.lmplot(x="PE", y="Pass rate", data=grade_vpl)
x.fig.set_size_inches(12,8)
x.fig.suptitle('Trendline for Pass rate (r2 = %.2f)'%r2, fontsize=16, y=0.97)
x.set_xlabels("Average Grade of VPL Programming Exercises (PE) (%)", fontsize=14)
x.set_ylabels("Pass rate (%)", fontsize=14)
#x.set_xlim(0,100)
#x.set_xticks(range(1,32))

if SAVE_FIGS_PAPER:
  plt.savefig('fig04new.tif', dpi=DPI_resolution)
  files.download('fig04new.tif')
```

Salvo com sucesso




```
color = {'boxes': 'DarkGreen', 'whiskers': 'DarkOrange','medians': 'DarkBlue', 'c
ax = dfVPL.plot.box(y=col_pass_rate, figsize=(8, 6), showmeans=True, color=color)
ax.grid()
plt.title('VPL Used', size='15')

ax.set_ylabel(col_pass_rate, fontsize=14)

ax.text(1.3, dfVPL[col_pass_rate].mean()-0.5, 'mean: '+str(round(dfVPL[col_pass_r
ax.text(1.3, dfVPL[col_pass_rate].mean()-3.9, 'std: '+str(round(dfVPL[col_pass_ra
ax.text(1.3, dfVPL[col_pass_rate].mean()-7.2, 'n: '+str(dfVPL[col_pass_rate].coun
```

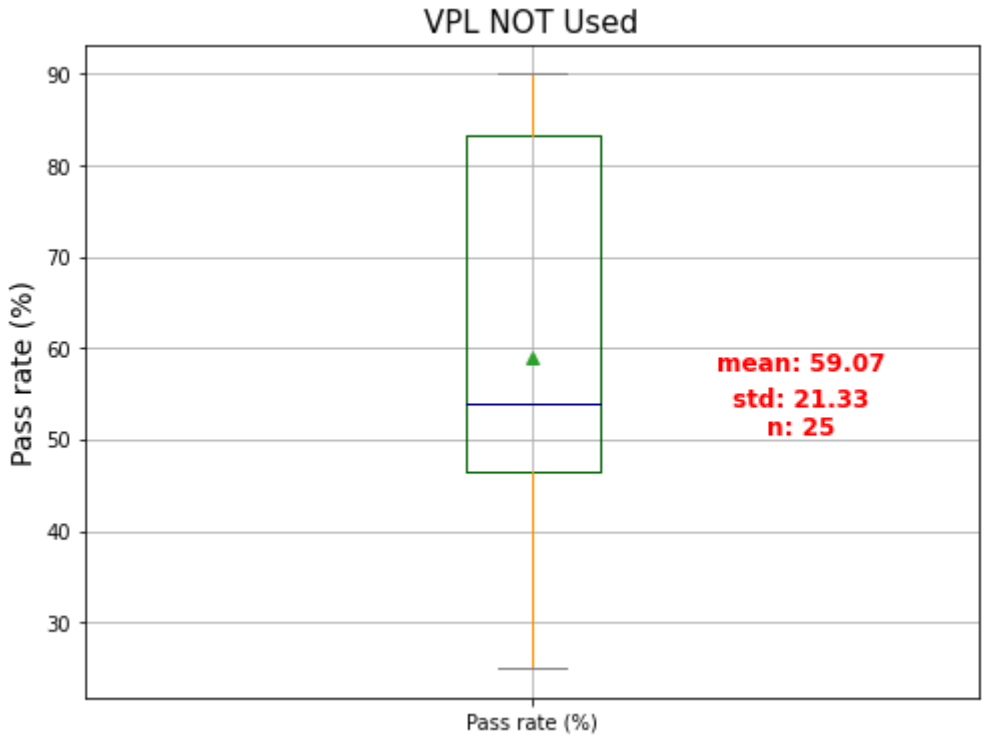
```
Text(1.3, 60.30535363157894, 'n: 19')
```



Salvo com sucesso

```
color = {'boxes': 'DarkGreen', 'whiskers': 'DarkOrange','medians': 'DarkBlue', 'c
ax = dfVPLNOT.plot.box(y=col_pass_rate, figsize=(8, 6), showmeans=True, color=col
ax.grid()
plt.title('VPL NOT Used', size='15')
ax.set_ylabel(col_pass_rate, fontsize=14)
ax.text(1.3, dfVPLNOT[col_pass_rate].mean()-1.5, 'mean: '+str(round(dfVPLNOT[col_
ax.text(1.3, dfVPLNOT[col_pass_rate].mean()-5.5, 'std: '+str(round(dfVPLNOT[col_p
ax.text(1.3, dfVPLNOT[col_pass_rate].mean()-8.5, 'n: '+str(dfVPLNOT[col_pass_rate
```

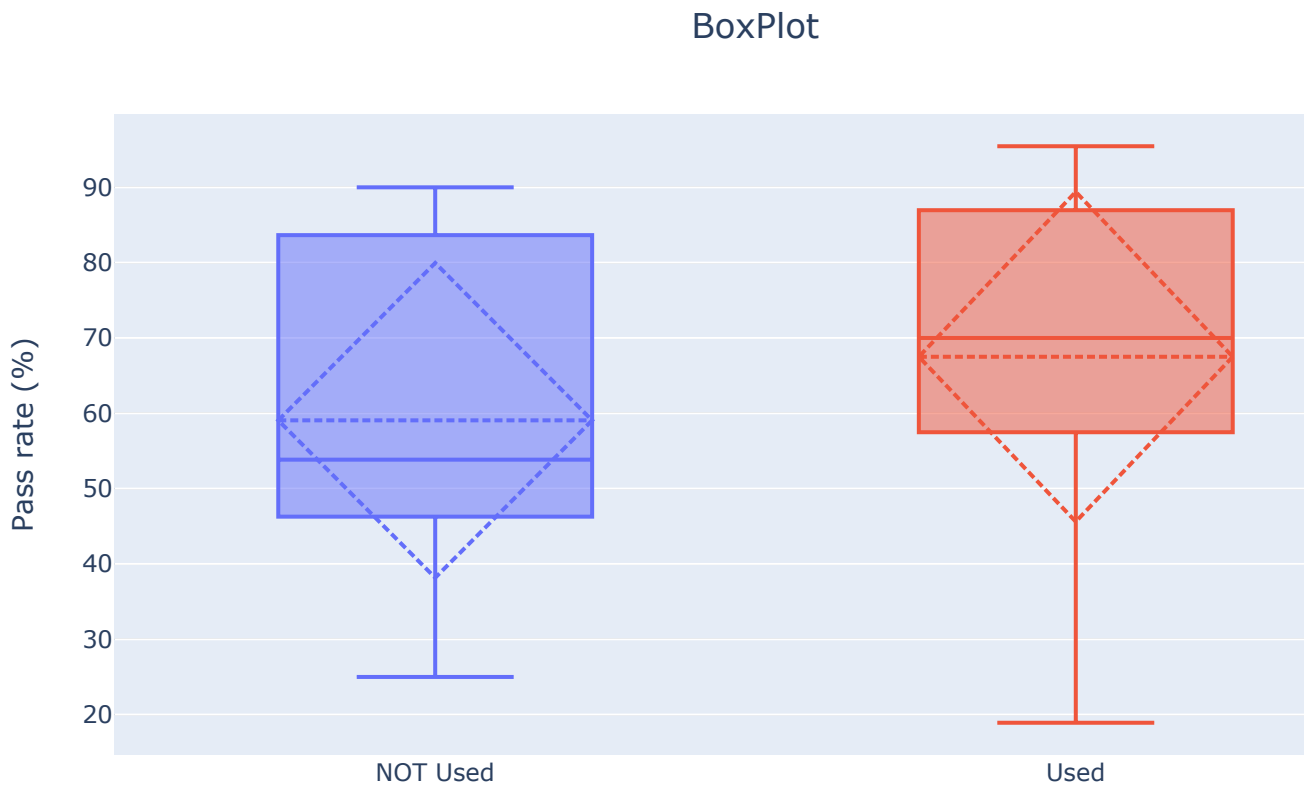
```
Text(1.3, 50.56526072, 'n: 25')
```




```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig1 = go.Box(
    y=dfVPLNOT[col_pass_rate].values,
    name='NOT Used',
    boxmean='sd'
    )
fig2 = go.Box(
    y=dfVPLNOT[col_pass_rate].values,
    name='Used',
    boxmean='sd'
    )

#boxfig= go.Figure(data=[fig1,fig2])
#fig = make_subplots(rows=1, cols=2)
#fig.add_trace(boxfig.data[0], row=1, col=1)
#fig.add_trace(boxfig.data[1], row=1, col=2)
```

Salvo com sucesso

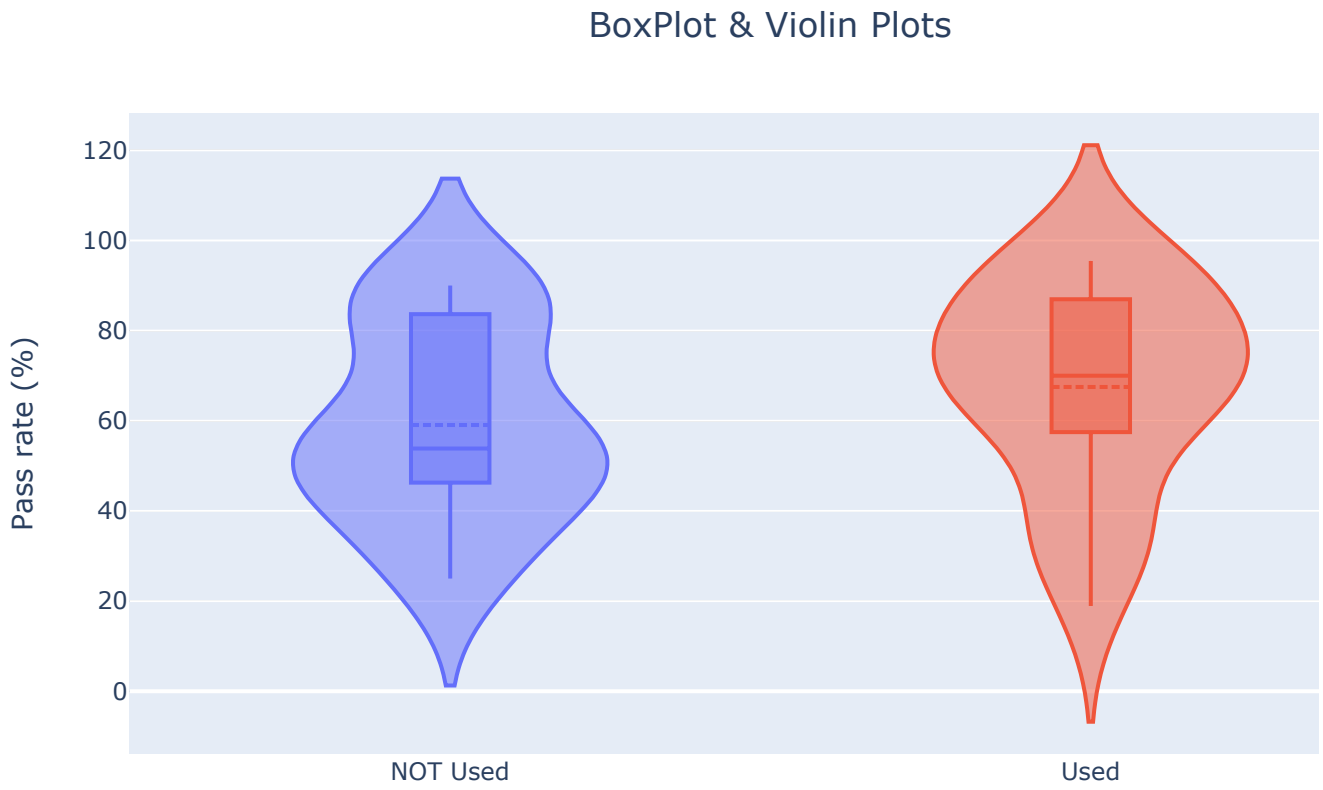
```
layout = go.Layout(
    width=800,height=500,showlegend=False,
    title={'text': "BoxPlot",'y':0.9,'x':0.5,'xanchor': 'center','yanchor': 'top'
    yaxis=dict(
        title=col_pass_rate
    ) )
fig = go.Figure(layout=layout)
fig.add_trace(fig1)
fig.add_trace(fig2)
fig.show()
```




```
import plotly.graph_objects as go
fig1 = go.Violin(meanline_visible=True,box_visible = True,
                 ].values,
                 )
fig2 = go.Violin(meanline_visible=True,box_visible = True,
    y=dfVPL[col_pass_rate].values,
    name='Used'
    )
#style layout
layout = go.Layout(
    width=800,height=500,showlegend=False,
    title={'text': "BoxPlot & Violin Plots",'y':0.9,'x':0.5,'xanchor': 'center','
```

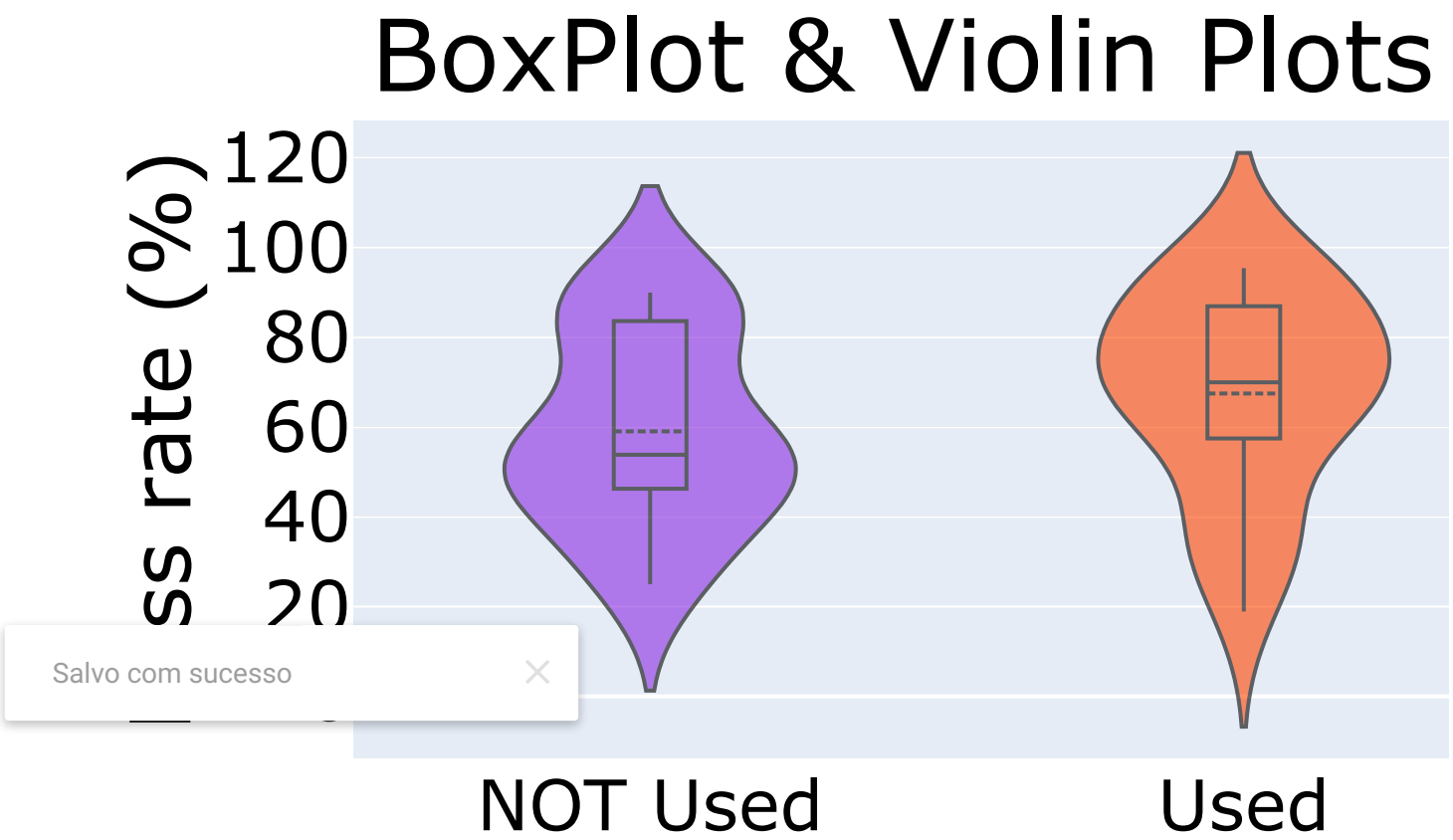
Salvo com sucesso

```
    yaxis=dict(
        title=col_pass_rate
    ) )
fig = go.Figure(layout=layout)
fig.add_trace(fig1)
fig.add_trace(fig2)
fig.show()

# kde(kernel density estimation).
```




```
import plotly.graph_objects as go
import matplotlib.pyplot as plt

Salvo com sucesso

                                    sible=True,box_visible=True,line_color='black',opacit
    y=dfVPLNOT[col_pass_rate],
    fillcolor='blueviolet',
    name='NOT Used'
)
fig2 = go.Violin(meanline_visible=True,box_visible=True,line_color='black',opacit
    y=dfVPL[col_pass_rate],
    fillcolor='orangered',
    name='Used'
)
```

```
,
15
16  #style layout
17  layout = go.Layout(title_font_size=50,
18      width=800,height=500,showlegend=False,
19      title={'text': "BoxPlot & Violin Plots",'y':0.9,'x':0.5,'xanchor': 'center','
20      font=dict(size=35, color = 'black'),
21      yaxis=dict(
22          title=col_pass_rate,
23      ) )
24
25  fig = go.Figure(layout=layout)
26  fig.add_trace(fig1)
27  fig.add_trace(fig2)
28  fig.show()
29
30  if SAVE_FIGS_PAPER:
31    from PIL import Image
32    fig.write_image('fig03.png', width=width_resolution, height=height_resolution)
33    img = Image.open('fig03.png')
34    img.save('fig03.tif')
35    files.download('fig03.tif')
```



Salvo com sucesso

Refs

[boxplot] Tukey, John W. Exploratory data analysis. Mosteller F, editor. United States Addison-Wesley. Publishing Company; (1977): 688 p.

[ViolinPlots] Hintze, Jerry L., and Ray D. Nelson. Violin plots: a box plot-density trace synergism. The American Statistician. 52.2 (1998): 181-184.

Cores:

```
aliceblue, antiquewhite, aqua, aquamarine, azure,
beige, bisque, black, blanchedalmond, blue,
blueviolet, brown, burlywood, cadetblue,
chartreuse, chocolate, coral, cornflowerblue,
cornsilk, crimson, cyan, darkblue, darkcyan,
darkgoldenrod, darkgray, darkgrey, darkgreen,
darkkhaki, darkmagenta, darkolivegreen, darkorange,
darkorchid, darkred, darksalmon, darkseagreen,
darkslateblue, darkslategray, darkslategrey,
darkturquoise, darkviolet, deeppink, deepskyblue,
dimgray, dimgrey, dodgerblue, firebrick,
floralwhite, forestgreen, fuchsia, gainsboro,
ghostwhite, gold, goldenrod, gray, grey, green,
greenyellow, honeydew, hotpink, indianred, indigo,
ivory, khaki, lavender, lavenderblush, lawngreen,
lemonchiffon, lightblue, lightcoral, lightcyan,
lightgoldenrodyellow, lightgray, lightgrey,
lightgreen, lightpink, lightsalmon, lightseagreen,
lightskyblue, lightslategray, lightslategrey,
lightsteelblue, lightyellow, lime, limegreen,
linen, magenta, maroon, mediumaquamarine,
mediumblue, mediumorchid, mediumpurple,
mediumseagreen, mediumslateblue, mediumspringgreen,
mediumturquoise, mediumvioletred, midnightblue,
mintcream, mistyrose, moccasin, navajowhite, navy,
                orange, orangered,
                green, paleturquoise,
palevioletred, papayawhip, peachpuff, peru, pink,
plum, powderblue, purple, red, rosybrown,
royalblue, rebeccapurple, saddlebrown, salmon,
sandybrown, seagreen, seashell, sienna, silver,
skyblue, slateblue, slategray, slategrey, snow,
springgreen, steelblue, tan, teal, thistle, tomato,
```

Salvo com sucesso

```
turquoise, violet, wheat, white, whitesmoke,
yellow, yellowgreen
```

Students' T Test

```
countVPL = dfVPL[['A','B','C','D','F','O']].sum().sum()
countVPLNOT = dfVPLNOT[['A','B','C','D','F','O']].sum().sum()
countVPL_BL = dfVPL_BL[['A','B','C','D','F','O']].sum().sum()
meanVPL = dfVPL[col_pass_rate].mean()
meanVPLNOT  = dfVPLNOT [col_pass_rate].mean()
meanVPL_BL = dfVPL_BL[col_pass_rate].mean()
stdVPL = dfVPL[col_pass_rate].std()
stdVPLNOT  = dfVPLNOT [col_pass_rate].std()
stdVPL_BL = dfVPL_BL[col_pass_rate].std()

```

```
print(countVPL_BL, meanVPL_BL)
print(countVPL, meanVPL)
print(countVPLNOT, meanVPLNOT)

```

```
171 70.376176
509 67.50535363157894
663 59.06526072
```

```
# calculate means
mean1, mean2 = meanVPL, meanVPLNOT
```

```
# calculate sample standard deviations
std1, std2 = stdVPL, stdVPLNOT
```

```
# calculate standard errors
import math
n1, n2 = countVPL, countVPLNOT
se1, se2 = std1/math.sqrt(n1), std2/math.sqrt(n2)
```

Salvo com sucesso

```
erence between the samples
se2**2.0)
```

```
# calculate the t statistic
t_stat = (mean1 - mean2) / sed
t_stat
```

```
6.512378158662304
```

```
# degrees of freedom
df = n1 + n2 - 2
df
```

```
1170
```

```
from scipy.stats import t
# calculate the critical value
alpha = 0.05
cv = t.ppf(1.0 - alpha, df)
cv
```

```
1.6461570318357832
```

```
# calculate the p-value
p = (1.0 - t.cdf(abs(t_stat), df)) * 2.0
'%.15f' % p
```

```
'0.000000000109572'
```

```
# confidence level
'%.10f' % ((1-p)*100)
```

```
'99.9999999890'
```

Similarity in PE

```
file3 = 'http://vision.ufabc.edu.br/MCTest/public/CAE2021/2019.1.similarity.csv'
df_similarity = pd.read_csv(file3)
df_similarity.sort_values(by=['PE'], inplace=True)
df_similarity['Similarity (%)'] = df_similarity.Similarity / df_similarity.Subscr
df_similarity.head(5)
```

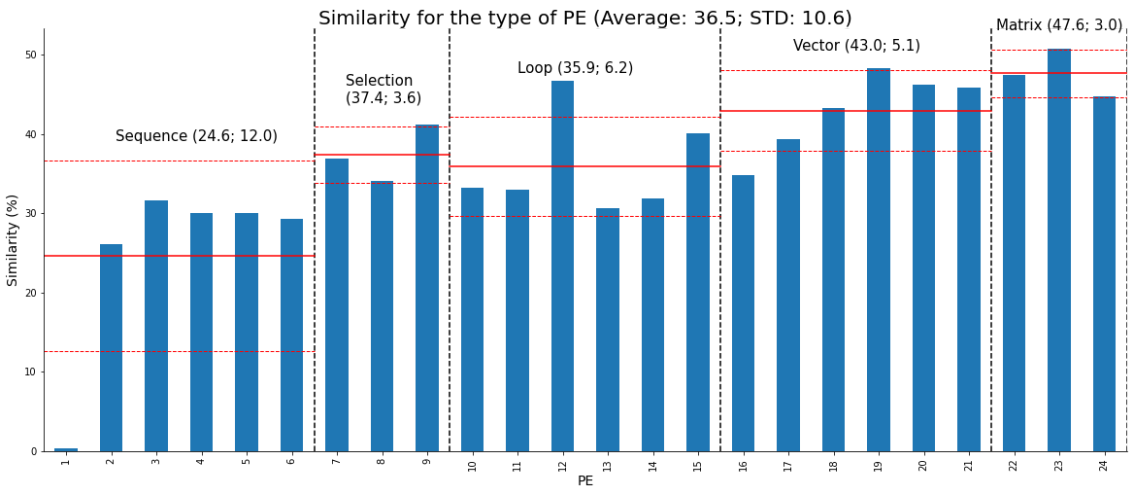
	PE	Similarity	Subscriptions	Corrections	Type	Similarity (%)
0	1	2	596	582	Sequence	0.335570
1	2	144	547	506	Sequence	26.142596
2	3	158	499	468	Sequence	31.663327
3	4	145	483	454	Sequence	30.020704
4	5	136	452	422	Sequence	30.088496

Salvo com sucesso

```
column = 'Similarity (%)'
xy = df_similarity[['PE',column]]
ax = xy.plot(kind='bar',x='PE',y=column)
```

```
ax = df.plot(kind='bar',x='Id',y='Column')
ax.get_legend().remove()
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False)

types = ['Sequence', 'Selection',  'Loop', 'Vector', 'Matrix']

col_sum = 0
for t in types:
  df_aux = df_similarity.query('Type == "'+ t + '"')
  mean = df_aux[column].mean()
  std = df_aux[column].std()
  dx,_ = df_aux.shape
  col_sum += dx
  x1 = col_sum - .5
  x0 = x1 - dx
  plt.axvline(x1, color="k", linestyle="--");
  plt.plot([x0, x1], [mean, mean], color='red', linewidth=1.5, linestyle="-")
  plt.plot([x0, x1], [mean-std, mean-std], color='red', linewidth=1, linestyle="-
  plt.plot([x0, x1], [mean+std, mean+std], color='red', linewidth=1, linestyle="-
  if t == 'Selection':
    ax.annotate('%s \n(%.1f; %.1f)' % (t, mean, std), xy=((x0+x1)/2-0.8, mean+stc
  elif t == 'Loop':
    ax.annotate('%s (%.1f; %.1f)' % (t, mean, std), xy=((x0+x1)/2-1.5, mean+std +
  else:
    ax.annotate('%s (%.1f; %.1f)' % (t, mean, std), xy=((x0+x1)/2-1.4, mean+std +

mean = df_similarity['Similarity (%)'].mean()
std = df_similarity['Similarity (%)'].std()
#print("mean:",str(mean),"std:",str(std))
plt.title('Similarity for the type of PE (Average: %.1f; STD: %.1f)'%(mean,std),
ax.set_xlabel("PE", fontsize=14)
ax.set_ylabel(column, fontsize=14)
ax.plot()

if SAVE_FIGS_PAPER:
  plt.savefig('fig07new.tif', dpi=DPI_resolution)
  files.download('fig07new.tif')
print('\n\n')
```

Salvo com sucesso




```
1  df_similarity.sample(3)
```

	PE	Similarity	Subscriptions	Corrections	Type	Similarity (%)
7	8	137	402	387	Selection	34.079602
6	7	153	414	397	Selection	36.956522
0	1	2	596	582	Sequence	0.335570

```
1
```

Salvo com sucesso

Salvo com sucesso