Visual Analysis for Instructors

Damien Bogan

2024-12-01

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
pd.set_option('display.max_columns', None) # display all columns
pd.set_option('display.expand_frame_repr', False) # use `...` for truncated
columns
pd.set_option('max_colwidth', None) # display full contents of a column

import seaborn as sns
from IPython.display import Markdown, display
from tabulate import tabulate
```

```
# #### Parameters

#########################################################
#    The following entries may require change
#########################################################

file_name = './data.csv'

number_sem_display = 10   # in portrait table
earliest_sem_in_plot = 201501

fig_width = 11.5
fig_height_bar = 10

WZ=6 # rolling window size
DEPT=["M", "S"]
top_major_n =5
# Define the set of all passing grades {A, B, C, A#, B#, C#, AU,B+,C+,B-, C-,
CR, S} as a Python set object named "passing_grades"
# Note the set of non-passing grades is the completement of the set of passing
grades
passing_grades = {"A", "B", "C", "A#", "B#", "C#", "AU", "B+", "C+", "B-", "C-",
```

```
"CR", "S"}
removed_course=[2]

Analyzed_course_subj =["M", "E", "S"]
remedial_courses =[6378, 6381, 6478, 6481, 6580, 6581]
Non_STEM = [6698, 6728, 6778, 6748, 8348]
STEM =[6798, 6808, 6828, 7228]
ENGR =[7288, 7298, 8308]
corequisites =[6700, 6730, 6780, 6800, 8350, 5698, 5728, 5748, 5798 ]


advanced =[9888,  9988,  9998, 10348, 10628,  9788,  9878,  9898,  9908,
        9978,  9718,  9728, 10078, 10148, 10278, 10602, 10610 ]
advanced_stat=[6688,  9828,  9838,  9848, 10178]


Analyzed_courses = sorted(
  remedial_courses + Non_STEM + corequisites +   STEM +
    ENGR  + advanced_stat+  advanced)


tt =[

]

le =[

]

ad =[

]

Analyzed_fac = sorted(tt + le + ad)

excluded_fac =[]

#################################################################
#   End of input information
#################################################################
```

```
df0 = pd.read_csv(file_name)
# check if df1 has any missing values and dulicated rows
df0.isnull().sum().sum(), df0.duplicated().sum()
```

Goal of My Project: *Analyze* the passing rate/rolling passing rate for a group of instructors in each semester, hence automatically the code should work for an individual instructor when the group has a size 1. You should have at least two examples. One is for an individual, and one is for a group.

Overview of the Instructors

The display of instructors are given below:

```
print(np.unique(df0['Instructor']))
n_instructors = df0["Instructor"].unique().shape[0]
print(f"The number of unique instructors in the dataset are: {n_instructors}")
```

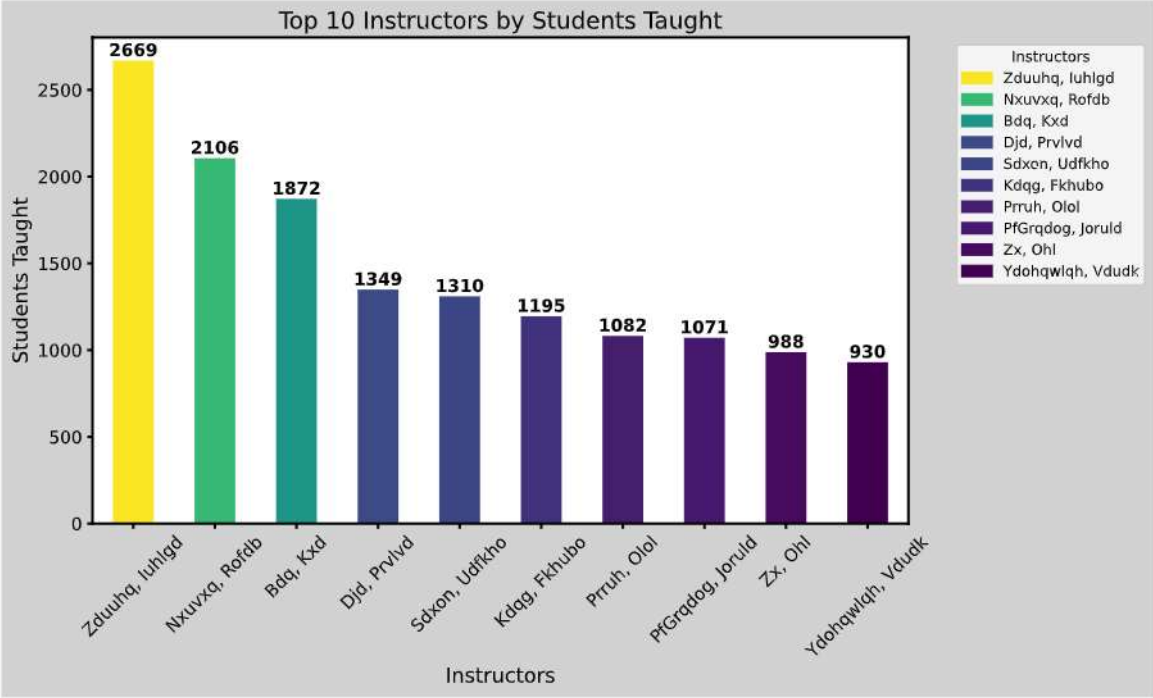
```
['Bdq, Kxd' 'Bhshc Pduwlqhc, Pdxulflr M' 'Ckdqj, Blihl' 'Djd, Prvlvd'
 'Dodwwdu, Dvkudi' 'Doeuhh, Dqvrq' 'Doohq, Hchoo' 'Dphuvrq, Jdub'
 'Dvkrndq, Vkduplod' 'Ehoolr Ulqfrq, Jlrydqql' 'Ehuub, Mrugdq'
 'Ekdwwdfkdubd, Vxwdqx' 'Ekdwwdfkdubd, Wdwkdjdwd' 'Erer, Obqghh'
 'Erkdqqrq, Mrkq' 'Eurzq, Vfrww' 'Exhqjhu, Dqwkrqb'
 'Fdvlplu- Sdwwrq, Eholqgd' 'Fkhq, Bxh' 'Fkhq, Wldqudq' 'Fodun, Vdoob'
 'Free, Zdqjvkx' 'Fxhyd-Sduud, Oxlv' 'Fxl, Bxh' 'Gdubdqdql, Mrkqqb'
 'Gduohb, Dxuhold' 'Gdylv, Fdol' 'Gdylv, Sdxo' 'Glqf, Vhplk'
 'Grxjkhuwb, Pdub' 'Gxqfdq, Eubfh' 'Jdr, Cklplq' 'Jrggdug, Mhurph'
 'Jrgzlq, Sulvflood' 'Jrnxo, Qludqmdq' 'Jxkd, Dwdqx' 'Kdqg, Fkhubo'
 'Kduglqj, Kdqqdk' 'Kduprq, Pdub' 'Krrg, Vdpdqwkd' 'Kruqh, Wzlod'
 'Kxdqj, Fkhqj' 'Kxdqj, Fkhqj-Fkl' 'Mdfnvrq, Grqqd' 'Mhiihuvrq, Eduedud'
 'Mxqnhu, Mrho' 'Ndudelehu, Idwlk' 'Nrwdpudmx, Ndvkbds' 'Nxuvxq, Rofdb'
 'Ohh, Mlqsbr' 'Ohh, Nlq' 'Ol, Zhl' 'Olqduhv, Mhvxv' 'Olskdp, Gdylg'
 'Olx, Brqjalq' 'Or, Wlqj Wr' 'PfGrqdog, Joruld' 'Prqlqvnl, Dqwkrqb'
 'Prruh, Olol' 'Pruwrq, Vfrww' 'Sdsh, Sdwulfn' 'Sdun, Mlhxq'
 'Sdxon, Udfkho' 'Shhoh, Ukrghv' 'Shuhjrb, Mduhg' 'Srzhoo, Mrdq'
 'Sudekdndu, Dqqlh' 'Udjodqg, Pdwwkhz' 'Udkedulqld, Ededn' 'Udwwdq, Fduro'
 'Uhhg, Qlfroh' 'Vdklqrjox, Phkphw' 'Vfkplgw, Ehwwlqd' 'Vplwk, Ixupdq'
 'Vplwk, Oxnh' 'Vroohb, Zdbqh' 'Vshdu, Pdub' 'Vwdqilhog, Ohwkhqxdo'
 'Wdqj, Zhq' 'Wldq, Clbdq' 'Wxoolv, Vkdbod' 'Wxun, Pbod'
 'Xqghuzrrg, Urehuw' 'Ydohqwlqh, Vdudk' 'Yhqndwudp, Sudprg' 'Zdqj, Bl'
 'Zdqj, Fkhqjihl' 'Zdqj, Fkhqjjdqj' 'Zduuhq, Iuhlgd' 'Zlqwhu, Urgqhb'
 'Zx, Ohl']
The number of unique instructors in the dataset are: 91
```

There are 91 instructors, but I want to concentrate on the top ten, using one as an example for the passing rate and rolling passing rates and two as my group of instructors to compare their passing rates and rolling passing rates.

The number of students taught by the top 10 instructors are plotted below.

```
data = df0['Instructor'].value_counts().head(10)
cmap = plt.get_cmap("viridis")
norm = mcolors.Normalize(vmin=data.min(), vmax=data.max())
custom_color = [cmap(norm(value)) for value in data]
display(Markdown(data.to_markdown()))
fig, ax = plt.subplots(figsize=(10, 6), facecolor='lightgray')

barplot = data.plot(kind='bar', color=custom_color, ax=ax, xlabel="")
ax.set_title('Top 10 Instructors by Students Taught', fontsize=16)
ax.set_xlabel('Instructors', fontsize=14)
ax.set_ylabel('Students Taught', fontsize=14)

for bar in barplot.patches:
    ax.annotate(f'{bar.get_height()}',
          xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()), xytext=(0,
3), # 3 points vertical offset
          textcoords="offset points", ha='center', fontsize=12, color='black',
weight='bold')
# tick params
ax.tick_params(axis='x', labelrotation=45, labelsize=12)
ax.tick_params(axis='y', labelsize=12)

handles = [plt.Rectangle((0, 0), 1, 1, color=custom_color[i]) for i in
range(len(data))]
labels = data.index.tolist()
ax.legend(handles, labels, title='Instructors', bbox_to_anchor=(1.05, 1),
loc='upper left')

plt.show()
```

Instructor	count
Zduuhq, Iuhlgd	2669
Nxuvxq, Rofdb	2106
Bdq, Kxd	1872
Djd, Prvlvd	1349
Sdxon, Udfkho	1310
Kdqg, Fkhubo	1195
Prruh, Olol	1082
PfGrqdog, Joruld	1071
Zx, Ohl	988
Ydohqwlqh, Vdudk	930




```
passing_grades = {"A", "B", "C", "A#", "B#", "C#", "AU", "B+", "C+", "B-", "C-",
"CR", "S"}
all_grades = set(df0['Grade'].unique())
failing_grades = all_grades - passing_grades

passing_counts              =              df0[df0['Grade'].isin(passing_grades)]
['Grade'].value_counts()
failing_counts = df0[df0['Grade'].isin(failing_grades)]['Grade'].value_counts()

passing_table                                                                  =
pd.DataFrame(passing_counts).reset_index().rename(columns={'index':    'Grade',
'Grade': 'Passing Count'})
failing_table                                                                  =
pd.DataFrame(failing_counts).reset_index().rename(columns={'index':    'Grade',
'Grade': 'Failing Count'})

display(Markdown("### Passing Grades"))
display(Markdown(passing_table.to_markdown()))
display(Markdown("### Failing Grades"))
display(Markdown(failing_table.to_markdown()))
```

Passing Grades

	Passing Count	count
0	A	10181

	Passing Count	count
1	B	4544
2	C	2806
3	B+	2423
4	CR	2145
5	B#	1470
6	A#	1193
7	C#	1169
8	C+	932
9	S	932
10	AU	16

Failing Grades

	Failing Count	count
0	W	1830
1	D	1043
2	F#	978
3	F	827
4	FAN	774
5	F*	345
6	FA	299
7	D+	254
8	U	171
9	NC	147
10	F**	140
11	D*	86
12	FA*	72
13	FA**	22
14	D#	12
15	I	7
16	D+*	4
17	IP	1

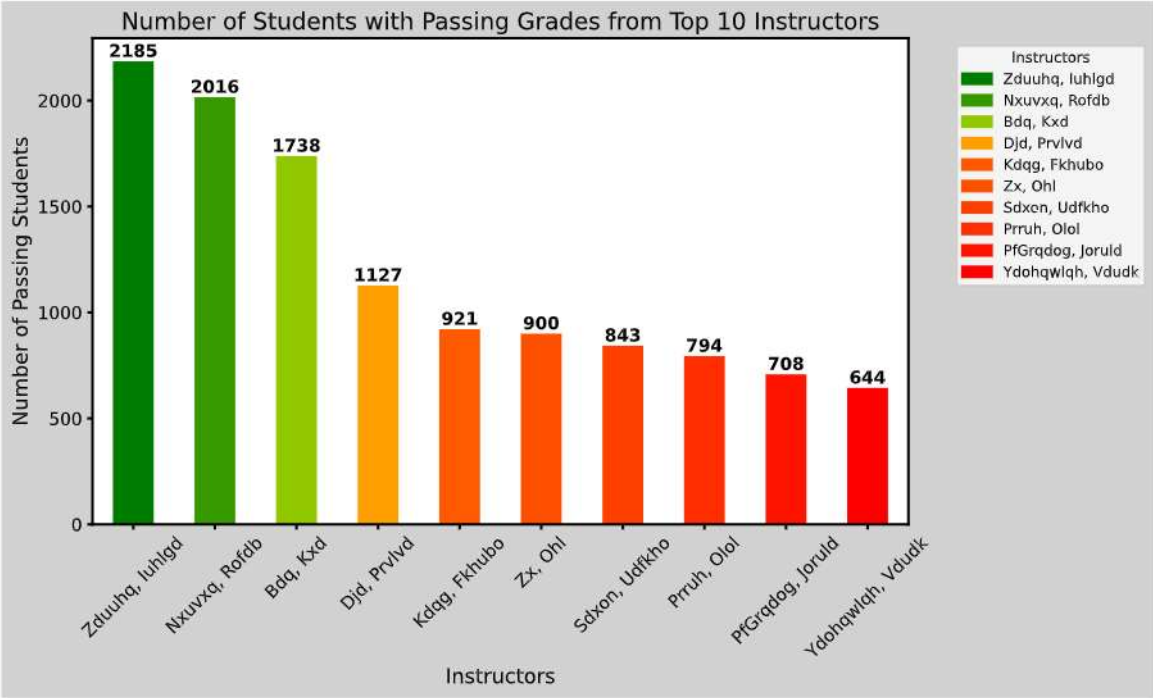
	Failing Count	count
18	NR	1

The number of students with passing grades from top 10 instructors

```
# number of students in each subject code

# display(Markdown(df0['SubjectCode'].value_counts().head(20).to_markdown()))
#plot the top 20 subject codes
# df0['SubjectCode'].value_counts().head(20).plot(kind='bar',xlabel="");

top_10_instructors = df0['Instructor'].value_counts().head(10)
filtered_df = df0[df0['Instructor'].isin(top_10_instructors.index)]
passing_grades = {"A", "B", "C", "A#", "B#", "C#", "AU", "B+", "C+", "B-", "C-",
"CR", "S"}
passing_students = filtered_df[filtered_df['Grade'].isin(passing_grades)]
passing_counts = passing_students['Instructor'].value_counts().head(10)

cmap = mcolors.LinearSegmentedColormap.from_list("custom", ["red", "yellow",
"green"])
norm_passing = mcolors.Normalize(vmin=passing_counts.min(),
vmax=passing_counts.max())
colors_passing = [cmap(norm_passing(value)) for value in passing_counts]

# display(Markdown(top_10_instructors.to_markdown()))
display(Markdown(passing_counts.to_markdown()))

fig, ax = plt.subplots(figsize=(10, 6), facecolor='lightgray')
bars = passing_counts.plot(kind='bar', color=colors_passing, ax=ax, xlabel="")
ax.set_title('Number of Students with Passing Grades from Top 10 Instructors',
fontsize=16)
ax.set_xlabel('Instructors', fontsize=14)
ax.set_ylabel('Number of Passing Students', fontsize=14)

for bar in bars.patches:
    ax.annotate(f'{bar.get_height()}',
          xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()), xytext=(0,
3), # 3 points vertical offset
          textcoords="offset points", ha='center', fontsize=12, color='black',
weight='bold', zorder=3)

ax.tick_params(axis='x', labelrotation=45, labelsize=12)
ax.tick_params(axis='y', labelsize=12)

handles = [plt.Rectangle((0, 0), 1, 1,
color=cmap(norm_passing(passing_counts.iloc[i]))) for i in
range(len(passing_counts))]
```

```
labels = passing_counts.index.tolist()
ax.legend(handles, labels, title='Instructors', bbox_to_anchor=(1.05, 1),
loc='upper left')

plt.show()
```

Instructor	count
Zduuhq, Iuhlgd	2185
Nxuvxq, Rofdb	2016
Bdq, Kxd	1738
Djd, Prvlvd	1127
Kdqg, Fkhubo	921
Zx, Ohl	900
Sdxon, Udfkho	843
Prruh, Olol	794
PfGrqdog, Joruld	708
Ydohqwlqh, Vdudk	644

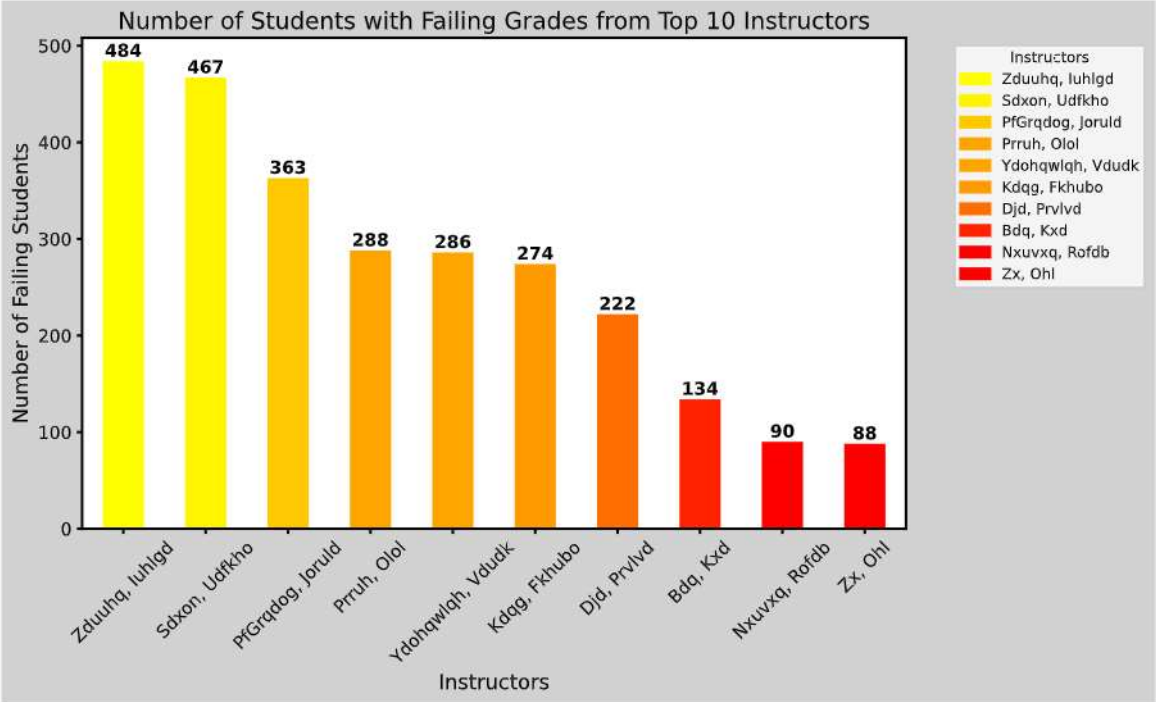


The number of students with failing grades from the top 10 instructors

```
top_10_instructors = df0['Instructor'].value_counts().head(10)
filtered_df = df0[df0['Instructor'].isin(top_10_instructors.index)]
passing_grades = {"A", "B", "C", "A#", "B#", "C#", "AU", "B+", "C+", "B-", "C-",
"CR", "S"}
all_grades = set(df0['Grade'].unique())
failing_grades = all_grades - passing_grades
failing_students = filtered_df[filtered_df['Grade'].isin(failing_grades)]
failing_counts = failing_students['Instructor'].value_counts().head(10)

cmap = mcolors.LinearSegmentedColormap.from_list("custom", ["red", "orange",
"yellow"])
norm_failing = mcolors.Normalize(vmin=failing_counts.min(),
vmax=failing_counts.max())
colors_failing = [cmap(norm_failing(value)) for value in failing_counts]

display(Markdown(failing_counts.to_markdown()))

fig, ax = plt.subplots(figsize=(10, 6), facecolor='lightgray')
barplot = failing_counts.plot(kind='bar', color=colors_failing, ax=ax,
xlabel="")
ax.set_title('Number of Students with Failing Grades from Top 10 Instructors',
fontsize=16)
ax.set_xlabel('Instructors', fontsize=14)
ax.set_ylabel('Number of Failing Students', fontsize=14)

for bar in barplot.patches:
    ax.annotate(f'{bar.get_height()}',
           xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()), xytext=(0,
3), # 3 points vertical offset
          textcoords="offset points", ha='center', fontsize=12, color='black',
weight='bold', zorder=3)

ax.tick_params(axis='x', labelrotation=45, labelsize=12)
ax.tick_params(axis='y', labelsize=12)

handles = [plt.Rectangle((0, 0), 1, 1,
color=cmap(norm_failing(failing_counts.iloc[i]))) for i in
range(len(failing_counts))]
labels = failing_counts.index.tolist()
ax.legend(handles, labels, title='Instructors', bbox_to_anchor=(1.05, 1),
loc='upper left')
# ax.legend(title='Instructors', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.show()
```

Instructor	count
Zduuhq, Iuhlgd	484

Instructor	count
Sdxon, Udfkho	467
PfGrqdog, Joruld	363
Prruh, Olol	288
Ydohqwlqh, Vdudk	286
Kdqg, Fkhubo	274
Djd, Prvlvd	222
Bdq, Kxd	134
Nxuvxq, Rofdb	90
Zx, Ohl	88

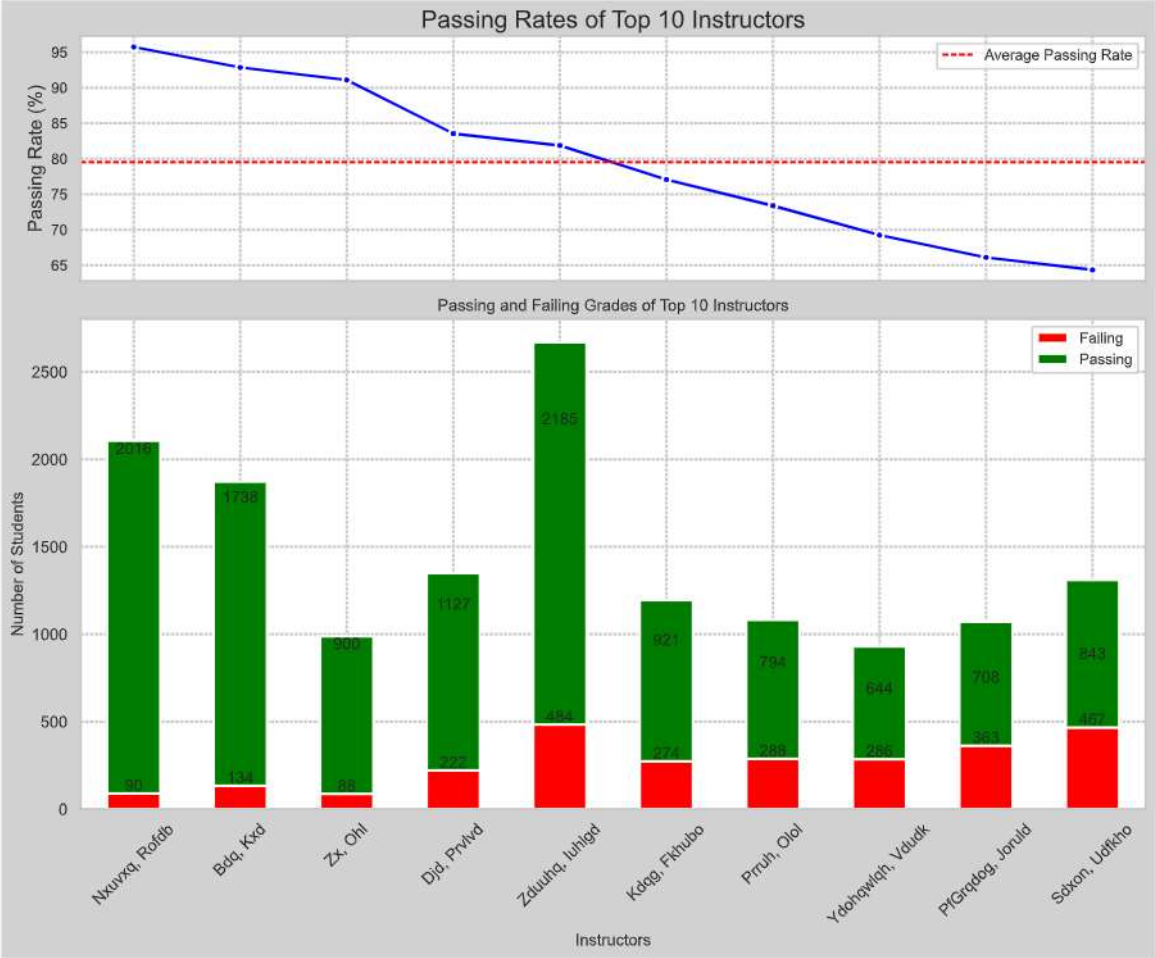


Passing Rates of the Top 10 Instructors vs. the Passing and Failing students taught by the Top 10 Instructors.

```
top_10_instructors = df0['Instructor'].value_counts().head(10)
filtered_df = df0[df0['Instructor'].isin(top_10_instructors.index)]
passing_grades = {"A", "B", "C", "A#", "B#", "C#", "AU", "B+", "C+", "B-", "C-",
"CR", "S"}
all_grades = set(df0['Grade'].unique())
failing_grades = all_grades - passing_grades
```

```
passing_counts                                                                =
filtered_df[filtered_df['Grade'].isin(passing_grades)].groupby('Instructor')
['Grade'].count()
failing_counts                                                                =
filtered_df[filtered_df['Grade'].isin(failing_grades)].groupby('Instructor')
['Grade'].count()

passing_rates = passing_counts / (passing_counts + failing_counts) * 100

combined_df = pd.DataFrame({
        'Passing': passing_counts,
        'Failing': failing_counts,
        'Passing Rate': passing_rates }).fillna(0)

combined_df = combined_df.sort_values(by='Passing Rate', ascending=False)

sns.set(style="whitegrid")

fig,  (ax1,  ax2)  =  plt.subplots(2,  1,  figsize=(12,  10),  sharex=True,
gridspec_kw={'height_ratios': [1, 2]}, facecolor='lightgray')

#line plot passing rate(top)
sns.lineplot(data=combined_df['Passing     Rate'],     ax=ax1,     marker='o',
color='blue', linewidth=2)
ax1.set_title('Passing Rates of Top 10 Instructors', fontsize=18)
ax1.set_ylabel('Passing Rate (%)', fontsize=14)
ax1.grid(True, which='both', linestyle='--', linewidth=0.7)
ax1.axhline(y=combined_df['Passing Rate'].mean(), color='red', linestyle='--',
linewidth=1.2, label='Average Passing Rate')
ax1.legend(loc='upper right')

#stacked bar plot(bottom)
combined_df[['Failing',   'Passing']].plot(kind='bar',   stacked=True,   ax=ax2,
color=['red', 'green'])
ax2.set_title('Passing and Failing Grades of Top 10 Instructors')
ax2.set_xlabel('Instructors')
ax2.set_ylabel('Number of Students')

for bar in ax2.patches:
    ax2.annotate(f'{int(bar.get_height())}',
        (bar.get_x() + bar.get_width() / 2., bar.get_height()),
        ha='center', va='center', xytext=(0, 5), textcoords='offset points')

ax2.tick_params(axis='x', rotation=45, labelsize=12)
ax2.tick_params(axis='y', labelsize=12)
ax2.grid(True, which='both', linestyle='--', linewidth=0.7)
```

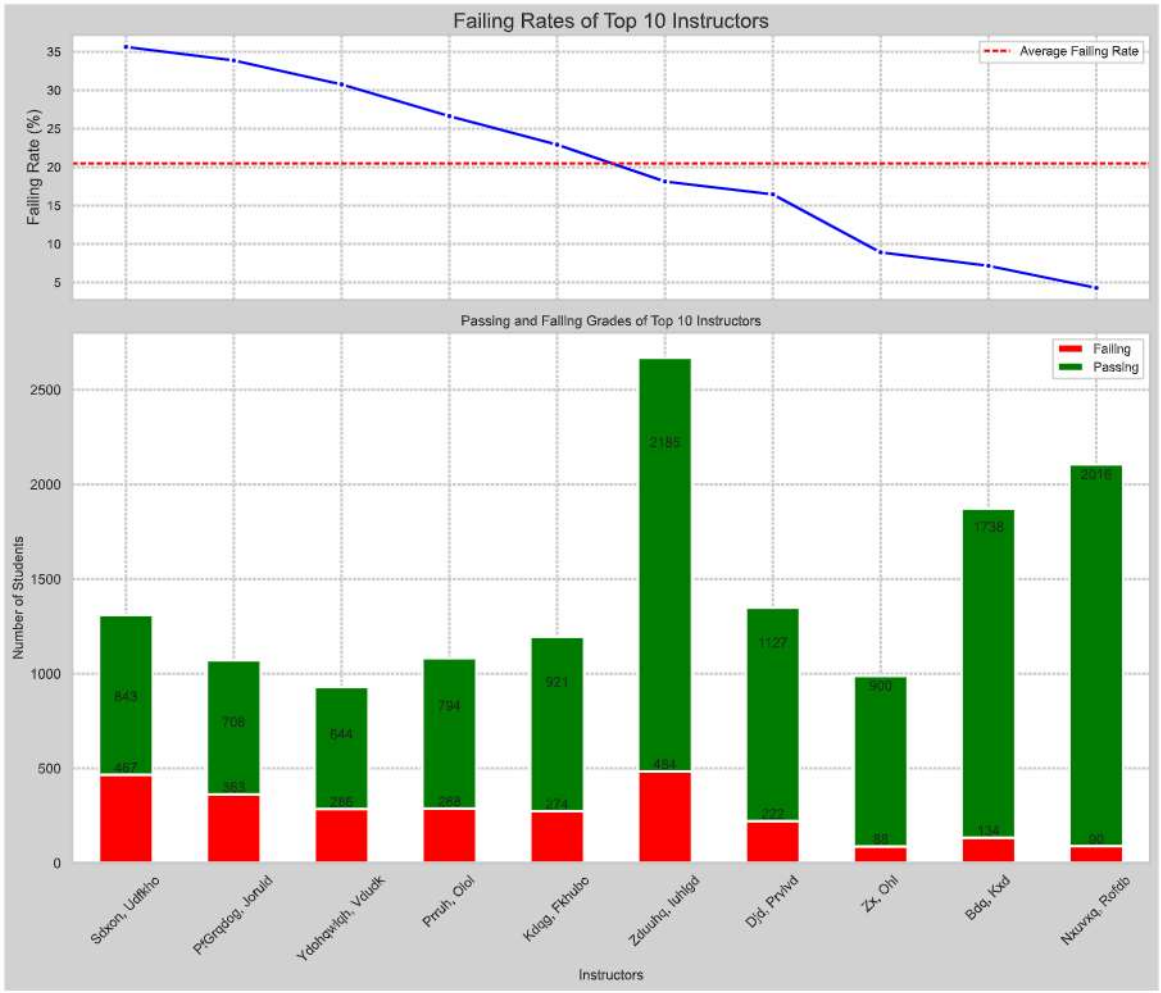
```
plt.tight_layout()
plt.show()
```



Failing Rates of the Top 10 Instructors vs. the Passing and Failing students taught by the Top 10 Instructors.

```
top_10_instructors = df0['Instructor'].value_counts().head(10)
filtered_df = df0[df0['Instructor'].isin(top_10_instructors.index)]
passing_grades = {"A", "B", "C", "A#", "B#", "C#", "AU", "B+", "C+", "B-", "C-",
"CR", "S"}
all_grades = set(df0['Grade'].unique())
failing_grades = all_grades - passing_grades

passing_counts                                                              =
filtered_df[filtered_df['Grade'].isin(passing_grades)].groupby('Instructor')
['Grade'].count()
failing_counts                                                              =
```

```
filtered_df[filtered_df['Grade'].isin(failing_grades)].groupby('Instructor')
['Grade'].count()

failing_rates = failing_counts / (passing_counts + failing_counts) * 100

combined_df = pd.DataFrame({
        'Passing': passing_counts,
        'Failing': failing_counts,
        'Failing Rate': failing_rates }).fillna(0)

combined_df = combined_df.sort_values(by='Failing Rate', ascending=False)

sns.set(style="whitegrid")

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 12), sharex=True,
gridspec_kw={'height_ratios': [1, 2]}, facecolor='lightgray')

#line plot failing rates(top)
sns.lineplot(data=combined_df['Failing Rate'], ax=ax1, marker='o',
color='blue', linewidth=2)
ax1.set_title('Failing Rates of Top 10 Instructors', fontsize=18)
ax1.set_ylabel('Failing Rate (%)', fontsize=14)
ax1.grid(True, which='both', linestyle='--', linewidth=0.7)
ax1.axhline(y=combined_df['Failing Rate'].mean(), color='red', linestyle='--',
linewidth=1.2, label='Average Failing Rate')
ax1.legend(loc='upper right')

#stacked bar plot(bottom)
combined_df[['Failing', 'Passing']].plot(kind='bar', stacked=True, ax=ax2,
color=['red', 'green'])
ax2.set_title('Passing and Failing Grades of Top 10 Instructors')
ax2.set_xlabel('Instructors')
ax2.set_ylabel('Number of Students')

for bar in ax2.patches:
    ax2.annotate(f'{int(bar.get_height())}',
        (bar.get_x() + bar.get_width() / 2., bar.get_height()),
        ha='center', va='center', xytext=(0, 5), textcoords='offset points')

ax2.tick_params(axis='x', rotation=45, labelsize=12)
ax2.tick_params(axis='y', labelsize=12)
ax2.grid(True, which='both', linestyle='--', linewidth=0.7)

plt.tight_layout()
plt.show()
```



Now that we have a taste of visuals for Instructors passing and failing rates of their students taught. I am now going to dive into my visual analysis for the passing rate/rolling passing rate for a group of instructors, but before I do so let's look at the tables for the passing/rolling rate of three instructors. The two instructors I will use are 'Nxuvxq, Rofdb' (highest passing rate), 'Bdq, Kxd' (second highest passing rate), and 'Djd, Prvlvd'.

```
# extract all columns but the two columns PartTerm and  Date into a new df1
df1 = df0.drop(columns=["PartTerm", "Date"])
#remove the courses in the removed_course list
df1 = df1[~df1['CourseNumber'].isin(removed_course)]

# create a column named "pass" if the grade is in the set passing grades
df1["pass"] = df1["Grade"].isin(passing_grades)
```

```
# Define a function that calculate the rolling average totals of Terms ending
with 01, 02, 03 separtely with a window size WZ, and then adding it as a new
colun to df_department
def rolling_average_total(df, wz=WZ):
    df['TermGroup'] = df.index.map(lambda x: str(x)[-2:])
    df['rolling_total'] = df.groupby('TermGroup')['total'].transform(lambda x:
x.rolling(window=wz, min_periods=1).mean())
    df.drop(columns='TermGroup', inplace=True)
    return df

# Define a function that calculate the rolling average totals of Terms ending
with 01, 02, 03 separtely with a window size WZ, and then adding it as a new
colun to df_department
def  rolling_average_total_by_(df,  group_keys=['SubjectCode','CourseNumber'],
wz=WZ//3):
    df['TermGroup'] = df.Term.map(lambda x: str(x)[-2:])
              df['rolling_total']    =    df.groupby(group_keys+['TermGroup'])
['total'].transform(lambda x: x.rolling(window=wz, min_periods=1).mean())
    df.drop(columns='TermGroup', inplace=True)
    return df
```

```
# Analyze the passing rate grouped by Term, SubjectCode, and Course Number
df_by_course=     df1.groupby(["Term",     "SubjectCode",     "CourseNumber"])
["pass"].agg([("passing","sum"),  ("total","size"), ("rate", "mean")])
df_by_course["rate"] = df_by_course['rate']*100

df_by_course_department=df_by_course.query("SubjectCode in @DEPT")
df_by_course_term                                                          =
df_by_course_department.swaplevel(0,2).swaplevel(0,1).sort_index()
df_by_course_term = df_by_course_term.reset_index("Term")

#calculate the rolling passing rates grouped by SubjectCode and CourseNumber
df_by_course_term['rolling_rate']  =  df_by_course_term.groupby(['SubjectCode',
'CourseNumber']).apply(
     lambda  x:  100  *  x['passing'].rolling(window=WZ,  min_periods=1).sum()  /
x['total'].rolling(window=WZ, min_periods=1).sum()
).reset_index(level=[0,1], drop=True)
df_by_course_term = rolling_average_total_by_(df_by_course_term)

# calculate the department summary statistics for each group in df4
df_department           =           df_by_course_department.groupby(['Term'])
[['passing','total']].sum()
df_department['rate'] = 100* df_department['passing'] / df_department['total']
#calculate the rolling passing grades for df_department with window size of 6
df_department['rolling_rate'] = \
    100*df_department['passing'].rolling(window=WZ,min_periods=1).sum()\
        /df_department['total'].rolling(window=WZ,min_periods=1).sum()
```

```
# Apply the function rolling_average_total to df_department to create a new col
called 'rolling_total'
df_department = rolling_average_total(df_department, WZ//3)
```

Analysis by Instructors

```
def calculate_group_df(df=df1, group="School", wz=WZ//3):
          df   =   df.groupby(["Term",   group])["pass"].agg([("passing","sum"),
("total","size"), ("rate", "mean")])
    df["rate"] = df['rate']*100
    df = df.swaplevel(0,1).sort_index()
    df = df.reset_index("Term")
    df['rolling_rate'] = df.groupby([group]).apply(
        lambda x: 100 * x['passing'].rolling(window=WZ, min_periods=1).sum() /
x['total'].rolling(window=WZ, min_periods=1).sum()
        ).reset_index(level=[0], drop=True)
    df=rolling_average_total_by_(df,group_keys=[group],wz=wz)
    return df

def   group_table(df,   Query=['Nxuvxq,   Rofdb','Bdq,   Kxd','Djd,   Prvlvd'],
col="Instructor", var='rate', number_sem_display=number_sem_display):

    df_course = df.loc[Query]
        df_course_rate  =  df_course.pivot_table(index=[col],  columns="Term",
values=var)
    df_course_rolling_rate = df_course.pivot_table(index=[col], columns="Term",
values="rolling_"+ var)
    df_course_rate = df_course_rate.iloc[:, -number_sem_display:]
              df_course_rolling_rate    =    df_course_rolling_rate.iloc[:,    -
number_sem_display:]
    df_course_rate.loc['Dept'] = df_department[var]
    df_course_rolling_rate.loc['Dept'] = df_department['rolling_' + var]
    return df_course_rate, df_course_rolling_rate
```

```
df_by_instructor =  calculate_group_df(df=df1, group="Instructor")
rate,rolling_rate     =     group_table(df=df_by_instructor,     Query=['Nxuvxq,
Rofdb','Bdq, Kxd','Djd, Prvlvd'], number_sem_display=number_sem_display)
```

```
# with pd.option_context('display.precision', 1):
display(Markdown("### Passing Rate Table"))
display(rate.fillna("**").style.set_properties(**{'font-size':
'10pt'}).format(precision=1))
```

Passing Rate Table

Term	202403	202501	202502	202503	202601	202602	202603	202701	202702	202703
Instructor										
Bdq, Kxd	94.3	90.6	82.7	98.1	92.6	93.6	98.8	96.7	95.3	100.0
Djd, Prvlvd	92.3	82.2	85.4	85.4	80.2	71.0	80.0	81.6	86.4	83.3
Nxuvxq, Rofdb	**	93.8	90.9	100.0	96.4	98.9	97.5	96.4	96.4	95.0
Dept	70.9	64.8	62.6	70.0	64.2	64.8	76.7	70.0	68.5	86.7

```
# with pd.option_context('display.precision', 1):
display(Markdown("### Rolling Rate Table"))
display(rolling_rate.fillna("**").style.set_properties(**{'font-size':
'10pt'}).format(precision=1))
```

Rolling Rate Table

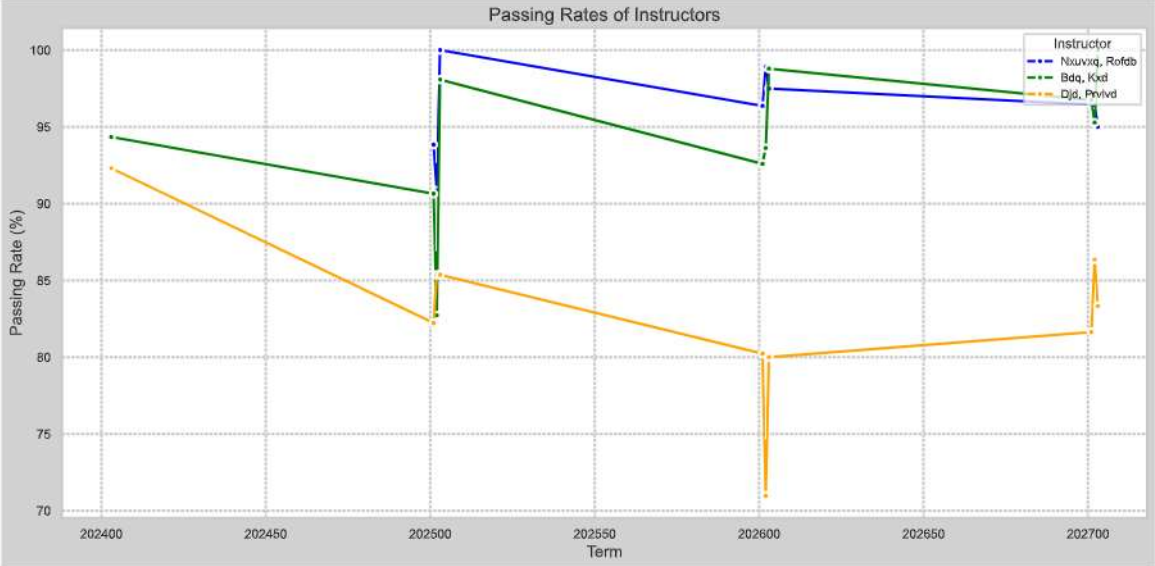
Term	202403	202501	202502	202503	202601	202602	202603	202701	202702	202703
Instructor										
Bdq, Kxd	90.8	90.7	88.9	89.7	89.6	91.2	91.8	94.1	95.7	95.7
Djd, Prvlvd	83.3	83.0	81.9	82.5	83.5	81.1	80.0	79.9	80.5	80.2
Nxuvxq, Rofdb	**	93.8	91.9	93.4	94.5	95.4	95.5	95.9	97.3	96.9
Dept	77.6	75.1	70.0	69.3	66.9	64.8	65.2	66.4	67.5	68.2

Passing Rates of group of 3 instructors

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

#passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
```

```
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

# Combining data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})

sns.set(style="whitegrid")


fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

#first instructor
sns.lineplot(x='Term',     y='rate_1',     data=combined_data,     marker='o',
color='blue', ax=ax, label=instructors[0])

#second instructor
sns.lineplot(x='Term',     y='rate_2',     data=combined_data,     marker='o',
color='green', ax=ax, label=instructors[1])

# third instructor
sns.lineplot(x='Term',     y='rate_3',     data=combined_data,     marker='o',
color='orange', ax=ax, label=instructors[2])

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

#control missing data
for i in range(len(combined_data) - 1):
    if pd.isnull(combined_data['rate_1'].iloc[i + 1]):
        ax.annotate('Missing Data',
                                            xy=(combined_data['Term'].iloc[i],
combined_data['rate_1'].iloc[i]),
                                        xytext=(combined_data['Term'].iloc[i],
combined_data['rate_1'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))
    if pd.isnull(combined_data['rate_2'].iloc[i + 1]):
        ax.annotate('Missing Data',
                                            xy=(combined_data['Term'].iloc[i],
combined_data['rate_2'].iloc[i]),
                                        xytext=(combined_data['Term'].iloc[i],
```

```
combined_data['rate_2'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))
    if pd.isnull(combined_data['rate_3'].iloc[i + 1]):
        ax.annotate('Missing Data',
                                            xy=(combined_data['Term'].iloc[i],
combined_data['rate_3'].iloc[i]),
                                        xytext=(combined_data['Term'].iloc[i],
combined_data['rate_3'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))

#legend
ax.legend(title='Instructor', loc='upper right', fontsize=10)


plt.tight_layout()
plt.show()
```

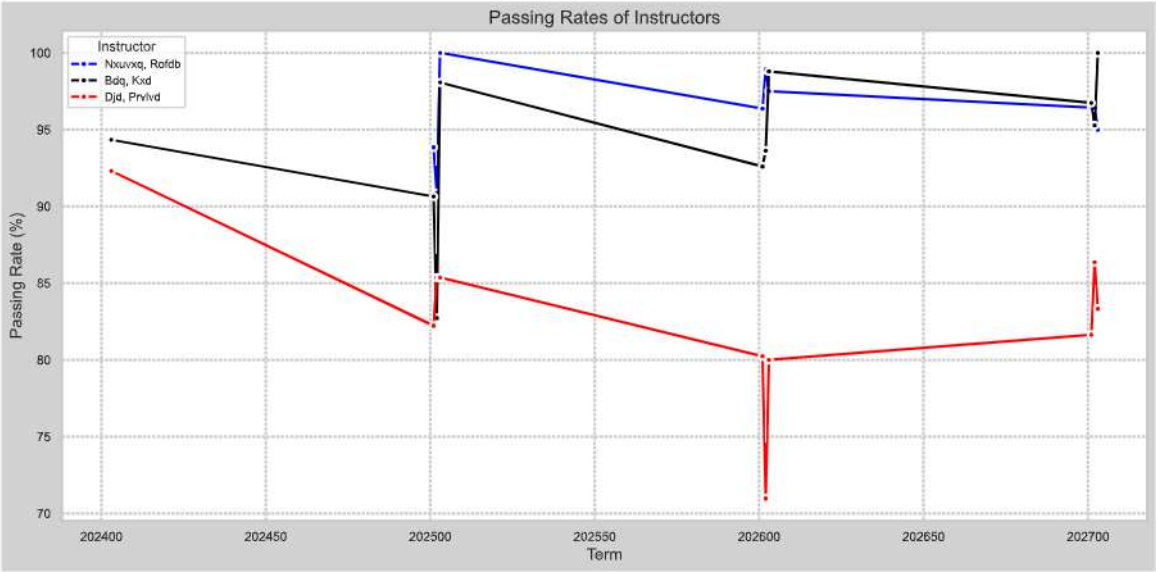


3 Instructors Passing rate plot 1.1

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

# passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
```

```
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

# Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")

#subplots
fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

#first instructor
sns.lineplot(x='Term',     y='rate_1',     data=combined_data,     marker='o',
color='blue', ax=ax, label=instructors[0])

#second instructor
sns.lineplot(x='Term',     y='rate_2',     data=combined_data,     marker='o',
color='black', ax=ax, label=instructors[1])

# third instructor
sns.lineplot(x='Term', y='rate_3', data=combined_data, marker='o', color='red',
ax=ax, label=instructors[2])

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

# control  missing data
for i in range(len(combined_data) - 1):
    if pd.isnull(combined_data['rate_1'].iloc[i + 1]):
        ax.annotate('Missing Data',
                                            xy=(combined_data['Term'].iloc[i],
combined_data['rate_1'].iloc[i]),
                                        xytext=(combined_data['Term'].iloc[i],
combined_data['rate_1'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))
    if pd.isnull(combined_data['rate_2'].iloc[i + 1]):
        ax.annotate('Missing Data',
                                            xy=(combined_data['Term'].iloc[i],
combined_data['rate_2'].iloc[i]),
                                        xytext=(combined_data['Term'].iloc[i],
combined_data['rate_2'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))
```

```
    if pd.isnull(combined_data['rate_3'].iloc[i + 1]):
        ax.annotate('Missing Data',
                                            xy=(combined_data['Term'].iloc[i],
combined_data['rate_3'].iloc[i]),
                                         xytext=(combined_data['Term'].iloc[i],
combined_data['rate_3'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))


ax.legend(title='Instructor', loc='upper left', fontsize=10)


plt.tight_layout()
plt.show()
```

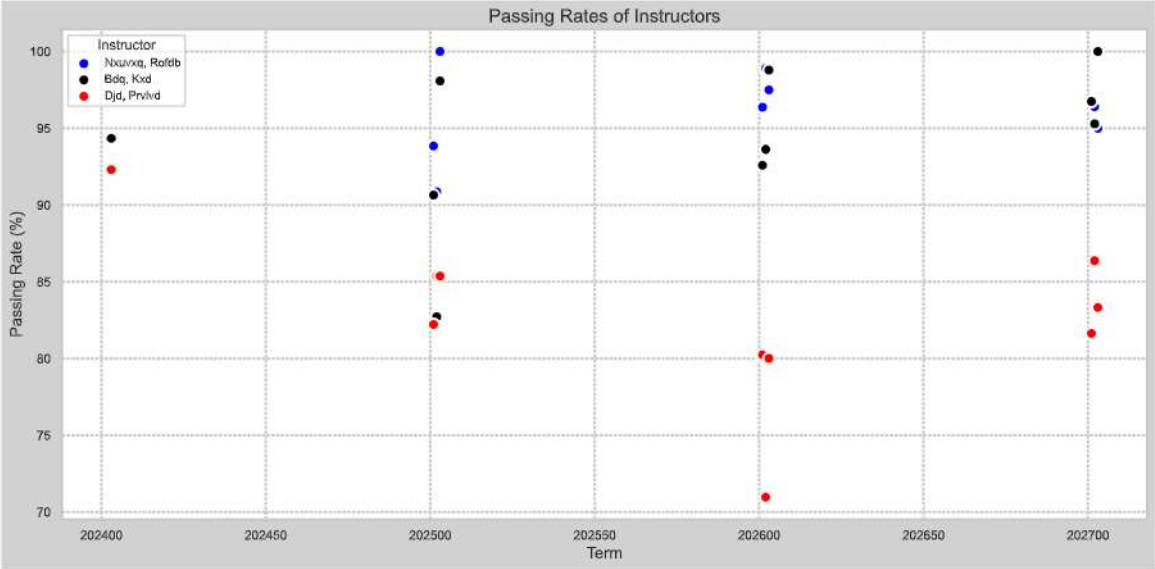


1.2

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

#passing rates
data_instructor1                                                              =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                              =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                              =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})
```

```
# Combine data
combined_data = pd.merge(data_instructor1, data_instructor2, on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data = pd.merge(combined_data, data_instructor3, on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")


fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

# first instructor
sns.scatterplot(x='Term', y='rate_1', data=combined_data, color='blue', ax=ax,
label=instructors[0], s=100)

#second instructor
sns.scatterplot(x='Term', y='rate_2', data=combined_data, color='black', ax=ax,
label=instructors[1], s=100)

#third instructor
sns.scatterplot(x='Term', y='rate_3', data=combined_data, color='red', ax=ax,
label=instructors[2], s=100)

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

#control missing data
for i in range(len(combined_data) - 1):
    if pd.isnull(combined_data['rate_1'].iloc[i + 1]):
        ax.annotate('Missing Data',
                                            xy=(combined_data['Term'].iloc[i],
combined_data['rate_1'].iloc[i]),
                                        xytext=(combined_data['Term'].iloc[i],
combined_data['rate_1'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))
    if pd.isnull(combined_data['rate_2'].iloc[i + 1]):
        ax.annotate('Missing Data',
                                            xy=(combined_data['Term'].iloc[i],
combined_data['rate_2'].iloc[i]),
                                        xytext=(combined_data['Term'].iloc[i],
combined_data['rate_2'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))
    if pd.isnull(combined_data['rate_3'].iloc[i + 1]):
        ax.annotate('Missing Data',
```

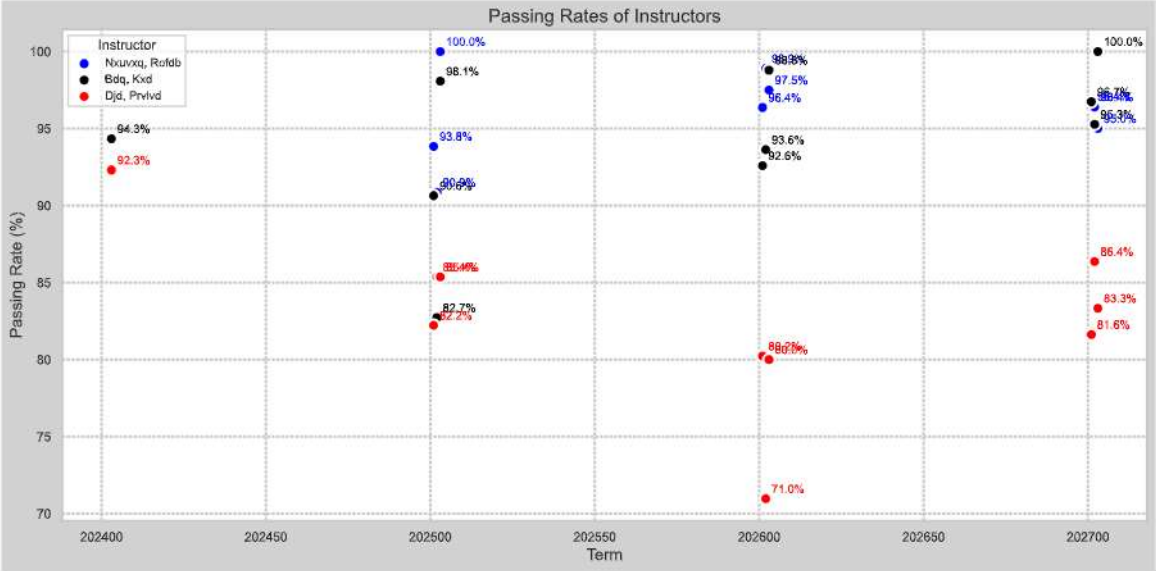
```
                                                xy=(combined_data['Term'].iloc[i],
combined_data['rate_3'].iloc[i]),
                                              xytext=(combined_data['Term'].iloc[i],
combined_data['rate_3'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))

#legend
ax.legend(title='Instructor', loc='upper left', fontsize=10)

plt.tight_layout()
plt.show()
```



1.3

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

#passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

# Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
```

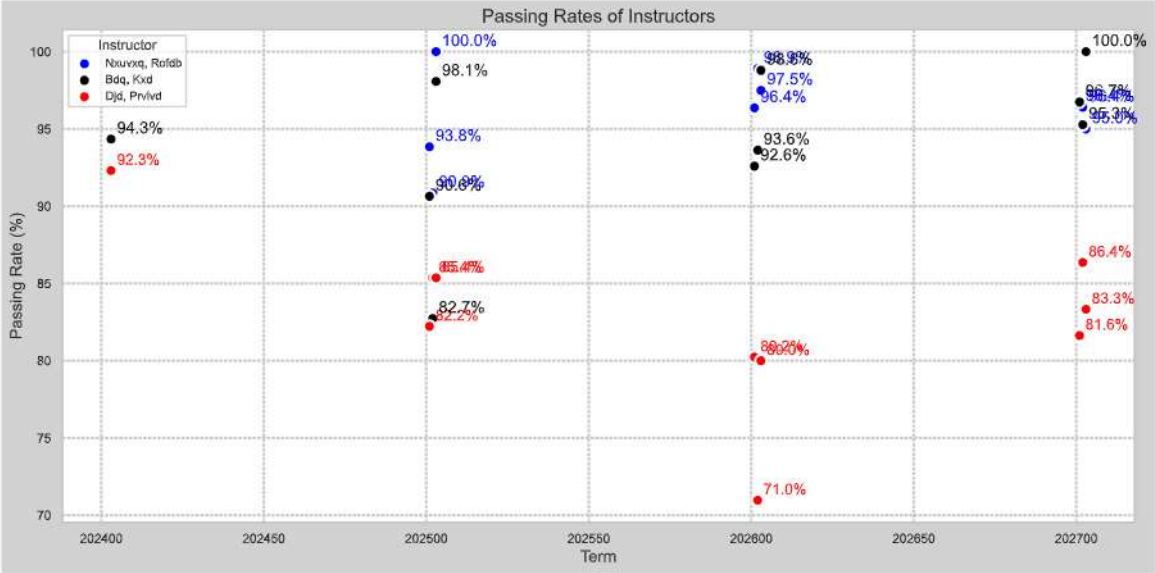
```
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")


fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

#first instructor
sns.scatterplot(x='Term', y='rate_1', data=combined_data, color='blue', ax=ax,
label=instructors[0], s=100)

#sec instructor
sns.scatterplot(x='Term', y='rate_2', data=combined_data, color='black', ax=ax,
label=instructors[1], s=100)

#third instructor
sns.scatterplot(x='Term',  y='rate_3',  data=combined_data,  color='red',  ax=ax,
label=instructors[2], s=100)

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

#passing rates
for index, row in combined_data.iterrows():
    if not pd.isnull(row['rate_1']):
         ax.annotate(f'{row["rate_1"]:.1f}%', xy=(row['Term'], row['rate_1']),
xytext=(5, 5), textcoords='offset points', fontsize=10, color='blue')
    if not pd.isnull(row['rate_2']):
         ax.annotate(f'{row["rate_2"]:.1f}%', xy=(row['Term'], row['rate_2']),
xytext=(5, 5), textcoords='offset points', fontsize=10, color='black')
    if not pd.isnull(row['rate_3']):
         ax.annotate(f'{row["rate_3"]:.1f}%', xy=(row['Term'], row['rate_3']),
xytext=(5, 5), textcoords='offset points', fontsize=10, color='red')

#legend
ax.legend(title='Instructor', loc='upper left', fontsize=10)


plt.tight_layout()
plt.show()
```



1.4

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

# passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

# Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")

#subplots
fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

#first instructor
sns.scatterplot(x='Term', y='rate_1', data=combined_data, color='blue', ax=ax,
label=instructors[0], s=100)
```

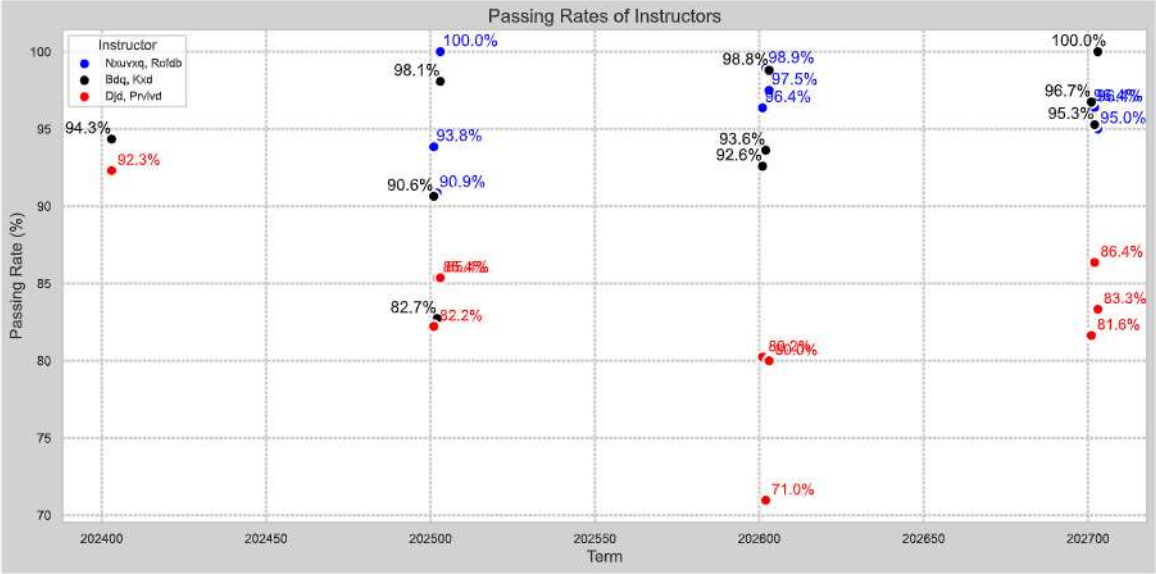
```
# second instructor
sns.scatterplot(x='Term', y='rate_2', data=combined_data, color='black', ax=ax,
label=instructors[1], s=100)

# third instructor
sns.scatterplot(x='Term', y='rate_3', data=combined_data, color='red', ax=ax,
label=instructors[2], s=100)

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

#passing rates
for index, row in combined_data.iterrows():
    if not pd.isnull(row['rate_1']):
         ax.annotate(f'{row["rate_1"]:.1f}%', xy=(row['Term'], row['rate_1']),
xytext=(5, 5), textcoords='offset points', fontsize=14, color='blue')
    if not pd.isnull(row['rate_2']):
         ax.annotate(f'{row["rate_2"]:.1f}%', xy=(row['Term'], row['rate_2']),
xytext=(5, 5), textcoords='offset points', fontsize=14, color='black')
    if not pd.isnull(row['rate_3']):
         ax.annotate(f'{row["rate_3"]:.1f}%', xy=(row['Term'], row['rate_3']),
xytext=(5, 5), textcoords='offset points', fontsize=13, color='red')


ax.legend(title='Instructor', loc='upper left', fontsize=10)


plt.tight_layout()
plt.show()
```



1.5

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

# passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

#Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")

#subplots
fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

# first instructor
sns.scatterplot(x='Term', y='rate_1', data=combined_data, color='blue', ax=ax,
label=instructors[0], s=100)
```

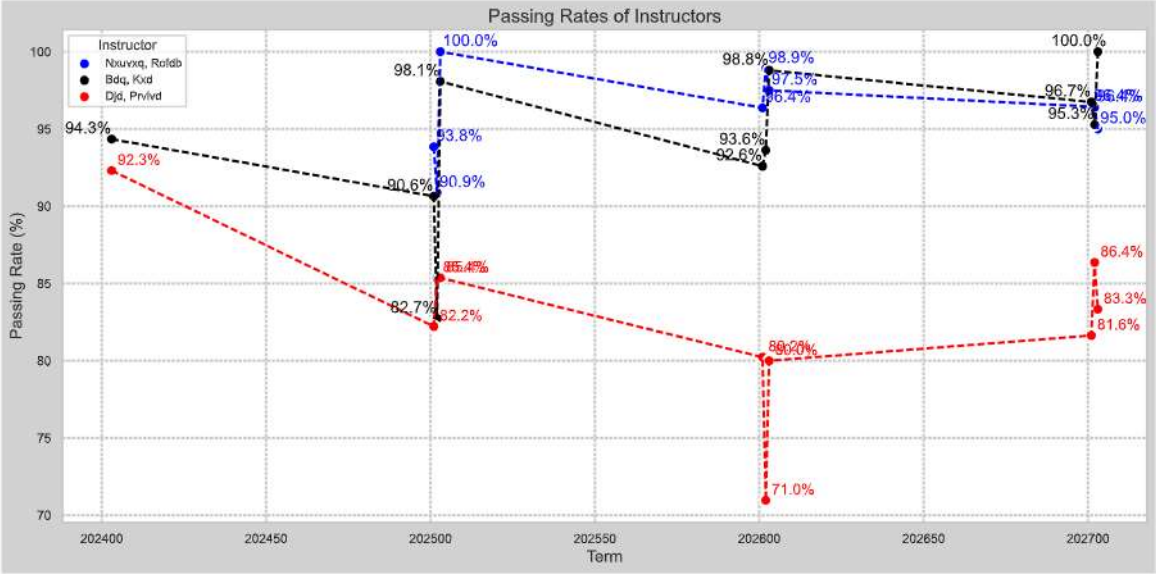
```
# second instructor
sns.scatterplot(x='Term', y='rate_2', data=combined_data, color='black', ax=ax,
label=instructors[1], s=100)

# third instructor
sns.scatterplot(x='Term', y='rate_3', data=combined_data, color='red', ax=ax,
label=instructors[2], s=100)

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

# passing rates
for index, row in combined_data.iterrows():
    if not pd.isnull(row['rate_1']):
         ax.annotate(f'{row["rate_1"]:.1f}%', xy=(row['Term'], row['rate_1']),
xytext=(2, 5), textcoords='offset points', fontsize=14, color='blue')
    if not pd.isnull(row['rate_2']):
         ax.annotate(f'{row["rate_2"]:.1f}%', xy=(row['Term'], row['rate_2']),
xytext=(-40, 5), textcoords='offset points', fontsize=14, color='black')
    if not pd.isnull(row['rate_3']):
         ax.annotate(f'{row["rate_3"]:.1f}%', xy=(row['Term'], row['rate_3']),
xytext=(5, 5), textcoords='offset points', fontsize=13, color='red')

#legend
ax.legend(title='Instructor', loc='upper left', fontsize=10)


plt.tight_layout()
plt.show()
```



1.6

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

#passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

# Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")


fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

#first instructor
sns.scatterplot(x='Term', y='rate_1', data=combined_data, color='blue', ax=ax,
label=instructors[0], s=100)
```

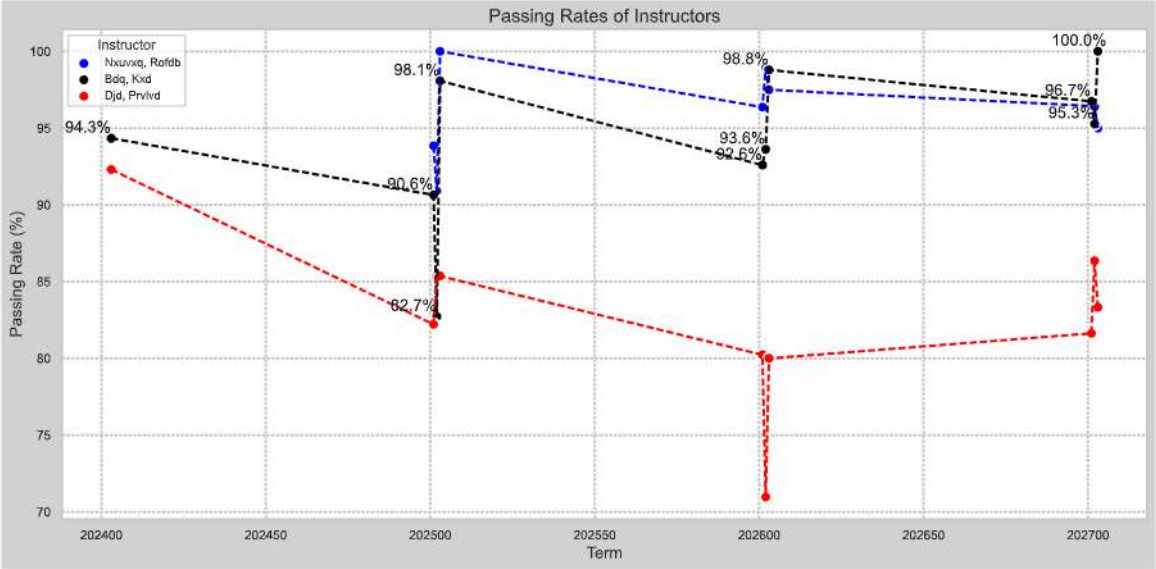
```
ax.plot(combined_data['Term'],     combined_data['rate_1'],     linestyle='--',
color='blue')

# second instructor
sns.scatterplot(x='Term', y='rate_2', data=combined_data, color='black', ax=ax,
label=instructors[1], s=100)
ax.plot(combined_data['Term'],     combined_data['rate_2'],     linestyle='--',
color='black')

#third instructor
sns.scatterplot(x='Term',  y='rate_3', data=combined_data, color='red', ax=ax,
label=instructors[2], s=100)
ax.plot(combined_data['Term'],     combined_data['rate_3'],     linestyle='--',
color='red')

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

#passing rates
for index, row in combined_data.iterrows():
    if not pd.isnull(row['rate_1']):
          ax.annotate(f'{row["rate_1"]:.1f}%', xy=(row['Term'], row['rate_1']),
xytext=(2, 5), textcoords='offset points', fontsize=14, color='blue')
    if not pd.isnull(row['rate_2']):
          ax.annotate(f'{row["rate_2"]:.1f}%', xy=(row['Term'], row['rate_2']),
xytext=(-40, 5), textcoords='offset points', fontsize=14, color='black')
    if not pd.isnull(row['rate_3']):
          ax.annotate(f'{row["rate_3"]:.1f}%', xy=(row['Term'], row['rate_3']),
xytext=(5, 5), textcoords='offset points', fontsize=13, color='red')


ax.legend(title='Instructor', loc='upper left', fontsize=10)

plt.tight_layout()
plt.show()
```



1.7

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

#passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

#Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")


fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

# first instructor
sns.scatterplot(x='Term', y='rate_1', data=combined_data, color='blue', ax=ax,
label=instructors[0], s=100)
```

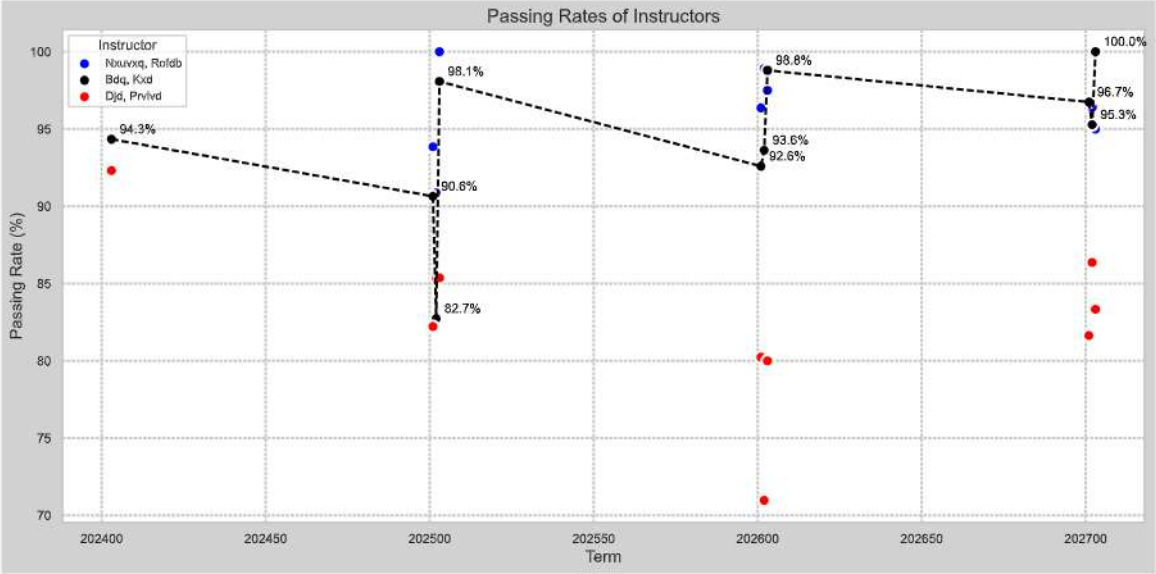
```
ax.plot(combined_data['Term'],     combined_data['rate_1'],     linestyle='--',
color='blue')

# second instructor
sns.scatterplot(x='Term', y='rate_2', data=combined_data, color='black', ax=ax,
label=instructors[1], s=100)
ax.plot(combined_data['Term'],     combined_data['rate_2'],     linestyle='--',
color='black')

#third instructor
sns.scatterplot(x='Term',  y='rate_3', data=combined_data, color='red', ax=ax,
label=instructors[2], s=100)
ax.plot(combined_data['Term'],     combined_data['rate_3'],     linestyle='--',
color='red')

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

# Annotate passing rate values for all instructors
for index, row in combined_data.iterrows():
    #if not pd.isnull(row['rate_1']):
        #ax.annotate(f'{row["rate_1"]:.1f}%', xy=(row['Term'], row['rate_1']),
xytext=(2, 5), textcoords='offset points', fontsize=14, color='blue')
    if not pd.isnull(row['rate_2']):
         ax.annotate(f'{row["rate_2"]:.1f}%', xy=(row['Term'], row['rate_2']),
xytext=(-40, 5), textcoords='offset points', fontsize=14, color='black')
    # if not pd.isnull(row['rate_3']):
        #ax.annotate(f'{row["rate_3"]:.1f}%', xy=(row['Term'], row['rate_3']),
xytext=(5, 5), textcoords='offset points', fontsize=13, color='red')

#legend
ax.legend(title='Instructor', loc='upper left', fontsize=10)

plt.tight_layout()
plt.show()
```



1.8

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

#passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

# Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")

# subplots
fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

# first instructor
sns.scatterplot(x='Term', y='rate_1', data=combined_data, color='blue', ax=ax,
label=instructors[0], s=100)
```

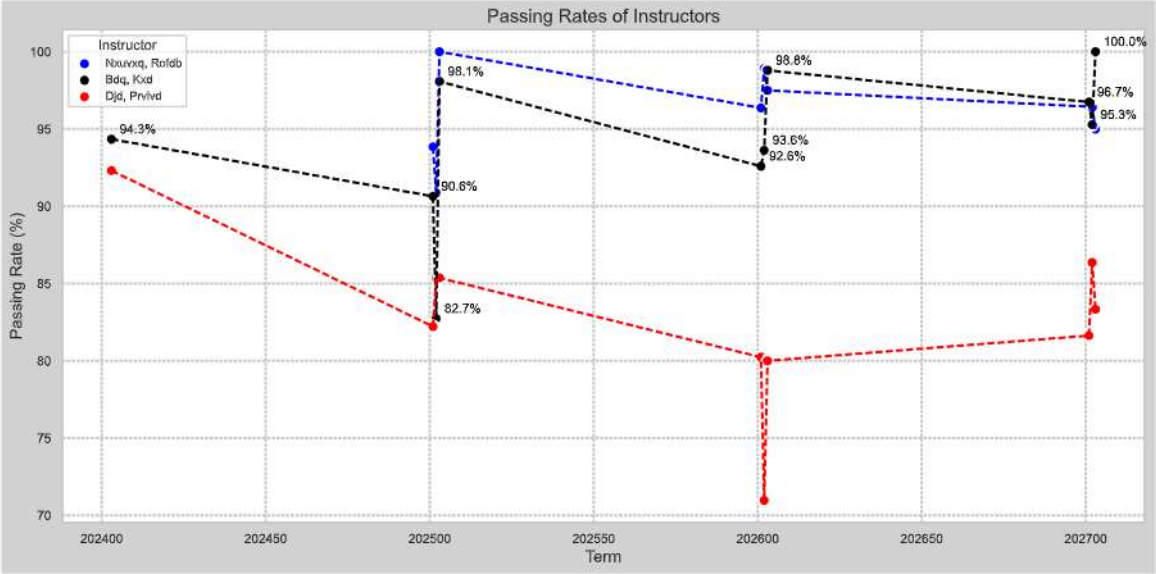
```
#ax.plot(combined_data['Term'],    combined_data['rate_1'],    linestyle='--',
color='blue')

#second instructor
sns.scatterplot(x='Term', y='rate_2', data=combined_data, color='black', ax=ax,
label=instructors[1], s=100)
ax.plot(combined_data['Term'],    combined_data['rate_2'],    linestyle='--',
color='black')

#third instructor
sns.scatterplot(x='Term', y='rate_3', data=combined_data, color='red', ax=ax,
label=instructors[2], s=100)
#ax.plot(combined_data['Term'],    combined_data['rate_3'],    linestyle='--',
color='red')

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

# Annotate passing rate values for all instructors
for index, row in combined_data.iterrows():
    #if not pd.isnull(row['rate_1']):
        #ax.annotate(f'{row["rate_1"]:.1f}%', xy=(row['Term'], row['rate_1']),
xytext=(2, 5), textcoords='offset points', fontsize=14, color='blue')
    if not pd.isnull(row['rate_2']):
         ax.annotate(f'{row["rate_2"]:.1f}%', xy=(row['Term'], row['rate_2']),
xytext=(7, 5), textcoords='offset points', fontsize=11, color='black')
    # if not pd.isnull(row['rate_3']):
        #ax.annotate(f'{row["rate_3"]:.1f}%', xy=(row['Term'], row['rate_3']),
xytext=(5, 5), textcoords='offset points', fontsize=13, color='red')

#legend
ax.legend(title='Instructor', loc='upper left', fontsize=10)

plt.tight_layout()
plt.show()
```



1.9

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

# passing rates
data_instructor1                                                    =
rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rate'})
data_instructor2                                                    =
rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rate'})
data_instructor3                                                    =
rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rate'})

# Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rate': 'rate_3'})


sns.set(style="whitegrid")

#subplots
fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

# first instructor
sns.scatterplot(x='Term', y='rate_1', data=combined_data, color='blue', ax=ax,
label=instructors[0], s=100)
```

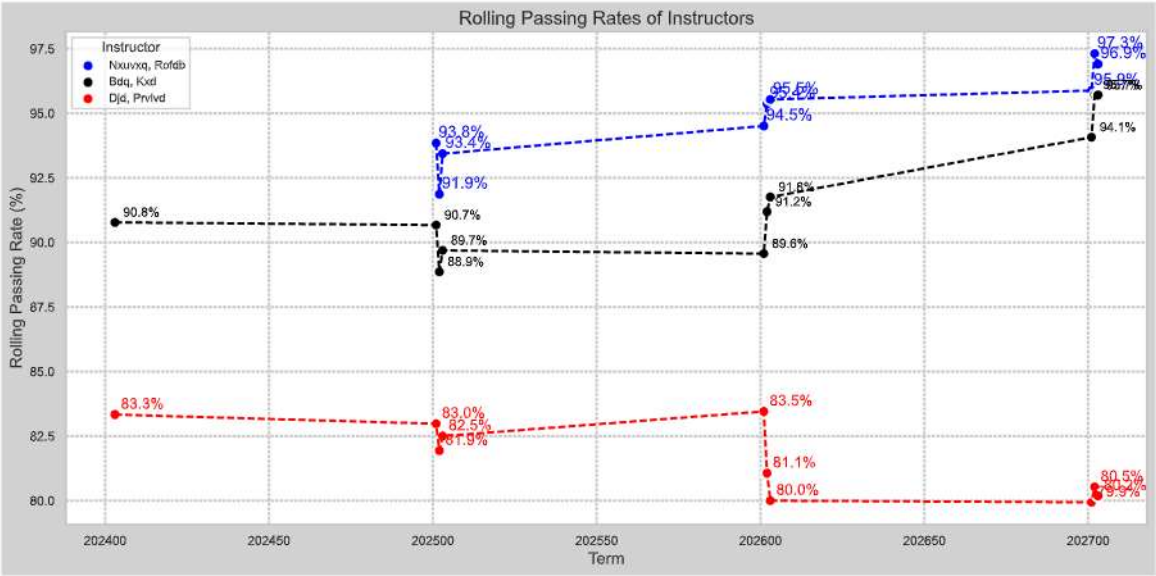
```
ax.plot(combined_data['Term'],     combined_data['rate_1'],     linestyle='--',
color='blue')

#second instructor
sns.scatterplot(x='Term', y='rate_2', data=combined_data, color='black', ax=ax,
label=instructors[1], s=100)
ax.plot(combined_data['Term'],     combined_data['rate_2'],     linestyle='--',
color='black')

# third instructor
sns.scatterplot(x='Term',  y='rate_3', data=combined_data, color='red', ax=ax,
label=instructors[2], s=100)
ax.plot(combined_data['Term'],     combined_data['rate_3'],     linestyle='--',
color='red')

ax.set_title('Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

# Annotate passing rate values for all instructors
for index, row in combined_data.iterrows():
    #if not pd.isnull(row['rate_1']):
        #ax.annotate(f'{row["rate_1"]:.1f}%', xy=(row['Term'], row['rate_1']),
xytext=(2, 5), textcoords='offset points', fontsize=14, color='blue')
    if not pd.isnull(row['rate_2']):
         ax.annotate(f'{row["rate_2"]:.1f}%', xy=(row['Term'], row['rate_2']),
xytext=(7, 5), textcoords='offset points', fontsize=11, color='black')
    # if not pd.isnull(row['rate_3']):
        #ax.annotate(f'{row["rate_3"]:.1f}%', xy=(row['Term'], row['rate_3']),
xytext=(5, 5), textcoords='offset points', fontsize=13, color='red')

#legend
ax.legend(title='Instructor', loc='upper left', fontsize=10)


plt.tight_layout()
plt.show()
```



Rolling Passing Rate for 3 Instructors

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

# rolling passing rates
data_instructor1 =
rolling_rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rolling_rate'})
data_instructor2 =
rolling_rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rolling_rate'})
data_instructor3 =
rolling_rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rolling_rate'})

# Combine data
combined_data = pd.merge(data_instructor1, data_instructor2, on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data = pd.merge(combined_data, data_instructor3, on='Term',
how='outer').rename(columns={'rolling_rate': 'rolling_rate_3'})


sns.set(style="whitegrid")

#subplots
fig, ax = plt.subplots(figsize=(14, 7), facecolor='lightgray')

#first instructor
sns.scatterplot(x='Term', y='rolling_rate_1', data=combined_data, color='blue',
ax=ax, label=instructors[0], s=100)
```

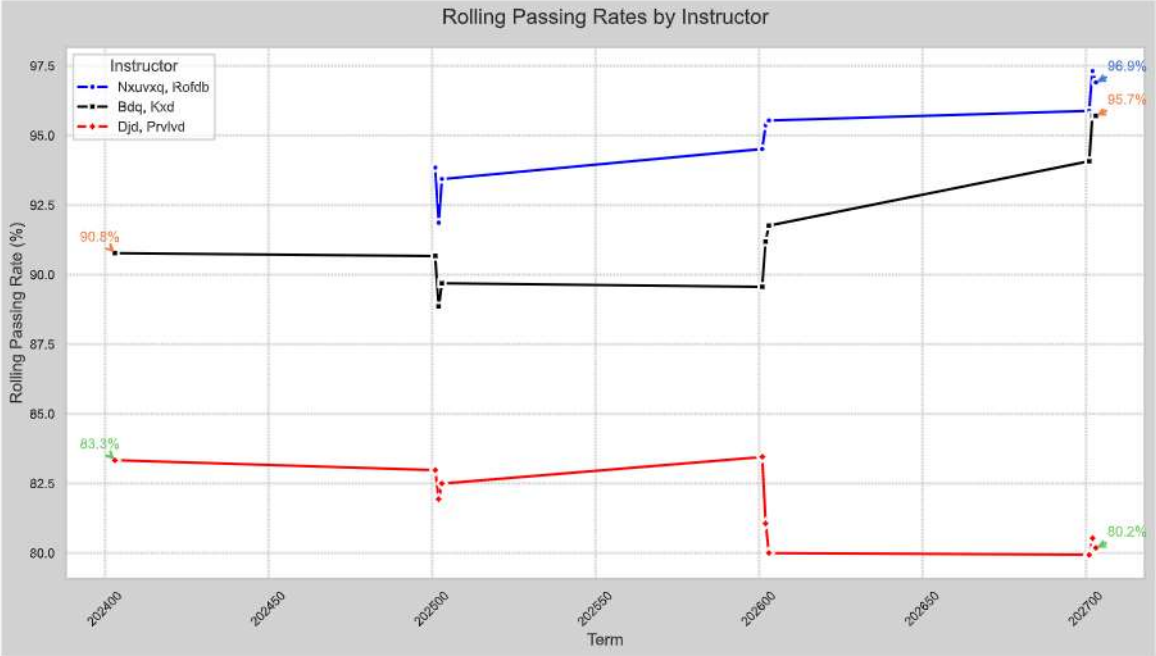
```
ax.plot(combined_data['Term'], combined_data['rolling_rate_1'], linestyle='--',
color='blue')

#second instructor
sns.scatterplot(x='Term',        y='rolling_rate_2',        data=combined_data,
color='black', ax=ax, label=instructors[1], s=100)
ax.plot(combined_data['Term'], combined_data['rolling_rate_2'], linestyle='--',
color='black')

#third instructor
sns.scatterplot(x='Term', y='rolling_rate_3', data=combined_data, color='red',
ax=ax, label=instructors[2], s=100)
ax.plot(combined_data['Term'], combined_data['rolling_rate_3'], linestyle='--',
color='red')

ax.set_title('Rolling Passing Rates of Instructors', fontsize=16)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Rolling Passing Rate (%)', fontsize=14)
ax.grid(True, which='both', linestyle='--', linewidth=0.7)

# Annotations for rolling passing rate
for index, row in combined_data.iterrows():
    if not pd.isnull(row['rolling_rate_1']):
                ax.annotate(f'{row["rolling_rate_1"]:.1f}%',  xy=(row['Term'],
row['rolling_rate_1']), xytext=(2, 5), textcoords='offset points', fontsize=14,
color='blue')
    if not pd.isnull(row['rolling_rate_2']):
                ax.annotate(f'{row["rolling_rate_2"]:.1f}%',  xy=(row['Term'],
row['rolling_rate_2']), xytext=(7, 5), textcoords='offset points', fontsize=11,
color='black')
    if not pd.isnull(row['rolling_rate_3']):
                ax.annotate(f'{row["rolling_rate_3"]:.1f}%',  xy=(row['Term'],
row['rolling_rate_3']), xytext=(5, 5), textcoords='offset points', fontsize=13,
color='red')


ax.legend(title='Instructor', loc='upper left', fontsize=10)


plt.tight_layout()
plt.show()
```



Rolling Passing rate for 3 Instructors(chart)

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

# rolling passing rates
data_instructor1                                                    =
rolling_rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rolling_rate'})
data_instructor2                                                    =
rolling_rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rolling_rate'})
data_instructor3                                                    =
rolling_rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rolling_rate'})

# Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rolling_rate': 'rolling_rate_3'})

sns.set_theme(style="whitegrid", palette="muted")

#subplots
fig, ax = plt.subplots(figsize=(14, 8), facecolor='lightgray')
colors = sns.color_palette("muted", 3)

# instructor rolling rate
sns.lineplot(x='Term', y='rolling_rate_1', data=combined_data, label='Nxuvxq,
Rofdb', color='blue', marker='o', ax=ax)
```

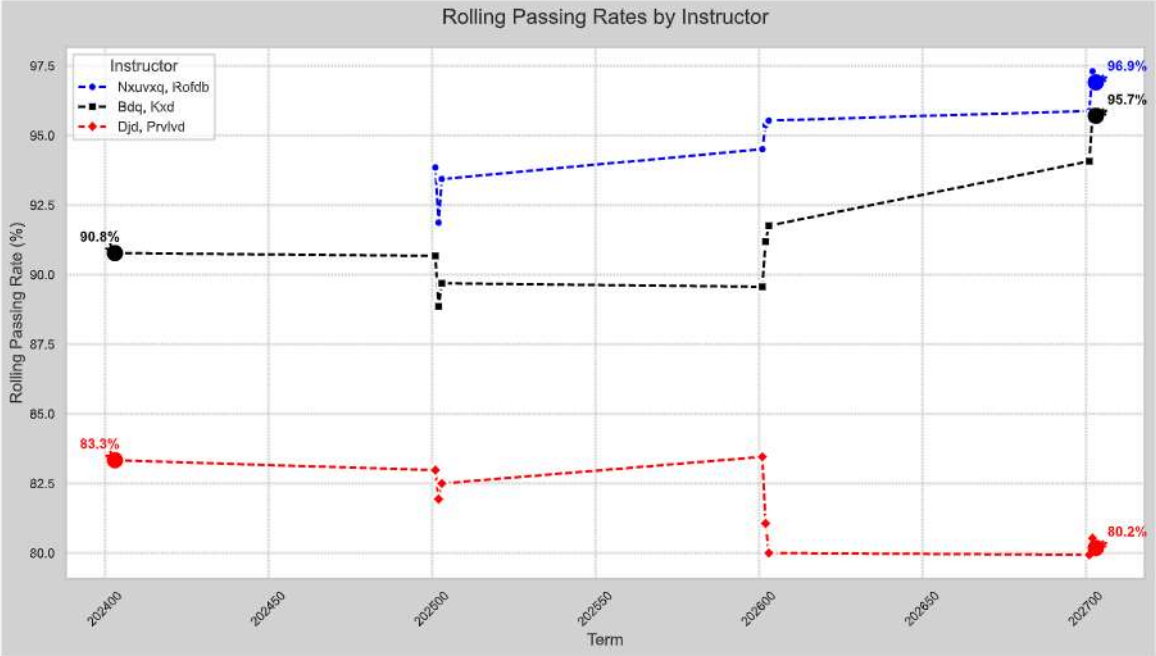
```
sns.lineplot(x='Term',  y='rolling_rate_2',  data=combined_data,  label='Bdq,
Kxd', color='black', marker='s', ax=ax)
sns.lineplot(x='Term',  y='rolling_rate_3',  data=combined_data,  label='Djd,
Prvlvd', color='red', marker='D', ax=ax)

# Add annotations
for col, color in zip(['rolling_rate_1', 'rolling_rate_2', 'rolling_rate_3'],
colors):
    ax.annotate(f'{combined_data[col].iloc[0]:.1f}%',
                xy=(combined_data['Term'].iloc[0], combined_data[col].iloc[0]),
                     xytext=(-30, 10), textcoords='offset points', color=color,
fontsize=12, arrowprops=dict(arrowstyle="->", color=color))
    ax.annotate(f'{combined_data[col].iloc[-1]:.1f}%',
               xy=(combined_data['Term'].iloc[-1], combined_data[col].iloc[-1]),
                      xytext=(10, 10), textcoords='offset points', color=color,
fontsize=12, arrowprops=dict(arrowstyle="->", color=color))

# axis labels
ax.set_title('Rolling Passing Rates by Instructor', fontsize=18, pad=20)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Rolling Passing Rate (%)', fontsize=14)

ax.legend(title='Instructor',  title_fontsize=14,  fontsize=12,  loc='best',
frameon=True)
plt.xticks(rotation=45)

#transparency
ax.grid(visible=True, which='both', linestyle='--', linewidth=0.5, alpha=0.7)


plt.tight_layout()

plt.show()
```



1.1

```
instructors = ['Nxuvxq, Rofdb', 'Bdq, Kxd', 'Djd, Prvlvd']

# Extract rolling passing rates
data_instructor1                                                    =
rolling_rate.loc[instructors[0]].reset_index().rename(columns={instructors[0]:
'rolling_rate'})
data_instructor2                                                    =
rolling_rate.loc[instructors[1]].reset_index().rename(columns={instructors[1]:
'rolling_rate'})
data_instructor3                                                    =
rolling_rate.loc[instructors[2]].reset_index().rename(columns={instructors[2]:
'rolling_rate'})

# Combine data
combined_data   =   pd.merge(data_instructor1,   data_instructor2,   on='Term',
how='outer', suffixes=('_1', '_2'))
combined_data    =    pd.merge(combined_data,    data_instructor3,    on='Term',
how='outer').rename(columns={'rolling_rate': 'rolling_rate_3'})

sns.set_theme(style="whitegrid", palette="muted")

# subplots
fig, ax = plt.subplots(figsize=(14, 8), facecolor='lightgray')
colors = sns.color_palette("muted", 3)
```

```
# Plot rolling rates
sns.lineplot(x='Term', y='rolling_rate_1', data=combined_data, label='Nxuvxq,
Rofdb', color='blue', marker='o', linestyle='--', ax=ax, markersize=8)
sns.lineplot(x='Term',  y='rolling_rate_2',  data=combined_data,  label='Bdq,
Kxd', color='black', marker='s', linestyle='--', ax=ax, markersize=8)
sns.lineplot(x='Term',  y='rolling_rate_3',  data=combined_data,  label='Djd,
Prvlvd', color='red', marker='D', linestyle='--', ax=ax, markersize=8)

# Emphasize first and last points
for col, color in zip(['rolling_rate_1', 'rolling_rate_2', 'rolling_rate_3'],
['blue', 'black', 'red']):
      ax.plot(combined_data['Term'].iloc[0], combined_data[col].iloc[0], 'o',
markersize=12, color=color)
    ax.annotate(f'{combined_data[col].iloc[0]:.1f}%',
              xy=(combined_data['Term'].iloc[0], combined_data[col].iloc[0]),
                   xytext=(-30, 10), textcoords='offset points', color=color,
fontsize=12, fontweight='bold', arrowprops=dict(arrowstyle="->", color=color))

     ax.plot(combined_data['Term'].iloc[-1], combined_data[col].iloc[-1], 'o',
markersize=12, color=color)
    ax.annotate(f'{combined_data[col].iloc[-1]:.1f}%',
             xy=(combined_data['Term'].iloc[-1], combined_data[col].iloc[-1]),
                    xytext=(10, 10), textcoords='offset points', color=color,
fontsize=12, fontweight='bold', arrowprops=dict(arrowstyle="->", color=color))

# axis labels
ax.set_title('Rolling Passing Rates by Instructor', fontsize=18, pad=20)
ax.set_xlabel('Term', fontsize=14)
ax.set_ylabel('Rolling Passing Rate (%)', fontsize=14)

ax.legend(title='Instructor',   title_fontsize=14,   fontsize=12,   loc='best',
frameon=True)
plt.xticks(rotation=45)

# transparency
ax.grid(visible=True, which='both', linestyle='--', linewidth=0.5, alpha=0.7)

plt.tight_layout()

plt.show()
```



Now that I have an example of the passing rate/rolling passing rate for a group of instructors, now I will do the analysis for one individual instructor, 'Nxuvxq, Rofdb'. ##

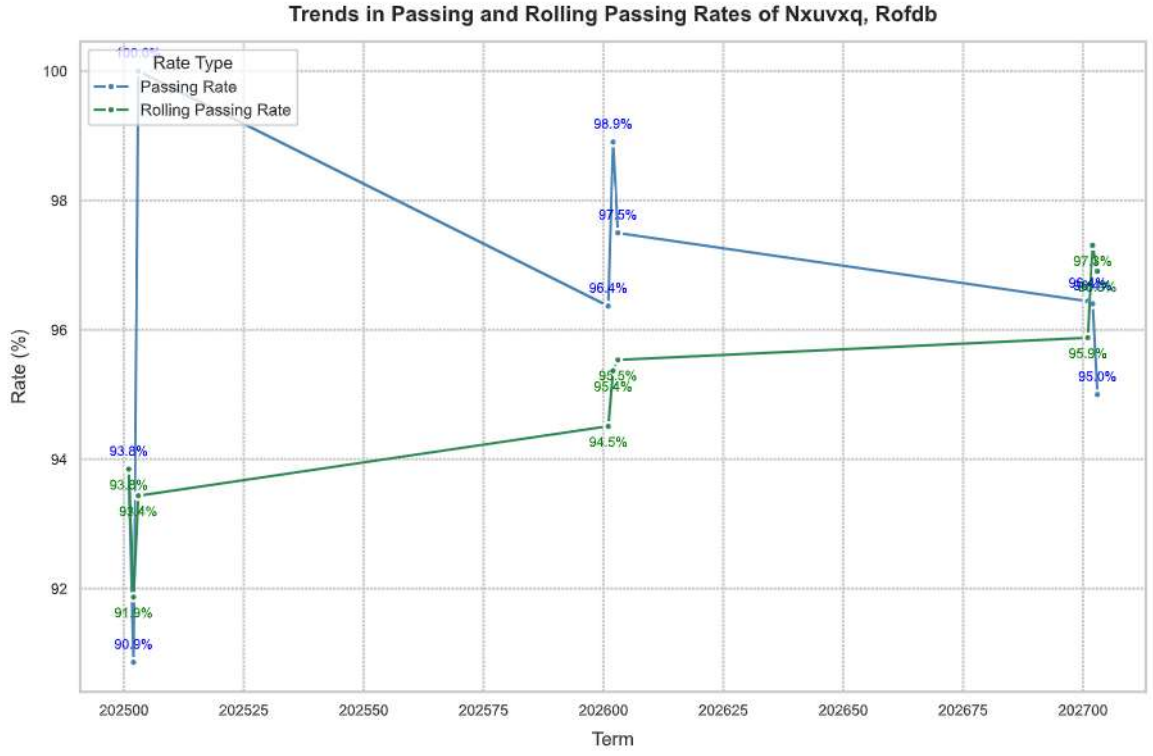
```
individual_instructor = ['Nxuvxq, Rofdb']

# Extract passing rates
individual_data_passing = rate.loc[individual_instructor[0]].reset_index().rename(columns={indivic
'rate'})
individual_data_rolling = rolling_rate.loc[individual_instructor[0]].reset_index().rename(columns=
'rolling_rate'})

sns.set(style="whitegrid")

# subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7), sharey=True,
facecolor='lightgray')

# Line plot passing rate
sns.lineplot(x='Term', y='rate', data=individual_data_passing, marker='o',
color='blue', ax=ax1, label=individual_instructor[0])
ax1.set_title(f'Passing Rates of {individual_instructor[0]}', fontsize=16)
ax1.set_xlabel('Term', fontsize=14)
ax1.set_ylabel('Passing Rate (%)', fontsize=14)
ax1.grid(True, which='both', linestyle='--', linewidth=0.7)
```

```
# Handle missing data (passing rate)
for i in range(len(individual_data_passing) - 1):
    if pd.isnull(individual_data_passing['rate'].iloc[i + 1]):
        ax1.annotate('Missing Data',
                                    xy=(individual_data_passing['Term'].iloc[i],
individual_data_passing['rate'].iloc[i]),
                                xytext=(individual_data_passing['Term'].iloc[i],
individual_data_passing['rate'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))

# Line plot for the rolling passing rate
sns.lineplot(x='Term',     y='rolling_rate',     data=individual_data_rolling,
marker='o', color='green', ax=ax2, label=individual_instructor[0])
ax2.set_title(f'Rolling   Passing   Rates   of   {individual_instructor[0]}',
fontsize=16)
ax2.set_xlabel('Term', fontsize=14)
ax2.grid(True, which='both', linestyle='--', linewidth=0.7)

# point missing data (rolling passing rate)
for i in range(len(individual_data_rolling) - 1):
    if pd.isnull(individual_data_rolling['rolling_rate'].iloc[i + 1]):
        ax2.annotate('Missing Data',
                                    xy=(individual_data_rolling['Term'].iloc[i],
individual_data_rolling['rolling_rate'].iloc[i]),
                                xytext=(individual_data_rolling['Term'].iloc[i],
individual_data_rolling['rolling_rate'].iloc[i] - 5),
                    arrowprops=dict(facecolor='red', shrink=0.05))


ax1.legend(loc='upper right')
ax2.legend(loc='upper left')

plt.tight_layout()
plt.show()
```

1.1

```
individual_instructor = 'Nxuvxq, Rofdb'

# Extract passing rates for individual instructor
individual_data_passing = rate.loc[individual_instructor].reset_index().rename(columns={individual
'rate'})
individual_data_rolling = rolling_rate.loc[individual_instructor].reset_index().rename(columns={in
'rolling_rate'})

# slope chart
slope_data = pd.DataFrame({
    'Term': individual_data_passing['Term'],
    'Passing Rate': individual_data_passing['rate'],
    'Rolling Passing Rate': individual_data_rolling['rolling_rate']
}).melt(id_vars='Term', var_name='Rate Type', value_name='Rate')

# Plot slope chart
sns.set(style="whitegrid")
fig, ax = plt.subplots(figsize=(12, 8))

sns.lineplot(
    data=slope_data,
    x='Term',
    y='Rate',
    hue='Rate Type',
    marker='o',
    ax=ax,
    palette={'Passing Rate': 'steelblue', 'Rolling Passing Rate': 'seagreen'}
)
```

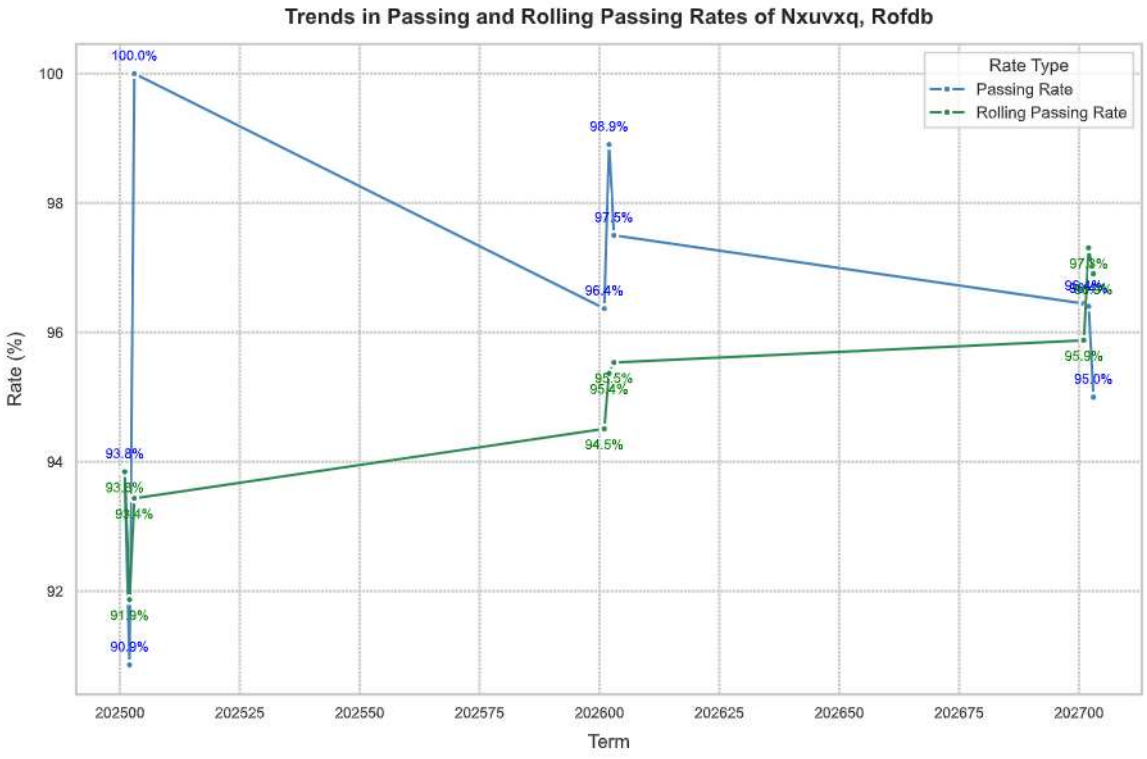
```
# title and labels
ax.set_title(f'Trends in Passing and Rolling Passing Rates of
{individual_instructor}', fontsize=16, fontweight='bold', pad=15)
ax.set_xlabel('Term', fontsize=14, labelpad=10)
ax.set_ylabel('Rate (%)', fontsize=14, labelpad=10)
ax.grid(visible=True, linestyle='--', linewidth=0.6)
ax.legend(title='Rate Type', fontsize=12, title_fontsize=13, loc='upper left')


for term, passing, rolling in zip(slope_data['Term'].unique(),
        individual_data_passing['rate'],
        individual_data_rolling['rolling_rate']):
    if not pd.isnull(passing) and not pd.isnull(rolling):
          ax.annotate(f'{passing:.1f}%', xy=(term, passing), xytext=(0, 10),
textcoords='offset points', ha='center', fontsize=10, color='blue')
         ax.annotate(f'{rolling:.1f}%', xy=(term, rolling), xytext=(0, -15),
textcoords='offset points', ha='center', fontsize=10, color='green')

plt.tight_layout()
plt.show()
```



1.2

```
individual_instructor = 'Nxuvxq, Rofdb'

# Extract passing rates for individual instructor
individual_data_passing = rate.loc[individual_instructor].reset_index().rename(columns={individual
'rate'})
individual_data_rolling = rolling_rate.loc[individual_instructor].reset_index().rename(columns={in
'rolling_rate'})

# slope chart
slope_data = pd.DataFrame({
    'Term': individual_data_passing['Term'],
    'Passing Rate': individual_data_passing['rate'],
    'Rolling Passing Rate': individual_data_rolling['rolling_rate']
}).melt(id_vars='Term', var_name='Rate Type', value_name='Rate')

# Plot slope chart
sns.set(style="whitegrid")
fig, ax = plt.subplots(figsize=(12, 8))

sns.lineplot(
    data=slope_data,
    x='Term',
    y='Rate',
    hue='Rate Type',
    marker='o',
    ax=ax,
    palette={'Passing Rate': 'steelblue', 'Rolling Passing Rate': 'seagreen'}
)

# title and labels
ax.set_title(f'Trends in Passing and Rolling Passing Rates of
{individual_instructor}', fontsize=16, fontweight='bold', pad=15)
ax.set_xlabel('Term', fontsize=14, labelpad=10)
ax.set_ylabel('Rate (%)', fontsize=14, labelpad=10)
ax.grid(visible=True, linestyle='--', linewidth=0.6)
ax.legend(title='Rate Type', fontsize=12, title_fontsize=13, loc='upper right')


for term, passing, rolling in zip(slope_data['Term'].unique(),
        individual_data_passing['rate'],
        individual_data_rolling['rolling_rate']):
    if not pd.isnull(passing) and not pd.isnull(rolling):
            ax.annotate(f'{passing:.1f}%', xy=(term, passing), xytext=(0, 10),
textcoords='offset points', ha='center', fontsize=10, color='blue')
          ax.annotate(f'{rolling:.1f}%', xy=(term, rolling), xytext=(0, -15),
textcoords='offset points', ha='center', fontsize=10, color='green')
```

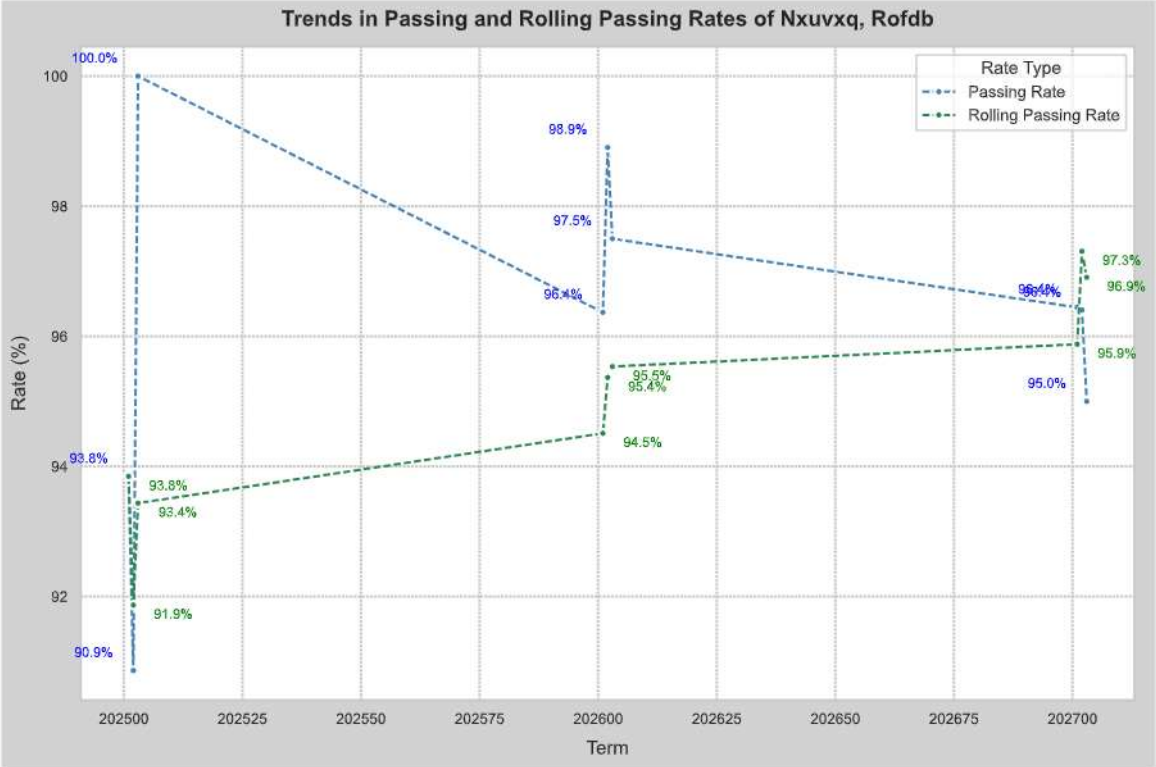
```
plt.tight_layout()
plt.show()
```



1.3

```
individual_instructor = 'Nxuvxq, Rofdb'

# Extract passing rates for individual instructor
individual_data_passing = rate.loc[individual_instructor].reset_index().rename(columns={individual
'rate'})
individual_data_rolling = rolling_rate.loc[individual_instructor].reset_index().rename(columns={in
'rolling_rate'})

# slope chart
slope_data = pd.DataFrame({
    'Term': individual_data_passing['Term'],
    'Passing Rate': individual_data_passing['rate'],
    'Rolling Passing Rate': individual_data_rolling['rolling_rate']
}).melt(id_vars='Term', var_name='Rate Type', value_name='Rate')

# Plot slope chart
sns.set(style="whitegrid")
fig, ax = plt.subplots(figsize=(12, 8), facecolor='lightgray')
```

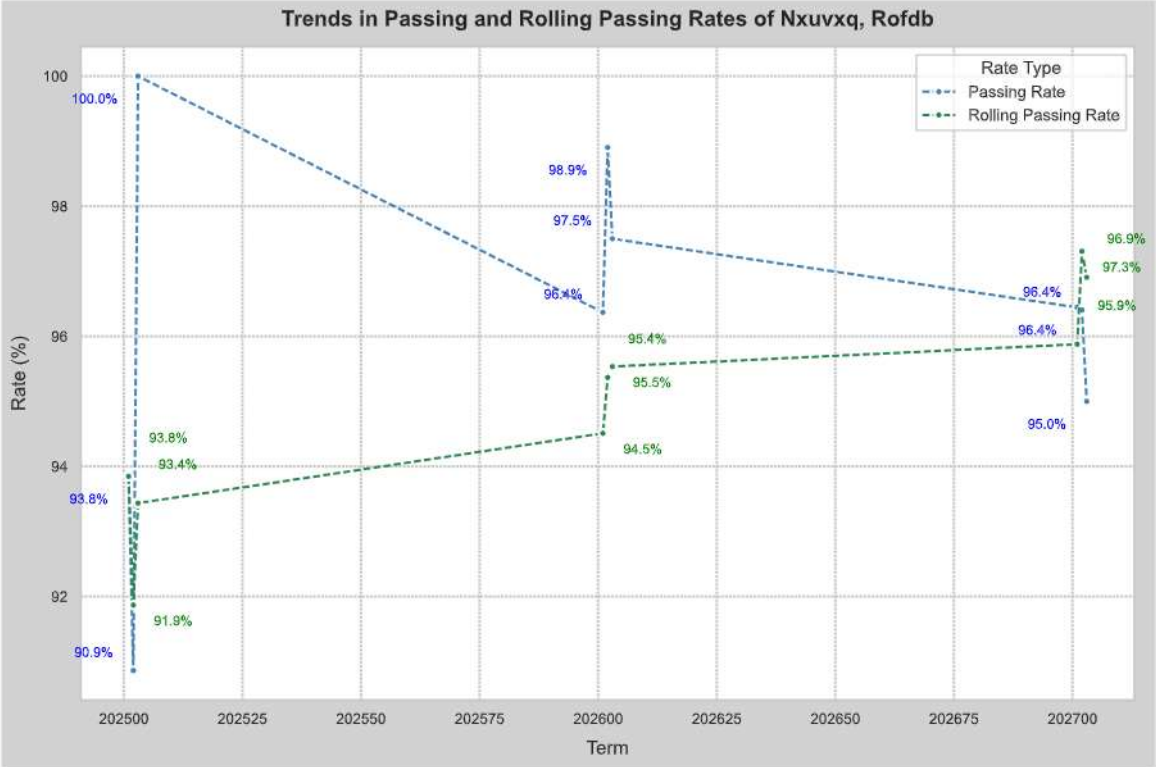
```
sns.lineplot(
    data=slope_data,
    x='Term',
    y='Rate',
    hue='Rate Type',
    marker='o',
    ax=ax,
    palette={'Passing Rate': 'steelblue', 'Rolling Passing Rate': 'seagreen'},
    linestyle='--'
)

#title and labels
ax.set_title(f'Trends in Passing and Rolling Passing Rates of
{individual_instructor}', fontsize=16, fontweight='bold', pad=15)
ax.set_xlabel('Term', fontsize=14, labelpad=10)
ax.set_ylabel('Rate (%)', fontsize=14, labelpad=10)
ax.grid(visible=True, linestyle='--', linewidth=0.6)
ax.legend(title='Rate Type', fontsize=12, title_fontsize=13, loc='upper right')

# adjusted offsets
for term, passing, rolling in zip(slope_data['Term'].unique(),
        individual_data_passing['rate'],
        individual_data_rolling['rolling_rate']):
    if not pd.isnull(passing):
          ax.annotate(f'{passing:.1f}%', xy=(term, passing), xytext=(-15, 10),
textcoords='offset points', ha='right', fontsize=10, color='blue')
    if not pd.isnull(rolling):
          ax.annotate(f'{rolling:.1f}%', xy=(term, rolling), xytext=(15, -10),
textcoords='offset points', ha='left', fontsize=10, color='green')

plt.tight_layout()
plt.show()
```



1.4

```
individual_instructor = 'Nxuvxq, Rofdb'

# Extract the passing rates of instructor
individual_data_passing = rate.loc[individual_instructor].reset_index().rename(columns={individual
'rate'})
individual_data_rolling = rolling_rate.loc[individual_instructor].reset_index().rename(columns={in
'rolling_rate'})

# slope chart
slope_data = pd.DataFrame({
    'Term': individual_data_passing['Term'],
    'Passing Rate': individual_data_passing['rate'],
    'Rolling Passing Rate': individual_data_rolling['rolling_rate']
}).melt(id_vars='Term', var_name='Rate Type', value_name='Rate')

# plot slope chart
sns.set(style="whitegrid")
fig, ax = plt.subplots(figsize=(12, 8), facecolor='lightgray')

sns.lineplot(
    data=slope_data,
    x='Term',
```

```
    y='Rate',
    hue='Rate Type',
    marker='o',
    ax=ax,
    palette={'Passing Rate': 'steelblue', 'Rolling Passing Rate': 'seagreen'},
    linestyle='--'
)

# titles and labels
ax.set_title(f'Trends in Passing and Rolling Passing Rates of
{individual_instructor}', fontsize=16, fontweight='bold', pad=15)
ax.set_xlabel('Term', fontsize=14, labelpad=10)
ax.set_ylabel('Rate (%)', fontsize=14, labelpad=10)
ax.grid(visible=True, linestyle='--', linewidth=0.6)
ax.legend(title='Rate Type', fontsize=12, title_fontsize=13, loc='upper right')

# Annotating clarity and positioning of texts
for i, (term, passing, rolling) in enumerate(zip(slope_data['Term'].unique(),
    individual_data_passing['rate'],
    individual_data_rolling['rolling_rate'])):
    if not pd.isnull(passing):
        y_offset = 10 if i % 2 == 0 else -20
      ax.annotate(f'{passing:.1f}%', xy=(term, passing), xytext=(-15, y_offset),
textcoords='offset points', ha='right', fontsize=10, color='blue')
    if not pd.isnull(rolling):
        y_offset = -15 if i % 2 == 0 else 25
      ax.annotate(f'{rolling:.1f}%', xy=(term, rolling), xytext=(15, y_offset),
textcoords='offset points', ha='left', fontsize=10, color='green')

plt.tight_layout()
plt.show()
```

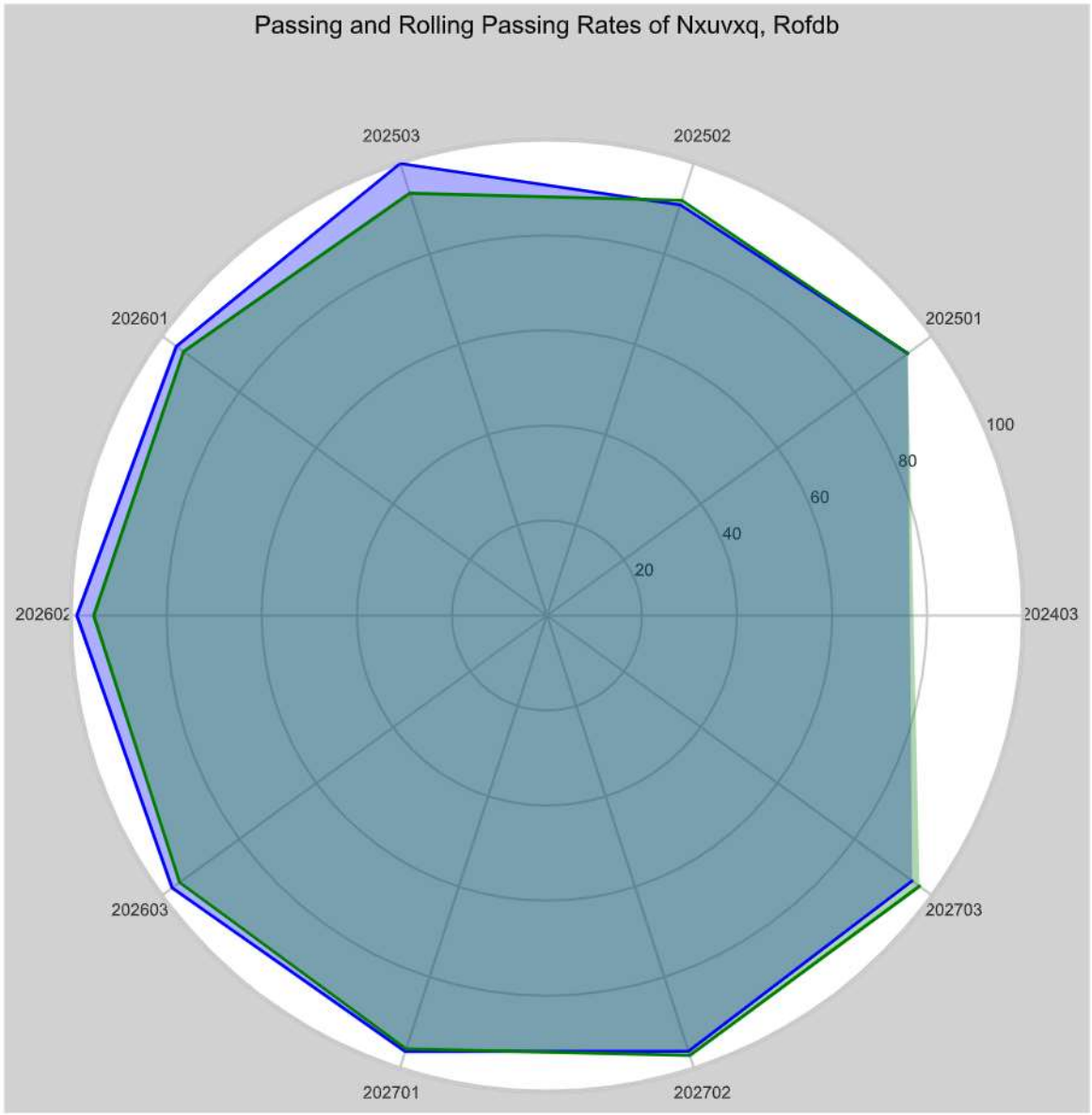


1.5

```
from math import pi

individual_instructor = 'Nxuvxq, Rofdb'

# Extract passing rates individual instructor
individual_data_passing = rate.loc[individual_instructor].reset_index().rename(columns={individual
'rate'})
individual_data_rolling = rolling_rate.loc[individual_instructor].reset_index().rename(columns={in
'rolling_rate'})


radar_data = pd.DataFrame({
    'Term': individual_data_passing['Term'],
    'Passing Rate': individual_data_passing['rate'],
    'Rolling Passing Rate': individual_data_rolling['rolling_rate']
})

# variables
categories = radar_data['Term']
N = len(categories)

# angle of each axis in the plot
```

```
angles = [n / float(N) * 2 * pi for n in range(N)]
angles += angles[:1]

# spider plot
fig,   ax   =   plt.subplots(figsize=(10,   10),   subplot_kw=dict(polar=True),
facecolor='lightgray')

# Plot
def add_to_radar(data, label, color):
    values = data.tolist()
    values += values[:1]
    ax.plot(angles, values, color=color, linewidth=2, linestyle='solid')
    ax.fill(angles, values, color=color, alpha=0.3)
         ax.text(angles[-1],  values[-1],  f'{values[-1]:.1f}%',  color=color,
fontsize=12)

add_to_radar(radar_data['Passing Rate'], 'Passing Rate', 'blue')
add_to_radar(radar_data['Rolling  Passing  Rate'],  'Rolling  Passing  Rate',
'green')


ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories)
ax.set_title(f'Passing and Rolling Passing Rates of {individual_instructor}',
size=16, color='black', y=1.1)
plt.tight_layout()
plt.show()
```

```
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```



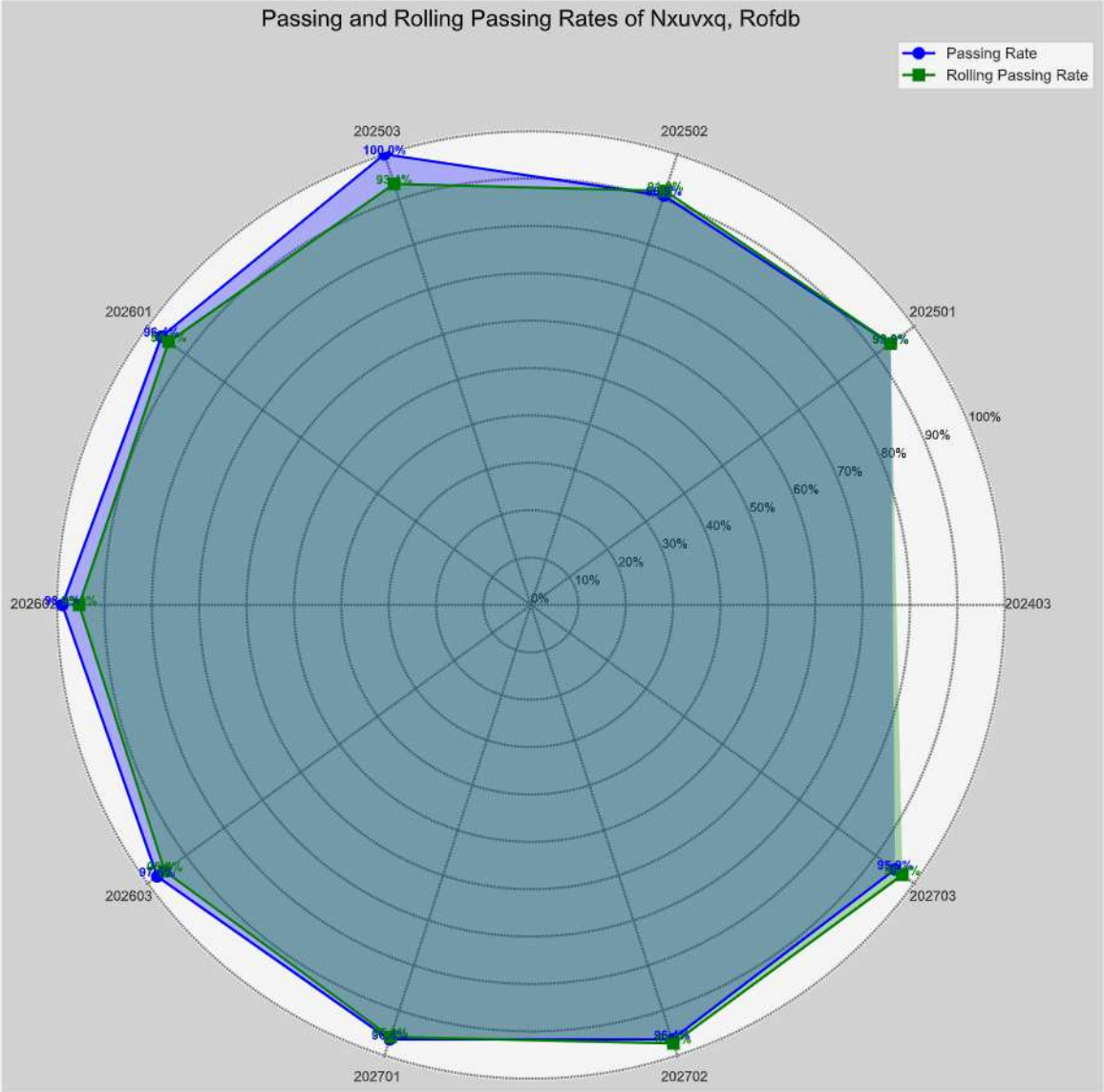
1.6

```
from math import pi

individual_instructor = 'Nxuvxq, Rofdb'

# Extract the passing rates individual instructor
individual_data_passing = rate.loc[individual_instructor].reset_index().rename(columns={individual
'rate'})
individual_data_rolling = rolling_rate.loc[individual_instructor].reset_index().rename(columns={in
'rolling_rate'})
```

```
# radar chart
radar_data = pd.DataFrame({
    'Term': individual_data_passing['Term'],
    'Passing Rate': individual_data_passing['rate'],
    'Rolling Passing Rate': individual_data_rolling['rolling_rate']
})

# variables
categories = radar_data['Term']
N = len(categories)

# angle of each axis in the plot
angles = [n / float(N) * 2 * pi for n in range(N)]
angles += angles[:1]

# spider plot
fig, ax = plt.subplots(figsize=(12, 12), subplot_kw=dict(polar=True),
facecolor='lightgray')

# Plot for data
def add_to_radar(data, label, color, marker):
    values = data.tolist()
    values += values[:1]
       ax.plot(angles, values, color=color, linewidth=2, linestyle='solid',
label=label, marker=marker, markersize=8)
    ax.fill(angles, values, color=color, alpha=0.3)
    for i in range(len(values) - 1):
                        ax.text(angles[i], values[i], f'{values[i]:.1f}%',
horizontalalignment='center', size=10, color=color, weight='semibold')

add_to_radar(radar_data['Passing Rate'], 'Passing Rate', 'blue', 'o')
add_to_radar(radar_data['Rolling Passing Rate'], 'Rolling Passing Rate',
'green', 's')


ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories)
ax.set_title(f'Passing and Rolling Passing Rates of {individual_instructor}',
size=18, color='black', y=1.1)

# gridlines
ax.yaxis.set_tick_params(labelsize=10)
ax.yaxis.set_ticks(np.arange(0, 101, 10))
ax.yaxis.set_ticklabels([f'{int(x)}%' for x in np.arange(0, 101, 10)])

# ax.spine and background
ax.grid(color='gray', linestyle='--', linewidth=0.5)
ax.spines['polar'].set_visible(False)
```

```
ax.set_facecolor('whitesmoke')

#legend
plt.legend(loc='upper    right',    bbox_to_anchor=(1.1,    1.1),    fontsize=12,
frameon=True)

plt.tight_layout()
plt.show()
```

```
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```



1.7

```
from math import pi


individual_instructor = 'Nxuvxq, Rofdb'

# Extract the passing rates individual instructor
individual_data_passing = rate.loc[individual_instructor].reset_index().rename(columns={individual
'rate'})
individual_data_rolling = rolling_rate.loc[individual_instructor].reset_index().rename(columns={in
'rolling_rate'})
```

```
# radar chart
radar_data = pd.DataFrame({
    'Term': individual_data_passing['Term'],
    'Passing Rate': individual_data_passing['rate'],
    'Rolling Passing Rate': individual_data_rolling['rolling_rate']
})

# variables
categories = radar_data['Term']
N = len(categories)

# angle of each axis in the plot
angles = [n / float(N) * 2 * pi for n in range(N)]
angles += angles[:1]

# spider plot
fig, ax = plt.subplots(figsize=(12, 12), subplot_kw=dict(polar=True),
facecolor='lightgray')

# Plot for data
def add_to_radar(data, label, color, marker):
    values = data.tolist()
    values += values[:1]
       ax.plot(angles, values, color=color, linewidth=2, linestyle='solid',
label=label, marker=marker, markersize=8)
    ax.fill(angles, values, color=color, alpha=0.3)

add_to_radar(radar_data['Passing Rate'], 'Passing Rate', 'blue', 'o')
add_to_radar(radar_data['Rolling Passing Rate'], 'Rolling Passing Rate',
'green', 's')

#labels
ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories)
ax.set_title(f'Passing and Rolling Passing Rates of {individual_instructor}',
size=18, color='black', y=1.1)

#gridlines
ax.yaxis.set_tick_params(labelsize=10)
ax.yaxis.set_ticks(np.arange(0, 101, 10))
ax.yaxis.set_ticklabels([f'{int(x)}%' for x in np.arange(0, 101, 10)])

#ax.spine and background
ax.grid(color='gray', linestyle='--', linewidth=0.5)
ax.spines['polar'].set_visible(False)
ax.set_facecolor('whitesmoke')

#legend
```

```
plt.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1), fontsize=12,
frameon=True)

plt.tight_layout()
plt.show()
```