Table of Contents

[Introduction 1](#_Toc92115742)

[Use Transpose (T) When Presenting DataFrames in Jupyter Notebooks 1](#_Toc92115743)

[Correlation Heatmaps 3](#_Toc92115744)

[Plotly 4](#_Toc92115745)

[Line Plots 4](#_Toc92115746)

[Stacked Bar Charts 6](#_Toc92115747)

[Histograms and Multiple Plots 8](#_Toc92115748)

[3D Plots 10](#_Toc92115749)

[DataFrame Graphics in Notebooks 12](#_Toc92115750)

[Colouring Cell Backgrounds 12](#_Toc92115751)

[Inline Bar Plots within Dataframe Cells 13](#_Toc92115752)

[Colouring and Bolding Cell Fonts 14](#_Toc92115753)

# Introduction

The browser-based nature of Python notebooks presents some challenges. However, the browser-based display also allows for some advanced graphics which are not possible in console applications. This document will provide some tips to help optimize your visual displays with browser notebooks.

## Use Transpose (T) When Presenting DataFrames in Jupyter Notebooks

When presenting data frames in notebooks an undesirable wrap can occur as shown in Figure 1.

Figure : Difficult-to-read Word Wrap



To fix this unsightly wrap, you can:

1. use transpose (T) to swap the columns and rows to fit the data better.

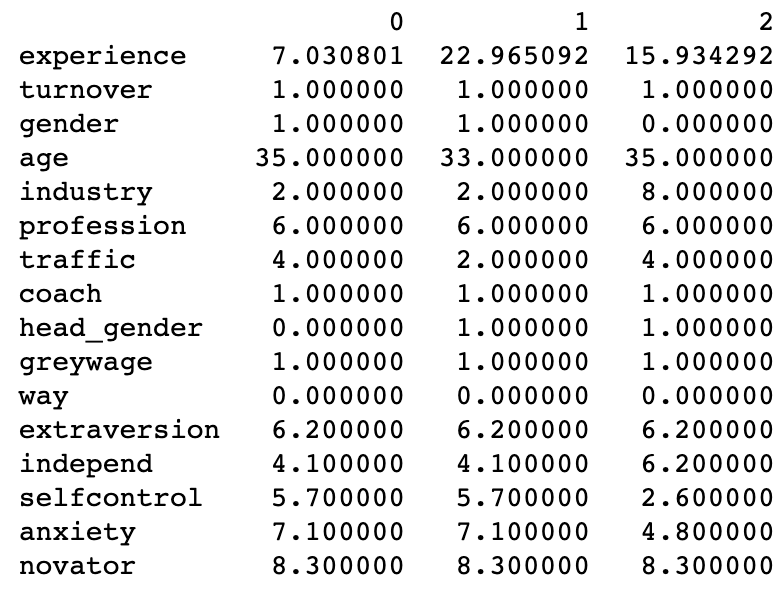
OR

1. Enable a horizontal scroll

Example : Presenting Dataframes in Notebooks with a Transpose

This example shows that the dataframe contents can be transposed to enable a more reader friendly display in Jupyter Notebooks. Sample transposed output is contained in Figure 2.

Figure : Comparing Un-Transposed Versus Transposed Data



Here is the code which displays the transposed heatmap.

|  |
| --- |
| import plotly.express as px  import pandas as pd  # load the data  CSV\_FILE = '/users/pm/desktop/daydocs/data/employee\_turnover.csv'  df = pd.read\_csv(CSV\_FILE)  pd.set\_option('display.max\_columns', None)  pd.set\_option('display.width', 1000)  # Show transposed data  print(df.head(3).T) |

Example : Presenting Dataframes in Notebooks with Horizontal Scroll

This example shows how to enable a horizontal scroll in Jupyter Notebook.

|  |
| --- |
| from IPython.core.display import display, HTML  def view(\_df):  with pd.option\_context('display.max\_rows', None, 'display.max\_columns', None, 'display.width', None):  display(HTML(\_df.to\_html()))  import pandas as pd  df = pd.DataFrame(data={'someText':["This is Trevor's cool horizontal scroller and it will let you scroll for miles and miles and miles and miles and miles and then you can see everything with perfect clarity."]})  view(df) |

Exercise (1 mark)

Show a screenshot of your notebook after running the code to enable horizontal scrolling. However, change the name from Trevor to your name.

|  |
| --- |
|  |

## Correlation Heatmaps

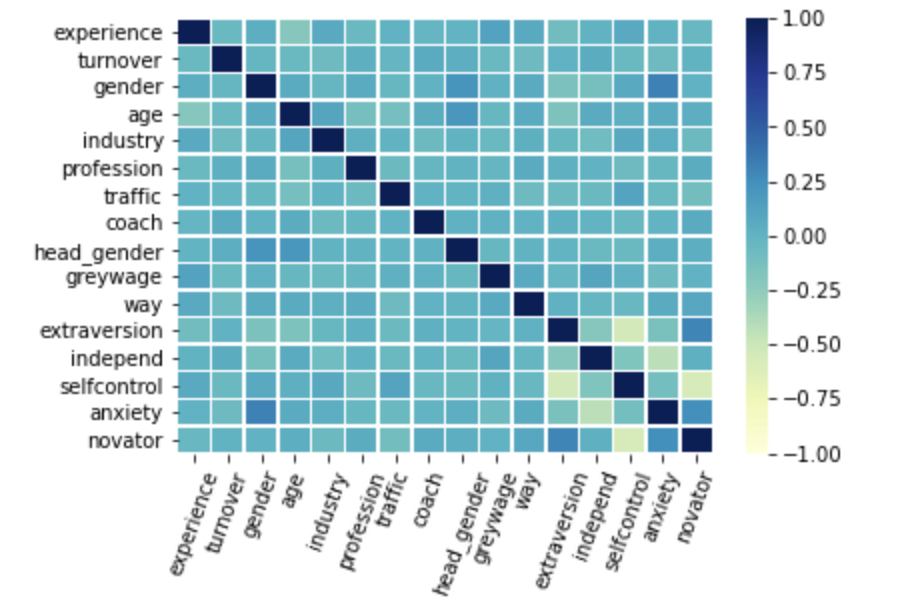
You have already used heatmaps but here are some tips:

* Remember to show a legend which displays the proper range of values. For this case, **vmin=-1** and **vmax=1**.
* Select a professional looking palette with the cmap property. Figure 3 highlights negative values in light yellow and positive values in darkening shades of blue. See <https://likegeeks.com/seaborn-heatmap-tutorial/> for sequential palettes.
* In any graph, sometimes visually it is easier to view x axis labels if they are rotated.

**(plt.xticks(rotation=70))**

Example : Correlation Heatmaps

Figure : Heatmap with a Divergent Palette



|  |
| --- |
| import matplotlib.pyplot as plt  import seaborn as sns  # get the correlation matrix  corr = df.corr()  ax = sns.heatmap(corr, linewidth=0.5, vmin=-1, vmax=1, cmap="YlGnBu")  plt.xticks(rotation=70)  plt.show() |

Example : Single Column Heatmap

Sometimes presenting the full matrix in a heatmap is distracting. For some cases it may make more sense to display the target variable correlations only. Setting the ***figsize*** property before assigning seaborn heatmap properties restricts the width and height proportion of the graphic output. To build this solution, add the following code to Example 3.

|  |  |
| --- | --- |
| # get the correlation matrix  corr = df.corr()  plt.figure(figsize = (0.5,4))  ax = sns.heatmap(corr[['turnover']],  linewidth=0.5, vmin=-1,  vmax=1, cmap="YlGnBu")  plt.show() |  |

Exercise (1 mark)

Modify Example 4 to show the correlations of *selfcontrol* with all other attributes in the employee\_turnover set. Change the cmap property to another colour palette which has a divergent set of colours that clearly distinguishes positive and negative values. Show a screenshot of the output here:

|  |
| --- |
| Chart  Description automatically generated |

## Plotly

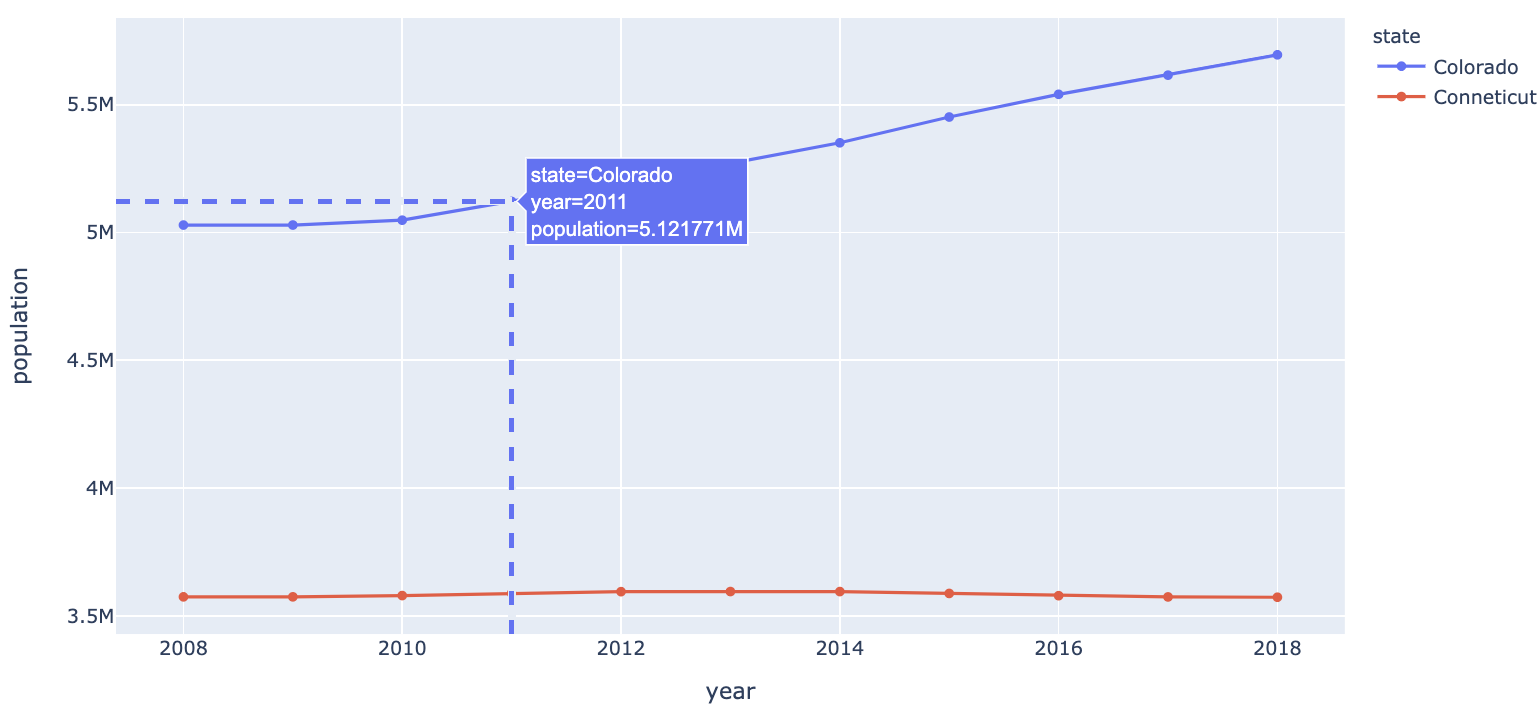
Plotly is a company that develops open-source data analytics and visualization tools for machine learning, data science, and engineering. The Plotly graphing library, which makes high quality, visually pleasing, interactive graphs that can be **viewed in the browser**. This library is excellent if you are using Python Notebooks.

### Line Plots

Plotly automates the inclusion of tool tips in web-based graphs. These tool tips appear when hovering over the graph with a cursor.

Example : Line Plot in Plotly

This graph shows populations for the states of Colorado and Conneticut.

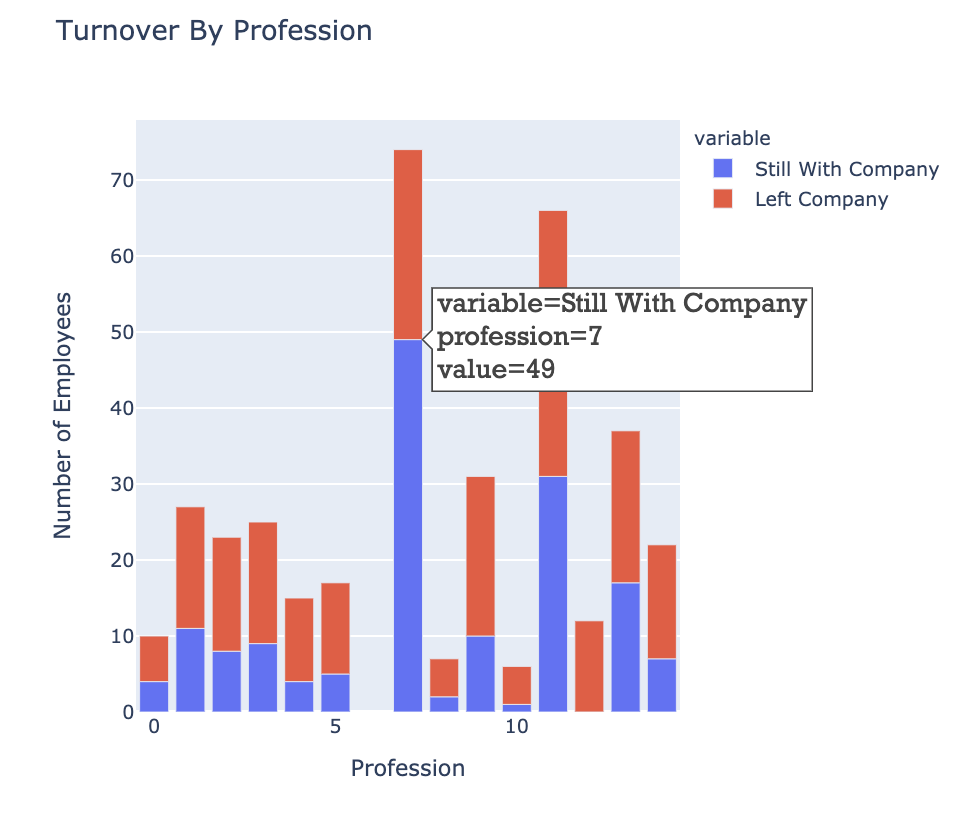


|  |
| --- |
| import plotly.express as px  import pandas as pd  years = [2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018]  colorado = [5029196,5029316,5048281,5121771,5193721,5270482,5351218,5452107,  5540921,5615902,5695564]  conneticut = [3574097,3574147,3579125,3588023,3594395,3594915,3594783,3587509,  3578674,3573880,3572665]  coloradoDf = pd.DataFrame(data={  "year":years, "population":colorado  })  coloradoDf['state']="Colorado"  connDf = pd.DataFrame(data={"year":years, "population":conneticut})  connDf['state'] = "Conneticut"  df = coloradoDf.append(connDf)  print(df)  fig = px.line(df, x="year", y="population", color="state" )  fig.update\_traces(mode="markers+lines")  fig.update\_xaxes(showspikes=True)  fig.update\_yaxes(showspikes=True)  fig.show() |

### Stacked Bar Charts

Stacked bar charts are a great way to visualize data while separating the data for two or more groups. The bar chart in Figure 4 compares counts of employees who stay versus counts of persons who leave for different professions within a company. Plotly stacked bar charts treat each column as a separate stack in a bar. Hovering over any sections of the bar chart triggers an informative tool tip.

Figure : Using Plotly to Show Employee Turnover by Profession



Example : Preparing Data for Grouped Bar Chart

This example shows how to prepare the dataframe so it can be used to draw the Plotly stacked bar chart shown in Figure 4. The resulting data frame is generated so it can be used to draw the stacked bar chart (see Table 1).

Table : Dataframe to be used in Stacked Bar Chart

|  |
| --- |
| Left Company Still With Company  profession  0 6 4.0  1 16 11.0  2 15 8.0  3 16 9.0  4 11 4.0  5 12 5.0  7 25 49.0  8 5 2.0  9 21 10.0  10 5 1.0  11 35 31.0  12 12 0.0  13 20 17.0  14 15 7.0 |

Here is the code that prepares the data frame.

|  |
| --- |
| import pandas as pd  # load the data  CSV\_FILE = '/users/pm/desktop/daydocs/data/employee\_turnover.csv'  df = pd.read\_csv(CSV\_FILE)  pd.set\_option('display.max\_columns', None)  pd.set\_option('display.width', 1000)  # view the data  print(df)  # get count of employees who have left, grouped by profession  left = df[df['turnover'] == 1].groupby('profession')['profession'].count() \  .reset\_index(name="Left Company")  # get count of employees who have stayed, grouped by profession  stayed = df[df['turnover'] == 0].groupby('profession')['profession'].count() \  .reset\_index(name="Still With Company")  stayed = stayed.set\_index('profession')  print("\nStill with company")  print(stayed)  print("\nLeft company")  left = left.set\_index('profession')  print(left)  print("\nMerged")  result = pd.merge(left, stayed, on='profession', how='outer')  # Sort by index and replace null values.  result = result.sort\_index()  result = result.fillna(0)  # Dropped profession 6 to avoid distorting scale for other professions.  result = result[result.index!=6]  print(result) |

Example : Showing Stacked Bar Chart with Plotly

This code can be added to Example 6 to draw the bar chart.

|  |
| --- |
| import plotly.express as px  # create the stacked bar chart  fig = px.bar(result,  x=result.index,  # specify columns to use for the stacked bars  y=[ "Still With Company", "Left Company"],  title="Turnover By Profession",  )  # adjust layout properties of the figure  fig.update\_layout(  yaxis=dict(title="Number of Employees"),  xaxis=dict(title="Profession"),  width=600,  height=550,  hoverlabel=dict(  bgcolor="white",  font\_size=16,  font\_family="Rockwell"  )  )  # show the stacked bar chart  fig.show() |

Exercise (4 marks)

Generate a plot where each bar compares counts of employees who left versus counts of employees who stayed. Show a stacked bar for each **gender** in the employee\_turnover set. Colour the background of the tooltip yellow. Show your code here:

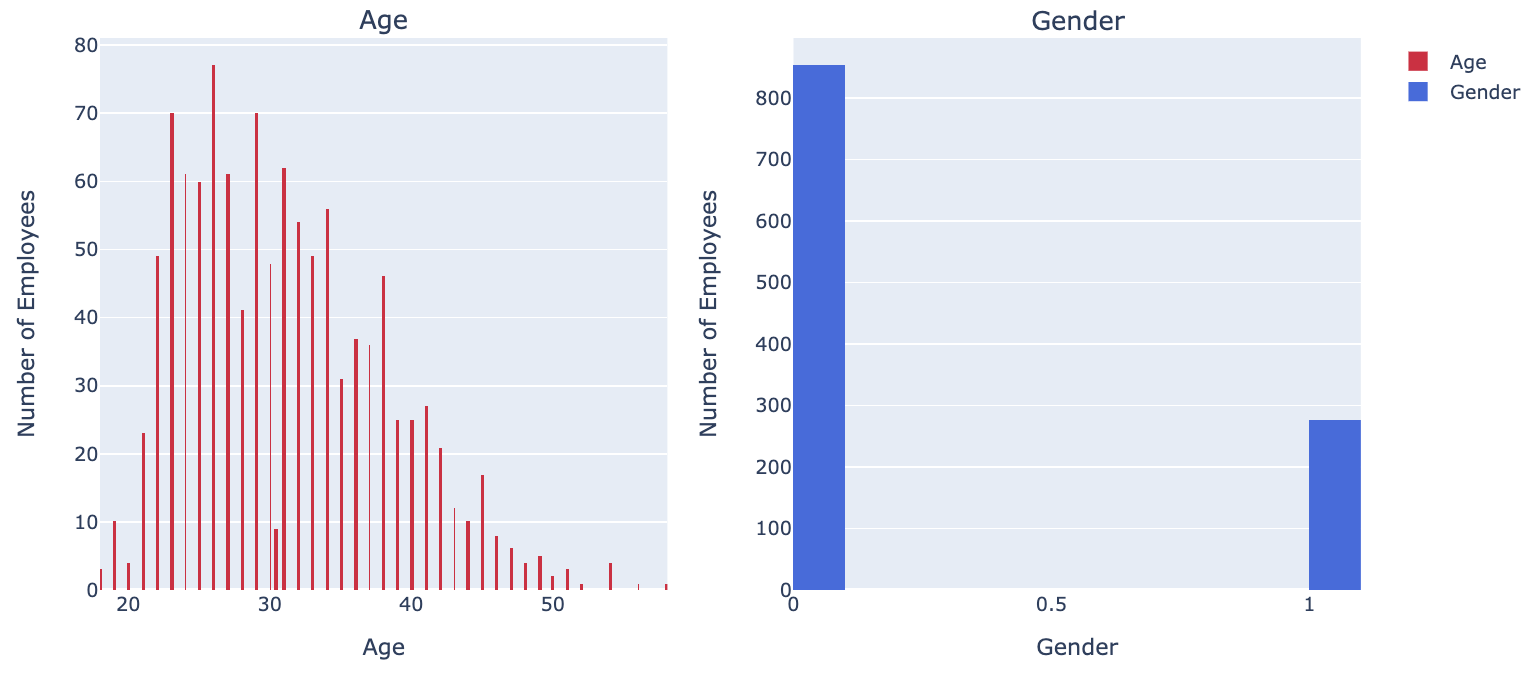
|  |
| --- |
|  |

Show your plot here:

|  |
| --- |
| import pandas as pd  # load the data CSV\_FILE = '/Users/hyerimshin/PycharmProjects/BigData/datasets//employee\_turnover.csv' df = pd.read\_csv(CSV\_FILE) pd.set\_option('display.max\_columns', None) pd.set\_option('display.width', 1000)  # view the data print(df)  # get count of employees who have left, grouped by profession left = df[df['turnover'] == 1].groupby('gender')['gender'].count() \  .reset\_index(name="Left Company")  # get count of employees who have stayed, grouped by profession stayed = df[df['turnover'] == 0].groupby('gender')['gender'].count() \  .reset\_index(name="Still With Company") stayed = stayed.set\_index('gender') print("\nStill with company") print(stayed)  print("\nLeft company") left = left.set\_index('gender') print(left)  print("\nMerged") result = pd.merge(left, stayed, on='gender', how='outer')  # Sort by index and replace null values. result = result.sort\_index() result = result.fillna(0)  # Dropped profession 6 to avoid distorting scale for other professions. result = result[result.index!=6] print(result)  import plotly.express as px  # create the stacked bar chart fig = px.bar(result,  x=result.index,  # specify columns to use for the stacked bars  y=[ "Still With Company", "Left Company"],  title="Turnover By Gender",  )  # adjust layout properties of the figure fig.update\_layout(  yaxis=dict(title="Number of Employees"),  xaxis=dict(title="Gender"),  width=600,  height=550,  hoverlabel=dict(  bgcolor="yellow",  font\_size=16,  font\_family="Rockwell"  ) ) # show the stacked bar chart fig.show() |

### Histograms and Multiple Plots

Figures with multiple plots can be achieved using the plotly.express and plotly.graph\_objects modules. Graph objects are Python classes that represent parts of a figure.



Example : Drawing Multiple Plots Together

This example shows how to draw two graphics plots in one row. The number of **rows** and **columns** are set in the **make\_subplots()** function. **suplot\_titles** are set here as well.

fig = make\_subplots(rows=1, cols=2,

# indicate the position of the first plot

start\_cell="bottom-left",

subplot\_titles=["Age", "Gender"])

When drawing each subplot, the row and column number are identified to set the placement of the plot within the series.

# create histograms and add them to the figure

fig.add\_trace(go.Histogram(

x=df['age'],

name='Age',

marker=dict(color='crimson'),

xbins=dict(start=0, end=100, size=0.1)), # define bins

row=1, col=1)

Here is the full program.

|  |
| --- |
| import plotly.graph\_objects as go  from plotly.subplots import make\_subplots  import pandas as pd  # load the data  PATH = '/users/pm/desktop/daydocs/data/'  CSV\_FILE = 'employee\_turnover.csv'  df = pd.read\_csv(PATH + CSV\_FILE)  pd.set\_option('display.max\_columns', None)  pd.set\_option('display.width', 1000)  # create 1 by 2 figure of 2 subplots  fig = make\_subplots(rows=1, cols=2,  # indicate the position of the first plot  start\_cell="bottom-left",  subplot\_titles=["Age", "Gender"])  # create histograms and add them to the figure  fig.add\_trace(go.Histogram(  x=df['age'],  name='Age',  marker=dict(color='crimson'),  xbins=dict(start=0, end=100, size=0.1)), # define bins  row=1, col=1)  fig.add\_trace(go.Histogram(  x=df['gender'],  name='Gender',  marker=dict(color='royalblue'),  xbins=dict(start=0, end=2, size=0.1)),  row=1, col=2)  # Update x-axis properties  fig.update\_xaxes(title\_text="Age", row=1, col=1)  fig.update\_xaxes(title\_text="Gender", row=1, col=2)  # Update y-axis properties  fig.update\_yaxes(title\_text="Number of Employees", row=1, col=1)  fig.update\_yaxes(title\_text="Number of Employees", row=1, col=2)  # show the histograms  fig.show() |

Exercise (4 **marks)**

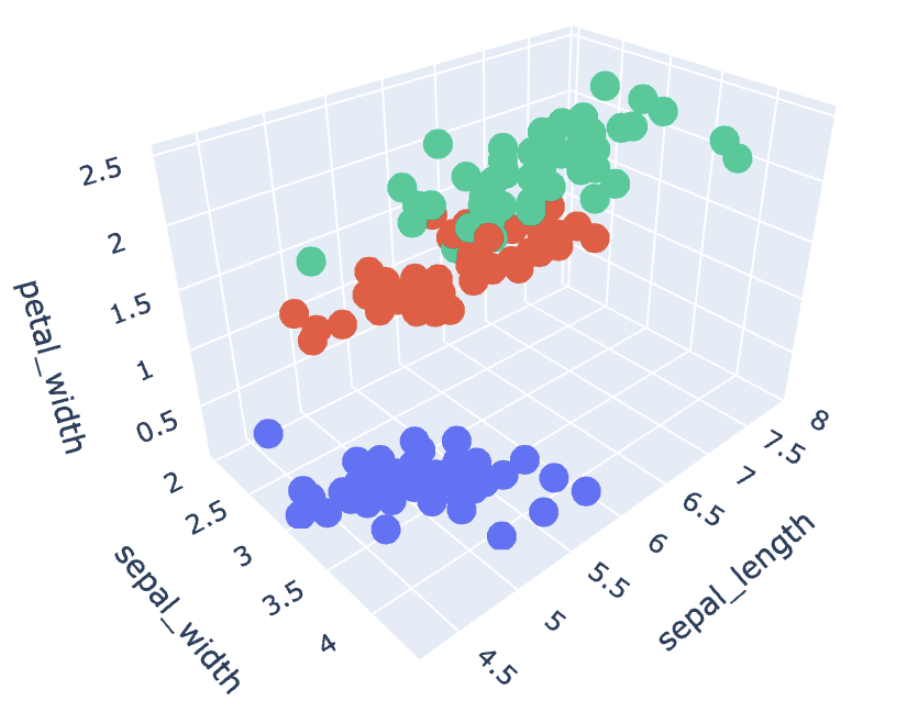
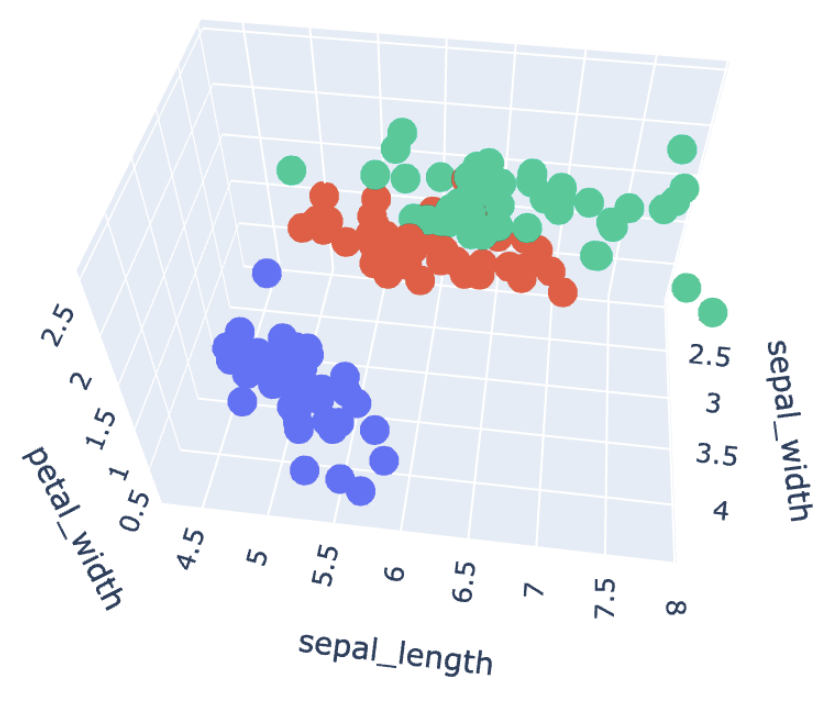
Using **babysamp-98.txt**, show histograms for the MomAge, MomEducation, MomMarried values. Show all histograms in one row with the help of Plotly’s make\_subplots() function. Show a screenshot of your plotly output here:

|  |
| --- |
| Graphical user interface, application  Description automatically generated |

## 3D Plots

Example : 3D Plots with Plotly

This example shows a novel but interesting way to plot three dimensions together in Jupyter Notebooks. The plot that is generated is rotateable.



|  |
| --- |
| import plotly.express as px  import plotly.graph\_objects as go  import dash  import dash\_core\_components as dcc  from dash import html  from dash.dependencies import Input, Output  df = px.data.iris()  fig\_3d = px.scatter\_3d(  df, x='sepal\_width', y='sepal\_length', z='petal\_width',  color='species')  fig\_3d.show() |

Exercise (4 marks)

Starting with the following code:

|  |
| --- |
| import pandas as pd  import numpy as np  # Setup data.  candidates = {'gmat': [780,750,690,710,680,730,690,720,  740,690,610,690,710,680,770,610,580,650,540,590,620,  600,550,550,570,670,660,580,650,660,640,620,660,660,  680,650,670,580,590,690],  'gpa': [4,3.9,3.3,3.7,3.9,3.7,2.3,3.3,  3.3,1.7,2.7,3.7,3.7,3.3,3.3,3,2.7,3.7,2.7,2.3,  3.3,2,2.3,2.7,3,3.3,3.7,2.3,3.7,3.3,3,2.7,4,  3.3,3.3,2.3,2.7,3.3,1.7,3.7],  'work\_experience': [3,4,3,5,4,6,1,4,5,  1,3,5,6,4,3,1,4,6,2,3,2,1,4,1,2,6,4,2,6,5,1,2,4,6,  5,1,2,1,4,5],  'admitted': [1,1,1,1,1,1,0,1,1,0,0,1,  1,1,1,0,0,1,0,0,0,0,0,0,0,1,1,0,1,1,0,0,1,1,1,0,0,  0,0,1]}  df = pd.DataFrame(candidates,columns= ['gmat', 'gpa',  'work\_experience','admitted'])  print(df) |

Create a 3D plot which displays gpa, work\_experience and gmat dimensions on each axis Color the samples based on their admission status. Show a screenshot of the plot here:

|  |
| --- |
| Chart, scatter chart  Description automatically generated |

## DataFrame Graphics in Notebooks

Python Notebooks enable much richer and more flexible formatting options for displaying Dataframes compared to other console applications.

### Colouring Cell Backgrounds

To colour a cell background, you can choose from many palettes:

<https://matplotlib.org/stable/gallery/color/colormap_reference.html>

Example : Colouring Cell Backgrounds (code changed: check learning hub)

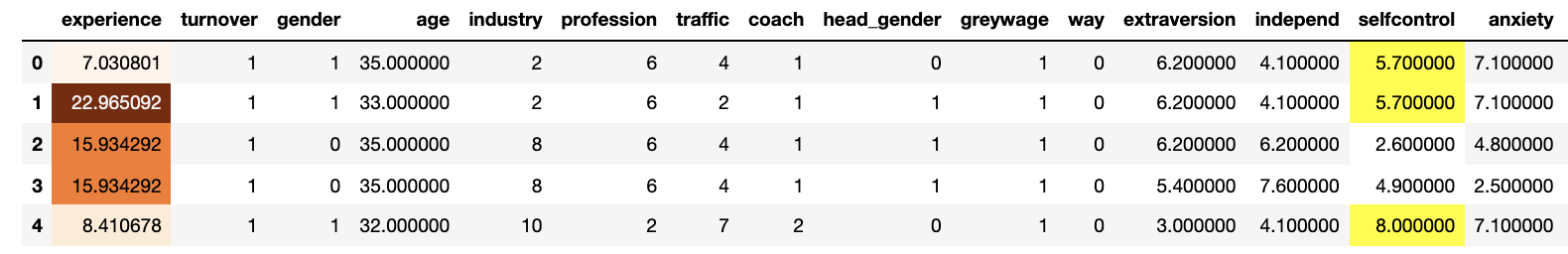
This example shows how to implement a background gradient for numeric values in the experience *column*.

.background\_gradient(cmap='Oranges', subset=['experience'])

Valid **cmap** properties can be referenced at <https://matplotlib.org/stable/gallery/color/colormap_reference.html>

The *selfcontrol* column shows cell backgrounds that are set conditionally.

Figure : Cell background color



|  |
| --- |
| import plotly.graph\_objects as go  from plotly.subplots import make\_subplots  import pandas as pd  # load the data  CSV\_FILE = '/users/pm/desktop/daydocs/data/employee\_turnover.csv'  df = pd.read\_csv(CSV\_FILE)  pd.set\_option('display.max\_columns', None)  pd.set\_option('display.width', 1000)  def highlight\_greaterthan\_5(column):  if column.selfcontrol > 5.0:  return ['background-color: yellow']  else:  return ['background-color: white']  df.head(5).style\  .background\_gradient(cmap='Oranges', subset=['experience'])\  .apply(highlight\_greaterthan\_1, axis=1, subset=["selfcontrol"]) |

Exercise (5 marks)

Modify Example 10 so that:

Anxiety less than 3 is coloured green. Anxiety between 3 and 5 is coloured yellow. Anxiety above 7 is coloured red.

Also, apply a background gradient to the numbers in the extraversion and independent columns. Use a different cmap property than Oranges. Refer to <https://matplotlib.org/stable/gallery/color/colormap_reference.html>

Show a screenshot of your output here:

|  |
| --- |
|  |

### Inline Bar Plots within Dataframe Cells

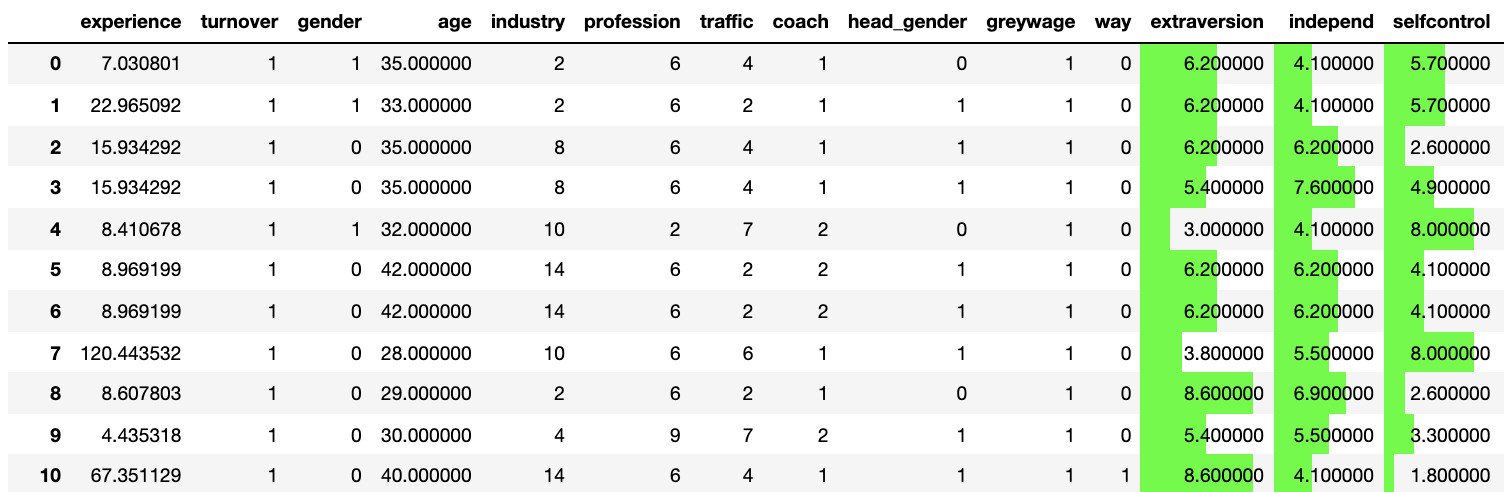
(notebook is better for the presentation, concise and easy to look at)

Example : Bar Plots

Adding inline bar plots into numeric cells within Jupyter Notebooks is easy:

df.style.bar(subset=["extraversion","selfcontrol","independ"],

color='lightgreen')



Here is the code:

|  |
| --- |
| from plotly.subplots import make\_subplots  import pandas as pd  # load the data  CSV\_FILE = '/users/pm/desktop/daydocs/data/employee\_turnover.csv'  df = pd.read\_csv(CSV\_FILE)  pd.set\_option('display.max\_columns', None)  pd.set\_option('display.width', 1000)  df.style.bar(subset=["extraversion","selfcontrol","independ"], color='#00ff00') |

Exercise (1 mark)

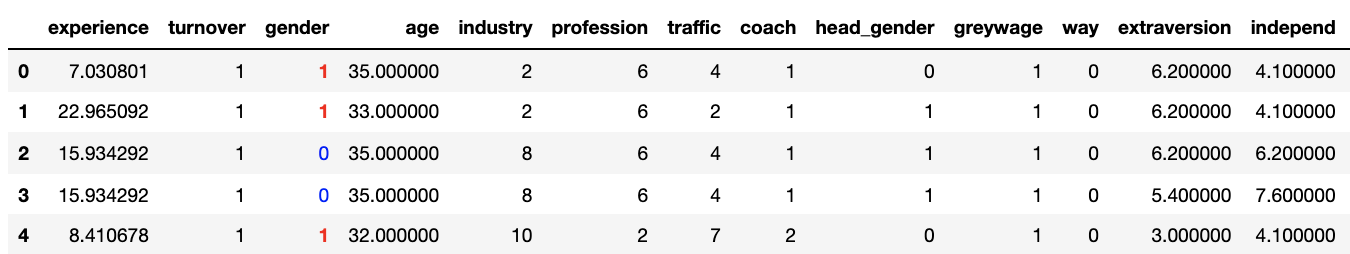
Modify Example 11 so it also includes shaded bar plots for the experience column. Show a screenshot of your graphical output here after the changes.

|  |
| --- |
| Table  Description automatically generated |

### Colouring and Bolding Cell Fonts

Example : Colouring and Bolding Fonts

This example shows how to conditionally colour and bold cell values.



Here is the code.

|  |
| --- |
| import pandas as pd  # load the data  CSV\_FILE = '/users/pm/desktop/daydocs/data/employee\_turnover.csv'  df = pd.read\_csv(CSV\_FILE)  pd.set\_option('display.max\_columns', None)  pd.set\_option('display.width', 1000)  import seaborn as sns  cm = sns.light\_palette("green", as\_cmap=True)  def bold\_red(val):  bold = ''  if val >0:  bold = 'bold'  return 'font-weight: %s' % bold  def color\_negative\_red(value):  color = 'red'    if value == 0:  color = 'blue'  return 'color: %s' % color  df.head(5).style.applymap(color\_negative\_red, subset=['gender'])\  .applymap(bold\_red, subset=['gender']) |

Exercise (4 marks)

Conditionally colour and bold the extraversion and indep columns. Colour and bold numbers under 5 red. Color numbers between 5 and 7 yellow. Color numbers over 7 green. Show a screenshot of the output here:

|  |
| --- |
|  |