In [55]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
import seaborn as sns
import plotly
import plotly.graph_objects as go
```

In [56]:
```
PhD_v3 = pd.read_csv("C:\\Users\\SCD UM\\Downloads\\Visualisation\\PhD_v3
```

In [57]:
```
PhD_v3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 448047 entries, 0 to 448046
Data columns (total 23 columns):
 #   Column                                  Non-Null Count   Dtype  
---  ------                                  --------------   -----  
 0   Unnamed: 0                              448047 non-null  int64  
 1   Auteur                                  448047 non-null  object 
 2   Identifiant auteur                      317700 non-null  object 
 3   Titre                                   448040 non-null  object 
 4   Directeur de these                      448034 non-null  object 
 5   Directeur de these (nom prenom)         448034 non-null  object 
 6   Identifiant directeur                   448047 non-null  object 
 7   Etablissement de soutenance             448046 non-null  object 
 8   Identifiant etablissement               430965 non-null  object 
 9   Discipline                              448047 non-null  object 
 10  Statut                                  448047 non-null  object 
 11  Date de premiere inscription en doctorat  64331 non-null  object 
 12  Date de soutenance                      390961 non-null  object 
 13  Year                                    390961 non-null  float64
 14  Langue de la these                      448047 non-null  object 
 15  Identifiant de la these                 448047 non-null  object 
 16  Accessible en ligne                     448047 non-null  object 
 17  Publication dans theses.fr              448047 non-null  object 
 18  Mise a jour dans theses.fr              447870 non-null  object 
 19  Discipline_prÃ©di                       448047 non-null  object 
 20  Genre                                   448047 non-null  object 
 21  etablissement_rec                       444973 non-null  object 
 22  Langue_rec                              383927 non-null  object 
dtypes: float64(1), int64(1), object(21)
memory usage: 78.6+ MB
```

In [58]:
```
PhD_v3=  PhD_v3.rename(columns={"Discipline_prÃ©di":"Discipline_pre"})
```

In [59]:
```
PhD_v3 = PhD_v3.drop('Unnamed: 0', axis=1)
```

In [60]:
```
PhD_v3.isnull().sum().sort_values(ascending=False)
```

Out[60]:
```
Date de premiere inscription en doctorat    383716
Identifiant auteur                          130347
Langue_rec                                   64120
Year                                         57086
Date de soutenance                           57086
Identifiant etablissement                    17082
etablissement_rec                             3074
Mise a jour dans theses.fr                     177
Directeur de these (nom prenom)                 13
Directeur de these                              13
Titre                                            7
Etablissement de soutenance                      1
```

```
Identifiant directeur                          0
Publication dans theses.fr                     0
Genre                                          0
Discipline_pre                                 0
Langue de la these                             0
Accessible en ligne                            0
Identifiant de la these                        0
Statut                                         0
Discipline                                     0
Auteur                                         0
dtype: int64
```

In [61]:
```
print("Le nombre de Discipline_pre unique est de: ", len(PhD_v3['Discipli
```

```
Le nombre de Discipline_pre unique est de:  15
```

In [62]:
```
# Verif avec la méthode value_counts()
len(PhD_v3['Discipline_pre'].value_counts())
```

Out[62]:
```
15
```

In [63]:
```
# Verif des NaN dans la variable
PhD_v3['Discipline_pre'].isnull().sum()
```

Out[63]:
```
0
```

In [64]:
```
# les disciplines qui correspondent aux années sans date (valeurs NaN)
missing_year = PhD_v3['Year'].isnull()
print("les disciplines qui correspondent aux années sans date (valeurs Na
      len(PhD_v3[missing_year]['Discipline_pre']))
print("Il faut les supprimer")
```

```
les disciplines qui correspondent aux années sans date (valeurs NaN) sont
: 57086
Il faut les supprimer
```

In [65]:
```
PhD_v3 = PhD_v3.dropna(subset=['Year'])
```

In [66]:
```
PhD_v3['Year']  = PhD_v3['Year'].astype(int)
```

In [67]:
```
PhD_v3['Year'].value_counts().sort_values(ascending=True)
```

Out[67]:
```
1979         1
1980         1
1982         1
1972         1
1976         1
1971         1
1973         1
1984         6
2020      1073
1985      3007
1986      5162
1987      8439
2002      9396
2001      9468
2003      9857
2004     10250
2005     10562
1995     10569
2019     10718
1991     10831
```

```
2000    10855
2006    10975
1999    10982
1990    11011
1998    11023
1988    11045
1989    11102
1996    11354
1997    11669
2007    11697
2008    11854
2009    12039
1992    12065
1993    12309
2010    12516
2018    12805
2016    12965
1994    12991
2015    13023
2017    13123
2011    13128
2014    13226
2013    13868
2012    13991
Name: Year, dtype: int64
```

In [68]:
```
PhD_v3[['Year']].describe()
```

Out[68]:

	Year
count	390961.000000
mean	2003.249455
std	9.845705
min	1971.000000
25%	1994.000000
50%	2004.000000
75%	2012.000000
max	2020.000000

In [69]:
```
PhD_v3['Discipline_pre'].describe(include='object')
```

Out[69]:
```
count        390961
unique           15
top        Biologie
freq          89142
Name: Discipline_pre, dtype: object
```

In [70]:
```
#Création du DataFrame groupé:
grouped_df = PhD_v3.groupby(['Year', 'Discipline_pre']).size().reset_inde
# la période 1985-2018
grouped_df = grouped_df[(grouped_df['Year'] >= 1985) & (grouped_df['Year'
#Pivoter le DataFrame pour obtenir les disciplines en tant que colonnes:
pivot_df = grouped_df.pivot(index='Year', columns='Discipline_pre', value
```

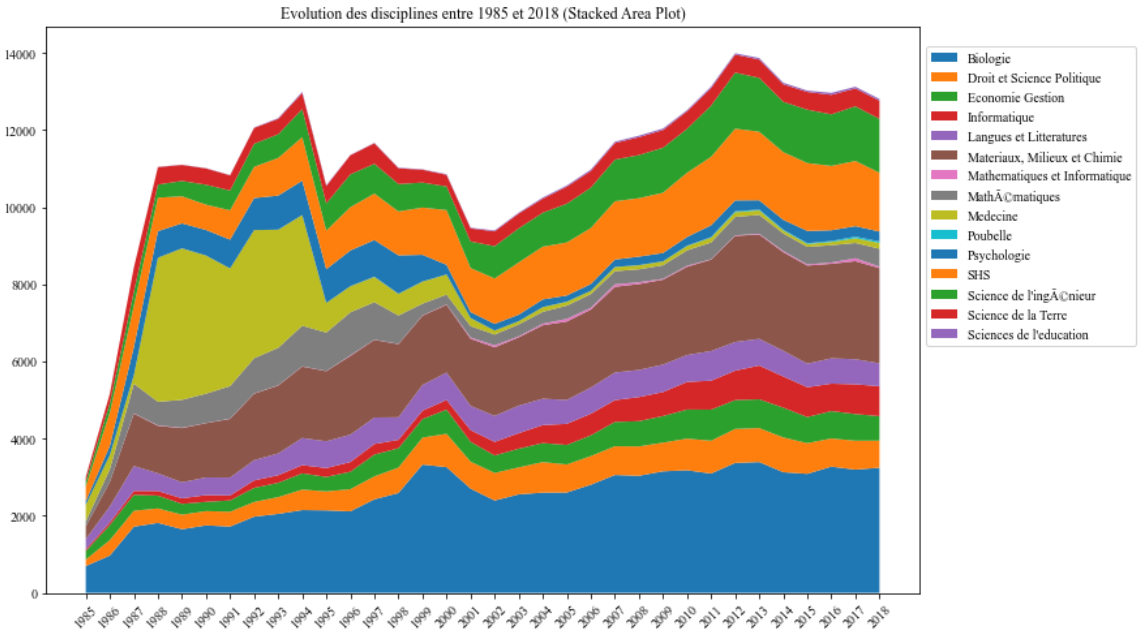
In [71]:
```
print(grouped_df.min())
print(grouped_df.max())
```

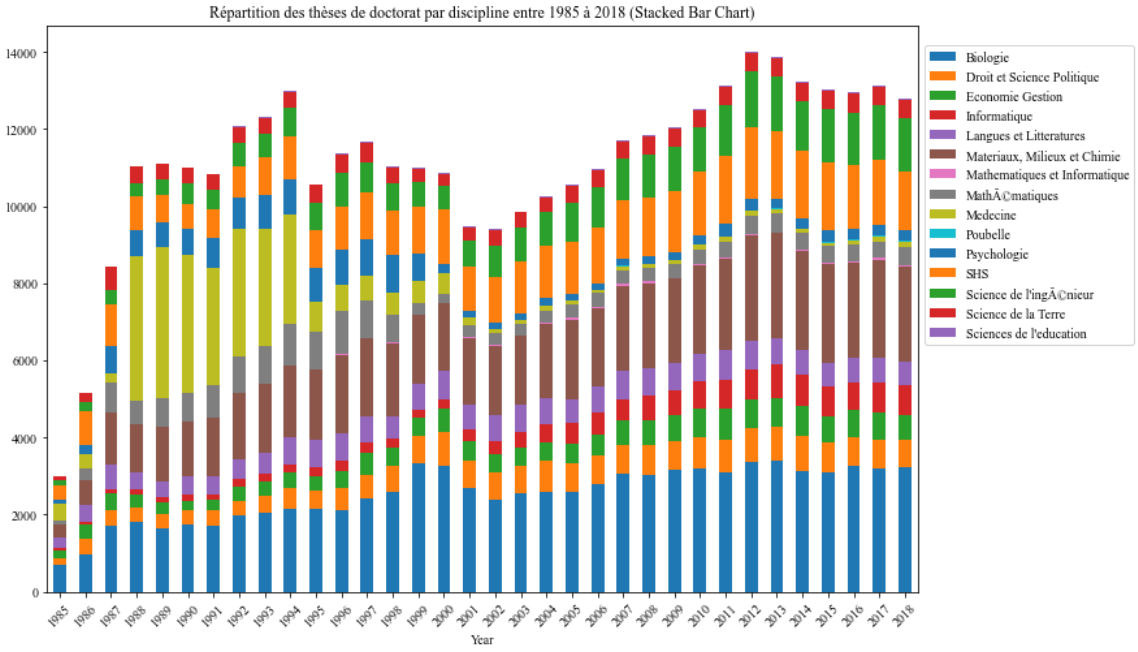
```
Year                    1985
Discipline_pre     Biologie
```

```
counts                      1
dtype: object
Year                                     2018
Discipline_pre     Sciences de l'education
counts                                   3926
dtype: object
```

3.1 Exercice 1

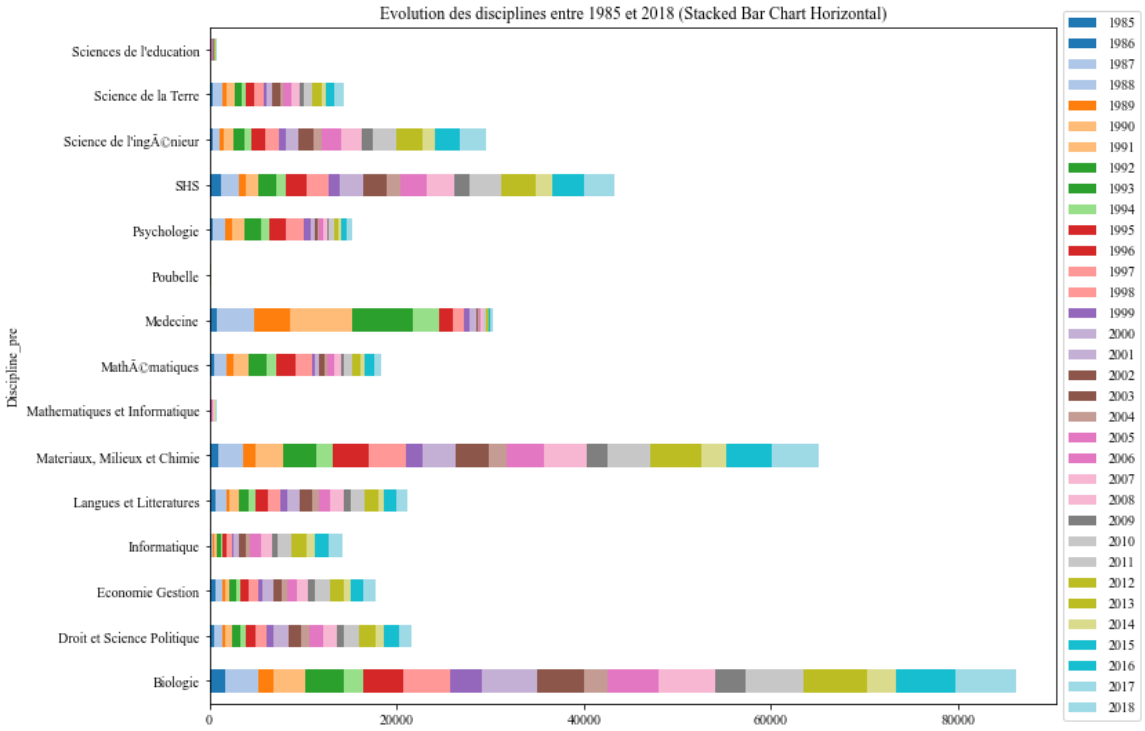
In [72]:
```
# 1- Stacked area plot:
plt.figure(figsize=(12,8))
plt.stackplot(pivot_df.index, pivot_df.values.T, labels=pivot_df.columns)
# Ajout des ticks pour toutes les années, y compris 2018
plt.xticks(np.arange(1985, 2019, 1), rotation=45)
# Place la légende à l'extérieur du graphe
plt.legend(loc='center left', bbox_to_anchor=(1, 0.7))
plt.title("Evolution des disciplines entre 1985 et 2018 (Stacked Area Plo
plt.show()


# 2- Stacked bar chart:
plt.figure(figsize=(12,8))
pivot_df.plot(kind='bar', stacked=True, ax=plt.gca())
plt.xticks(np.arange(len(pivot_df.index)), pivot_df.index, rotation=45)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.7))
plt.title("Répartition des thèses de doctorat par discipline entre 1985 à
plt.show()
```





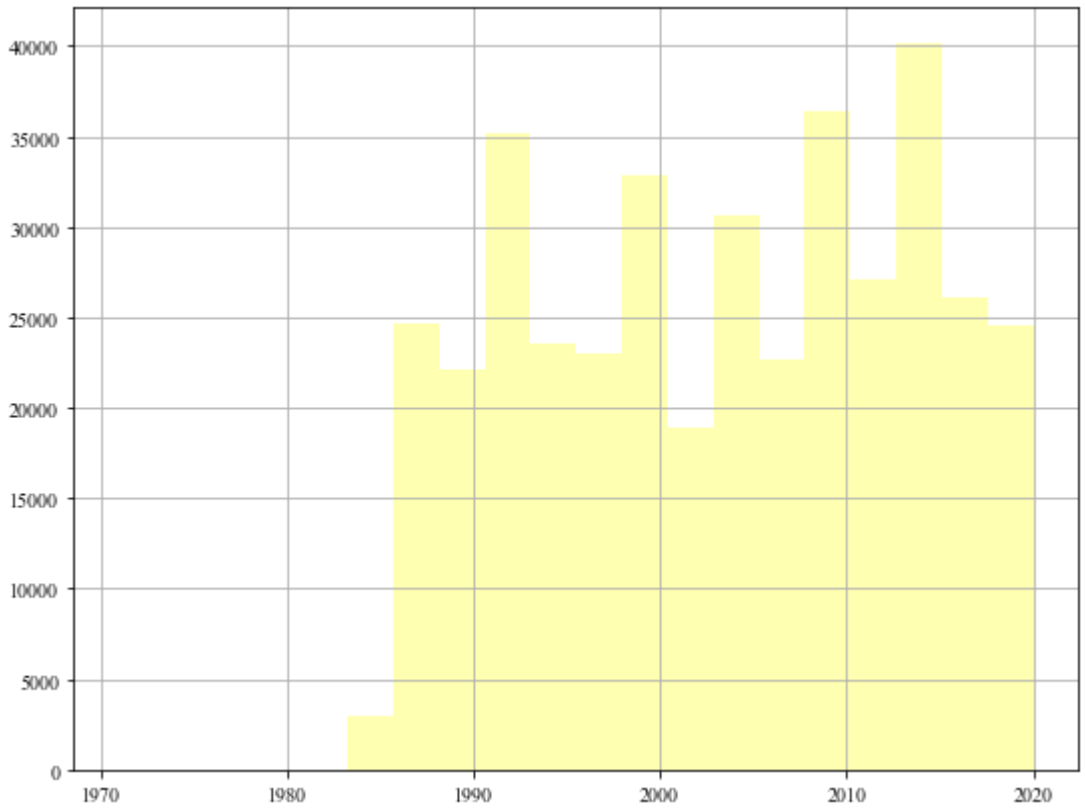
Graphique analogue "stacked barplot"

In [73]:
```
plt.figure(figsize=(11,9))
# Transposition du PhD_v3 pour avoir les années en tant que colonnes (néc
pivot_df.T.plot(kind='barh', stacked=True, ax=plt.gca(), colormap='tab20'
plt.yticks(np.arange(len(pivot_df.columns)), pivot_df.columns)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.title("Evolution des disciplines entre 1985 et 2018 (Stacked Bar Char
plt.show()
```



3.2 Exercice 2

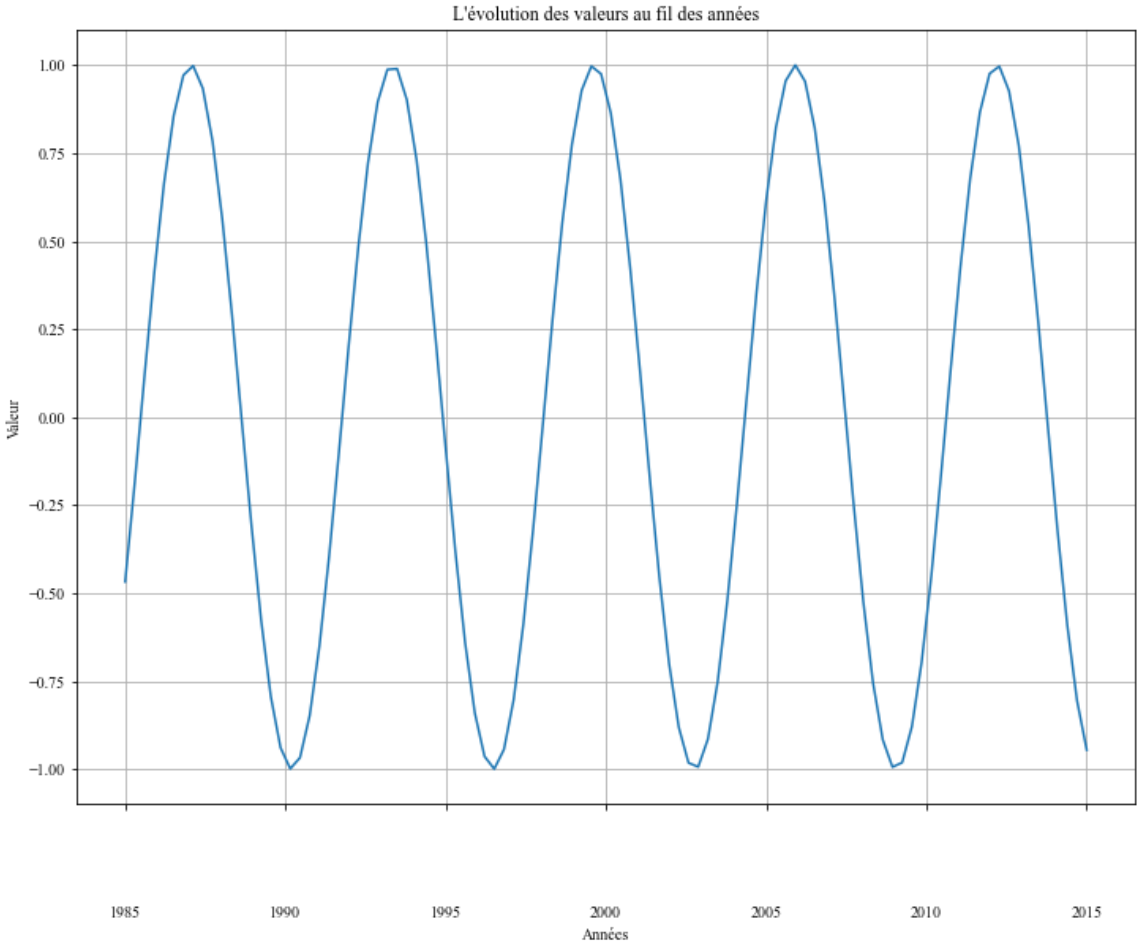
In [74]:
```
fig, ax = plt.subplots(figsize=(9,7))  # Création d'une figure et d'un ax
ax.grid(True)  # Ajoute une grille
ax.hist(PhD_v3['Year'], alpha=0.3, bins=20, color='yellow')  # Trace l'hi
plt.show()
```



3.3 Exercice 3

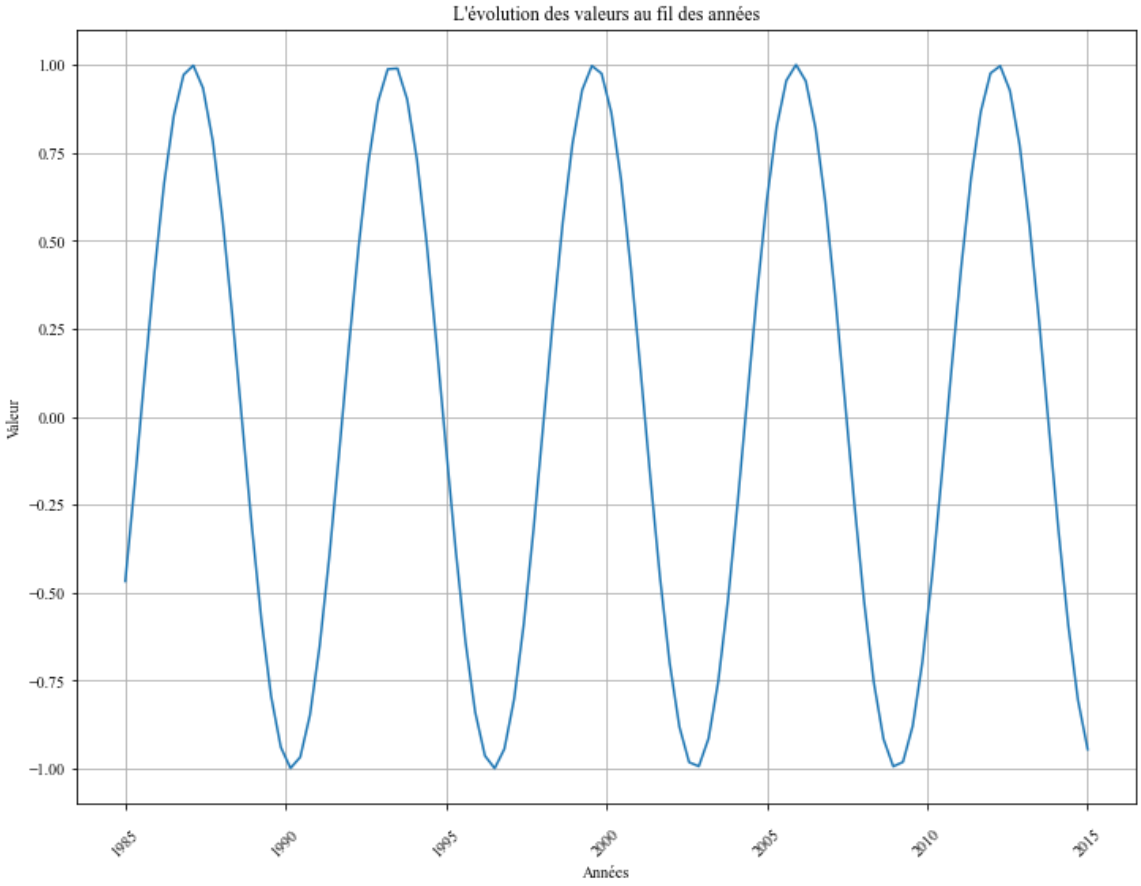
In [75]:

```
fig, ax = plt.subplots(figsize=(12,9))

# Ajouter une grille
ax.grid(True)

# Simuler des données
x = np.linspace(1985, 2015, 100)
y = np.sin(x)

# Tracer les données
ax.plot(x, y)

# Ajuster la position des labels de l'axe des x avec un pad = 60
ax.tick_params(axis='x', which='major', pad=60)

ax.set_xlabel("Années")
ax.set_ylabel("Valeur")
ax.set_title("L'évolution des valeurs au fil des années")

plt.show()
```



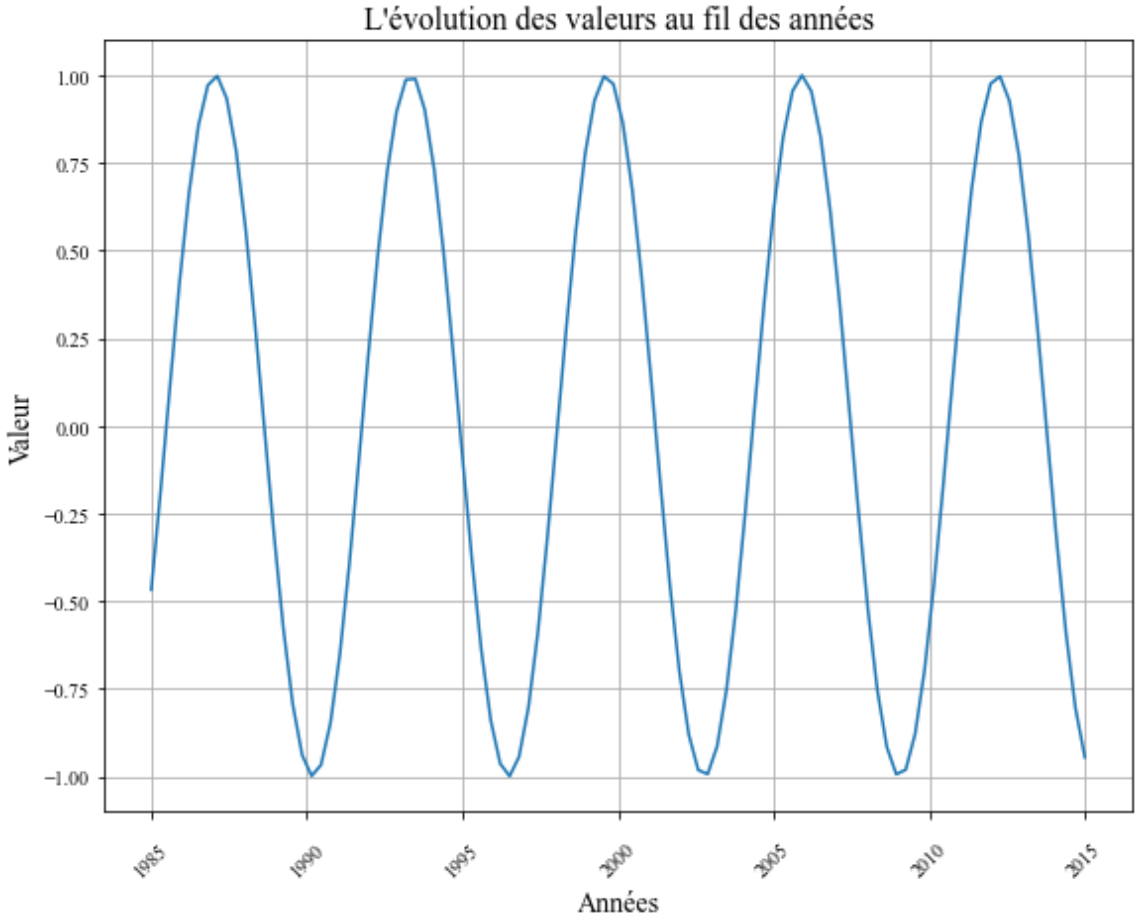
Rotation 45°

In [76]:
```
fig, ax = plt.subplots(figsize=(12,9))
ax.grid(True)
x = np.linspace(1985, 2015, 100)
y = np.sin(x)
ax.plot(x, y)
# Ajuster la position et l'inclinaison des labels de l'axe des x à 45°
ax.tick_params(axis='x', which='major', pad=10, rotation=45)
ax.set_xlabel("Années")
ax.set_ylabel("Valeur")
ax.set_title("L'évolution des valeurs au fil des années")
plt.show()
```

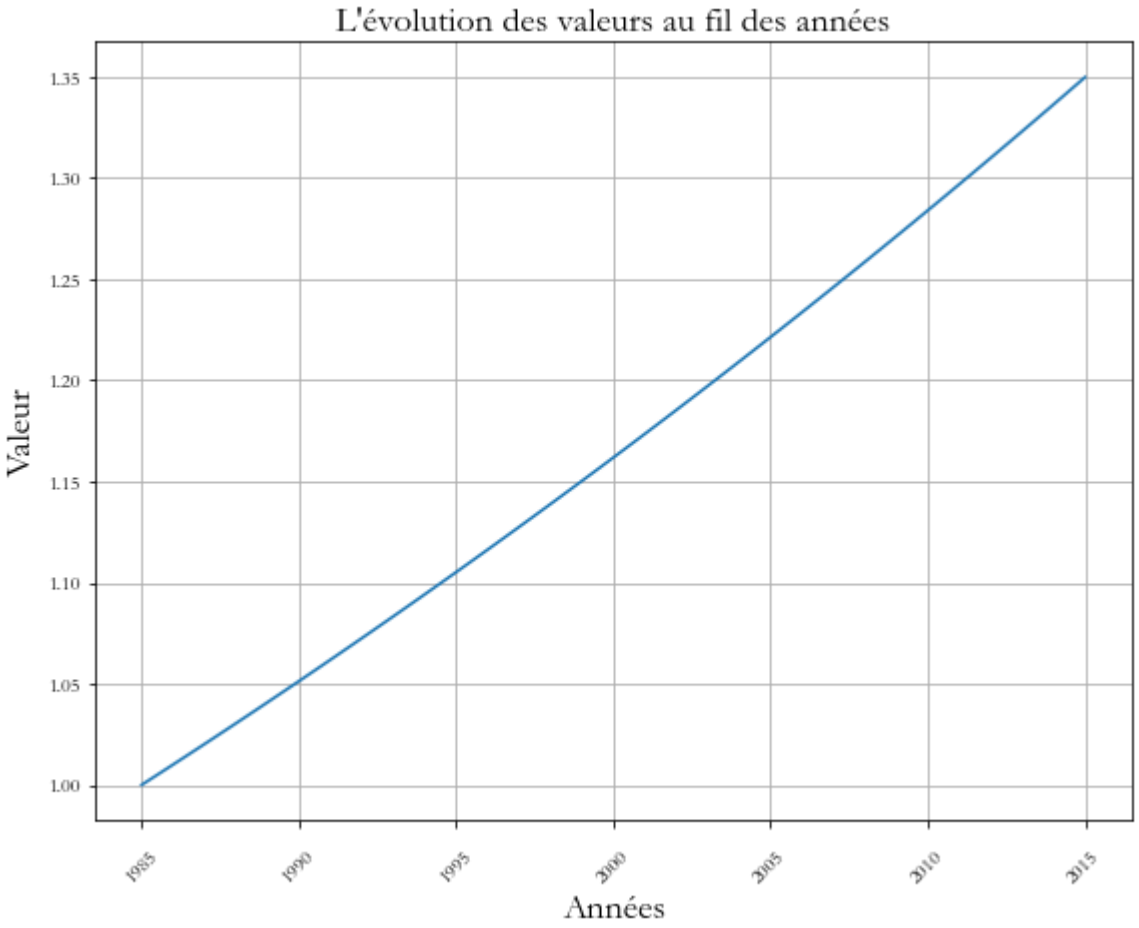


3.4 Exercice 4

In [77]:
```
# Changer la police globale pour toutes les étiquettes
plt.rcParams['font.family'] = 'Times New Roman'  # ou 'Garamond'
fig, ax = plt.subplots(figsize=(12,9))
ax.grid(True)
x = np.linspace(1985, 2015, 100)
y = np.sin(x)
# Tracer les données
ax.plot(x, y)
ax.tick_params(axis='x', which='major', pad=10, rotation=45)
# Ajuster la taille de la police pour les étiquettes des axes
ax.set_xlabel("Années", fontsize=14)
ax.set_ylabel("Valeur", fontsize=14)
# Ajuster la taille de la police pour le titre
ax.set_title("L'évolution des valeurs au fil des années", fontsize=16)
# Ajuster les marges pour "écraser" le graphique vers le centre
plt.subplots_adjust(left=0.2, right=0.8, bottom=0.2, top=0.8)
plt.show()
```

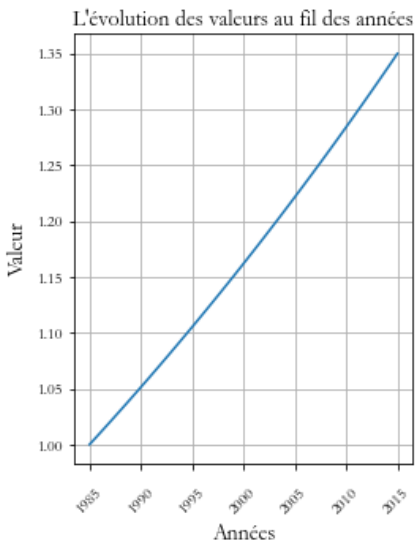


In [78]:
```
# Changement de la police globale pour toutes les étiquettes
plt.rcParams['font.family'] = 'Garamond' # ou 'Times New Roman'
fig, ax = plt.subplots(figsize=(12,9))
ax.grid(True)
x = np.linspace(1985, 2015, 100)
y = np.exp((x - 1985) / 100)  # ici j'ai décidé de changer linspace pour
ax.plot(x, y)
ax.tick_params(axis='x', which='major', pad=10, rotation=45)
ax.set_xlabel("Années", fontsize=18)
ax.set_ylabel("Valeur", fontsize=18)
ax.set_title("L'évolution des valeurs au fil des années", fontsize=18)
# Ajuster les marges pour "écraser" le graphique vers le centre
plt.subplots_adjust(left=0.2, right=0.8, bottom=0.2, top=0.8)
plt.show()
```



Jouer sur la taille des marges

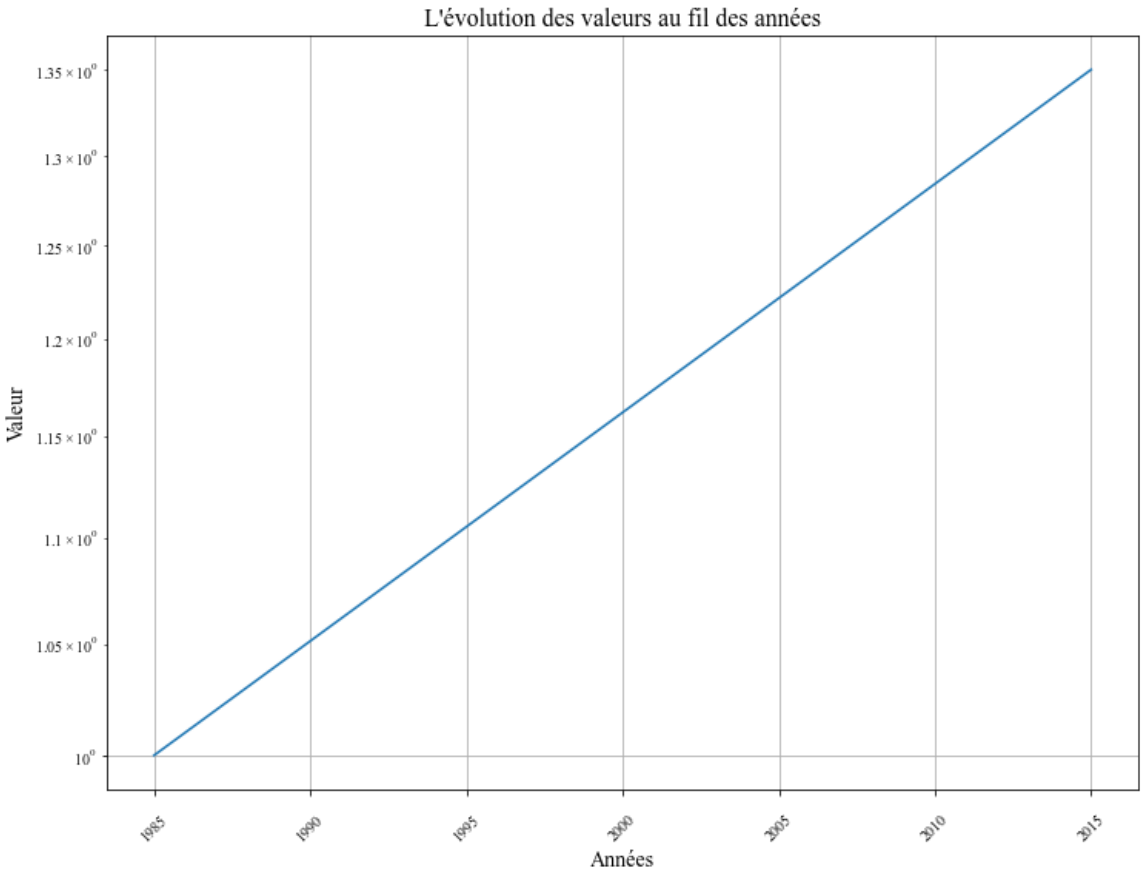
In [79]:
```
plt.rcParams['font.family'] = 'Garamond' # ou 'Times New Roman'

# Création de la grille spécifique pour contrôler la taille de chaque sou
gs = GridSpec(1, 2, width_ratios=[3, 2])  # Ici, la première sous-figure

fig = plt.figure(figsize=(12,5))
ax1 = plt.subplot(gs[0])
ax1.grid(True)
x1 = np.linspace(1985, 2015, 100)
y1 = np.exp((x1 - 1985) / 100)
ax1.plot(x1, y1)
ax1.tick_params(axis='x', which='major', pad=10, rotation=45)
ax1.set_xlabel("Années", fontsize=18)
ax1.set_ylabel("Valeur", fontsize=18)
ax1.set_title("L'évolution des valeurs au fil des années", fontsize=18)

ax2 = plt.subplot(gs[1])
ax2.grid(True)
x2 = np.linspace(1985, 2015, 100)
y2 = np.exp((x2 - 1985) / 100)
ax2.plot(x2, y2)
ax2.tick_params(axis='x', which='major', pad=10, rotation=45)
ax2.set_xlabel("Années", fontsize=14)
ax2.set_ylabel("Valeur", fontsize=14)
ax2.set_title("L'évolution des valeurs au fil des années", fontsize=14)
fig.subplots_adjust(wspace=0.5)

plt.show()
```



3.5 Exercice 5

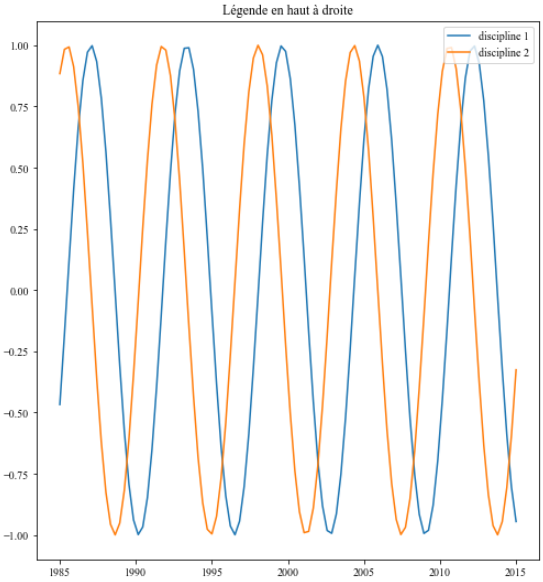
L'échelle de l'axe Y pour une échelle logarithmique en utilisant la fonction set_yscale('log').

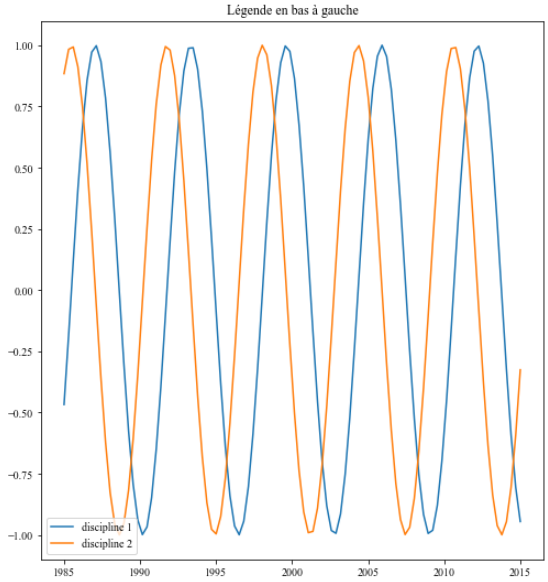
In [80]:
```
# Changement de la police globale pour toutes les étiquettes
plt.rcParams['font.family'] = 'Times New Roman'
fig, ax = plt.subplots(figsize=(12,9))
ax.grid(True)
x = np.linspace(1985, 2015, 100)
y = np.exp((x - 1985) / 100)
ax.plot(x, y)
ax.tick_params(axis='x', which='major', pad=10, rotation=45)
ax.set_xlabel("Années", fontsize=14)
ax.set_ylabel("Valeur", fontsize=14)
ax.set_title("L'évolution des valeurs au fil des années", fontsize=16)
# Changer l'échelle de l'axe Y à une échelle logarithmique
ax.set_yscale('log')

plt.show()
```



3.6 Exercice 6

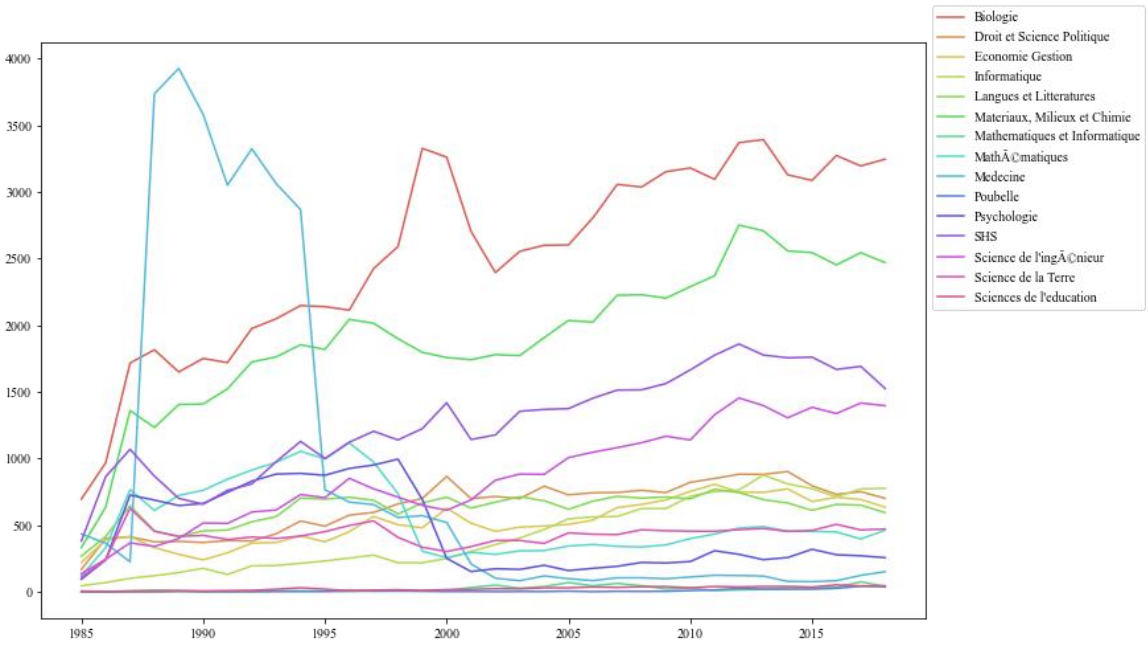
In [81]:
```
x = np.linspace(1985, 2015, 100)
y1 = np.sin(x)   # discipline 1: Juste à titre exemple
y2 = np.cos(x)   # discipline 2

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18,9))
# Première sous-figure
ax1.plot(x, y1, label='discipline 1')
ax1.plot(x, y2, label='discipline 2')
ax1.legend(loc='upper right')  # Position de la légende en haut à droite
ax1.set_title("Légende en haut à droite")

# Deuxième sous-figure
ax2.plot(x, y1, label='discipline 1')
ax2.plot(x, y2, label='discipline 2')
ax2.legend(loc='lower left')  # Position de la légende en bas à gauche
ax2.set_title("Légende en bas à gauche")

plt.show()
```





3.7 Exercice 7

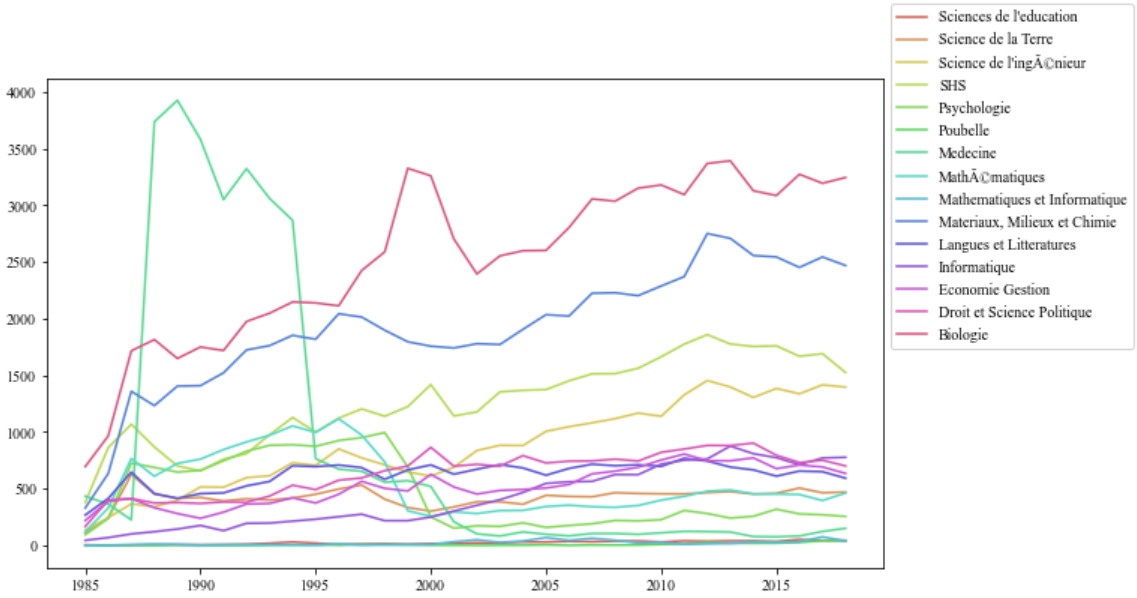
In [82]:
```
# Définir la palette de couleurs
palette = sns.color_palette("hls", pivot_df.columns.size)
fig, ax = plt.subplots(figsize=(12, 8))
# Tracer les données avec différentes couleurs pour chaque discipline
for i, column in enumerate(pivot_df.columns):
    ax.plot(pivot_df.index, pivot_df[column], color=palette[i], label=col
# Ajouter une légende
plt.legend(loc='center left', bbox_to_anchor=(1, 0.8))

plt.show()
```



3.8 Exercice 8

In [83]:
```
palette = sns.color_palette("hls", pivot_df.columns.size)
fig, ax = plt.subplots(figsize=(10, 6))

# Inverser l'ordre des colonnes
pivot_df = pivot_df.iloc[:, ::-1]
# Tracer les données avec différentes couleurs pour chaque discipline
for i, column in enumerate(pivot_df.columns):
```

```
    ax.plot(pivot_df.index, pivot_df[column], color=palette[i], label=col

# Ajouter une légende
plt.legend(loc='center left', bbox_to_anchor=(1, 0.8))

plt.show()
```

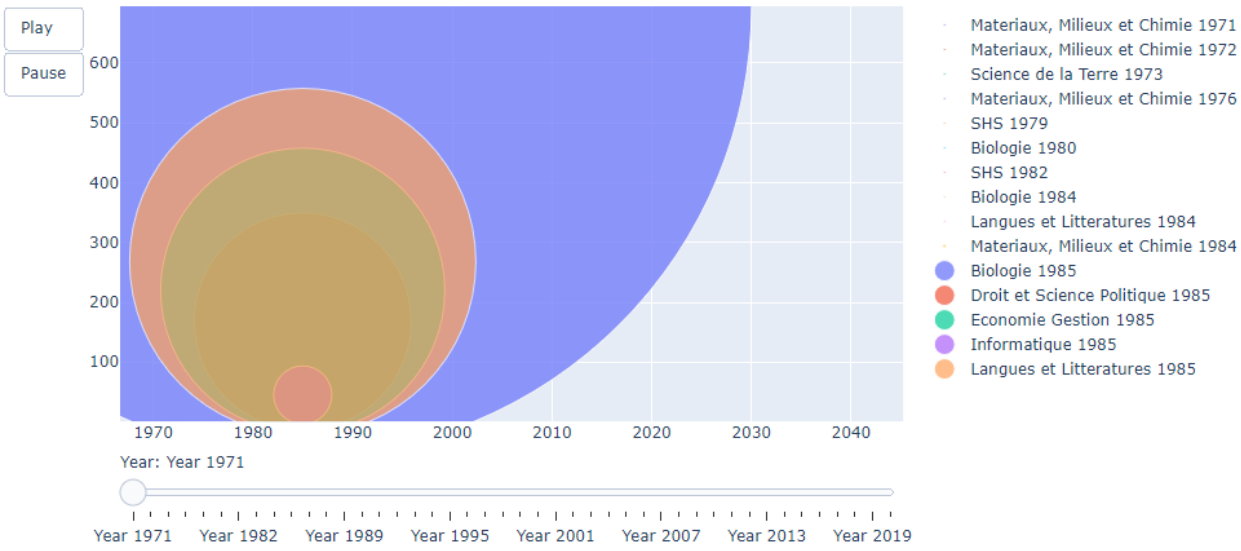


3.9 Exercice 9

In [84]:

```
# Grouper par année et discipline et compter le nombre de thèses
grouped_df = PhD_v3.groupby(['Year', 'Discipline_pre']).size().reset_inde

# Créer une figure vide
fig = go.Figure()

# Pour chaque année, ajouter un scatter plot pour chaque discipline
for year in grouped_df['Year'].unique():
    df_year = grouped_df[grouped_df['Year'] == year]
    for discipline in df_year['Discipline_pre'].unique():
        fig.add_trace(
            go.Scatter(
                visible=(year == grouped_df['Year'].min()),
                mode='markers',
                name=f"{discipline} {year}",
                x=df_year['Year'],
                y=df_year[df_year['Discipline_pre'] == discipline]['count
                marker=dict(size=df_year[df_year['Discipline_pre'] == dis
            )
        )

# Créer un slider
steps = []
for i, year in enumerate(grouped_df['Year'].unique()):
    step = dict(
        method="restyle",
        args=["visible", [False]*len(fig.data)],
        label=f"Year {year}"
    )
    step["args"][1][i*len(grouped_df['Discipline_pre'].unique()):(i+1)*le
    steps.append(step)

sliders = [dict(
    active=0,
```

```
    currentvalue={"prefix": "Year: "},
    steps=steps
)]

fig.update_layout(
    sliders=sliders,
    updatemenus=[
        dict(
            type="buttons",
            showactive=False,
            buttons=[
                dict(
                    label="Play",
                    method="animate",
                    args=[None, {"fromcurrent": True, "frame": {"duration
                ),
                dict(
                    label="Pause",
                    method="animate",
                    args=[[None], {"frame": {"duration": 0, "redraw": Fal
                ),
            ],
        )
    ],
)

frames = [go.Frame(data=[go.Scatter(x=grouped_df[grouped_df['Year'] == ye
                                    y=grouped_df[grouped_df['Year'] == ye
                                    mode='markers',
                                    marker=dict(size=grouped_df[grouped_d
          for year in grouped_df['Year'].unique()]

fig.frames = frames

fig.show()
```



In [85]:

```
grouped_df = PhD_v3.groupby(['Year', 'Discipline_pre']).size().reset_inde
grouped_df = grouped_df[(grouped_df['Year'] >= 1985) & (grouped_df['Year'
pivot_df = grouped_df.pivot(index='Year', columns='Discipline_pre', value

fig = go.Figure()

# Pour chaque discipline, ajoutez une trace à la figure
for discipline in pivot_df.columns:
    fig.add_trace(
        go.Scatter(
            visible=True,  # change to False if you want all traces to be
            mode='lines',
            name=discipline,
            x=pivot_df.index,
            y=pivot_df[discipline],
        )
    )

# Création des boutons pour le "selector"
buttons = []
for i, discipline in enumerate(pivot_df.columns):
    visibility = [False] * len(pivot_df.columns)
    visibility[i] = True
    button = dict(
        label=discipline,
        method="update",
        args=[{"visible": visibility},
              {"title": f"Discipline: {discipline}"}])
    buttons.append(button)

# Ajout des boutons à la mise en page de la figure
updatemenus = [{"buttons": buttons}]

fig.update_layout(updatemenus=updatemenus)
fig.show()
```