Beyond Standardization: A Comprehensive Review of Topic Modeling Validation Methods for Computational Social Science Research

Analysis Replication File

This jupyter notebook runs all the analysis steps that have been taken to get the descriptive numbers and visualizations. Please note that graphs might look different on your machine, depending on your settings. The code is not commented on in detail, however, the headings should give an insight into what the code is doing. The variable labels have been given during coding, so some of them are not as readable as others. In case there are any errors or inconsistencies please contact [the authors].

In [1]:
```
# Load all required packages

# for data handling
import pandas as pd

# for analysis
from scipy.stats import entropy
import numpy as np
import itertools
from itertools import combinations


# for plotting
import seaborn as sns
sns.set(context='paper',
        style='whitegrid',
        font_scale=1.2,
        rc={'figure.figsize':(14,7),
            'font.family':'sans-serif',
            'font.size':15,
            'axes.titlesize':15,
            'axes.labelsize':10})

colors = ['#0063A6',  '#A71C49', '#DD4814', '#F6A800', '#94C154', '#11
sns.set_palette(sns.color_palette(colors))

import matplotlib.pyplot as plt
import matplotlib.ticker as mtick

import pywaffle
from pywaffle import Waffle
```

In [2]:
```
# Load dataset
path_to_dataset = 'ReplicationDataset.csv'
df = pd.read_csv(path_to_dataset, sep=';')
print("Overall we have", len(df), "articles in our sample.")

rows_count = df[df['Year'] >= 2011].shape[0]
print(f"For some graphs we will only look at more recent papers, the r

# fill empty cells with 0
methods = ['Cross-Validation',
       'diffMethods', 'Split Train Test Set', 'Baseline Model',
       'CHI (Calinski-Harabarz Index)', 'Coherence', 'Exclusivity', 'F
       'Cohesion', 'Topic Uniqueness (TU)', 'Downstream Tasks', 'Accur
       'AUC-ROC', 'Error 1 & 2', 'F-Score', 'Fowlkes and Mallows Index
       'Homogeneity Index', 'Mean Absolute Deviation (MAD)',
       'Mean Absolute Error (MAE)', 'Mean Squared Error (MSE)',
       'Precision & Recall',
       'Recall-Oriented Understudy for Gisting Evaluation (ROUGE)',
       'Root Mean Scared Error (RMSE)', 'Completeness', 'Experts',
       'Human Analysis', 'Human Topic Interpretation',
       'Theoretical Considerations', 'Top Documents', 'Topic Labeling'
       'Word Intrusion', 'Real Life Dynamics', 'Topic Intrusion', 'Ent
       'Jensen-Shannon Divergence (JSD)', 'Kullback-Leibler Divergence
       'Mutual Information (MI)', 'Perplexity',
       'Variation of Information (VI)', 'Information Retrieval (IR)',
       'Akaike Information Criterion (AIC)', 'Information Gain (IG)',
       'Cluster Distance', 'DBI (Davies-Bouldin Index)', 'ARI',
       'Jaccard Coefficient', 'Silhouette', 'Similarity']
df[methods] = df[methods].fillna(0)

# add categories
df['ComparingMethods'] = df['Cross-Validation'] + df['diffMethods'] +
df['Distinctivness'] = df['Coherence'] + df['Exclusivity'] + df['Purit
df['Downstream'] = df['Downstream Tasks']
df['ErrorRateAnalysis'] = df['Accuracy'] + df['AUC-ROC'] + df['Error 1
df['Human_int'] = df['Experts'] + df['Human Topic Interpretation'] + d
df['Human_ext'] = df['Human Analysis'] + df['Theoretical Considerations
df['Info'] = df['Entropy'] + df['Jensen-Shannon Divergence (JSD)'] + d
df['Similar'] = df['Jaccard Coefficient'] + df['Silhouette'] + df['Sim

categories = ['ComparingMethods', 'Distinctivness', 'Downstream', 'Err


# Add binary categories
df['binary_ComparingMethods'] = (df['Cross-Validation'] + df['diffMeth
df['binary_Distinctivness'] = (df['Coherence'] + df['Exclusivity'] + d
df['binary_Downstream'] = (df['Downstream Tasks']).apply(lambda x: 0 i
df['binary_ErrorRateAnalysis'] = (df['Accuracy'] + df['AUC-ROC'] + df
df['binary_Human_int'] = (df['Experts'] + df['Human Topic Interpretati
df['binary_Human_ext'] = (df['Human Analysis'] + df['Theoretical Consi
df['binary_Info'] = (df['Entropy'] + df['Jensen-Shannon Divergence (JS
df['binary_Similar'] = (df['Jaccard Coefficient'] + df['Silhouette'] +

binary_categories = ['binary_ComparingMethods', 'binary_Human_int', 'b

# add needed counters
df['method_count'] = df[methods].sum(axis=1)
df['category_count'] = df[categories].sum(axis = 1, numeric_only=True)
df['binary_category_count'] = df[binary_categories].sum(axis = 1, nume
```

```
# add needed subsets
substantive_df = df[df['Substantive'] == 1]
methodological_df = df[df['Methodological'] == 1]
core_sosci_df = df[df['Field'] == 'Social Science']
peripheral_sosci_df = df[df['Field'] != 'Social Science']
```

```
Overall we have 789 articles in our sample.
For some graphs we will only look at more recent papers, the number
of studies published since 2011 is: 750
```

In [3]:
```
df
```

Out[3]:

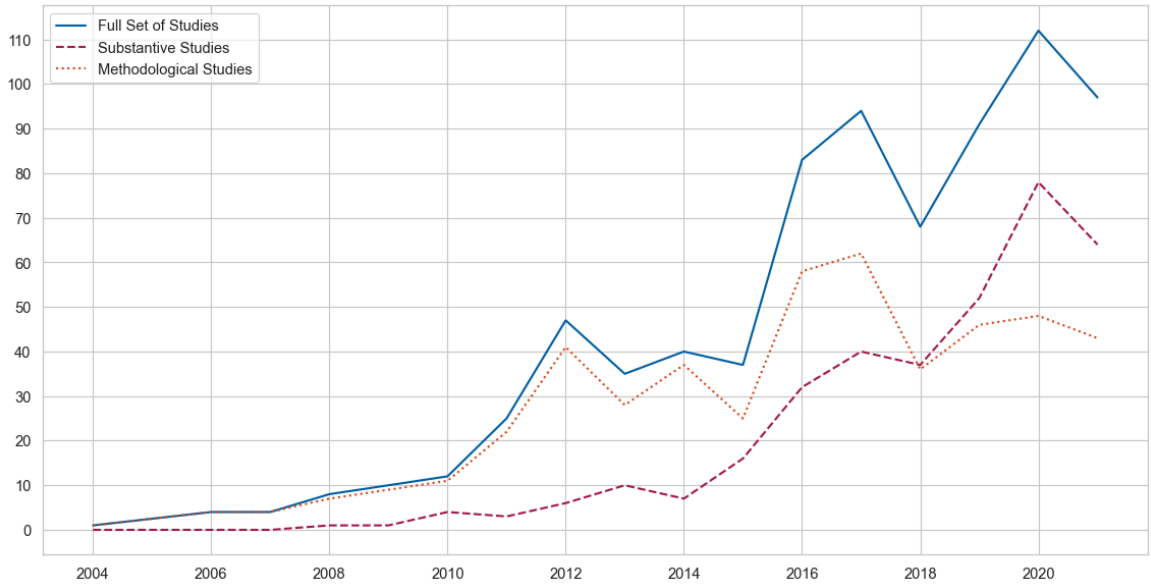
	index	RfE	Year	Methodological	Substantive	Multigoal	Publication	Publication-short	
0	1	0	2016	0.0	1	1	IEEE INTERNATIONAL REQUIREMENTS ENGINEERING CO...	C	
1	4	0	2020	0.0	1	1	International Conference on Mining Software Re...	C	
2	5	0	2020	0.0	1	1	International Conference on World Wide Web	C	
3	6	0	2010	0.0	1	1	ACM Transactions on the Web	J	
4	8	0	2019	0.0	1	1	International Conference on Artificial Intelli...	C	
...	...	...	...	...	...	...	...	...	
784	1550	0	2021	1.0	0	1	Expert Systems with Applications	J	
785	1551	0	2014	0.0	1	1	DATA & KNOWLEDGE ENGINEERING	J	
786	1553	0	2015	0.0	1	1	Working Conference on Mining Software Reposito...	C	
787	1555	0	2016	1.0	0	1	ACM SIGKDD International Conference on Knowled...	C	
788	1556	0	2016	1.0	0	1	KNOWLEDGE AND INFORMATION SYSTEMS	J	

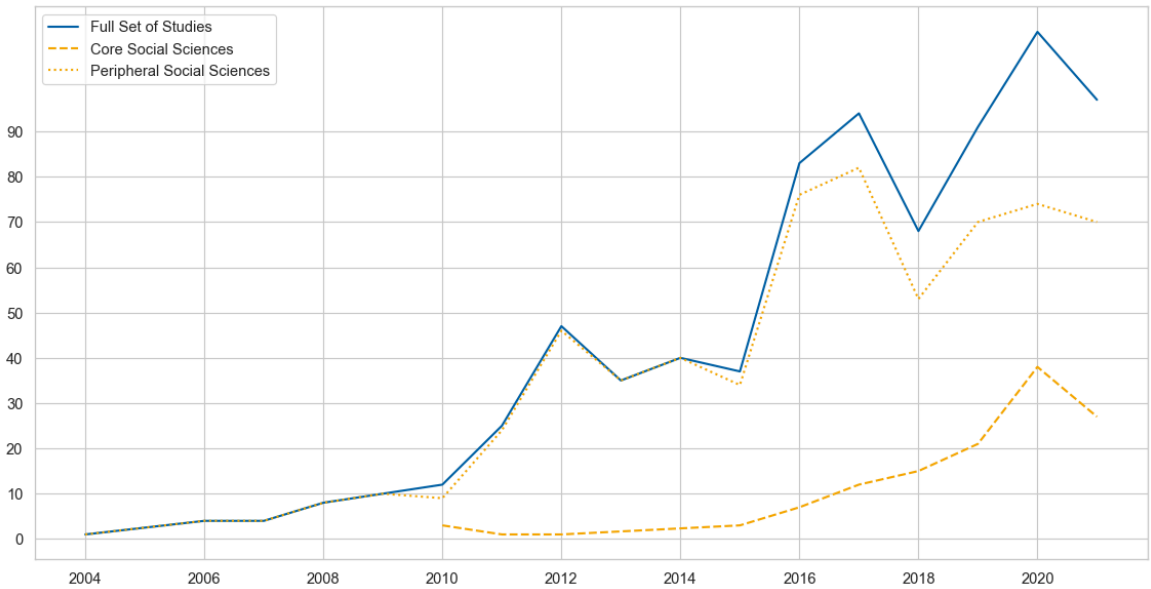
789 rows × 80 columns

In [4]:
```
## get overview over how many methods a study uses

# raw numbers
print(f'The average study uses {df["method_count"].mean():.2f} validat
      f'The average study uses {df["binary_category_count"].mean():.2f

# Print the averages for each subset
print(f'The average substantive study uses {substantive_df["method_cou
      f'The average substantive study uses {substantive_df["binary_cat

print(f'The average methodological study uses {methodological_df["meth
      f'The average methodological study uses {methodological_df["bina

print(f'The average core social science study uses {core_sosci_df["met
      f'The average core social science study uses {core_sosci_df["bin

print(f'The average peripheral social science study uses {peripheral_s
      f'The average peripheral social science study uses {peripheral_s


# Plot 1: Substantive vs. Methodological Studies Over Time
df_year = df['Year'].value_counts().sort_index()
df_fields = pd.DataFrame({
    'substantive studies': df['Substantive'],
    'methodological studies': df['Methodological'],
    'Year': df['Year']
})
tab = df_fields.groupby('Year').sum()
tab['all studies'] = df_year
tab = tab[['all studies', 'substantive studies', 'methodological studi

plt.figure()  # Create a new figure for the first plot
vis1 = sns.lineplot(data=tab['all studies'], label='Full Set of Studie
sns.lineplot(data=tab['substantive studies'], label='Substantive Studi
sns.lineplot(data=tab['methodological studies'], label='Methodologica
vis1.set_xticks(range(2004, 2022, 2))
vis1.set_yticks(range(0, 120, 10))
vis1.set(xlabel=None, ylabel=None)
fig1 = vis1.get_figure()
fig1.savefig('overtime_numberOfStudies.png', bbox_inches='tight')

# Plot 2: Core vs. Peripheral Social Sciences Over Time
core_sosci_counts = core_sosci_df.groupby('Year').size()
peripheral_sosci_counts = peripheral_sosci_df.groupby('Year').size()
tab_core_peripheral = pd.DataFrame({
    'Core Social Sciences': core_sosci_counts,
    'Peripheral Social Sciences': peripheral_sosci_counts
})
tab_core_peripheral['all studies'] = df_year
tab_core_peripheral = tab_core_peripheral[['all studies', 'Core Social

plt.figure()  # Create a new figure for the second plot
core_color = colors[3]
peripheral_color = colors[3]
vis2 = sns.lineplot(data=tab_core_peripheral['all studies'], label='Fu
sns.lineplot(data=tab_core_peripheral['Core Social Sciences'], label=
sns.lineplot(data=tab_core_peripheral['Peripheral Social Sciences'],
vis2.set_xticks(range(2004, 2022, 2))
vis2.set_yticks(range(0, 100, 10))
vis2.set(xlabel=None, ylabel=None)
fig2 = vis2.get_figure()
```

```
fig2.savefig('overtime_core_vs_peripheral_social_sciences.png', bbox_i
```

```
The average study uses 4.07 validation methods.
The average study uses 2.78 binary validation categories.

The average substantive study uses 3.74 validation methods.
The average substantive study uses 2.67 binary validation categorie
s.

The average methodological study uses 4.34 validation methods.
The average methodological study uses 2.88 binary validation categor
ies.

The average core social science study uses 3.61 validation methods.
The average core social science study uses 2.50 binary validation ca
tegories.

The average peripheral social science study uses 4.16 validation met
hods.
The average peripheral social science study uses 2.84 binary validat
ion categories.
```





In [5]:
```
### Print an Overview of the most frequent publication spots:

print('Overall: \n', df.Publication.value_counts().head(10), '\n')

df_journals = df[df['Publication-short']== 'J']
print('Journals: \n', df_journals.Publication.value_counts().head(10),

df_conf = df[df['Publication-short']== 'C']
print('Conferences: \n', df_conf.Publication.value_counts().head(10),

print('Core Social Science: \n', core_sosci_df.Publication.value_count

print('Peripheral Social Science: \n', peripheral_sosci_df.Publicatio
```

```
Overall:
 ACM International Conference on Information and Knowledge Managemen
t          38
International Conference on World Wide Web
34
ACM SIGKDD International Conference on Knowledge Discovery and Data
Mining    25
ACM SIGIR Conference on Research and Development in Information Retr
ieval     22
ACM International Conference on Web Search and Data Mining
18
Conference on Empirical Methods in Natural Language Processing
9
ACM Transactions on Knowledge Discovery from Data
9
IEEE ACCESS
9
ACM Transactions on Intelligent Systems and Technology
9
ACM Transactions on Information Systems
8
Name: Publication, dtype: int64

Journals:
 IEEE ACCESS
9
ACM Transactions on Intelligent Systems and Technology             9
ACM Transactions on Knowledge Discovery from Data                  9
ACM Transactions on Information Systems                            8
International Journal of Communication                             8
Journal of Machine Learning Research                               8
Communication Methods & Measures                                   7
IEEE/ACM Transactions on Audio, Speech, and Language Processing    7
Political Communication                                            6
Marketing Science                                                  6
Name: Publication, dtype: int64

Conferences:
 ACM International Conference on Information and Knowledge Managemen
t                     38
International Conference on World Wide Web
34
ACM SIGKDD International Conference on Knowledge Discovery and Data
Mining              25
ACM SIGIR Conference on Research and Development in Information Retr
ieval               22
ACM International Conference on Web Search and Data Mining
18
Conference on Empirical Methods in Natural Language Processing
9
ACM Conference on Web Science
7
ACM Conference on Computer Supported Cooperative Work and Social Com
puting               5
IEEE/ACM International Conference on Advances in Social Networks Ana
lysis and Mining     5
Annual Meeting of the Association for Computational Linguistics
5
Name: Publication, dtype: int64

Core Social Science:
```

```
 International Journal of Communication                      8
Communication Methods & Measures                            7
Marketing Science                                           6
Political Communication                                     6
Environmental Communication                                 4
Journalism Studies                                          4
Journal of Broadcasting & Electronic Media                  3
International Conference on Social Media and Society        2
American Sociological Review                                2
International Conference on Digital Government Research     2
Name: Publication, dtype: int64

Peripheral Social Science:
 ACM International Conference on Information and Knowledge Managemen
t          38
International Conference on World Wide Web
34
ACM SIGKDD International Conference on Knowledge Discovery and Data
Mining    25
ACM SIGIR Conference on Research and Development in Information Retr
ieval     22
ACM International Conference on Web Search and Data Mining
18
ACM Transactions on Knowledge Discovery from Data
9
ACM Transactions on Intelligent Systems and Technology
9
IEEE ACCESS
9
Conference on Empirical Methods in Natural Language Processing
9
ACM Transactions on Information Systems
8
Name: Publication, dtype: int64
```

In [6]:

```
### Get an Overview of the Frequencies

publication_counts = df['Publication-short'].value_counts()
publication_frequencies = df['Publication-short'].value_counts(normali

j_count = publication_counts.get('J', 0)
j_frequency = publication_frequencies.get('J', 0)

c_count = publication_counts.get('C', 0)
c_frequency = publication_frequencies.get('C', 0)


print(f"Journals: Absolute = {j_count}, Frequency = {j_frequency:.2%}'
print(f"Conferences: Absolute = {c_count}, Frequency = {c_frequency:.2
```

```
Journals: Absolute = 362, Frequency = 45.88%
Conferences: Absolute = 426, Frequency = 53.99%
```

In [7]:
```
### Insights into use of individual methods:

separator = '-' * 40
for v in methods:
    print(f"\nMethod: {v}\n{separator}")
    print(df[v].value_counts().sort_index().to_string())
    print(separator)
```

```
----------------------------------------
0.0    652
1.0    135
2.0      2
----------------------------------------

Method: AUC-ROC
----------------------------------------
0.0    764
1.0     21
2.0      3
3.0      1
----------------------------------------

Method: Error 1 & 2
----------------------------------------
0.0    777
1.0      5
2.0      6
5.0      1
```

In [8]:
```
### Insights into use of categories:

sorted_categories = sorted(categories)

print("\nValidation Categories Summary\n" + separator)
for c in sorted_categories:
    print(f"{c}: {df[c].sum()}")
print(separator)

sorted_binary_categories = sorted(binary_categories)

print("\nBinary Validation Categories Summary\n" + separator)
for c in sorted_binary_categories:
    print(f"{c}: {df[c].sum()}")
print(separator)
```

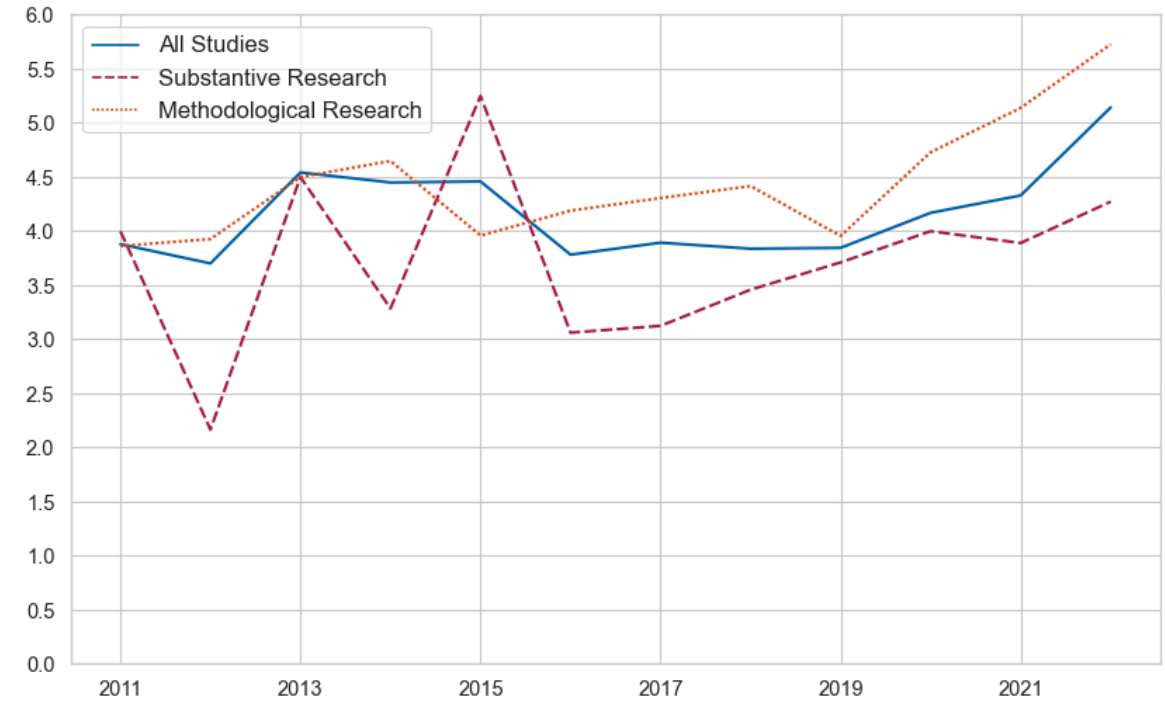
```
Validation Categories Summary
----------------------------------------
ComparingMethods: 763.0
Distinctivness: 274.0
Downstream: 334.0
ErrorRateAnalysis: 618.0
Human_ext: 210.0
Human_int: 634.0
Info: 223.0
Similar: 80.0
----------------------------------------

Binary Validation Categories Summary
----------------------------------------
binary_ComparingMethods: 483
binary_Distinctivness: 176
binary_Downstream: 320
binary_ErrorRateAnalysis: 351
binary_Human_ext: 177
binary_Human_int: 428
binary_Info: 191
binary_Similar: 66
----------------------------------------
```

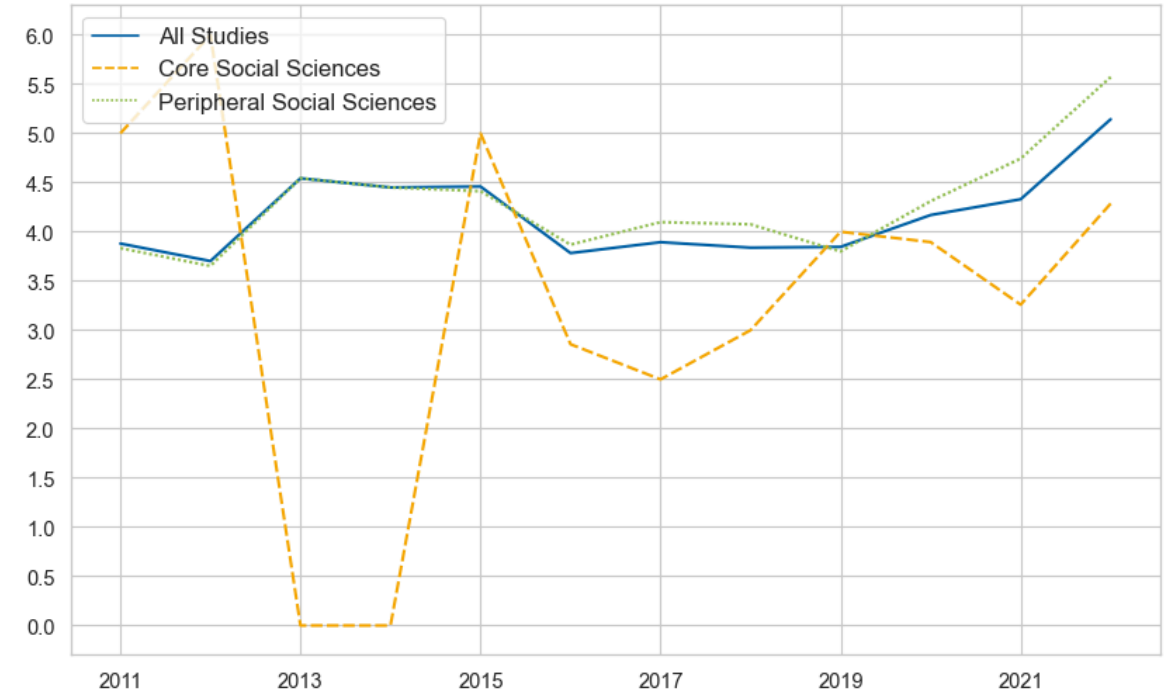
In [9]:
```
### waffle plots
df_binary_categories = pd.DataFrame({
    'category': binary_categories,
    'nonnullcount': df[binary_categories].sum(),
})

df_binary_categories['percentage'] = (df_binary_categories['nonnullcou
df_binary_categories['percentage'] = df_binary_categories['percentage

# Set up custom color palette and category names
colors = ['#0063A6',  '#A71C49', '#DD4814', '#F6A800', '#94C154', '#11
pal_ = list(sns.color_palette(colors).as_hex())
cat_names = [
    'Model Comparison', 'Internal Qualitative Inspection', 'Error Rate
    'Downstream Tasks', 'Information Theory Metrics', 'External Qualit
    'Distinctivness of Topwords', 'Similarity and Distance Measures'
]

# Plot waffle charts for each category
plt.rcParams['savefig.facecolor'] = 'white'
keep_sname = []

# Loop through the DataFrame using `enumerate()` to ensure integer ind
for idx, (index, row) in enumerate(df_binary_categories.iterrows()):
    fig = plt.figure(
        FigureClass=Waffle,
        rows=5, columns=20,
        values=[row['percentage'], 100 - row['percentage']],
        colors=[pal_[idx], 'gainsboro'],  # Use integer index `idx` to
        figsize=(3, 3)
    )

    plt.title(f"{cat_names[idx]}: {row['percentage']}% ({row['nonnullc
    keep_sname.append(f"waffle_{row['category']}.png")
    plt.tight_layout()

    # Save the figure as a PDF
    plt.savefig(
        f"waffle_{row['category']}.pdf",
        bbox_inches='tight', format='pdf'
    )
    plt.close(fig)

# Function to create waffle plots for subsets
def create_waffle_plots_for_subset(subset_df, subset_name):
    df_binary_categories = pd.DataFrame({
        'category': binary_categories,
        'nonnullcount': subset_df[binary_categories].sum(),
    })

    df_binary_categories['percentage'] = (df_binary_categories['nonnul
    df_binary_categories['percentage'] = df_binary_categories['percent

    # Set up color palette and category names
    colors = ['#0063A6',  '#A71C49', '#DD4814', '#F6A800', '#94C154',
    pal_ = list(sns.color_palette(colors).as_hex())
    cat_names = [
        'Model Comparison', 'Internal Qualitative Inspection', 'Error
        'Downstream Tasks', 'Information Theory Metrics', 'External Qu
        'Distinctivness of Topwords', 'Similarity and Distance Measure
    ]
```

```
    # Plot waffle charts for each category
    plt.rcParams['savefig.facecolor'] = 'white'
    keep_sname = []

    for idx, (index, row) in enumerate(df_binary_categories.iterrows()
        fig = plt.figure(
            FigureClass=Waffle,
            rows=5, columns=20,
            values=[row['percentage'], 100 - row['percentage']],
            colors=[pal_[idx], 'gainsboro'],  # Use integer index `idx
            figsize=(3, 3)
        )

        # Update title to include subset name
        plt.title(f"{cat_names[idx]} ({subset_name}): {row['percentage
        keep_sname.append(f"waffle_{row['category']}_{subset_name}.png
        plt.tight_layout()

        # Save the figure as a PDF
        plt.savefig(
            f"waffle_{row['category']}_{subset_name}.pdf",
            bbox_inches='tight', format='pdf'
        )
        plt.close(fig)  # Close the figure to free memory

# Create plots for each subset
create_waffle_plots_for_subset(substantive_df, 'substantive')
create_waffle_plots_for_subset(methodological_df, 'methodological')
create_waffle_plots_for_subset(core_sosci_df, 'core')
create_waffle_plots_for_subset(peripheral_sosci_df, 'peripheral')
```

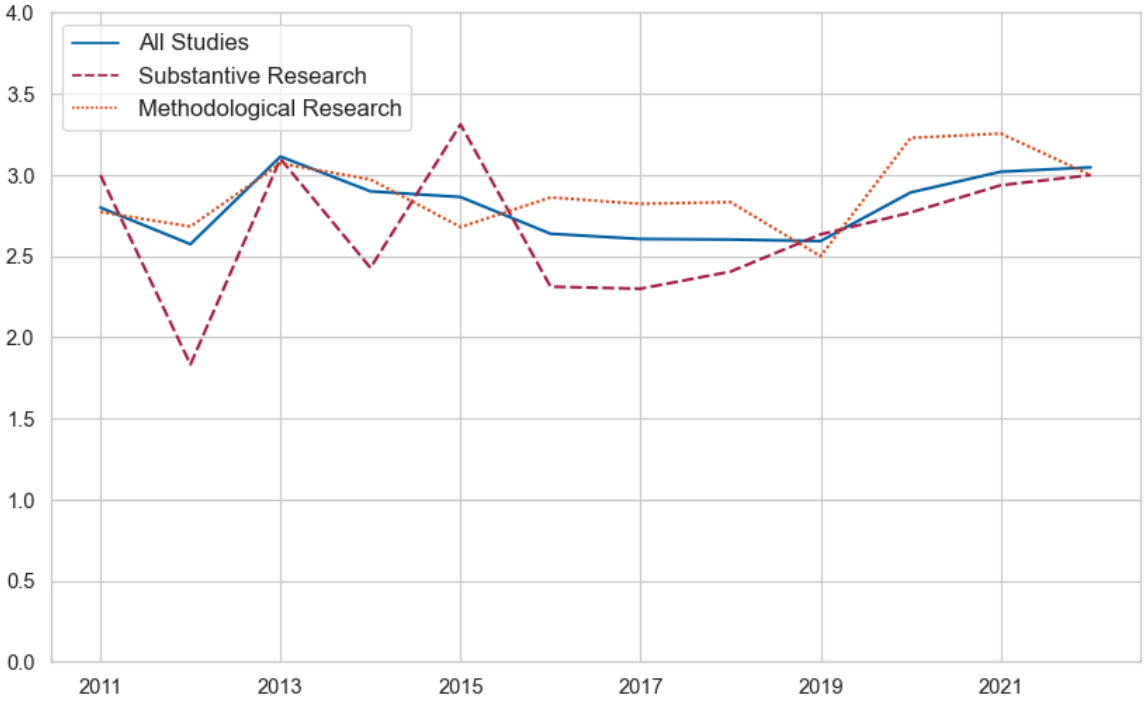
In [10]:
```
### Visualization of methods used over time

# Filter the DataFrame to include only studies published in 2011 or la
df_overtime = df[df['Year'] >= 2011]

grouped_means = df_overtime.groupby('Year').mean(numeric_only=True)
subst_val = df_overtime[df_overtime['Substantive'] == 1].groupby('Year
meth_val = df_overtime[df_overtime['Methodological'] == 1].groupby('Ye
core_val = df_overtime[df_overtime['Field'] == 'Social Science'].group
peripheral_val = df_overtime[df_overtime['Field'] != 'Social Science']

# Plot 1: Visualize the mean number of validation methods used per pap
df_year_val = pd.DataFrame({
    'Year': grouped_means.index,
    'All Studies': grouped_means['method_count'],
    'Substantive Research': subst_val,
    'Methodological Research': meth_val
}).fillna(0).reset_index(drop=True)

plt.figure(figsize=(10, 6))
vis = sns.lineplot(
    x='Year', y='value', hue='variable', style='variable',
    data=pd.melt(df_year_val, ['Year']), linewidth=1.5
)

handles, labels = vis.get_legend_handles_labels()
vis.legend(title=None, loc='upper left', handles=handles, labels=['All
vis.set_xticks(range(2011, 2023, 2))
vis.set_yticks([0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6])
vis.set(xlabel=None, ylabel=None)

plt.rcParams.update({'font.size': 25})
plt.savefig('meanovertime.pdf', bbox_inches='tight', format='pdf')

# Plot 2: Core vs. Peripheral Social Sciences over time

df_core_peripheral_val = pd.DataFrame({
    'Year': grouped_means.index,
    'All Studies': grouped_means['method_count'],
    'Core Social Sciences': core_val,
    'Peripheral Social Sciences': peripheral_val
}).fillna(0).reset_index(drop=True)

core_periphery_colors = [colors[0], colors[3], colors[4]]

plt.figure(figsize=(10, 6))
vis2 = sns.lineplot(
    x='Year', y='value', hue='variable', style='variable',
    data=pd.melt(df_core_peripheral_val, ['Year']), linewidth=1.5, pal
)

handles, labels = vis2.get_legend_handles_labels()
vis2.legend(title=None, loc='upper left', handles=handles, labels=['Al
vis2.set_xticks(range(2011, 2023, 2))
vis2.set_yticks([0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6])
vis2.set(xlabel=None, ylabel=None)

plt.rcParams.update({'font.size': 25})
```

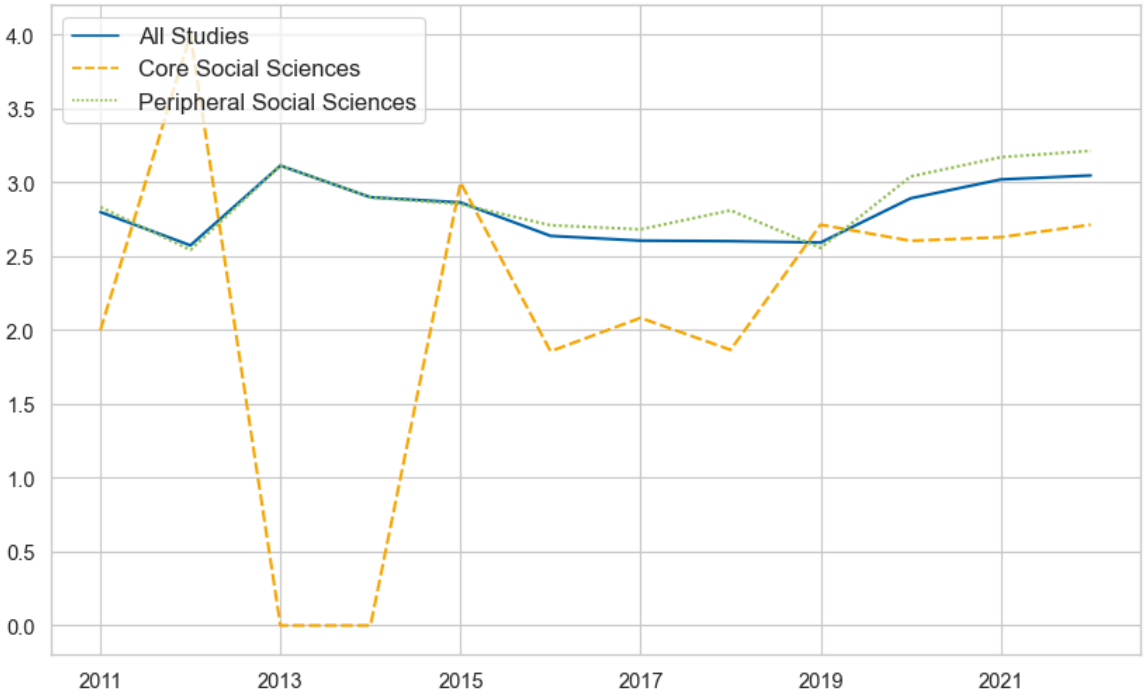
```
plt.savefig('core_vs_periphery_meanovertime.pdf', bbox_inches='tight',
```





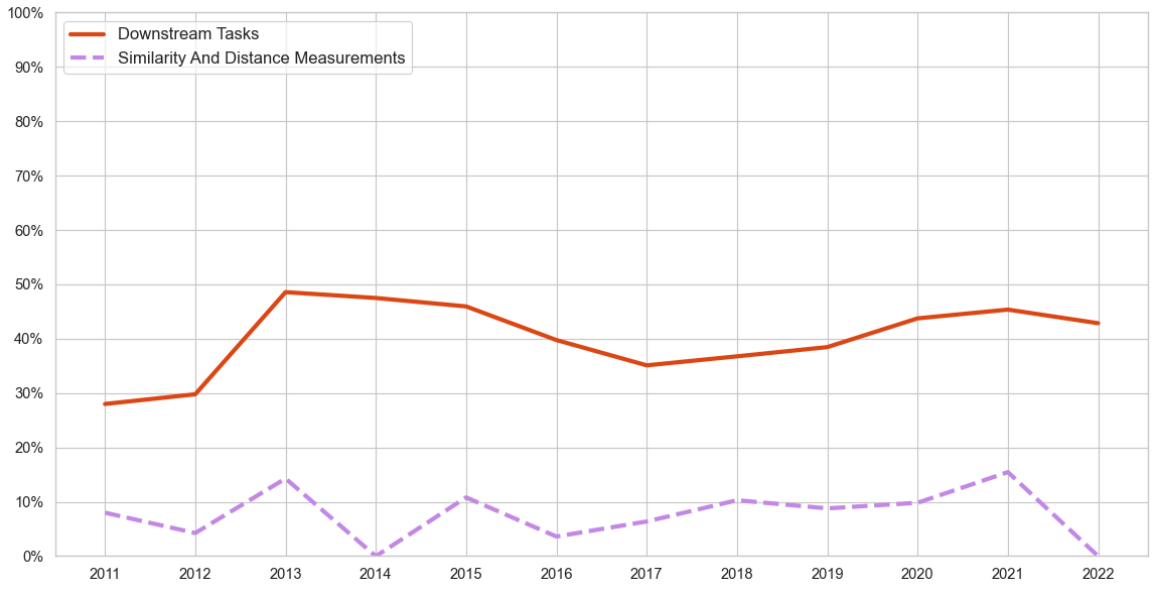
In [11]:
```
### Visualization of categories used over time


# Group by 'Year' and calculate means for different subsets (binary va
grouped_means_cat = df_overtime.groupby('Year').mean(numeric_only=True
subst_val_cat = df_overtime[df_overtime['Substantive'] == 1].groupby(
meth_val_cat = df_overtime[df_overtime['Methodological'] == 1].groupby
core_val_cat = df_overtime[df_overtime['Field'] == 'Social Science'].g
peripheral_val_cat = df_overtime[df_overtime['Field'] != 'Social Scier

# Plot 1: Visualize the mean number of binary validation categories us
df_year_val_cat = pd.DataFrame({
    'Year': grouped_means_cat.index,
    'All Studies': grouped_means_cat['binary_category_count'],
    'Substantive Research': subst_val_cat,
    'Methodological Research': meth_val_cat
}).fillna(0).reset_index(drop=True)

plt.figure(figsize=(10, 6))
vis_cat = sns.lineplot(
    x='Year', y='value', hue='variable', style='variable',
    data=pd.melt(df_year_val_cat, ['Year']), linewidth=1.5
)

handles, labels = vis_cat.get_legend_handles_labels()
vis_cat.legend(title=None, loc='upper left', handles=handles, labels=
vis_cat.set_xticks(range(2011, 2023, 2))
vis_cat.set_yticks([0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4])
vis_cat.set(xlabel=None, ylabel=None)

plt.rcParams.update({'font.size': 25})
plt.savefig('meanovertime_categories.pdf', bbox_inches='tight', format

# Plot 2: Core vs. Peripheral Social Sciences with All Studies (binary

df_core_peripheral_val_cat = pd.DataFrame({
    'Year': grouped_means_cat.index,
    'All Studies': grouped_means_cat['binary_category_count'],
    'Core Social Sciences': core_val_cat,
    'Peripheral Social Sciences': peripheral_val_cat
}).fillna(0).reset_index(drop=True)

core_periphery_colors_cat = [colors[0], colors[3], colors[4]]  # Using

plt.figure(figsize=(10, 6))
vis2_cat = sns.lineplot(
    x='Year', y='value', hue='variable', style='variable',
    data=pd.melt(df_core_peripheral_val_cat, ['Year']), linewidth=1.5,
)

handles, labels = vis2_cat.get_legend_handles_labels()
vis2_cat.legend(title=None, loc='upper left', handles=handles, labels=
vis2_cat.set_xticks(range(2011, 2023, 2))
vis2_cat.set_yticks([0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4])
vis2_cat.set(xlabel=None, ylabel=None)

plt.rcParams.update({'font.size': 25})
plt.savefig('meanovertime_categories_core_vs_periphery.pdf', bbox_inch
```

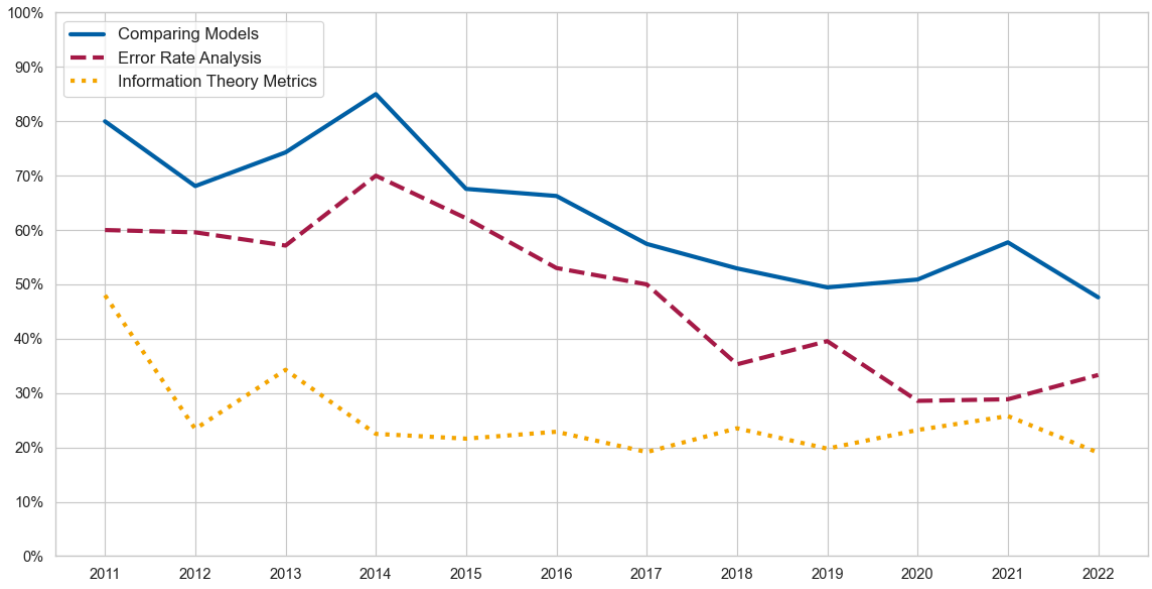

All Studies
Substantive Research
Methodological Research
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2011
2013
2015
2017
2019
2021

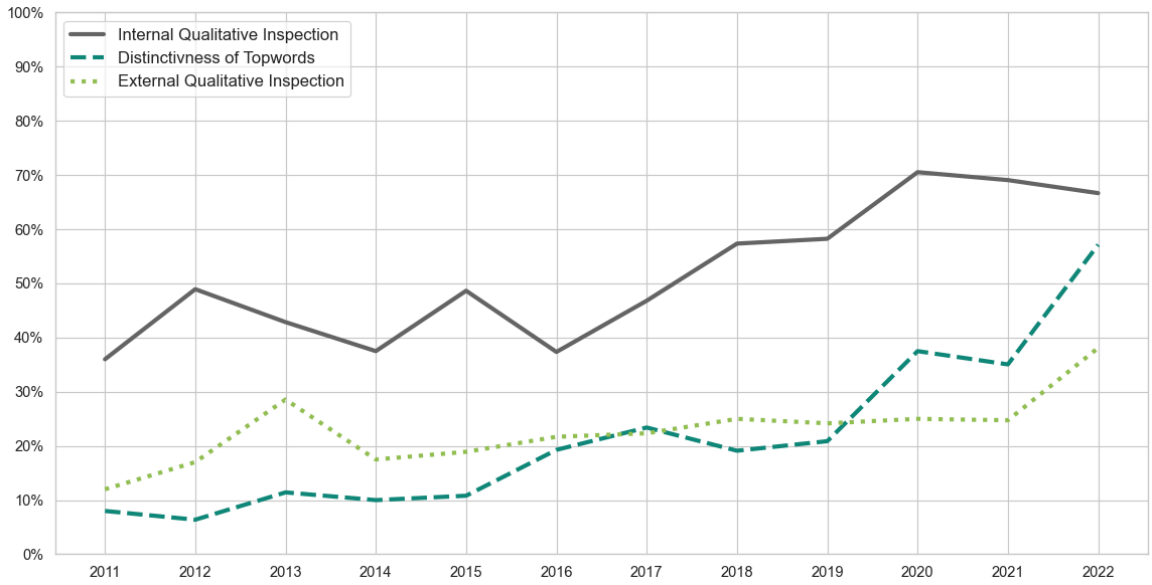

All Studies
Core Social Sciences
Peripheral Social Sciences
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2011
2013
2015
2017
2019
2021

In [12]:
```
# Set the corrected custom color palette
colors = ['#0063A6', '#666666', '#A71C49', '#DD4814', '#F6A800', '#94C
sns.set_palette(sns.color_palette(colors))

# Binary categories
binary_categories = [
    'binary_ComparingMethods', 'binary_Human_int', 'binary_ErrorRateAr
    'binary_Downstream', 'binary_Info', 'binary_Human_ext', 'binary_Di
]

# Prepare the data
df_overtime = df[df['Year'] > 2010]
n_year = df_overtime['Year'].value_counts().sort_index().reset_index(

# Group by Year and sum the binary categories
gr = df_overtime[['Year'] + binary_categories].groupby('Year').sum().r
gr['n_year'] = n_year['Year']

# Calculate percentages for each binary category
for val in binary_categories:
    gr[val] = (gr[val] * 100) / gr['n_year']

# Define a reusable plotting function
def plot_categories(x, y_data, labels, linestyles, colors, title, file
    fig, ax = plt.subplots()
    for y, label, linestyle, color in zip(y_data, labels, linestyles,
        plt.plot(x, y, label=label, linestyle=linestyle, color=color,
    ax.set_xticks(range(2011, 2023))
    ax.set_ylim([0, 100])
    ax.set_yticks(range(0, 101, 10))
    fmt = '%.0f%%'  # Percentage format for y-ticks
    ax.yaxis.set_major_formatter(mtick.FormatStrFormatter(fmt))
    plt.legend(prop={'size': 12}, loc='upper left')
    plt.savefig(filename, bbox_inches='tight', format='pdf')
    plt.show()
    plt.close(fig)

# X values (Years)
x = gr['Year']

# Plot 1: Consistent Categories
plot_categories(
    x=x,
    y_data=[gr['binary_Downstream'], gr['binary_Similar']],
    labels=['Downstream Tasks', 'Similarity And Distance Measurements'
    linestyles=['-', 'dashed'],
    colors=[colors[3], colors[7]],  # Corrected to use the new color p
    title='Over Time: Consistent Categories',
    filename='overtime_consistent.pdf'
)

# Plot 2: Decreasing Categories
plot_categories(
    x=x,
    y_data=[gr['binary_ComparingMethods'], gr['binary_ErrorRateAnalysi
    labels=['Comparing Models', 'Error Rate Analysis', 'Information Th
    linestyles=['-', 'dashed', 'dotted'],
    colors=[colors[0], colors[2], colors[4]],  # Corrected to use the
    title='Over Time: Decreasing Categories',
    filename='overtime_decrease.pdf'
)
```

```
# Plot 3: Increasing Categories
plot_categories(
    x=x,
    y_data=[gr['binary_Human_int'], gr['binary_Distinctivness'], gr['b
    labels=['Internal Qualitative Inspection', 'Distinctivness of Topv
    linestyles=['-', 'dashed', 'dotted'],
    colors=[colors[1], colors[6], colors[5]],  # Corrected to use the
    title='Over Time: Increasing Categories',
    filename='overtime_increase.pdf'
)
```



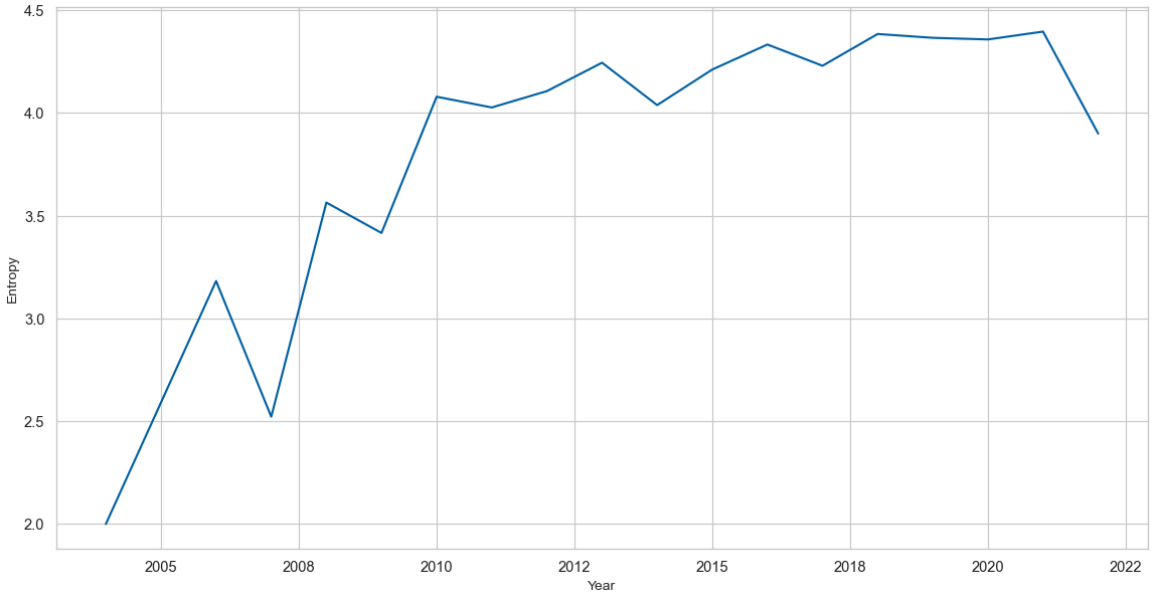




In [13]:
```
### Get information for heatmap
selected_df = df[binary_categories].copy()
selected_df['binary_category_count'] = df['binary_category_count']

co_occurrence_matrix = selected_df.T.dot(selected_df)
co_occurrence_matrix.to_csv('co_occurrence_matrix.csv')

solo_counts = {}
for column in binary_categories:
    solo_counts[column] = df.loc[(df[column] == 1) & (df['binary_categ
    print(f"The counter for {column} is: {solo_counts[column]}")
```

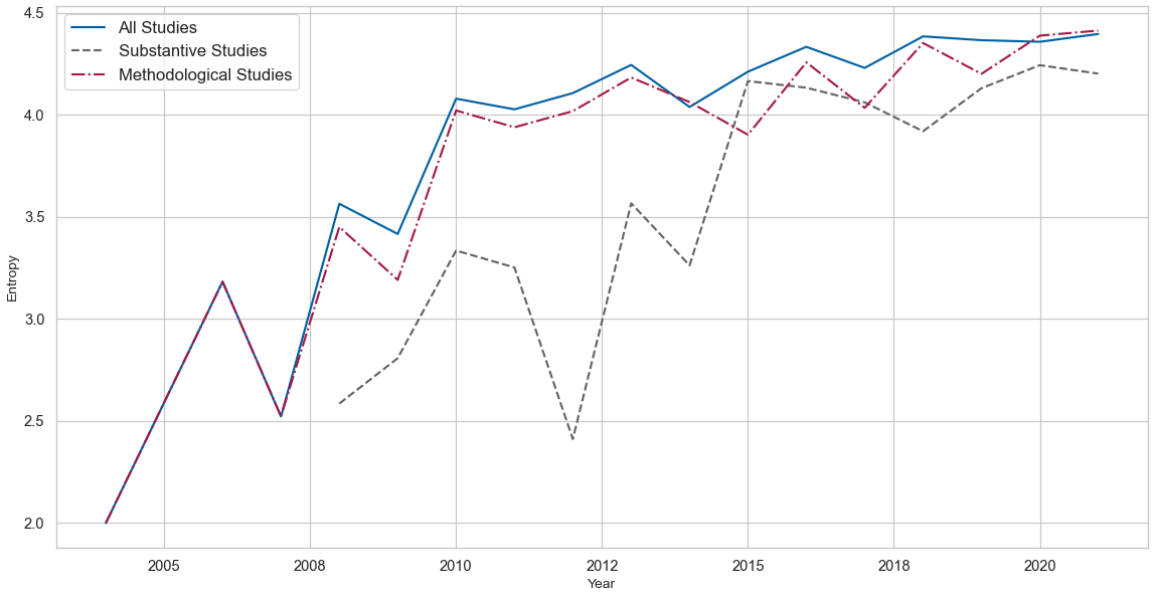
```
The counter for binary_ComparingMethods is: 18
The counter for binary_Human_int is: 39
The counter for binary_ErrorRateAnalysis is: 15
The counter for binary_Downstream is: 12
The counter for binary_Info is: 9
The counter for binary_Human_ext is: 6
The counter for binary_Distinctivness is: 1
The counter for binary_Similar is: 1
```

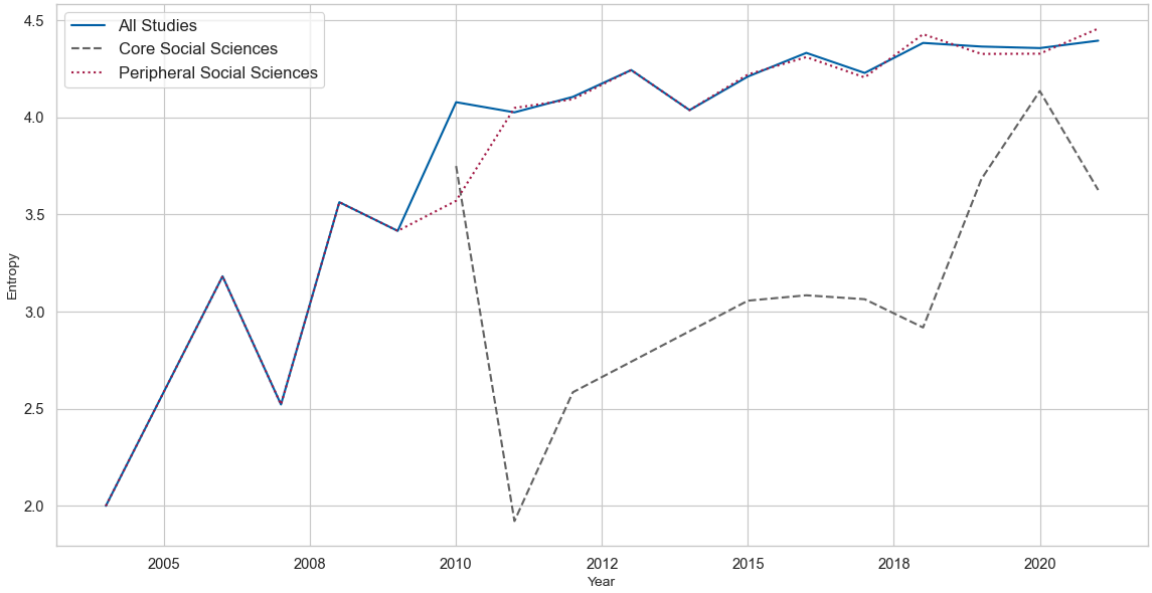
In [14]:
```
### Convergence Plots

# all methods all studies
def calculate_entropy_for_year(group, methods):
    """
    Calculates the entropy for a group of articles (grouped by year) b
    Ensures that missing methods are counted as zero occurrences.
    """
    # Create a frequency count for all methods, including missing ones
    method_counts = group[methods].sum().reindex(methods, fill_value=0

    # Normalize the counts to get probabilities
    probabilities = method_counts / method_counts.sum()

    # Calculate entropy (Shannon entropy)
    return entropy(probabilities, base=2)

# Group the dataset by year and apply the entropy calculation to each
entropy_per_year = df.groupby('Year').apply(lambda group: calculate_en

plt.figure(figsize=(14,7))
sns.lineplot(x=entropy_per_year.index, y=entropy_per_year.values, line
#plt.title('Entropy of Methods Applied in Articles Over Time')
plt.xlabel('Year')
plt.ylabel('Entropy')
plt.grid(True)
plt.gca().xaxis.set_major_formatter(mtick.StrMethodFormatter('{x:.0f}
plt.show()
plt.savefig('entropy_overtime.pdf', bbox_inches='tight', format='pdf')
```




```
<Figure size 1400x700 with 0 Axes>
```

In [15]:
```
# Calculate entropy for All Studies
df = df[df['Year'] != 2022]
entropy_all = df.groupby('Year').apply(lambda group: calculate_entropy

# Group 1: Substantive vs Methodological Studies
entropy_substantive = df[df['Substantive'] == 1].groupby('Year').apply
entropy_methodological = df[df['Methodological'] == 1].groupby('Year')

plt.figure(figsize=(14,7))
# Plot lines with different line styles
plt.plot(entropy_all.index, entropy_all.values, linestyle='-', linewid
plt.plot(entropy_substantive.index, entropy_substantive.values, linest
plt.plot(entropy_methodological.index, entropy_methodological.values,

#plt.title('Entropy of Methods Applied in All, Substantive, and Method
plt.xlabel('Year')
plt.ylabel('Entropy')
plt.grid(True)
plt.gca().xaxis.set_major_formatter(mtick.StrMethodFormatter('{x:.0f}
plt.legend(prop={'size': 12})
plt.savefig('entropy_all_substantive_vs_methodological.pdf', bbox_inch
plt.show()

# Group 2: Core vs Peripheral Social Sciences
entropy_core = df[df['Field'] == 'Social Science'].groupby('Year').app
entropy_peripheral = df[df['Field'] != 'Social Science'].groupby('Year


plt.figure(figsize=(14,7))
plt.plot(entropy_all.index, entropy_all.values, linestyle='-', linewid
plt.plot(entropy_core.index, entropy_core.values, linestyle='--', line
plt.plot(entropy_peripheral.index, entropy_peripheral.values, linesty

#plt.title('Entropy of Methods Applied in All, Core, and Peripheral Sc
plt.xlabel('Year')
plt.ylabel('Entropy')
plt.grid(True)
plt.gca().xaxis.set_major_formatter(mtick.StrMethodFormatter('{x:.0f}
plt.legend(prop={'size': 12})
plt.savefig('entropy_all_core_vs_peripheral.pdf', bbox_inches='tight',
plt.show()
```



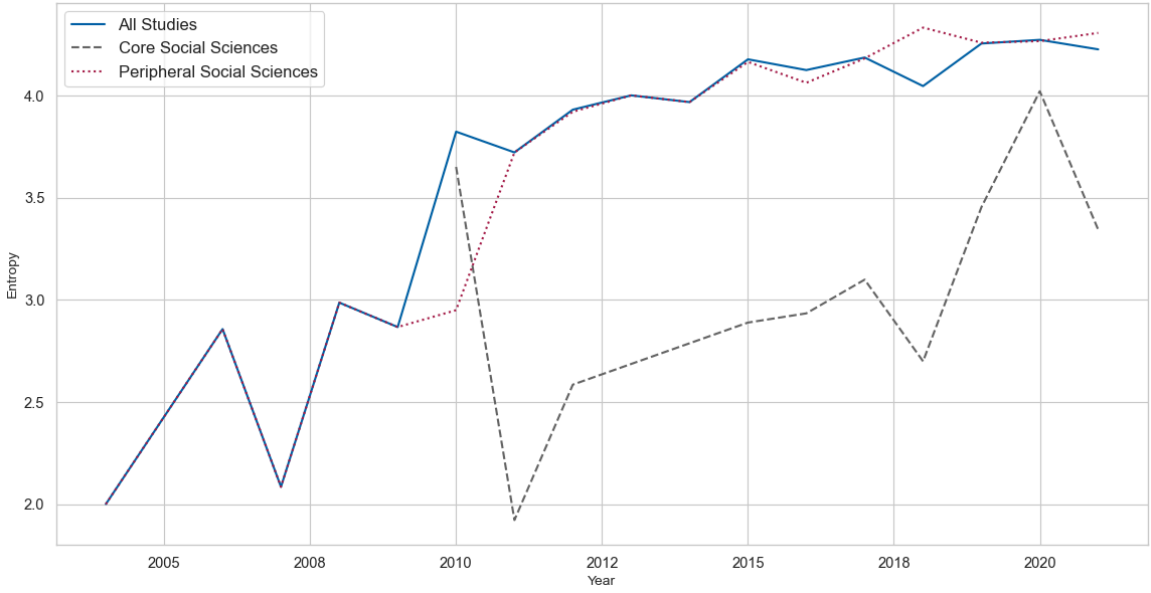


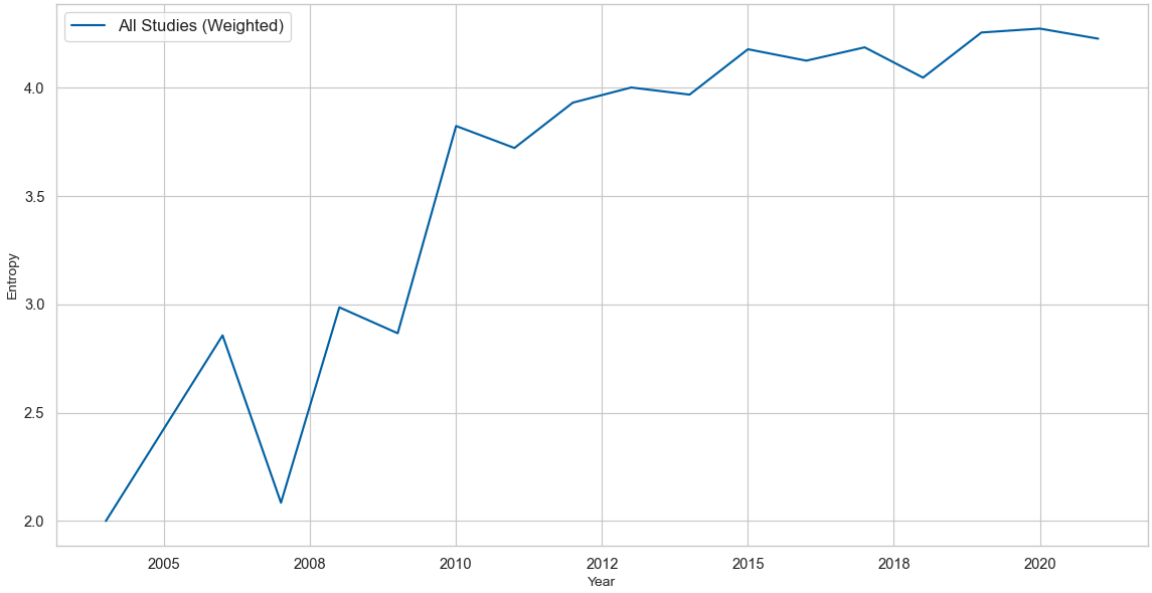
In [16]:
```
# Replace commas with dots and convert to numeric, using .loc for expl
df.loc[:, 'weight'] = df['weight'].str.replace(',', '.', regex=False)
df.loc[:, 'weight'] = pd.to_numeric(df['weight'], errors='coerce')
```

```
/var/folders/f2/wyd2lb6n62x4z40yhppm10s40000gp/T/ipykernel_6717/3846
634385.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy)
  df.loc[:, 'weight'] = df['weight'].str.replace(',', '.', regex=Fal
se)
/var/folders/f2/wyd2lb6n62x4z40yhppm10s40000gp/T/ipykernel_6717/3846
634385.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy)
  df.loc[:, 'weight'] = pd.to_numeric(df['weight'], errors='coerce')
/var/folders/f2/wyd2lb6n62x4z40yhppm10s40000gp/T/ipykernel_6717/3846
634385.py:3: DeprecationWarning: In a future version, `df.iloc[:, i]
= newvals` will attempt to set the values inplace instead of always
setting a new array. To retain the old behavior, use either `df[df.c
olumns[i]] = newvals` or, if columns are non-unique, `df.isetitem(i,
newvals)`
  df.loc[:, 'weight'] = pd.to_numeric(df['weight'], errors='coerce')
```

In [17]:
```
### Entropy with weights

# Define the entropy calculation function with weighting
def calculate_entropy_for_year(group, methods, weights_col):
    """
    Calculates the entropy for a group of articles (grouped by year) b
    incorporating the 'weight' column for weighted probabilities.
    """
    # Multiply methods by weights to get weighted counts
    weighted_methods = group[methods].multiply(group[weights_col], axi

    # Create a frequency count for all methods, including missing ones
    method_counts = weighted_methods.sum().reindex(methods, fill_value

    # Normalize the counts to get weighted probabilities
    probabilities = method_counts / method_counts.sum()

    # Calculate entropy (Shannon entropy)
    return entropy(probabilities, base=2)


# Calculate entropy for All Studies (including weights)
entropy_all = df.groupby('Year').apply(lambda group: calculate_entropy

# Group 1: Substantive vs Methodological Studies
entropy_substantive = df[df['Substantive'] == 1].groupby('Year').apply
entropy_methodological = df[df['Methodological'] == 1].groupby('Year')

plt.figure(figsize=(14,7))

# Plot lines with different line styles
plt.plot(entropy_all.index, entropy_all.values, linestyle='-', linewid
plt.plot(entropy_substantive.index, entropy_substantive.values, linest
plt.plot(entropy_methodological.index, entropy_methodological.values,

#plt.title('Entropy of Methods Applied in All, Substantive, and Method
plt.xlabel('Year')
plt.ylabel('Entropy')
plt.grid(True)
plt.gca().xaxis.set_major_formatter(mtick.StrMethodFormatter('{x:.0f}
plt.legend(prop={'size': 12})
plt.savefig('entropy_all_substantive_vs_methodological_weighted.pdf',
plt.show()

# Group 2: Core vs Peripheral Social Sciences
entropy_core = df[df['Field'] == 'Social Science'].groupby('Year').app
entropy_peripheral = df[df['Field'] != 'Social Science'].groupby('Year

plt.figure(figsize=(14,7))

# Plot lines with different line styles
plt.plot(entropy_all.index, entropy_all.values, linestyle='-', linewid
plt.plot(entropy_core.index, entropy_core.values, linestyle='--', line
plt.plot(entropy_peripheral.index, entropy_peripheral.values, linesty

#plt.title('Entropy of Methods Applied in All, Core, and Peripheral Sc
plt.xlabel('Year')
plt.ylabel('Entropy')
plt.grid(True)
plt.gca().xaxis.set_major_formatter(mtick.StrMethodFormatter('{x:.0f}
plt.legend(prop={'size': 12})
```

```
plt.savefig('entropy_all_core_vs_peripheral_weighted.pdf', bbox_inches
plt.show()

# Plot: All Studies Weighted Entropy over time
plt.figure(figsize=(14,7))

# Plot the weighted entropy for All Studies with solid line
plt.plot(entropy_all.index, entropy_all.values, linestyle='-', linewi

#plt.title('Entropy of Methods Applied in All Studies (Weighted) Over
plt.xlabel('Year')
plt.ylabel('Entropy')
plt.grid(True)
plt.gca().xaxis.set_major_formatter(mtick.StrMethodFormatter('{x:.0f}
plt.legend(prop={'size': 12})

plt.savefig('entropy_all_weighted.pdf', bbox_inches='tight', format='
plt.show()
```




All Studies (Weighted)
Entropy
4.0
3.5
3.0
2.5
2.0
2005
2008
2010
2012
2015
2018
2020
Year