Stats

April 6, 2025

[1]:
```python
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import os

# Global figure template settings for a scientific look
plotly_scientific_template = dict(
    layout=go.Layout(
        font=dict(
            family="Serif, Times New Roman, Georgia",
            size=16,
            color="black"
        ),
        title_font=dict(
            family="Serif, Times New Roman, Georgia",
            size=26,
            color="black"
        ),
        paper_bgcolor='white',
        plot_bgcolor='white',
        xaxis=dict(
            showgrid=True,
            gridcolor='lightgrey',
            zeroline=False,
            linecolor='black',
            ticks='outside',
            ticklen=5,
            mirror=True
        ),
        yaxis=dict(
            showgrid=True,
            gridcolor='lightgrey',
            zeroline=False,
            linecolor='black',
            ticks='outside',
```

```
            ticklen=5,
            mirror=True
        ),
        colorway=['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd'],  #␣
 ↪Scientific color palette
        legend=dict(
            bordercolor='black',
            borderwidth=1,
            bgcolor='white',
            font=dict(size=14)
        ),

    )
)

# Register the template globally
import plotly.io as pio
pio.templates['scientific'] = plotly_scientific_template
pio.templates.default = 'scientific'

own_path = os.getcwd()
#own_path = 'MAKai'

# get all sheet names
sheet_names = pd.ExcelFile(own_path+'\MasterarbeitenDatenAlleV2.xlsx').
 ↪sheet_names

color_palette = {1: 'rgb(248, 246, 245)', 2: 'rgb(228, 227, 221)', 3: 'rgb(77,␣
 ↪75, 70)', 4: 'rgb(134, 0, 71)', 5: 'rgb(179, 6, 44)', 6: 'rgb(277, 186,␣
 ↪15)', 7: 'rgb(115, 124, 69)',
                 8: 'rgb(0, 97, 143)', 9: 'rgb(173, 59, 118)', 10: 'rgb(201,␣
 ↪98, 21)', 11: 'rgb(247, 217, 38)', 12: 'rgb(165, 171, 82)', 13: 'rgb(72,␣
 ↪169, 218)'}


dfs = []
for sheet_name in sheet_names:
    if sheet_name == 'Codierung':
        codes = pd.read_excel(own_path+'\MasterarbeitenDatenAlleV2.xlsx',␣
 ↪sheet_name=sheet_name)
    else:
        df = pd.read_excel(own_path+'\MasterarbeitenDatenAlleV2.xlsx',␣
 ↪sheet_name=sheet_name)
        df['DT'] = sheet_name
        dfs.append(df)
```

```python
code_color = {
    0: 0,
    1: -1,
    2: -1,
    3: 0,
    4: 1,
    5: 0,
}

codes['code_color'] = codes['Codierung'].map(code_color)

# concatenate all dataframes
df = pd.concat(dfs, ignore_index=True)
df = df.set_index(['DT', 'Land']).melt(ignore_index=False).set_index('variable', append=True)
df = df.pivot_table(index=['Land', 'variable'], columns='DT', values='value')
df.index.names = ['Land', 'Jahr']
codes = codes.set_index('Land')
df = df.merge(codes, left_index=True, right_index=True).drop('Typ', axis=1)
df = df.set_index('Codierung', append=True).sort_index()

name_map = {
    'Kosten': ['Anteil  BIP Private', 'Anteil BIP Public ', 'Gesundheitsausgaben pro Kopf', 'Out of Pocket'],
    'Zugänglichkeit': ['Artzbesuche (pro Kopf)', 'Belegungsrate Akutpflegebet', 'Hospital beds', 'Practising doctors', 'Professional nurses'],
    'Qualität': ['Krebs M', 'Krebs W', 'Schlaganfall M', 'Schlaganfall W', 'Sterblichkeit ab 65 M', 'Sterblichkeit ab 65 W', 'Verhinderbare Sterblichkeitsrat']
}

score_map = {
    'Anteil  BIP Private': 1,
    'Anteil BIP Public ': 1,
    'Artzbesuche (pro Kopf)': 1,
    'Belegungsrate Akutpflegebet': -1,
    'Gesundheitsausgaben pro Kopf': 1,
    'Hospital beds': 1,
    'Krebs M': -1,
    'Krebs W': -1,
    'Out of Pocket': 1,
    'Practising doctors': 1,
    'Professional nurses': 1,
    'Schlaganfall M': -1,
    'Schlaganfall W': -1,
    'Sterblichkeit ab 65 M': 1,
    'Sterblichkeit ab 65 W': 1,
```

```
    'Verhinderbare Sterblichkeitsrat': -1
    }

# fill missing values
df = df.groupby(['Land']).ffill()

# z normalize over all years per variable
df = (df - df.mean()) / df.std()

# add columns for each category
for key in name_map.keys():
    tmp_sum = []
    for col in name_map[key]:
        tmp_sum.append(df[col] * score_map[col])

    tmp_sum = pd.concat(tmp_sum, axis=1).mean(axis=1)

    df[key] = tmp_sum

# filter year > 2010
df_f = df[df.index.get_level_values('Jahr') > 2010]
```

[2]:
```
corr_data = df_f.copy()

corr_data['Krebs'] = (corr_data['Krebs M'] + corr_data['Krebs W']) / 2
corr_data['Schlaganfall'] = (corr_data['Schlaganfall M'] +␣
 ↪corr_data['Schlaganfall W']) / 2
corr_data['Sterblichkeit'] = (corr_data['Sterblichkeit ab 65 M'] +␣
 ↪corr_data['Sterblichkeit ab 65 W']) / 2
corr_data['Practising Medical Staff'] = (corr_data['Practising doctors'] +␣
 ↪corr_data['Professional nurses']) / 2

corr_data = corr_data.drop(['Krebs M', 'Krebs W', 'Schlaganfall M',␣
 ↪'Schlaganfall W', 'Sterblichkeit ab 65 M', 'Sterblichkeit ab 65 W',␣
 ↪'Practising doctors', 'Professional nurses'], axis=1)
corr_data = corr_data[['Anteil  BIP Private', 'Anteil BIP Public ',␣
 ↪'Gesundheitsausgaben pro Kopf', 'Out of Pocket', 'Artzbesuche (pro Kopf)',␣
 ↪'Belegungsrate Akutpflegebet', 'Hospital beds', 'Practising Medical Staff',␣
 ↪'Krebs', 'Schlaganfall', 'Sterblichkeit', 'Verhinderbare Sterblichkeitsrat']]

colorscale = [[0, color_palette[13]], [0.5, color_palette[1]], [1,␣
 ↪color_palette[5]]]
```

[3]:
```
fig = px.imshow(
    corr_data.corr(),
    title='Korrelation der Variablen',
    labels=dict(x='Variable', y='Variable', color='Korrelation'),
```

```
    color_continuous_scale=colorscale
)

fig.update_layout(
    margin=dict(t=40, b=250, l=250, r=100),
    width=800, height=800
)

# Remove axis labels and rotate x-axis tick labels
fig.update_xaxes(title_text="", tickangle=90)
fig.update_yaxes(title_text="")

fig.show()
```

[4]:
```
# count data points per year
df.groupby('Jahr').count().mean(axis=1)
```

[4]:
```
Jahr
2006     1.25
2007    10.40
2008    24.20
2009    18.75
2010    28.95
2011    29.90
2012    30.65
2013    32.00
2014    33.15
2015    33.35
2016    34.10
2017    34.75
2018    34.20
2019    34.30
2020    35.45
2021    32.70
2022    35.90
dtype: float64
```

[5]:
```
vari = 'Kosten'
plot_bar = df_f[list(name_map.keys())].reset_index()
color_seq = [color_palette[3], color_palette[4], color_palette[5],␣
 ↪color_palette[7], color_palette[12], color_palette[13]]

fig = go.Figure()
fig = make_subplots(
    rows=2, cols=1,
)
tmp_codes = np.sort(plot_bar['Codierung'].unique())
```

```python
for code in tmp_codes:
    plot_bar_tmp = plot_bar[plot_bar['Codierung'] == code]
    x = plot_bar_tmp['Jahr']
    fig.add_trace(go.Box(
        x=x,
        y=plot_bar_tmp[vari],
        name=str(code),
        boxpoints=False,
        marker_color=color_seq[code]
    ),
        row=2, col=1
    )
    plot_line_tmp = plot_bar_tmp[['Jahr', vari]].groupby('Jahr').mean()
    fig.add_trace(go.Scatter(
        x=plot_line_tmp.index,
        y=plot_line_tmp[vari],
        name=str(code),
        marker_color=color_seq[code],
        mode='lines',
    ),
        row=1, col=1
    )

fig.update_xaxes(title_text="Jahr", row=1, col=1)
fig.update_yaxes(title_text=vari, row=1, col=1)
fig.update_xaxes(title_text="Jahr", row=2, col=1)
fig.update_yaxes(title_text=vari, row=2, col=1)

fig.update_traces(name='Typ 0', selector=dict(name="0"))
fig.update_traces(name='Typ 1', selector=dict(name="1"))
fig.update_traces(name='Typ 2', selector=dict(name="2"))
fig.update_traces(name='Typ 3', selector=dict(name="3"))
fig.update_traces(name='Typ 4', selector=dict(name="4"))
fig.update_traces(name='Typ 5', selector=dict(name="5"))

fig.update_layout(
    boxmode='group',
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=-0.3,
        xanchor="center",
        x=0.5
    ),
    title=f'{vari} nach Typ und Jahr',
    height=800,
```

```
    width=1000
)
fig.show()
```

```
[6]: vari = 'Qualität'
plot_bar = df_f[list(name_map.keys())].reset_index()
color_seq = [color_palette[3], color_palette[4], color_palette[5],␣
 ↪color_palette[7], color_palette[12], color_palette[13]]

fig = go.Figure()
fig = make_subplots(
    rows=2, cols=1,
)
tmp_codes = np.sort(plot_bar['Codierung'].unique())

for code in tmp_codes:
    plot_bar_tmp = plot_bar[plot_bar['Codierung'] == code]
    x = plot_bar_tmp['Jahr']
    fig.add_trace(go.Box(
        x=x,
        y=plot_bar_tmp[vari],
        name=str(code),
        boxpoints=False,
        marker_color=color_seq[code]
    ),
        row=2, col=1
    )
    plot_line_tmp = plot_bar_tmp[['Jahr', vari]].groupby('Jahr').mean()
    fig.add_trace(go.Scatter(
        x=plot_line_tmp.index,
        y=plot_line_tmp[vari],
        name=str(code),
        marker_color=color_seq[code],
        mode='lines',
    ),
        row=1, col=1
    )

fig.update_xaxes(title_text="Jahr", row=1, col=1)
fig.update_yaxes(title_text=vari, row=1, col=1)
fig.update_xaxes(title_text="Jahr", row=2, col=1)
fig.update_yaxes(title_text=vari, row=2, col=1)

fig.update_traces(name='Typ 0', selector=dict(name="0"))
fig.update_traces(name='Typ 1', selector=dict(name="1"))
fig.update_traces(name='Typ 2', selector=dict(name="2"))
fig.update_traces(name='Typ 3', selector=dict(name="3"))
```

```
fig.update_traces(name='Typ 4', selector=dict(name="4"))
fig.update_traces(name='Typ 5', selector=dict(name="5"))

fig.update_layout(
    boxmode='group',
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=-0.3,
        xanchor="center",
        x=0.5
    ),
    title=f'{vari} nach Typ und Jahr',
    height=800,
    width=1000
)
fig.show()
```

[7]:
```
vari = 'Zugänglichkeit'
plot_bar = df_f[list(name_map.keys())].reset_index()
color_seq = [color_palette[3], color_palette[4], color_palette[5],␣
 ↪color_palette[7], color_palette[12], color_palette[13]]

fig = go.Figure()
fig = make_subplots(
    rows=2, cols=1,
)
tmp_codes = np.sort(plot_bar['Codierung'].unique())

for code in tmp_codes:
    plot_bar_tmp = plot_bar[plot_bar['Codierung'] == code]
    x = plot_bar_tmp['Jahr']
    fig.add_trace(go.Box(
        x=x,
        y=plot_bar_tmp[vari],
        name=str(code),
        boxpoints=False,
        marker_color=color_seq[code]
    ),
        row=2, col=1
    )
    plot_line_tmp = plot_bar_tmp[['Jahr', vari]].groupby('Jahr').mean()
    fig.add_trace(go.Scatter(
        x=plot_line_tmp.index,
        y=plot_line_tmp[vari],
        name=str(code),
        marker_color=color_seq[code],
```

```
        mode='lines',
    ),
        row=1, col=1
    )

fig.update_xaxes(title_text="Jahr", row=1, col=1)
fig.update_yaxes(title_text=vari, row=1, col=1)
fig.update_xaxes(title_text="Jahr", row=2, col=1)
fig.update_yaxes(title_text=vari, row=2, col=1)

fig.update_traces(name='Typ 0', selector=dict(name="0"))
fig.update_traces(name='Typ 1', selector=dict(name="1"))
fig.update_traces(name='Typ 2', selector=dict(name="2"))
fig.update_traces(name='Typ 3', selector=dict(name="3"))
fig.update_traces(name='Typ 4', selector=dict(name="4"))
fig.update_traces(name='Typ 5', selector=dict(name="5"))

fig.update_layout(
    boxmode='group',
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=-0.3,
        xanchor="center",
        x=0.5
    ),
    title=f'{vari} nach Typ und Jahr',
    height=800,
    width=1000
)
fig.show()
```

[8]:
```
plot_bar_tmp = plot_bar[plot_bar['Codierung'] == 0]
plot_bar_line = plot_bar_tmp[['Jahr', 'Kosten']].groupby('Jahr').mean()
plt_data = df_f.groupby(['Jahr', 'Codierung']).mean()
err_data = df_f.groupby(['Jahr', 'Codierung']).std()
plt_data_lines = pd.concat([plt_data[list(name_map.keys())],␣
 ↪err_data[list(name_map.keys())].add_prefix('err')], axis=1).reset_index()
plt_data3D = plt_data[np.logical_and(plt_data.index.
 ↪get_level_values('Codierung') != 0,plt_data.index.
 ↪get_level_values('Codierung')!= 3)].reset_index()
```

[9]:
```
from plotly.subplots import make_subplots

# Create subplot layout (2x2)
fig = make_subplots(
    rows=2, cols=2,
```

```
    specs=[[{"type": "scatter"}, {"type": "scatter"}],
           [{"type": "scatter"}, {"type": "scatter3d"}]],  # 3D in bottom right
    horizontal_spacing = 0.1,
    vertical_spacing = 0.2,
    subplot_titles=[
        "Kosten vs Zugänglichkeit",
        "Kosten vs Qualität",
        "Zugänglichkeit vs Qualität",
        "Kosten vs Zugänglichkeit vs Qualität"
    ]
)

colorlist_3D = []
for i in plt_data3D["Codierung"]:
    colorlist_3D.append(color_seq[i])

for co in [1,2,4,5]:
    # 2D Scatter plots
    fig.add_trace(
        go.Scatter(
            x=plt_data3D[plt_data3D["Codierung"]==co]["Kosten"],
            y=plt_data3D[plt_data3D["Codierung"]==co]["Zugänglichkeit"],
            mode="markers",
            marker=dict(color=color_seq[co]),
            name=f"Typ {co}"
        ),
        row=1, col=1
    )

    fig.add_trace(
        go.Scatter(
            x=plt_data3D[plt_data3D["Codierung"]==co]["Kosten"],
            y=plt_data3D[plt_data3D["Codierung"]==co]["Qualität"],
            mode="markers",
            marker=dict(color=color_seq[co]),
            name=f"Typ {co}",
            showlegend=False
        ),
        row=1, col=2
    )

    fig.add_trace(
        go.Scatter(
            x=plt_data3D[plt_data3D["Codierung"]==co]["Zugänglichkeit"],
            y=plt_data3D[plt_data3D["Codierung"]==co]["Qualität"],
            mode="markers",
            marker=dict(color=color_seq[co]),
```

```
            name=f"Typ {co}",
            showlegend=False
        ),
        row=2, col=1
    )

# 3D Scatter plot
fig.add_trace(
    go.Scatter3d(
        x=plt_data3D["Kosten"],
        y=plt_data3D["Zugänglichkeit"],
        z=plt_data3D["Qualität"],
        mode="markers",
        marker=dict(size=4, color=colorlist_3D),
        name="Kosten vs Zugänglichkeit vs Qualität",
        showlegend=False
    ),
    row=2, col=2
)

fig.update_xaxes(title_text="Kosten", row=1, col=1)
fig.update_yaxes(title_text="Zugänglichkeit", row=1, col=1)

fig.update_xaxes(title_text="Kosten", row=1, col=2)
fig.update_yaxes(title_text="Qualität", row=1, col=2)

fig.update_xaxes(title_text="Zugänglichkeit", row=2, col=1)
fig.update_yaxes(title_text="Qualität", row=2, col=1)

fig.update_layout(annotations=[dict(font=dict(size=20))])

# Update layout
fig.update_layout(
    title="Gesundheitssystemmaße nach Typ",
    width=900, height=900,
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=-0.2,
        xanchor="center",
        x=0.5
    ),
    margin=dict(t=90, b=50)
)

# Show the plot
fig.show()
```

[10]:

```
fig = go.Figure(data=[
    go.Scatter3d(
        x=plt_data3D["Kosten"],
        y=plt_data3D["Zugänglichkeit"],
        z=plt_data3D["Qualität"],
        mode="markers",
        marker=dict(size=4, color=colorlist_3D),
        name="Kosten vs Zugänglichkeit vs Qualität",
        showlegend=False
    )])

# Adjust figure size
fig.update_layout(
    width=600,  # Increase width
    height=600,  # Increase height
    scene=dict(
        xaxis_title="Kosten",
        yaxis_title="Zugänglichkeit",
        zaxis_title="Qualität"
    )
)

# Show the plot
fig.show()
```

[11]:

```
# use statsmodels to do a linear regression with fixed effects for year and␣
 ↪group
import statsmodels.api as sm
import statsmodels.formula.api as smf

df_r = df_f.reset_index()
df_r['Jahr'] = df_r['Jahr'].astype('category')
df_r['Land'] = df_r['Land'].astype('category')
# drop codings 0 and 3
df_r = df_r[np.logical_and(df_r['Codierung'] != 0, df_r['Codierung'] != 3)]
df_r['Codierung'] = df_r['Codierung'].astype('category')
df_r['system'] = df_r['Codierung'].map(code_color)

# regression for quality_i,j = a + b * Kosten_i,j + c * Zugänglichkeit_i,j + d␣
 ↪* Jahr_i + e * Codierung_i
model = smf.ols('Qualität ~ Kosten + Zugänglichkeit + C(Jahr) + C(Codierung)',␣
 ↪data=df_r).fit()

print(model.summary())

with open("regression_summary.tex", "w") as f:
    f.write(model.summary().as_latex())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Qualität   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.580
Method:                 Least Squares   F-statistic:                     23.73
Date:                Sun, 06 Apr 2025   Prob (F-statistic):           3.92e-41
Time:                        00:01:46   Log-Likelihood:                -97.091
No. Observations:                 264   AIC:                             228.2
Df Residuals:                     247   BIC:                             289.0
Df Model:                          16
Covariance Type:            nonrobust
================================================================================
=====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept            0.0171      0.083      0.205      0.838      -0.147
0.181
C(Jahr)[T.2012]      0.0486      0.109      0.446      0.656      -0.166
0.263
C(Jahr)[T.2013]      0.2210      0.109      2.027      0.044       0.006
0.436
C(Jahr)[T.2014]      0.3003      0.109      2.756      0.006       0.086
0.515
C(Jahr)[T.2015]      0.1989      0.109      1.825      0.069      -0.016
0.413
C(Jahr)[T.2016]      0.1645      0.109      1.506      0.133      -0.051
0.380
C(Jahr)[T.2017]      0.2865      0.109      2.623      0.009       0.071
0.502
C(Jahr)[T.2018]      0.2426      0.110      2.215      0.028       0.027
0.458
C(Jahr)[T.2019]      0.3714      0.109      3.393      0.001       0.156
0.587
C(Jahr)[T.2020]      0.2932      0.110      2.676      0.008       0.077
0.509
C(Jahr)[T.2021]      0.3266      0.110      2.982      0.003       0.111
0.542
C(Jahr)[T.2022]      0.3407      0.109      3.115      0.002       0.125
0.556
C(Codierung)[T.2]   -0.0308      0.084     -0.366      0.715      -0.197
0.135
C(Codierung)[T.4]    0.2026      0.076      2.662      0.008       0.053
0.352
C(Codierung)[T.5]   -0.3369      0.063     -5.381      0.000      -0.460
-0.214
Kosten               0.4482      0.048      9.274      0.000       0.353
```

```
0.543
Zugänglichkeit        -0.3513      0.063     -5.549      0.000      -0.476
-0.227
==============================================================================
Omnibus:                        4.713   Durbin-Watson:                   0.245
Prob(Omnibus):                  0.095   Jarque-Bera (JB):                4.599
Skew:                          -0.237   Prob(JB):                        0.100
Kurtosis:                       3.441   Cond. No.                         14.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[12]:

```
# regression for quality_i,j = a + b * Kosten_i,j + c * Zugänglichkeit_i,j + d␣
↪* Jahr_i + e * Codierung_i
model = smf.ols('Qualität ~ Kosten + Zugänglichkeit + C(Jahr) + C(Codierung)',␣
↪data=df_r).fit()

print(model.summary())

with open("regression_summary.tex", "w") as f:
    f.write(model.summary().as_latex())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Qualität   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.580
Method:                 Least Squares   F-statistic:                     23.73
Date:                Sun, 06 Apr 2025   Prob (F-statistic):           3.92e-41
Time:                        00:01:46   Log-Likelihood:                -97.091
No. Observations:                 264   AIC:                             228.2
Df Residuals:                     247   BIC:                             289.0
Df Model:                          16
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept             0.0171      0.083      0.205      0.838      -0.147
0.181
C(Jahr)[T.2012]       0.0486      0.109      0.446      0.656      -0.166
0.263
C(Jahr)[T.2013]       0.2210      0.109      2.027      0.044       0.006
0.436
C(Jahr)[T.2014]       0.3003      0.109      2.756      0.006       0.086
```

```
0.515
C(Jahr)[T.2015]        0.1989      0.109      1.825      0.069     -0.016
0.413
C(Jahr)[T.2016]        0.1645      0.109      1.506      0.133     -0.051
0.380
C(Jahr)[T.2017]        0.2865      0.109      2.623      0.009      0.071
0.502
C(Jahr)[T.2018]        0.2426      0.110      2.215      0.028      0.027
0.458
C(Jahr)[T.2019]        0.3714      0.109      3.393      0.001      0.156
0.587
C(Jahr)[T.2020]        0.2932      0.110      2.676      0.008      0.077
0.509
C(Jahr)[T.2021]        0.3266      0.110      2.982      0.003      0.111
0.542
C(Jahr)[T.2022]        0.3407      0.109      3.115      0.002      0.125
0.556
C(Codierung)[T.2]     -0.0308      0.084     -0.366      0.715     -0.197
0.135
C(Codierung)[T.4]      0.2026      0.076      2.662      0.008      0.053
0.352
C(Codierung)[T.5]     -0.3369      0.063     -5.381      0.000     -0.460
-0.214
Kosten                 0.4482      0.048      9.274      0.000      0.353
0.543
Zugänglichkeit        -0.3513      0.063     -5.549      0.000     -0.476
-0.227
==============================================================================
Omnibus:                        4.713   Durbin-Watson:                   0.245
Prob(Omnibus):                  0.095   Jarque-Bera (JB):                4.599
Skew:                          -0.237   Prob(JB):                        0.100
Kurtosis:                       3.441   Cond. No.                         14.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[13]:
```python
# regression for quality_i,j = a + b * Kosten_i,j + c * Zugänglichkeit_i,j + d␣
 ↪* Jahr_i + e * Codierung_i
model = smf.ols('Zugänglichkeit ~ Qualität + Kosten + C(Jahr) + C(Codierung)',␣
 ↪data=df_r).fit()

print(model.summary())
with open("regression_summary.tex", "w") as f:
    f.write(model.summary().as_latex())
```

```
                            OLS Regression Results
```

```
==============================================================================
Dep. Variable:          Zugänglichkeit   R-squared:                       0.475
Model:                             OLS   Adj. R-squared:                  0.441
Method:                  Least Squares   F-statistic:                     13.95
Date:                 Sun, 06 Apr 2025   Prob (F-statistic):           1.88e-26
Time:                         00:01:46   Log-Likelihood:                -82.906
No. Observations:                  264   AIC:                             199.8
Df Residuals:                      247   BIC:                             260.6
Df Model:                           16
Covariance Type:             nonrobust
================================================================================
=====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept           -0.0924      0.079     -1.174      0.241      -0.247
0.063
C(Jahr)[T.2012]     -0.0221      0.103     -0.214      0.831      -0.226
0.181
C(Jahr)[T.2013]      0.0355      0.104      0.341      0.734      -0.170
0.241
C(Jahr)[T.2014]      0.0907      0.105      0.867      0.387      -0.115
0.297
C(Jahr)[T.2015]      0.0660      0.104      0.635      0.526      -0.139
0.271
C(Jahr)[T.2016]     -0.0534      0.104     -0.514      0.608      -0.258
0.151
C(Jahr)[T.2017]     -0.0154      0.105     -0.147      0.884      -0.222
0.191
C(Jahr)[T.2018]     -0.0581      0.105     -0.555      0.580      -0.264
0.148
C(Jahr)[T.2019]     -0.0172      0.106     -0.162      0.871      -0.226
0.192
C(Jahr)[T.2020]      0.0500      0.105      0.475      0.635      -0.157
0.257
C(Jahr)[T.2021]      0.1310      0.105      1.244      0.215      -0.076
0.338
C(Jahr)[T.2022]      0.0662      0.106      0.627      0.531      -0.142
0.274
C(Codierung)[T.2]   -0.2192      0.079     -2.789      0.006      -0.374
-0.064
C(Codierung)[T.4]    0.6079      0.062      9.793      0.000       0.486
0.730
C(Codierung)[T.5]    0.2760      0.060      4.585      0.000       0.157
0.395
Qualität            -0.3155      0.057     -5.549      0.000      -0.427
-0.204
```

```
Kosten                  0.2222      0.051      4.334      0.000       0.121
0.323
==============================================================================
Omnibus:                        5.665   Durbin-Watson:                   0.390
Prob(Omnibus):                  0.059   Jarque-Bera (JB):                3.376
Skew:                          -0.021   Prob(JB):                        0.185
Kurtosis:                       2.448   Cond. No.                         14.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[14]:
```python
# regression for quality_i,j = a + b * Kosten_i,j + c * Zugänglichkeit_i,j + d␣
 ↪* Jahr_i + e * Codierung_i
model = smf.ols('Kosten ~ Zugänglichkeit + Qualität + C(Jahr) + C(Codierung)',␣
 ↪data=df_r).fit()

print(model.summary())
with open("regression_summary.tex", "w") as f:
    f.write(model.summary().as_latex())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Kosten   R-squared:                       0.473
Model:                            OLS   Adj. R-squared:                  0.439
Method:                 Least Squares   F-statistic:                     13.86
Date:                Sun, 06 Apr 2025   Prob (F-statistic):           2.63e-26
Time:                        00:01:46   Log-Likelihood:                -130.23
No. Observations:                 264   AIC:                             294.5
Df Residuals:                     247   BIC:                             355.2
Df Model:                          16
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept              0.1352      0.094      1.439      0.151      -0.050
0.320
C(Jahr)[T.2012]       -0.0075      0.124     -0.061      0.952      -0.251
0.236
C(Jahr)[T.2013]       -0.1072      0.124     -0.862      0.390      -0.352
0.138
C(Jahr)[T.2014]       -0.1640      0.125     -1.312      0.191      -0.410
0.082
C(Jahr)[T.2015]       -0.0974      0.124     -0.784      0.434      -0.342
```

```
0.147
C(Jahr)[T.2016]      -0.0634      0.124     -0.510      0.610      -0.308
0.181
C(Jahr)[T.2017]      -0.1291      0.125     -1.030      0.304      -0.376
0.118
C(Jahr)[T.2018]      -0.0327      0.125     -0.261      0.795      -0.280
0.214
C(Jahr)[T.2019]      -0.1334      0.127     -1.053      0.293      -0.383
0.116
C(Jahr)[T.2020]       0.0036      0.126      0.029      0.977      -0.245
0.252
C(Jahr)[T.2021]      -0.0248      0.126     -0.196      0.845      -0.274
0.224
C(Jahr)[T.2022]      -0.0537      0.126     -0.425      0.671      -0.303
0.195
C(Codierung)[T.2]    -0.0791      0.095     -0.830      0.407      -0.267
0.109
C(Codierung)[T.4]     0.0328      0.087      0.375      0.708      -0.139
0.205
C(Codierung)[T.5]    -0.1815      0.074     -2.449      0.015      -0.328
-0.036
Zugänglichkeit        0.3180      0.073      4.334      0.000       0.173
0.463
Qualität              0.5762      0.062      9.274      0.000       0.454
0.699
==============================================================================
Omnibus:                        3.521   Durbin-Watson:                   0.305
Prob(Omnibus):                  0.172   Jarque-Bera (JB):                3.727
Skew:                           0.123   Prob(JB):                        0.155
Kurtosis:                       3.527   Cond. No.                         14.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[15]:

```python
# regression for quality_i,j = a + b * Kosten_i,j + c * Zugänglichkeit_i,j + d␣
↪* Jahr_i + e * Codierung_i
model = smf.ols('Qualität ~ Kosten + Zugänglichkeit + C(Jahr)', data=df_r).fit()

print(model.summary())
with open("regression_summary.tex", "w") as f:
    f.write(model.summary().as_latex())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Qualität   R-squared:                       0.508
Model:                            OLS   Adj. R-squared:                  0.483
```

```
Method:                 Least Squares   F-statistic:                     19.88
Date:                Sun, 06 Apr 2025   Prob (F-statistic):           9.85e-32
Time:                        00:01:46   Log-Likelihood:                -126.26
No. Observations:                 264   AIC:                             280.5
Df Residuals:                     250   BIC:                             330.6
Df Model:                          13
Covariance Type:            nonrobust
================================================================================
===
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept         -0.0776      0.086     -0.903      0.367      -0.247
0.092
C(Jahr)[T.2012]    0.0431      0.121      0.356      0.722      -0.195
0.281
C(Jahr)[T.2013]    0.2155      0.121      1.782      0.076      -0.023
0.454
C(Jahr)[T.2014]    0.2979      0.121      2.463      0.014       0.060
0.536
C(Jahr)[T.2015]    0.1944      0.121      1.607      0.109      -0.044
0.433
C(Jahr)[T.2016]    0.1557      0.121      1.285      0.200      -0.083
0.394
C(Jahr)[T.2017]    0.2765      0.121      2.283      0.023       0.038
0.515
C(Jahr)[T.2018]    0.2139      0.121      1.763      0.079      -0.025
0.453
C(Jahr)[T.2019]    0.3496      0.121      2.882      0.004       0.111
0.588
C(Jahr)[T.2020]    0.2481      0.121      2.044      0.042       0.009
0.487
C(Jahr)[T.2021]    0.2844      0.121      2.342      0.020       0.045
0.523
C(Jahr)[T.2022]    0.3034      0.121      2.502      0.013       0.065
0.542
Kosten             0.6412      0.046     13.943      0.000       0.551
0.732
Zugänglichkeit    -0.3870      0.055     -7.064      0.000      -0.495
-0.279
==============================================================================
Omnibus:                       14.435   Durbin-Watson:                   0.279
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               15.405
Skew:                          -0.530   Prob(JB):                     0.000452
Kurtosis:                       3.527   Cond. No.                         13.2
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[16]:
```
# regression for quality_i,j = a + b * Kosten_i,j + c * Zugänglichkeit_i,j + d␣
↪* Jahr_i + e * Codierung_i
model = smf.ols('Zugänglichkeit ~ Qualität + Kosten + C(Jahr)', data=df_r).fit()

print(model.summary())
with open("regression_summary.tex", "w") as f:
    f.write(model.summary().as_latex())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         Zugänglichkeit   R-squared:                       0.189
Model:                            OLS   Adj. R-squared:                  0.147
Method:                 Least Squares   F-statistic:                     4.493
Date:                Sun, 06 Apr 2025   Prob (F-statistic):           7.06e-07
Time:                        00:01:46   Log-Likelihood:                -140.16
No. Observations:                 264   AIC:                             308.3
Df Residuals:                     250   BIC:                             358.4
Df Model:                          13
Covariance Type:            nonrobust
================================================================================
===
                      coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept           0.0806      0.091      0.890      0.374      -0.098
0.259
C(Jahr)[T.2012]    -0.0165      0.128     -0.129      0.897      -0.268
0.235
C(Jahr)[T.2013]     0.0607      0.128      0.473      0.636      -0.192
0.313
C(Jahr)[T.2014]     0.1244      0.129      0.966      0.335      -0.129
0.378
C(Jahr)[T.2015]     0.0868      0.128      0.678      0.498      -0.165
0.339
C(Jahr)[T.2016]    -0.0316      0.128     -0.247      0.805      -0.284
0.221
C(Jahr)[T.2017]     0.0199      0.129      0.154      0.877      -0.234
0.274
C(Jahr)[T.2018]    -0.0334      0.129     -0.260      0.795      -0.287
0.220
C(Jahr)[T.2019]     0.0249      0.130      0.192      0.848      -0.231
0.281
C(Jahr)[T.2020]     0.0684      0.129      0.530      0.597      -0.186
```

```
0.322
C(Jahr)[T.2021]     0.1500      0.129      1.162      0.246      -0.104
0.404
C(Jahr)[T.2022]     0.0930      0.129      0.719      0.473      -0.162
0.348
Qualität           -0.4300      0.061     -7.064      0.000      -0.550
-0.310
Kosten              0.3448      0.061      5.667      0.000       0.225
0.465
==============================================================================
Omnibus:                        0.446   Durbin-Watson:                   0.282
Prob(Omnibus):                  0.800   Jarque-Bera (JB):                0.376
Skew:                           0.092   Prob(JB):                        0.829
Kurtosis:                       3.004   Cond. No.                         13.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[17]:
```python
# regression for quality_i,j = a + b * Kosten_i,j + c * Zugänglichkeit_i,j + d␣
↪* Jahr_i + e * Codierung_i
model = smf.ols('Kosten ~ Zugänglichkeit + Qualität + C(Jahr)', data=df_r).fit()

print(model.summary())
with open("regression_summary.tex", "w") as f:
    f.write(model.summary().as_latex())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Kosten   R-squared:                       0.456
Model:                            OLS   Adj. R-squared:                  0.428
Method:                 Least Squares   F-statistic:                     16.12
Date:                Sun, 06 Apr 2025   Prob (F-statistic):           1.69e-26
Time:                        00:01:47   Log-Likelihood:                -134.44
No. Observations:                 264   AIC:                             296.9
Df Residuals:                     250   BIC:                             346.9
Df Model:                          13
Covariance Type:            nonrobust
================================================================================
===
                      coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept           0.0763      0.089      0.861      0.390      -0.098
0.251
C(Jahr)[T.2012]    -0.0147      0.125     -0.118      0.906      -0.261
```

```
0.231
C(Jahr)[T.2013]    -0.1326      0.125     -1.058      0.291      -0.379
0.114
C(Jahr)[T.2014]    -0.1965      0.126     -1.564      0.119      -0.444
0.051
C(Jahr)[T.2015]    -0.1195      0.125     -0.954      0.341      -0.366
0.127
C(Jahr)[T.2016]    -0.0849      0.125     -0.678      0.498      -0.332
0.162
C(Jahr)[T.2017]    -0.1638      0.126     -1.302      0.194      -0.412
0.084
C(Jahr)[T.2018]    -0.0678      0.126     -0.538      0.591      -0.316
0.180
C(Jahr)[T.2019]    -0.1805      0.127     -1.425      0.155      -0.430
0.069
C(Jahr)[T.2020]    -0.0390      0.126     -0.309      0.757      -0.288
0.210
C(Jahr)[T.2021]    -0.0690      0.127     -0.545      0.586      -0.318
0.180
C(Jahr)[T.2022]    -0.0995      0.126     -0.787      0.432      -0.349
0.150
Zugänglichkeit      0.3302      0.058      5.667      0.000       0.215
0.445
Qualität            0.6822      0.049     13.943      0.000       0.586
0.779
==============================================================================
Omnibus:                        3.764   Durbin-Watson:                   0.286
Prob(Omnibus):                  0.152   Jarque-Bera (JB):                3.476
Skew:                           0.221   Prob(JB):                        0.176
Kurtosis:                       3.347   Cond. No.                         13.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[18]:

```python
# test hypothesis:

import scipy.stats as stats

t_texts = [
    'system 4 has higher quality than system 5',
    'system 4 has higher accessibility than system 1 and 2',
    'system 4 has higher costs than system 5',
    'system 1 and 2 has higher costs than system 5',
    'system 1 and 2 has higher quality than system 5',
    'system 1 and 2 has higher accessibility than system 5',
```

```
    'system 4 has higher quality than system 1 and 2',
    'system 4 has higher costs than system 1 and 2'
]

t_res = []

t_res.append([stats.ttest_ind(df_r[df_r['system'] == 1]['Qualität'],␣
 ↪df_r[df_r['system'] == 0]['Qualität']), df_r[df_r['system'] ==␣
 ↪1]['Qualität'].mean(), df_r[df_r['system'] == 0]['Qualität'].mean()])
t_res.append([stats.ttest_ind(df_r[df_r['system'] == 1]['Zugänglichkeit'],␣
 ↪df_r[df_r['system'] == -1]['Zugänglichkeit']), df_r[df_r['system'] ==␣
 ↪1]['Zugänglichkeit'].mean(), df_r[df_r['system'] == -1]['Zugänglichkeit'].
 ↪mean()])
t_res.append([stats.ttest_ind(df_r[df_r['system'] == 1]['Kosten'],␣
 ↪df_r[df_r['system'] == 0]['Kosten']), df_r[df_r['system'] == 1]['Kosten'].
 ↪mean(), df_r[df_r['system'] == 0]['Kosten'].mean()])
t_res.append([stats.ttest_ind(df_r[df_r['system'] == -1]['Kosten'],␣
 ↪df_r[df_r['system'] == 0]['Kosten']), df_r[df_r['system'] == -1]['Kosten'].
 ↪mean(), df_r[df_r['system'] == 0]['Kosten'].mean()])
t_res.append([stats.ttest_ind(df_r[df_r['system'] == -1]['Qualität'],␣
 ↪df_r[df_r['system'] == 0]['Qualität']), df_r[df_r['system'] ==␣
 ↪-1]['Qualität'].mean(), df_r[df_r['system'] == 0]['Qualität'].mean()])
t_res.append([stats.ttest_ind(df_r[df_r['system'] == -1]['Zugänglichkeit'],␣
 ↪df_r[df_r['system'] == 0]['Zugänglichkeit']), df_r[df_r['system'] ==␣
 ↪-1]['Zugänglichkeit'].mean(), df_r[df_r['system'] == 0]['Zugänglichkeit'].
 ↪mean()])
t_res.append([stats.ttest_ind(df_r[df_r['system'] == 1]['Qualität'],␣
 ↪df_r[df_r['system'] == -1]['Qualität']), df_r[df_r['system'] ==␣
 ↪1]['Qualität'].mean(), df_r[df_r['system'] == -1]['Qualität'].mean()])
t_res.append([stats.ttest_ind(df_r[df_r['system'] == 1]['Kosten'],␣
 ↪df_r[df_r['system'] == -1]['Kosten']), df_r[df_r['system'] == 1]['Kosten'].
 ↪mean(), df_r[df_r['system'] == -1]['Kosten'].mean()])

for res in t_res:
    print(f'{t_texts[t_res.index(res)]}: p-value: {round(res[0].pvalue,3)},␣
 ↪mean 1: {round(res[1],3)}, mean 2: {round(res[2],3)}')
```

```
system 4 has higher quality than system 5: p-value: 0.0, mean 1: 0.531, mean 2:
-0.255
system 4 has higher accessibility than system 1 and 2: p-value: 0.0, mean 1:
0.496, mean 2: -0.193
system 4 has higher costs than system 5: p-value: 0.0, mean 1: 0.564, mean 2:
-0.183
system 1 and 2 has higher costs than system 5: p-value: 0.0, mean 1: 0.229, mean
2: -0.183
system 1 and 2 has higher quality than system 5: p-value: 0.0, mean 1: 0.414,
mean 2: -0.255
```

system 1 and 2 has higher accessibility than system 5: p-value: 0.0, mean 1: -0.193, mean 2: 0.246
system 4 has higher quality than system 1 and 2: p-value: 0.01, mean 1: 0.531, mean 2: 0.414
system 4 has higher costs than system 1 and 2: p-value: 0.0, mean 1: 0.564, mean 2: 0.229