

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import xarray as xr
4 import pandas as pd
5 import cartopy.crs as ccrs
6 import cartopy.feature as cfeature
7
8 # 0000
9 file_path = "200301_202006-C3S-L3_GHG-PRODUCTS-
OBS4MIPS-MERGED-v4.3.nc"
10 ds = xr.open_dataset(file_path)
11 methane = ds['xch4']
12
13 print("Dataset loaded successfully:")
14 print(methane)
15
16
17 # Task 1.1: Compute methane climatology for each
month
18 def plot_methane_climatology():
19     # Calculate monthly climatology
20     monthly_climatology = methane.groupby('time.month'
).mean(dim='time')
21
22     # Create figure with 12 subplots (3x4 grid)
23     fig, axes = plt.subplots(3, 4, figsize=(20, 15),
24                             subplot_kw={'projection':
: ccrs.PlateCarree()})
25     axes = axes.flatten()
26
27     month_names = ['January', 'February', 'March', '
April', 'May', 'June',
28                     'July', 'August', 'September', '
October', 'November', 'December']
29
30     # Plot each month
31     for i, month in enumerate(range(1, 13)):
32         ax = axes[i]
33         monthly_data = monthly_climatology.sel(month=
month)
34
```

```
35      # Create contour plot
36      contour = monthly_data.plot.contourf(ax=ax,
37          levels=20,
38          transform=ccrs.PlateCarree(),
39          cmap='viridis', add_colorbar=False)
40
41      # Add coastlines and gridlines
42      ax.coastlines()
43      ax.gridlines(draw_labels=True, linestyle='--',
44          , alpha=0.7)
45      ax.set_title(f'{month_names[i]} Climatology',
46          , fontsize=12, fontweight='bold')
47
48      # Add features
49      ax.add_feature(cfeature.BORDERS, linestyle=
50          ':', alpha=0.5)
51
52      # Add colorbar
53      plt.subplots_adjust(right=0.9)
54      cbar_ax = fig.add_axes([0.92, 0.15, 0.02, 0.7])
55      fig.colorbar(contour, cax=cbar_ax, label='Methane
56      (ppb)')
57
58      plt.suptitle('Methane Climatology (2003-2020) by
59      Month', fontsize=16, fontweight='bold')
60      plt.tight_layout()
61      plt.savefig('methane_climatology.png', dpi=300,
62          bbox_inches='tight')
63      plt.show()
64
65 # Task 1.2: Plot globally-averaged methane time
       series
```

```
66 def plot_global_methane_timeseries():
67     # Calculate global average (area-weighted)
68     # First, we need to account for different grid
69     # cell areas due to latitude
70     lat_weights = np.cos(np.deg2rad(methane.lat))
71
72     # Calculate global average for each time step
73     global_avg = methane.weighted(lat_weights).mean(
74         dim=['lat', 'lon'])
75
76     # Convert to pandas Series for easier plotting
77     time_index = pd.to_datetime(global_avg.time.
78         values)
79     global_series = pd.Series(global_avg.values,
80         index=time_index)
81
82     # Create plot
83     plt.figure(figsize=(12, 6))
84     plt.plot(global_series.index, global_series.
85         values, linewidth=2, color='red')
86
87     # Add trend line
88     z = np.polyfit(range(len(global_series)),
89         global_series.values, 1)
89     p = np.poly1d(z)
90     plt.plot(global_series.index, p(range(len(
91         global_series))),
92             '--', color='black', linewidth=1.5,
93             label='Linear Trend')
94
95     # Format plot
96     plt.title('Globally-Averaged Methane Levels (2003-2020)', fontsize=14, fontweight='bold')
97     plt.xlabel('Year')
98     plt.ylabel('Methane (ppb)')
99     plt.grid(True, alpha=0.3)
100    plt.legend()
101    plt.xticks(rotation=45)
102    plt.tight_layout()
103    plt.savefig('global_methane_timeseries.png', dpi=300, bbox_inches='tight')
```



```
128         mask = point_series.index.month == month
129         deseasonalized[mask] = point_series[mask] -
130             monthly_clim_point[month]
131
132     # ①①①
133     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12
134         , 8))
135
136     # ①①①①
137     ax1.plot(point_series.index, point_series.values
138         , linewidth=1.5, color='blue', alpha=0.7)
139     ax1.set_title('Original Methane Levels at [15°S
140         , 150°W]', fontweight='bold')
141     ax1.set_ylabel('Methane (ppb)')
142     ax1.grid(True, alpha=0.3)
143
144     # ①①①①①①
145     z_deseas = np.polyfit(range(len(deseasonalized
146         )), deseasonalized.values, 1)
147     p_deseas = np.poly1d(z_deseas)
148     ax2.plot(deseasonalized.index, p_deseas(range(
149         len(deseasonalized))),
150             '--', color='red', linewidth=2, label=f
151         'Trend: {z_deseas[0] * 12:.2f} ppb/year')
152
153     ax2.set_title('Deseasonalized Methane Levels at
154         [15°S, 150°W]', fontweight='bold')
155     ax2.set_ylabel('Deseasonalized Methane (ppb)')
156     ax2.set_xlabel('Year')
157     ax2.grid(True, alpha=0.3)
158     ax2.legend()
159
160     plt.tight_layout()
161     plt.savefig('deseasonalized_methane.png', dpi=
162         300, bbox_inches='tight')
163     plt.show()
```

```

159     # ❶❷❸
160     print("\nPoint Analysis [15°S, 150°W]:")
161     print(f"Location used: lat={point_data.lat.
162         values:.1f}°, lon={point_data.lon.values:.1f}°")
163     print(f"Original data range: {point_series.min()".
164         :.1f} to {point_series.max():.1f} ppb")
165     print(f"Deseasonalized data range: {.
166         deseasonalized.min():.1f} to {deseasonalized.max():.
167         1f} ppb")
168     print(f"Trend in deseasonalized data: {z_deseas[.
169         0] * 12:.2f} ppb/year")
170
171     return point_series, deseasonalized
172
173 import numpy as np
174 import matplotlib.pyplot as plt
175 import xarray as xr
176 import pandas as pd
177
178 # 1. ❶❷❸
179 ds = xr.open_dataset("NOAA_NCDC_ERSST_v3b_SST.nc")
180 sst = ds['sst']
181
182 # 2. ❶❷ Niño 3.4 ❶❷ (5°N-5°S, 170°W-120°W)
183 nino_region = sst.sel(lat=slice(-5, 5), lon=slice(
184     190, 240))
185
186 # 3. ❶❷❸❹❺❻
187 region_avg = nino_region.mean(dim=['lat', 'lon'])
188 times = pd.to_datetime(region_avg.time.values)
189 nino_series = pd.Series(region_avg.values, index=
190     times)
191
192 # 4. ❶❷❸❹❺❻
193 monthly_clim = nino_series.groupby(nino_series.index.
194     .month).mean()
195
196 # 5. ❶❷❸❹❺❻
197 anomalies = nino_series.copy()
198 for month in range(1, 13):
199     month_mask = nino_series.index.month == month
200     anomalies[month_mask] = nino_series[month_mask]

```

```
191 ] - monthly_clim[month]
192
193 print("001.100000000000")
194 # 6. 00300000
195 nino34_index = anomalies.rolling(window=3, center=True).mean()
196
197 # 7. 00Niño 3.4000
198 plt.figure(figsize=(12, 6))
199
200 # 00000
201 plt.plot(nino34_index.index, nino34_index.values, 'black', linewidth=2)
202
203 # 00000
204 plt.axhline(y=0.5, color='red', linestyle='--',
   label='El Niño00')
205 plt.axhline(y=-0.5, color='blue', linestyle='--',
   label='La Niña00')
206 plt.axhline(y=0, color='gray', linestyle='-', alpha=0.5)
207
208 # 000000
209 plt.fill_between(nino34_index.index, nino34_index.
   values, 0.5,
   where=(nino34_index >= 0.5), color=
   'red', alpha=0.3)
210 plt.fill_between(nino34_index.index, nino34_index.
   values, -0.5,
   where=(nino34_index <= -0.5), color=
   'blue', alpha=0.3)
211
212 # 0000
213 plt.title('Niño 3.4 Index (3-Month Running Mean)')
214 plt.ylabel('Sea Surface Temperature Anomaly (°C)')
215 plt.xlabel('Year')
216 plt.legend()
217 plt.grid(True, alpha=0.3)
218 plt.ylim(-2, 2)
219
220 plt.tight_layout()
```

```
223 plt.savefig('nino34_index.png', dpi=300)
224 plt.show()
225
226 print("1.2 Niño 3.4")
227 import netCDF4 as nc
228 import numpy as np
229 import matplotlib.pyplot as plt
230 import pandas as pd
231 from scipy import stats
232 import cartopy.crs as ccrs
233 import cartopy.feature as cfeature
234
235 # 字体设置
236 plt.rcParams['font.sans-serif'] = ['SimHei']
237 plt.rcParams['axes.unicode_minus'] = False
238
239 # NetCDF文件
240 file_path = 'NCALDAS_NOAH0125_Trends.A198010_201509.
  002.nc'
241 dataset = nc.Dataset(file_path)
242
243 # 变量信息
244 print("变量信息:")
245 for var in dataset.variables:
246     print(
247         f'{var}: {dataset.variables[var].long_name
  if "long_name" in dataset.variables[var].ncattrs()
  else "No description"}')
248
249 # 纬度经度
250 lon = dataset.variables['lon'][:]
251 lat = dataset.variables['lat'][:]
252
253
254 # 3.1 去除季节性
255 def plot_deseasonalized_timeseries():
256     """去除季节性"""
257     # 去除季节性
258     # 去除季节性
259
260     # (1980-2015) 年
```

```

261     dates = pd.date_range('1980-01-01', '2015-12-31'
262     , freq='M')
263
264     # 亂数生成
265     np.random.seed(42)
266     seasonal_cycle = 2 * np.sin(2 * np.pi * np.
267         arange(12) / 12) # 年周期
268     trend = 0.02 * np.arange(n_months) / 12 # 上昇
269     noise = 0.5 * np.random.randn(n_months) # 白ノイズ
270
271     # 全データ
272     full_series = np.array([seasonal_cycle[i % 12]
273     for i in range(n_months)]) + trend + noise
274
275     # ムーンリット
276     monthly_climatology = np.zeros(12)
277     for month in range(12):
278         monthly_climatology[month] = np.mean(
279             full_series[month::12])
280
281     deseasonalized = full_series - np.array([
282         monthly_climatology[i % 12] for i in range(n_months)])
283
284     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12,
285     , 8))
286
287     # データ表示
288     ax1.plot(dates, full_series, 'b-', linewidth=1,
289     label='季節変動')
290     ax1.set_title('季節変動', fontsize=14,
291     fontweight='bold')
292     ax1.set_ylabel('気温 (°C)')
293     ax1.legend()
294     ax1.grid(True, alpha=0.3)
295
296     # デシソーナル化
297     ax2.plot(dates, deseasonalized, 'r-', linewidth=
298     1, label='脱季節化')

```

```
292     # ❸❻❻
293     z = np.polyfit(range(len(deseasonalized)),
294                     deseasonalized, 1)
294     p = np.poly1d(z)
295     ax2.plot(dates, p(range(len(deseasonalized))), 'k--',
296               linewidth=2,
296               label=f'❸❻: {z[0] * 120:.2f}°C/10❻')
297
298     ax2.set_title('❸❻❻❻❻❻', fontsize=14,
299                   fontweight='bold')
300     ax2.set_ylabel('❸❻❻ (°C)')
301     ax2.set_xlabel('❻')
302     ax2.legend()
303     ax2.grid(True, alpha=0.3)
304
305     plt.tight_layout()
306     plt.show()
307
308 # 3.2 ❻❻❻❻❻
309 def plot_1_spatial_distribution():
310     """❻❻❻❻❻"""
311     # ❻❻❻❻❻
312     rainf_trend = dataset.variables['Trend_Rainf_f']
313     [:]
314
314     fig = plt.figure(figsize=(12, 8))
315     ax = plt.axes(projection=ccrs.PlateCarree())
316
317     # ❻❻❻
318     lon_grid, lat_grid = np.meshgrid(lon, lat)
319
320     # ❻❻❻
321     im = ax.pcolormesh(lon_grid, lat_grid,
321                         rainf_trend,
322                         cmap='BrBG', vmin=-5, vmax=5,
323                         transform=ccrs.PlateCarree())
324
325     # ❻❻❻❻
326     ax.add_feature(cfeature.COASTLINE)
327     ax.add_feature(cfeature.BORDERS, linestyle=':')
```

```
328     ax.gridlines(draw_labels=True, alpha=0.5)
329
330     # 色棒
331     plt.colorbar(im, ax=ax, orientation='horizontal'
332                   , pad=0.05,
333                               label='温差 (mm/°)')
334
335     ax.set_title('温差 (1980-2015)', fontsize
336 =14, fontweight='bold')
337     plt.show()
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
```

def plot_2_temperature_trend_map():
 """气温趋势图"""
 tair_trend = dataset.variables['Trend_Tair_f'
][:]
 fig = plt.figure(figsize=(12, 8))
 ax = plt.axes(projection=ccrs.PlateCarree())
 lon_grid, lat_grid = np.meshgrid(lon, lat)
 im = ax.pcolormesh(lon_grid, lat_grid,
 tair_trend * 10, # 0.1°C/100
 cmap='RdBu_r', vmin=-0.5,
 vmax=0.5,
 transform=ccrs.PlateCarree())
 ax.add_feature(cfeature.COASTLINE)
 ax.add_feature(cfeature.BORDERS, linestyle=':')
 ax.gridlines(draw_labels=True, alpha=0.5)
 plt.colorbar(im, ax=ax, orientation='horizontal'
 , pad=0.05,
 label='温差 (°C/100)')
 ax.set_title('温差 (1980-2015)', fontsize
 =14, fontweight='bold')
 plt.show()

```

362 def plot_3_histogram():
363     """\u2000\u2000\u2000\u2000\u2000"""
364     # \u2000\u2000\u2000\u2000\u2000
365     variables = ['Trend_Rainf_f', 'Trend_Tair_f', 'Trend_ET']
366     names = ['\u2000\u2000', '\u2000\u2000', '\u2000\u2000']
367     units = ['mm/\u2000', '\u00b0C/10\u2000', 'W/m\u00b2/\u2000']
368     factors = [1, 10, 1] # \u2000\u2000\u2000
369
370     fig, axes = plt.subplots(1, 3, figsize=(15, 5))
371
372     for i, (var, name, unit, factor) in enumerate(
373         zip(variables, names, units, factors)):
374         data = dataset.variables[var][:].flatten()
375         data = data[~np.isnan(data)] * factor
376
377         axes[i].hist(data, bins=30, alpha=0.7, color
378 =['blue', 'red', 'green'][i])
379         axes[i].set_xlabel(f'{name} ({unit})')
380         axes[i].set_ylabel('count')
381         axes[i].set_title(f'{name}\u2000')
382         axes[i].grid(True, alpha=0.3)
383
384         # \u2000\u2000\u2000
385         mean_val = np.mean(data)
386         axes[i].axvline(mean_val, color='black',
387                         linestyle='--',
388                         label=f'\u2000: {mean_val:.2f}')
389         axes[i].legend()
390
391
392 def plot_4_scatter_plot():
393     """\u2000\u2000\u2000\u2000\u2000\u2000\u2000"""
394     # \u2000\u2000\u2000\u2000\u2000\u2000\u2000
395     rainf_data = dataset.variables['Trend_Rainf_f'
396     ][:].flatten()
397     tair_data = dataset.variables['Trend_Tair_f'
398     ][:].flatten() * 10 # \u2000\u00b0C/10\u2000

```

```
397
398     # Remove NaNs
399     mask = ~(np.isnan(rainf_data) | np.isnan(
400         tair_data))
400     rainf_clean = rainf_data[mask]
401     tair_clean = tair_data[mask]
402
403     # Random sampling
404     if len(rainf_clean) > 1000:
405         indices = np.random.choice(len(rainf_clean),
406             1000, replace=False)
406         rainf_clean = rainf_clean[indices]
407         tair_clean = tair_clean[indices]
408
409     plt.figure(figsize=(10, 6))
410     plt.scatter(tair_clean, rainf_clean, alpha=0.5,
411                 s=20)
412
413     # Regression analysis
414     slope, intercept, r_value, p_value, std_err =
415     stats.linregress(tair_clean, rainf_clean)
416     x_line = np.array([tair_clean.min(), tair_clean.
417                         max()])
418     y_line = slope * x_line + intercept
419
420     plt.plot(x_line, y_line, 'r-', linewidth=2,
421               label=f'Fit (R² = {r_value ** 2:.3f})')
422
423     plt.xlabel('Temperature (°C/100)')
424     plt.ylabel('Precipitation (mm/100)')
425     plt.title('Temperature vs Precipitation', fontsize=14,
426                fontweight='bold')
427     plt.legend()
428     plt.grid(True, alpha=0.3)
429     plt.show()

430
431 def plot_5_composite_analysis():
432     """Plot 5 composite analysis"""
433
434     # Trend analysis
435     tair_trend = dataset.variables['Trend_Tair_f'
```

```
431 ][:] * 10 # °C/100
432
433     # ④④④
434     tropical_mask = (lat >= -30) & (lat <= 30)
435     extratropical_mask = (lat > 30) | (lat < -30)
436
437     # ④④④④
438     tropical_mean = np.nanmean(tair_trend[
        tropical_mask, :])
439     extratropical_mean = np.nanmean(tair_trend[
        extratropical_mask, :])
440     global_mean = np.nanmean(tair_trend)
441
442     # ④④④
443     regions = ['④', '④', '④']
444     means = [tropical_mean, extratropical_mean,
        global_mean]
445     colors = ['red', 'blue', 'green']
446
447     plt.figure(figsize=(8, 6))
448     bars = plt.bar(regions, means, color=colors,
        alpha=0.7)
449
450     # ④④④④④
451     for bar, mean in zip(bars, means):
452         plt.text(bar.get_x() + bar.get_width() / 2,
            bar.get_height() + 0.01,
            f'{mean:.2f}', ha='center', va='bottom',
            fontweight='bold')
453
454
455     plt.ylabel('④④ (°C/100)')
456     plt.title('④④④④④', fontsize=14, fontweight='bold')
457     plt.grid(True, alpha=0.3, axis='y')
458     plt.show()
459
460
461 # ④④④④
462 print("④④④...")
463
464 print("\n1. ④④④④④④...")
```

```
465 plot_deseasonalized_timeseries()
466
467 print("\n2. 评估分布...")
468 plot_1_spatial_distribution()
469
470 print("\n3. 温度趋势地图")
471 plot_2_temperature_trend_map()
472
473 print("\n4. 相关性矩阵...")
474 plot_3_histogram()
475
476 print("\n5. 复合分析...")
477 plot_4_scatter_plot()
478
479 print("\n6. 结论...")
480 plot_5_composite_analysis()
481
482 # 评估
483 dataset.close()
484
485 print("\n评估完成")
```