

```

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 # TODO
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,
        levels=20,
37         transform=ccrs.PlateCarree(),
38                                     cmap='
        viridis', add_colorbar=False)
39
40         # Add coastlines and gridlines
41         ax.coastlines()
42         ax.gridlines(draw_labels=True, linestyle='--'
        , alpha=0.7)
43         ax.set_title(f'{month_names[i]} Climatology'
        , fontsize=12, fontweight='bold')
44
45         # Add features
46         ax.add_feature(cfeature.BORDERS, linestyle=
        ':', alpha=0.5)
47
48         # Add colorbar
49         plt.subplots_adjust(right=0.9)
50         cbar_ax = fig.add_axes([0.92, 0.15, 0.02, 0.7])
51         fig.colorbar(contour, cax=cbar_ax, label='Methane
        (ppb)')
52
53         plt.suptitle('Methane Climatology (2003-2020) by
        Month', fontsize=16, fontweight='bold')
54         plt.tight_layout()
55         plt.savefig('methane_climatology.png', dpi=300,
        bbox_inches='tight')
56         plt.show()
57
58         return monthly_climatology
59
60
61 # =====
62 monthly_climatology = plot_methane_climatology()
63
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
    cell areas due to latitude
69     lat_weights = np.cos(np.deg2rad(methane.lat))
70
71     # Calculate global average for each time step
72     global_avg = methane.weighted(lat_weights).mean(
    dim=['lat', 'lon'])
73
74     # Convert to pandas Series for easier plotting
75     time_index = pd.to_datetime(global_avg.time.
    values)
76     global_series = pd.Series(global_avg.values,
    index=time_index)
77
78     # Create plot
79     plt.figure(figsize=(12, 6))
80     plt.plot(global_series.index, global_series.
    values, linewidth=2, color='red')
81
82     # Add trend line
83     z = np.polyfit(range(len(global_series)),
    global_series.values, 1)
84     p = np.poly1d(z)
85     plt.plot(global_series.index, p(range(len(
    global_series))),
86             '--', color='black', linewidth=1.5,
    label='Linear Trend')
87
88     # Format plot
89     plt.title('Globally-Averaged Methane Levels (
    2003-2020)', fontsize=14, fontweight='bold')
90     plt.xlabel('Year')
91     plt.ylabel('Methane (ppb)')
92     plt.grid(True, alpha=0.3)
93     plt.legend()
94     plt.xticks(rotation=45)
95     plt.tight_layout()
96     plt.savefig('global_methane_timeseries.png', dpi
    =300, bbox_inches='tight')

```

```

97     plt.show()
98
99     # Print statistics
100     print("Global Methane Analysis:")
101     print(f"Start value (2003): {global_series.iloc[
0]:.1f} ppb")
102     print(f"End value (2020): {global_series.iloc[-1
]:.1f} ppb")
103     print(f"Total increase: {global_series.iloc[-1
] - global_series.iloc[0]:.1f} ppb")
104     print(f"Average annual increase: {(global_series
.iloc[-1] - global_series.iloc[0]) / 17.5:.1f} ppb/
year")
105
106     return global_series
107
108
109 # Execute global average analysis
110 global_methane = plot_global_methane_timeseries()
111
112
113 # Task 1.3: Plot deseasonalized methane at [15°S,
150°W]
114 def plot_deseasonalized_methane():
115     # [15°S, 150°W]
116     point_data = methane.sel(lat=-15, lon=-150,
method='nearest')
117
118     # 
119     time_index = pd.to_datetime(point_data.time.
values)
120     point_series = pd.Series(point_data.values,
index=time_index)
121
122     # 
123     monthly_clim_point = point_series.groupby(
point_series.index.month).mean()
124
125     # 
126     deseasonalized = point_series.copy()
127     for month in range(1, 13):

```

```

128         mask = point_series.index.month == month
129         deseasonalized[mask] = point_series[mask] -
        monthly_clim_point[month]
130
131         # 0000
132         fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12
        , 8))
133
134         # 00000
135         ax1.plot(point_series.index, point_series.values
        , linewidth=1.5, color='blue', alpha=0.7)
136         ax1.set_title('Original Methane Levels at [15°S
        , 150°W]', fontweight='bold')
137         ax1.set_ylabel('Methane (ppb)')
138         ax1.grid(True, alpha=0.3)
139
140         # Plot deseasonalized data
141         ax2.plot(deseasonalized.index, deseasonalized.
        values, linewidth=1.5, color='green')
142
143         # 000000000
144         z_deseas = np.polyfit(range(len(deseasonalized
        )), deseasonalized.values, 1)
145         p_deseas = np.poly1d(z_deseas)
146         ax2.plot(deseasonalized.index, p_deseas(range(
        len(deseasonalized))),
147                 '--', color='red', linewidth=2, label=f
        'Trend: {z_deseas[0] * 12:.2f} ppb/year')
148
149         ax2.set_title('Deseasonalized Methane Levels at
        [15°S, 150°W]', fontweight='bold')
150         ax2.set_ylabel('Deseasonalized Methane (ppb)')
151         ax2.set_xlabel('Year')
152         ax2.grid(True, alpha=0.3)
153         ax2.legend()
154
155         plt.tight_layout()
156         plt.savefig('deseasonalized_methane.png', dpi=
        300, bbox_inches='tight')
157         plt.show()
158

```

```

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

```

```

191 ] - monthly_clim[month]
192
193 print("1.1")
194 # 6. 3
195 nino34_index = anomalies.rolling(window=3, center=
    True).mean()
196
197 # 7. Niño 3.4
198 plt.figure(figsize=(12, 6))
199
200 #
201 plt.plot(nino34_index.index, nino34_index.values, '
    black', linewidth=2)
202
203 #
204 plt.axhline(y=0.5, color='red', linestyle='--',
    label='El Niño')
205 plt.axhline(y=-0.5, color='blue', linestyle='--',
    label='La Niña')
206 plt.axhline(y=0, color='gray', linestyle='-', alpha=
    0.5)
207
208 #
209 plt.fill_between(nino34_index.index, nino34_index.
    values, 0.5,
210                 where=(nino34_index >= 0.5), color=
    'red', alpha=0.3)
211 plt.fill_between(nino34_index.index, nino34_index.
    values, -0.5,
212                 where=(nino34_index <= -0.5), color
    ='blue', alpha=0.3)
213
214 #
215 plt.title('Niño 3.4 Index (3-Month Running Mean)')
216 plt.ylabel('Sea Surface Temperature Anomaly (°C)')
217 plt.xlabel('Year')
218 plt.legend()
219 plt.grid(True, alpha=0.3)
220 plt.ylim(-2, 2)
221
222 plt.tight_layout()

```

```

223 plt.savefig('nino34_index.png', dpi=300)
224 plt.show()
225
226 print("1.2Niño 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 #
236 plt.rcParams['font.sans-serif'] = ['SimHei']
237 plt.rcParams['axes.unicode_minus'] = False
238
239 # NetCDF
240 file_path = 'NCALDAS_NOAH0125_Trends.A198010_201509.
002.nc'
241 dataset = nc.Dataset(file_path)
242
243 #
244 print(":")
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 #
250 lon = dataset.variables['lon'][:]
251 lat = dataset.variables['lat'][:]
252
253
254 # 3.1
255 def plot_deseasonalized_timeseries():
256     """
257     #
258     #
259
260     # (1980-2015)

```



```

261     dates = pd.date_range('1980-01-01', '2015-12-31'
    , freq='M')
262     n_months = len(dates)
263
264     # 00000000000000000000
265     np.random.seed(42)
266     seasonal_cycle = 2 * np.sin(2 * np.pi * np.
    arange(12) / 12) # 000
267     trend = 0.02 * np.arange(n_months) / 12 # 0000
268     noise = 0.5 * np.random.randn(n_months) # 0000
269
270     # 0000000
271     full_series = np.array([seasonal_cycle[i % 12]
    for i in range(n_months)]) + trend + noise
272
273     # 000000000000000000
274     monthly_climatology = np.zeros(12)
275     for month in range(12):
276         monthly_climatology[month] = np.mean(
    full_series[month::12])
277
278     deseasonalized = full_series - np.array([
    monthly_climatology[i % 12] for i in range(n_months
    )])
279
280     # 00
281     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12
    , 8))
282
283     # 0000000
284     ax1.plot(dates, full_series, 'b-', linewidth=1,
    label='0000')
285     ax1.set_title('000000000000000000', fontsize=14,
    fontweight='bold')
286     ax1.set_ylabel('0000 (°C)')
287     ax1.legend()
288     ax1.grid(True, alpha=0.3)
289
290     # 0000000000000000
291     ax2.plot(dates, deseasonalized, 'r-', linewidth=
    1, label='000000')

```

```

292     # 拟合
293     z = np.polyfit(range(len(deseasonalized)),
                     deseasonalized, 1)
294     p = np.poly1d(z)
295     ax2.plot(dates, p(range(len(deseasonalized))), '
    k--', linewidth=2,
296             label=f'拟合: {z[0] * 120:.2f}°C/10°')
297
298     ax2.set_title('全球变暖趋势', fontsize=14,
                    fontweight='bold')
299     ax2.set_ylabel('温度 (°C)')
300     ax2.set_xlabel('年份')
301     ax2.legend()
302     ax2.grid(True, alpha=0.3)
303
304     plt.tight_layout()
305     plt.show()
306
307
308 # 3.2 全球降水分布
309 def plot_1_spatial_distribution():
310     """全球降水分布"""
311     # 数据准备
312     rainf_trend = dataset.variables['Trend_Rainf_f'
    ][: ]
313
314     fig = plt.figure(figsize=(12, 8))
315     ax = plt.axes(projection=ccrs.PlateCarree())
316
317     # 网格
318     lon_grid, lat_grid = np.meshgrid(lon, lat)
319
320     # 绘图
321     im = ax.pcolormesh(lon_grid, lat_grid,
    rainf_trend,
322                       cmap='BrBG', vmin=-5, vmax=5,
323                       transform=ccrs.PlateCarree())
324
325     # 添加海岸线
326     ax.add_feature(cfeature.COASTLINE)
327     ax.add_feature(cfeature.BORDERS, linestyle=':')

```

```

328     ax.gridlines(draw_labels=True, alpha=0.5)
329
330     # 颜色条
331     plt.colorbar(im, ax=ax, orientation='horizontal'
332 , pad=0.05,
333                 label='降水 (mm/天)')
334
335     ax.set_title('全球平均降水 (1980-2015)', fontsize
336 =14, fontweight='bold')
337     plt.show()
338
339
340 def plot_2_temperature_trend_map():
341     """全球平均温度趋势"""
342     tair_trend = dataset.variables['Trend_Tair_f'
343 ][:]
344
345     fig = plt.figure(figsize=(12, 8))
346     ax = plt.axes(projection=ccrs.PlateCarree())
347
348     lon_grid, lat_grid = np.meshgrid(lon, lat)
349
350     im = ax.pcolormesh(lon_grid, lat_grid,
351 tair_trend * 10, # 温度/10度
352                 cmap='RdBu_r', vmin=-0.5,
353                 vmax=0.5,
354                 transform=ccrs.PlateCarree())
355
356     ax.add_feature(cfeature.COASTLINE)
357     ax.add_feature(cfeature.BORDERS, linestyle=':')
358     ax.gridlines(draw_labels=True, alpha=0.5)
359
360     plt.colorbar(im, ax=ax, orientation='horizontal'
361 , pad=0.05,
362                 label='温度趋势 (°C/10天)')
363
364     ax.set_title('全球平均温度趋势 (1980-2015)', fontsize
365 =14, fontweight='bold')
366     plt.show()
367
368

```

```

362 def plot_3_histogram():
363     """Histogram of variables"""
364     # Histogram of variables
365     variables = ['Trend_Rainf_f', 'Trend_Tair_f', '
Trend_ET']
366     names = ['Rainf', 'Tair', 'ET']
367     units = ['mm/d', '°C/10d', 'W/m²/d']
368     factors = [1, 10, 1] # Conversion factors
369
370     fig, axes = plt.subplots(1, 3, figsize=(15, 5))
371
372     for i, (var, name, unit, factor) in enumerate(
zip(variables, names, units, factors)):
373         data = dataset.variables[var][:].flatten()
374         data = data[~np.isnan(data)] * factor
375
376         axes[i].hist(data, bins=30, alpha=0.7, color
=['blue', 'red', 'green'][i])
377         axes[i].set_xlabel(f'{name} ({unit})')
378         axes[i].set_ylabel('Count')
379         axes[i].set_title(f'{name} {unit}')
380         axes[i].grid(True, alpha=0.3)
381
382         # Mean value
383         mean_val = np.mean(data)
384         axes[i].axvline(mean_val, color='black',
linestyle='--',
385                        label=f'{name}: {mean_val:.2f}')
386         axes[i].legend()
387
388     plt.tight_layout()
389     plt.show()
390
391
392 def plot_4_scatter_plot():
393     """Scatter plot of Rainf vs Tair"""
394     # Scatter plot of Rainf vs Tair
395     rainf_data = dataset.variables['Trend_Rainf_f']
][:].flatten()
396     tair_data = dataset.variables['Trend_Tair_f']
][:].flatten() * 10 # Convert to °C/10d

```

```

397
398     # NaN
399     mask = ~(np.isnan(rainf_data) | np.isnan(
    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
    ), 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,
    s=20)
411
412     # Linear Regression
413     slope, intercept, r_value, p_value, std_err =
    stats.linregress(tair_clean, rainf_clean)
414     x_line = np.array([tair_clean.min(), tair_clean.
    max()])
415     y_line = slope * x_line + intercept
416
417     plt.plot(x_line, y_line, 'r-', linewidth=2,
    label=f'Trend (R² = {r_value ** 2:.3f})')
418
419
420     plt.xlabel('Tair (°C/10)')
421     plt.ylabel('Rainf (mm/10)')
422     plt.title('Tair vs Rainf', fontsize=14,
    fontweight='bold')
423     plt.legend()
424     plt.grid(True, alpha=0.3)
425     plt.show()
426
427
428 def plot_5_composite_analysis():
429     """Composite Analysis"""
430     # Data Loading
431     tair_trend = dataset.variables['Trend_Tair_f']

```

```

431 ][:] * 10 # °C/100
432
433 # 00000
434 tropical_mask = (lat >= -30) & (lat <= 30)
435 extratropical_mask = (lat > 30) | (lat < -30)
436
437 # 000000
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 # 00000
443 regions = ['00', '00', '00']
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 # 000000000
451 for bar, mean in zip(bars, means):
452     plt.text(bar.get_x() + bar.get_width() / 2,
bar.get_height() + 0.01,
453             f'{mean:.2f}', ha='center', va='
bottom', fontweight='bold')
454
455 plt.ylabel('0000 (°C/100)')
456 plt.title('000000000000', fontsize=14, fontweight
='bold')
457 plt.grid(True, alpha=0.3, axis='y')
458 plt.show()
459
460
461 # 000000000
462 print("000000...")
463
464 print("\n1. 0000000000000000...")

```

```
465 plot_deseasonalized_timeseries()
466
467 print("\n2. 0000000000...")
468 plot_1_spatial_distribution()
469
470 print("\n3. 0000000000...")
471 plot_2_temperature_trend_map()
472
473 print("\n4. 0000000000...")
474 plot_3_histogram()
475
476 print("\n5. 0000000000...")
477 plot_4_scatter_plot()
478
479 print("\n6. 00000000...")
480 plot_5_composite_analysis()
481
482 # 00000
483 dataset.close()
484
485 print("\n0000000000")
486
487
```