PS3 报告说明

引用说明

任务 1：1.时间数据转换（`num2date`与`pandas`） 将 NetCDF 的时间变量（数值）转为可读的日期格式，采用 netCDF4 时间转换教程中 num2date`函数的参数（单位、日历类型）以及 Pandas 时间序列基础学习`pd.to_datetime`转换和时间属性提取 2. 数组计算与缺失值处理（`numpy`），按条件筛选数据（如`month_mask`）、计算平均值（`np.nanmean`）、处理缺失值（`np.nan`）。采用 NumPy 基础教程中数组索引、条件筛选、聚合函数（`mean`、`nanmean`）以及 NumPy 缺失值处理理解`np.nanmean`如何忽略缺失值。3. 绘图（`matplotlib`）多面板绘图（`subplots`）、彩色图（`pcolormesh`）、时间序列图（`plot`）、添加标签/图例。Matplotlib 官方入门教程涵盖基础绘图流程和参数设置。Matplotlib 多子图示例对应 12 个面板的绘图逻辑。 4. 经纬度索引查找（`numpy.argmin`） 通过`np.abs(...).argmin()`找到最接近目标经纬度的网格点。 NumPy argmin 文档理解如何用该函数找最小值索引（对应最接近的点）。

任务 2：

1.`groupby`计算气候态与异常（核心气象分析逻辑）`groupby('time.month').mean()` 计算月度气候态；`groupby('time.month') - 气候态` 计算 SST 异常。xarray 官方文档中 GroupBy 操作，详解`groupby`的分组、聚合、减法操作，附带"气候态与异常计算"的示例，与代码中 Niño 3.4 指数的计算逻辑完全一致。2.Pangeo 气候数据教程：气候态与异常以实际气候数据为例，讲解如何用 xarray 计算气候态、异常值，直接支撑代码中"SST 异常"的计算步骤。3.Niño 3.4 指数的气候学背景与定义，区域平均（`mean(dim=['lat', 'lon'])`）得到 Niño 3.4 指数；与 El Niño/La Niña 阈值（±0.5℃）对比。 NOAA 官方：ENSO 与 Niño 指数定义解释 Niño 3.4 区域的地理范围、指数计算方法，以及 El Niño/La Niña 的阈值标准。NCAR 气候数据指南：Niño 指数说明详细对比不同 Niño 区域（1+2、3、3.4、4）的定义，以及指数在气候研究中的应用，帮助理解代码的科学意义。 4.matplotlib 可视化与中文字体设置 matplotlib 官方文档：基础绘图，涵盖折线图、参考线、图例、网格的基础用法，代码中的可视化元素均可在此找到对应示例。 matplotlib 官方文档：字体设置详解如何设置中文字体（如"Microsoft YaHei"），解决代码中"中文标题/标签显示异常"的问题。针对国内用户常见的"matplotlib 中文乱码"问题，分步讲解`font.sans-serif`和`axes.unicode_minus`的设置逻辑。

任务 3：

1.Unidata 气象数据处理教程针对气象数据（如再分析、观测数据）的 xarray 操作，与代码中"气象变量（t2m、skt）处理"场景高度契合。2.气候态与异常计算（`groupby`核心逻辑），groupby('valid_time.month').mean()` 计算月度气候态 - `原始数据.groupby(...) - 气候态` 去除季节周期，xarray 官方 GroupBy 文档详细解释`groupby`的分组、聚合、减法操作，附带"气候态异常计算"的示例，与代码中"去除季节周期"的逻辑完全一致。 Pangeo 气候数据分析教程以实际气候数据为例，讲解如何用 xarray 计算气候态、异常值，直接支撑代码中"t2m 异常时间序列"的计算步骤。3. 空间分布（如"某一时刻的 t2m 空间分布"xarray 可视化文档，详解 xarray 与 matplotlib 的集成，包括二维数据的空间分布（`plot()`/`pcolormesh()`）、色标选择（`cmap='coolwarm'`）等，对应代码中的"空间分布图"。 时间序列折线图（如"t2m 全球平均时间序列"） matplotlib 官方折线图教程涵盖折线图的绘制、标题/标签设置、网格调整，与代码中的"时间序列图"完全对应。数值分布直方图（如"t2m 数值分布"）matplotlib 官方直方图教程讲解直方图的分箱（`bins=50`）、概率密度（`density=True`）、颜色/透明度设置，对应代码中的"直方图"。 多子图与色条统一（如"季节气候态空间分布对比"） matplotlib 官方子图与色条教程 详解多子图布局（`subplots`/`flatten`）、色条的统一添加与位置调整，对应代码中"季节气候态多图对比"的实现。散点图（如"t2m 与 skt 的散点图"）matplotlib 官方散点图教程讲解散点图的点密度（`alpha=0.6`）、

颜色选择、标题标签设置，对应代码中的“变量间散点图”。4.中文字体与警告处理，中文字体设置（`font.sans-serif`、`axes.unicode_minus`）matplotlib 官方字体教程 + 中文社区解决方案 - 官方字体文档解决 matplotlib 中文显示异常的问题，包括字体选择（如“Microsoft YaHei”）、负号显示等，与代码中的字体配置完全对应。警告过滤 Python 官方警告处理文档 讲解如何过滤特定模块/类别的警告，对应代码中“忽略 matplotlib 用户警告”的逻辑。

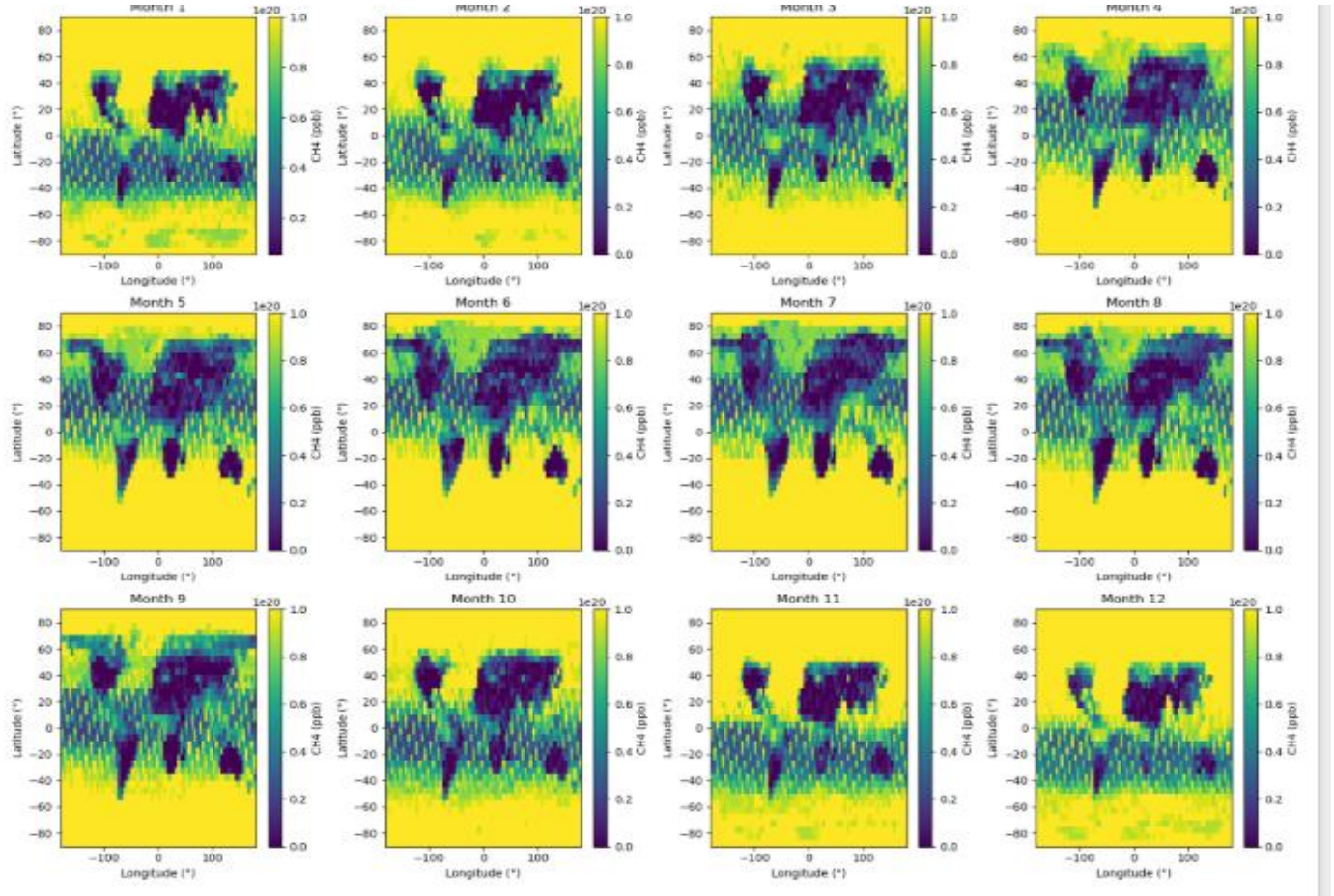
运行结果

任务 1：

```
#任务1
import netCDF4 as nc
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from netCDF4 import num2date
import os

nc_path = r'C:\Users\Morchaim&Hwachi\200301_202006-C3S-L3_GHG-PRODUCTS-OBS4MIPS-MERGED-v4.3.nc'
if not os.path.exists(nc_path):
    print(f"错误：文件不存在！路径：{nc_path}")
    exit()
try:
    ds = nc.Dataset(nc_path)
except Exception as e:
    print(f"打开文件失败：{e}")
    exit()

print("可用变量：", list(ds.variables.keys()))

# 处理时间变量（强制转为pandas datetime）
time = ds.variables['time'][:]
time_units = ds.variables['time'].units
time_calendar = ds.variables['time'].calendar if hasattr(ds.variables['time'], 'calendar') else 'gregorian'
# 先转为datetime对象列表，再强制转为pandas datetime
dates = num2date(time, units=time_units, calendar=time_calendar)
df_time = pd.DataFrame({'date': dates})
df_time['date'] = pd.to_datetime(df_time['date'].astype(str))  # 强制转换为pandas datetime
months = df_time['date'].dt.month  # 提取月份

# 处理甲烷变量（确认维度和类型）
methane_var_name = 'xch4'
xch4 = ds.variables[methane_var_name][:].astype(np.float64)  # 强制转为浮点型
print("xch4维度：", xch4.shape)  # 确认维度为 [time, lat, lon]
```

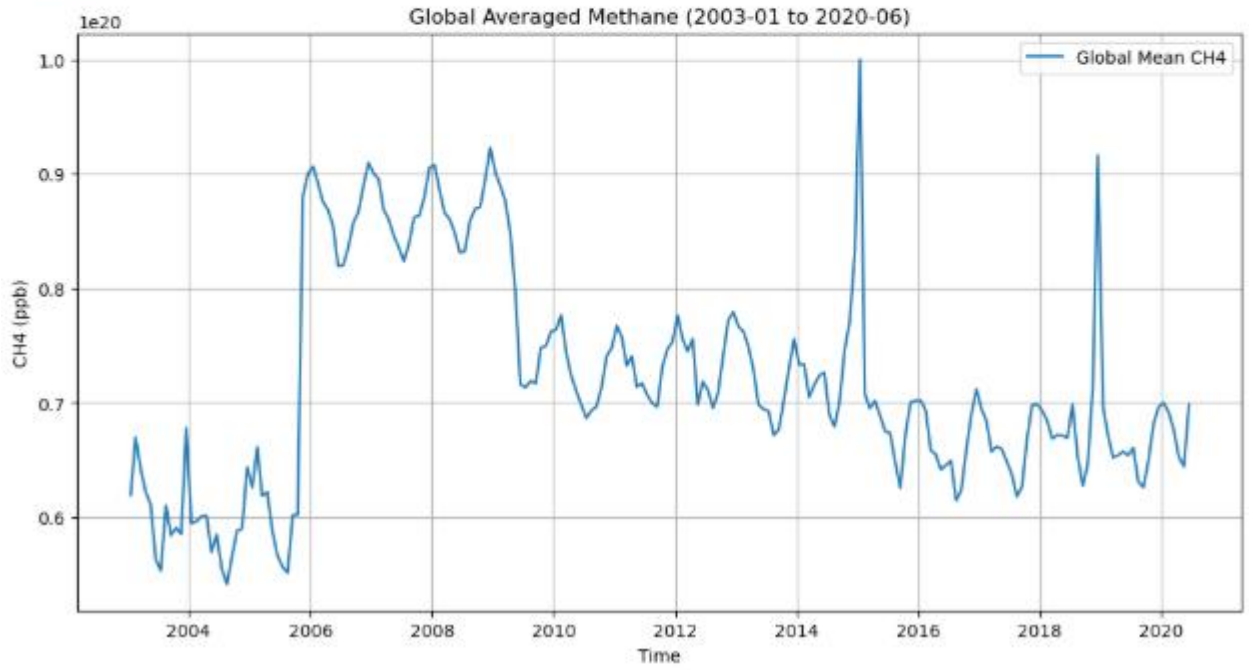
1.1：

```
# 任务1.1：  每月甲烷气候学 （12面板地图）

lon = ds.variables['lon'][:]
lat = ds.variables['lat'][:]

climatology = []
for month in range(1, 13):
    month_mask = (months == month)
    if not np.any(month_mask):
        print(f"警告：无第{month}月数据，将用0填充")
        climatology.append(np.zeros((len(lat), len(lon))))
        continue
    monthly_data = xch4[month_mask, :, :]
    monthly_mean = np.nanmean(monthly_data, axis=0)  # 沿时间轴平均
    climatology.append(monthly_mean)
climatology = np.array(climatology)

fig, axes = plt.subplots(3, 4, figsize=(16, 12))
axes = axes.flatten()
for i in range(12):
    im = axes[i].pcolormesh(lon, lat, climatology[i], cmap='viridis')
    axes[i].set_title(f'Month {i+1}')
    axes[i].set_xlabel('Longitude (°)')
    axes[i].set_ylabel('Latitude (°)')
    fig.colorbar(im, ax=axes[i], label='CH4 (ppb)')
plt.tight_layout()
plt.savefig('monthly_climatology.png')
plt.show()
```

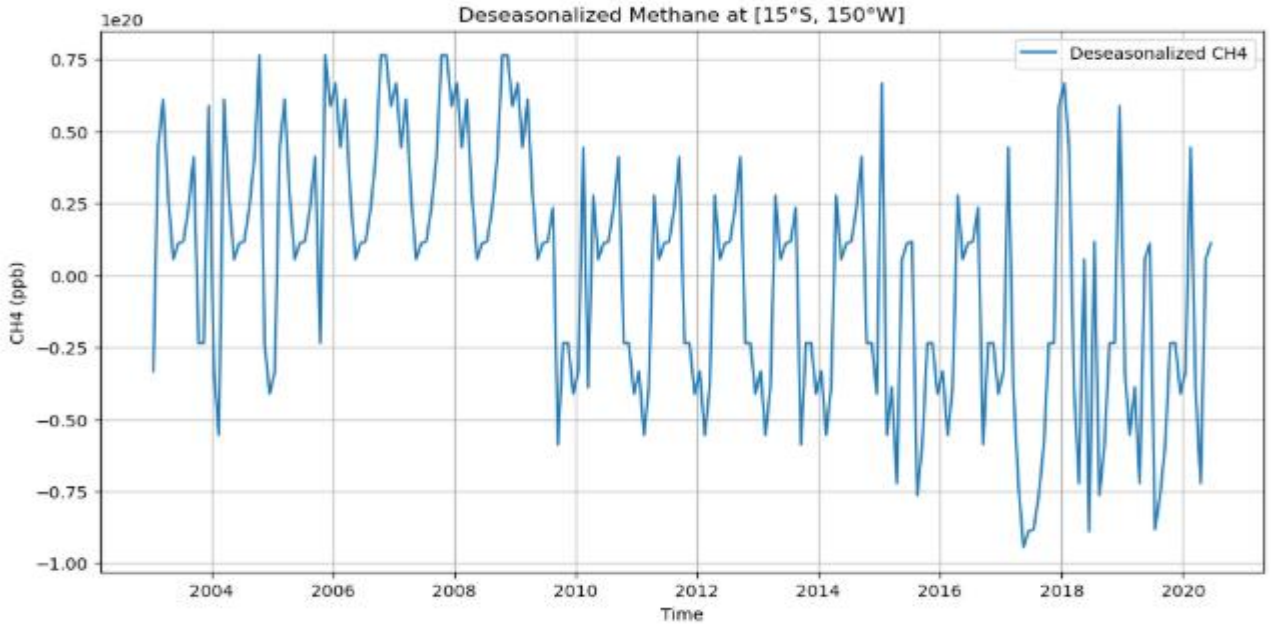


1.2：

```
# 任务1.2： 全球平均甲烷时间序列
global_means = []
for t in range(len(df_time)):
# xch4维度是 [time, lat, lon]，对lat平均用axis=0，lon平均用axis=1
    lat_mean = np.nanmean(xch4[t, :, :], axis=0)  # 纬度方向（第1轴）平均
    global_mean = np.nanmean(lat_mean)            # 经度方向（第0轴）平均
    global_means.append(global_mean)

plt.figure(figsize=(12, 6))
plt.plot(df_time['date'], global_means, label='Global Mean CH4')
plt.xlabel('Time')
plt.ylabel('CH4 (ppb)')
plt.title('Global Averaged Methane (2003-01 to 2020-06)')
plt.grid(True)
plt.legend()
plt.savefig('global_mean_time_series.png')
plt.show()
```



1.3：



结果描述：该点去季节化后甲烷浓度年际波动显著，在 - 1.0~0.75 ppb 区间内震荡，无明显长期趋势，体现了年际尺度因素对该区域甲烷浓度的影响。

任务 2：

```
#任务2
import xarray as xr
import matplotlib.pyplot as plt

file_path = r'C:\Users\Morchaim&Hwachi\NOAA_NCDC_ERSST_v3b_SST.nc'
ds = xr.open_dataset(file_path)
# 查看数据结构（确认经纬度范围）
print(ds)
print("经度范围：", ds.lon.min().values, ds.lon.max().values)
print("纬度范围：", ds.lat.min().values, ds.lat.max().values)
```

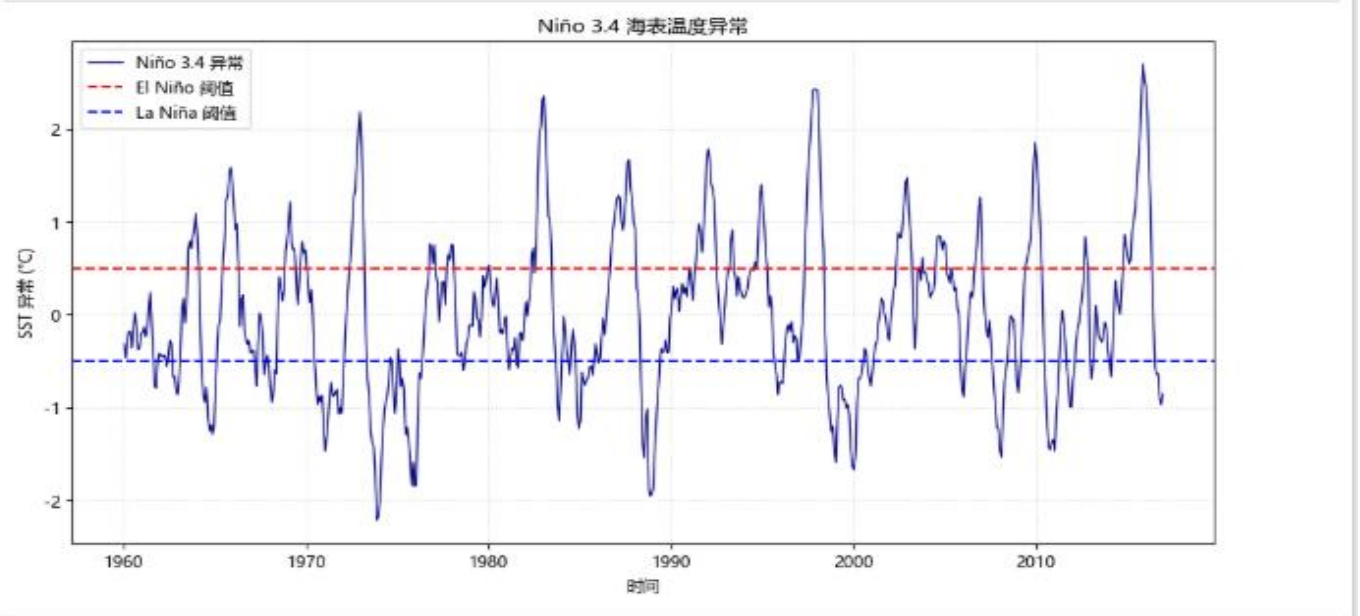
```
<xarray.Dataset> Size: 44MB
Dimensions:  (lat: 89, lon: 180, time: 684)
Coordinates:
  * lat      (lat) float32 356B -88.0 -86.0 -84.0 -82.0 ... 82.0 84.0 86.0 88.0
  * lon      (lon) float32 720B 0.0 2.0 4.0 6.0 8.0 ... 352.0 354.0 356.0 358.0
  * time     (time) datetime64[ns] 5kB 1960-01-15 1960-02-15 ... 2016-12-15
Data variables:
    sst      (time, lat, lon) float32 44MB ...
Attributes:
    Conventions:  IRIDL
    source:       https://iridl.ldeo.columbia.edu/SOURCES/.NOAA/.NCDC/.ERSST/...
    history:      extracted and cleaned by Ryan Abernathey for Research Compu...
经度范围： 0.0 358.0
纬度范围： -88.0 88.0
```

[6]:
```
# 定义Niño 3.4区域
lat_bounds = [-5, 5]
# 经度：170°W至120°W，数据中经度是0~360，转换为190°E至240°E
lon_bounds = [190, 240]

# 选择区域内的SST数据
sst = ds['sst'].sel(
    lat=slice(lat_bounds[0], lat_bounds[1]),
    lon=slice(lon_bounds[0], lon_bounds[1])
)
```

[7]:
```
# 计算月度气候态（Monthly Climatology）
sst_clim = sst.groupby('time.month').mean(dim='time')
```

[8]:
```
#4. 计算SST异常（Anomalies）
sst_anomaly = sst.groupby('time.month') - sst_clim

# 计算Niño 3.4指数（区域平均异常）
nino34 = sst_anomaly.mean(dim=['lat', 'lon'])
```

[9]:
```
#可视化Niño 3.4指数
plt.figure(figsize=(12, 6))

nino34.plot(linewidth=1, color='navy', label='Niño 3.4 异常')
plt.axhline(y=0.5, color='red', linestyle='--', label='El Niño 阈值')
plt.axhline(y=-0.5, color='blue', linestyle='--', label='La Niña 阈值')

# 字体更换为微软雅黑字体
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False

plt.title('Niño 3.4 海表温度异常')
plt.ylabel('SST 异常 (°C)')
plt.xlabel('时间')
plt.legend()
plt.grid(linestyle=':', alpha=0.5)
plt.show()
```



任务 3：

```
[11]: #任务3
import xarray as xr
import matplotlib.pyplot as plt
import numpy as np
import warnings  # 导入警告处理模块
from matplotlib import font_manager  # 正确导入
#读取数据集并查看变量
file_path = r'C:\Users\Morchaim&Hwachi\data_stream-moda.nc'
ds = xr.open_dataset(file_path)

# 打印数据集基本信息和所有变量，确认可用变量名
print("数据集结构：")
print(ds)
print("\n所有可用变量：", list(ds.variables))
```

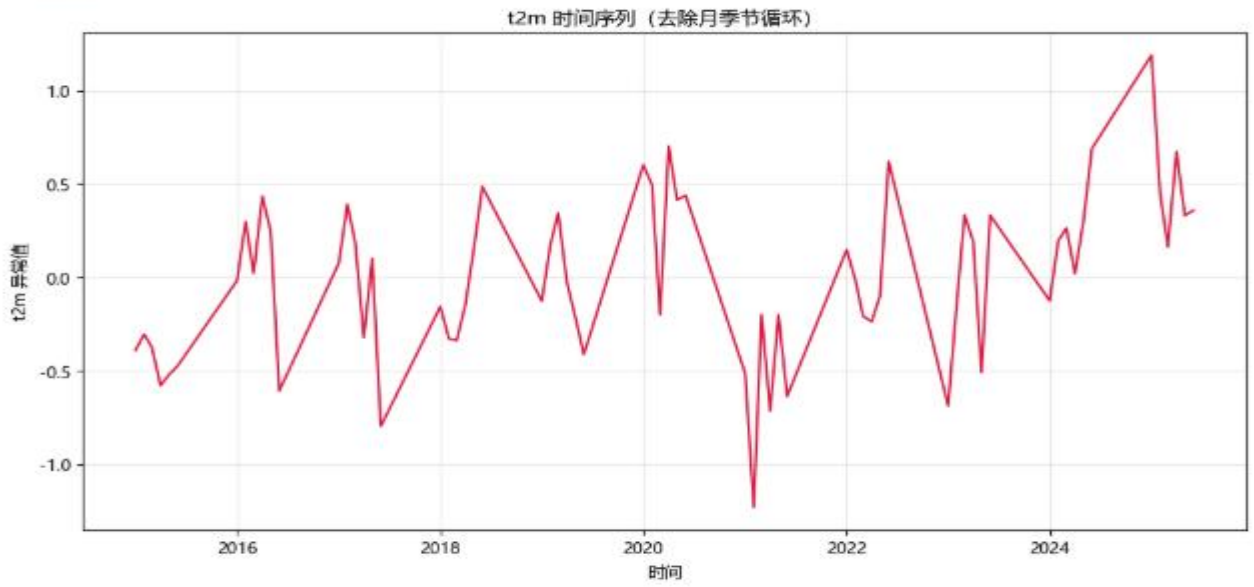
```
数据集结构：
<xarray.Dataset> Size: 19GB
Dimensions:     (valid_time: 66, latitude: 1801, longitude: 3600)
Coordinates:
    number      int64 8B ...
  * valid_time  (valid_time) datetime64[ns] 528B 2015-01-01 ... 2025-06-01
  * latitude    (latitude) float64 14kB 90.0 89.9 89.8 ... -89.8 -89.9 -90.0
  * longitude   (longitude) float64 29kB -180.0 -179.9 -179.8 ... 179.8 179.9
    expver      (valid_time) <U4 1kB ...
Data variables:
    t2m         (valid_time, latitude, longitude) float32 2GB ...
    skt         (valid_time, latitude, longitude) float32 2GB ...
    stl1        (valid_time, latitude, longitude) float32 2GB ...
    lblt        (valid_time, latitude, longitude) float32 2GB ...
    asn         (valid_time, latitude, longitude) float32 2GB ...
    swvl1       (valid_time, latitude, longitude) float32 2GB ...
    ssr         (valid_time, latitude, longitude) float32 2GB ...
    evabs       (valid_time, latitude, longitude) float32 2GB ...
    u10         (valid_time, latitude, longitude) float32 2GB ...
    tp          (valid_time, latitude, longitude) float32 2GB ...
    lai_hv      (valid_time, latitude, longitude) float32 2GB ...
Attributes:
    GRIB_centre:             ecmf
    GRIB_centreDescription:  European Centre for Medium-Range Weather Forecasts
    GRIB_subCentre:          0
    Conventions:             CF-1.7
    institution:             European Centre for Medium-Range Weather Forecasts
    history:                 2025-11-12T13:41 GRIB to CDM+CF via cfgrib-0.9.1...

所有可用变量： ['number', 'valid_time', 'latitude', 'longitude', 'expver', 't2m', 'skt', 'stl1', 'lblt', 'asn', 'swvl1', 'ssr', 'evabs', 'u10', 'tp', 'l
ai_hv']
```

任务 3.1：

```
[12]: # 抑制字体警告警告
warnings.filterwarnings("ignore", category=UserWarning, module="matplotlib")
# 字体设置
plt.rcParams["font.family"] = ["Microsoft YaHei", "SimHei"]
plt.rcParams["axes.unicode_minus"] = False  # 确保负号正常显示
```

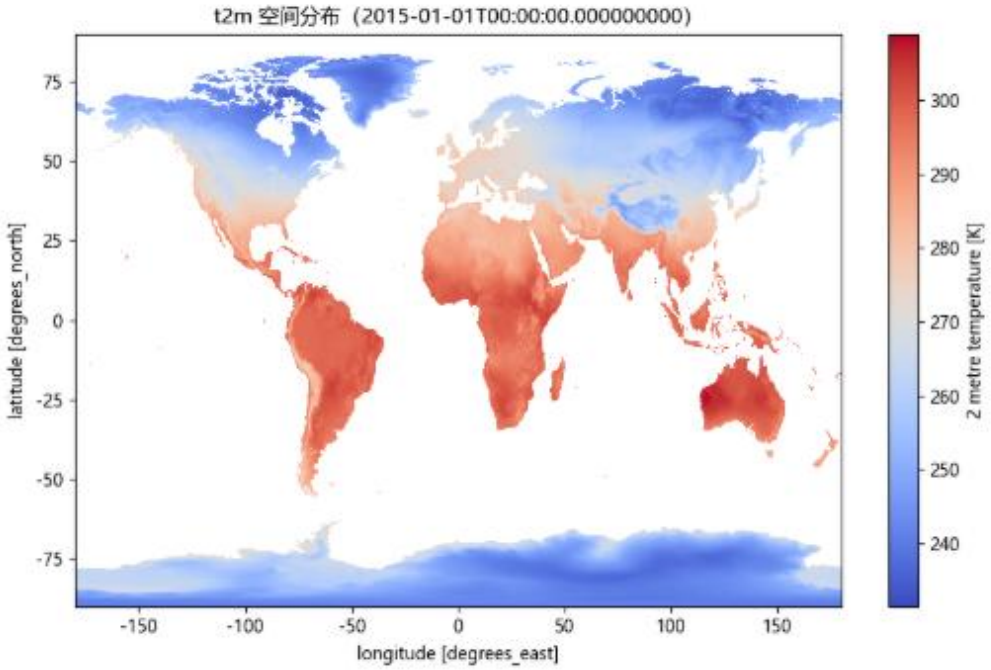
```
[24]: #读取数据集并选择变量（如t2m）
file_path = r'C:\Users\Morchaim&Hwachi\data_stream-moda.nc'
ds = xr.open_dataset(file_path)
var_name = 't2m'
data = ds[var_name]
time_dim = 'valid_time'  # 明确数据集的时间维度名
```

```
[14]: # 任务3.1：绘制"去除月季节循环"的时间序列
# 计算月气候态（多年同月平均值）
climatology = data.groupby('valid_time.month').mean(dim='valid_time')
# 去除季节循环（原数据 - 对应月气候态）
anomaly = data.groupby('valid_time.month') - climatology
# 空间平均（全局平均）
anomaly_mean = anomaly.mean(dim=['latitude', 'longitude'])

plt.figure(figsize=(12, 6))
anomaly_mean.plot(linewidth=1.5, color='crimson')
plt.title(f'{var_name} 时间序列（去除月季节循环）')
plt.xlabel('时间')
plt.ylabel(f'{var_name} 异常值')
plt.grid(alpha=0.3)
plt.show()
```

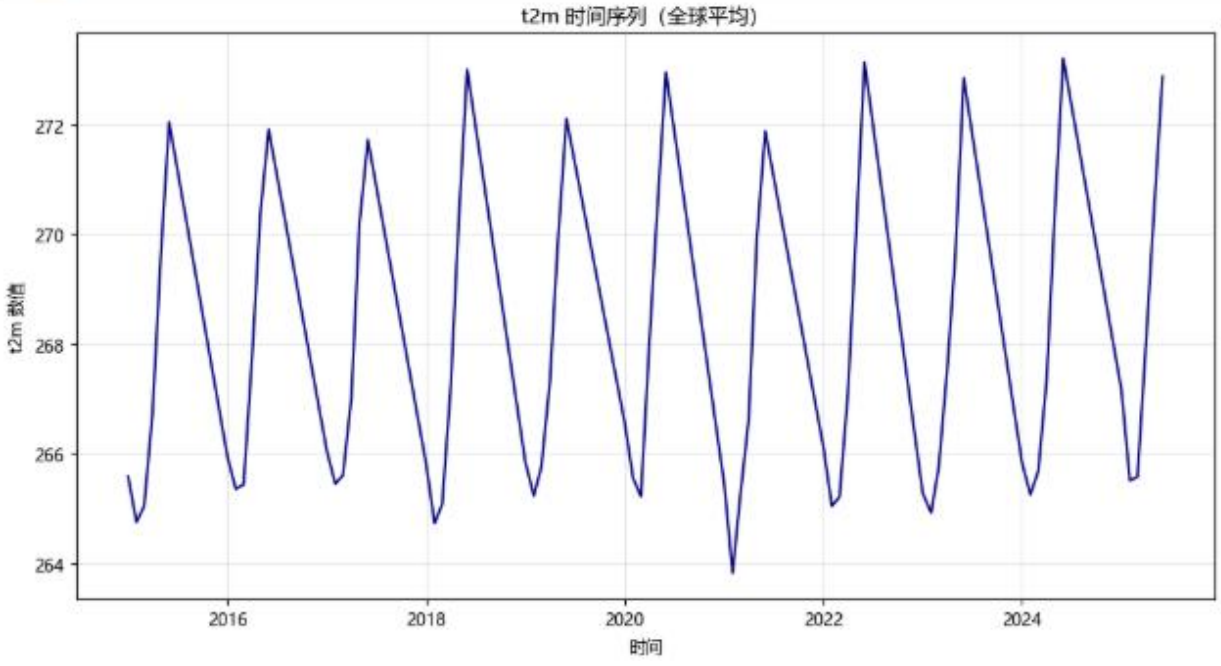


任务 3.2：

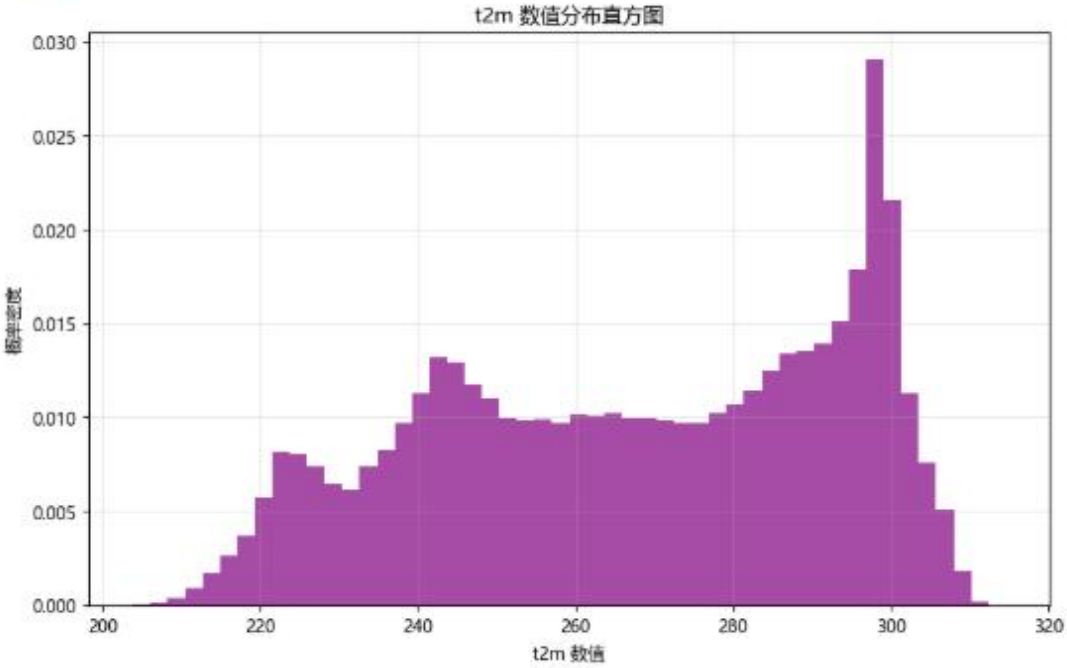
```
[15]: #任务3.2：绘制5种不同的图
# 图1：某一时刻的空间分布图（第一个时间步）
data_first = data.isel(valid_time=0)
plt.figure(figsize=(10, 6))
data_first.plot(cmap='coolwarm')  # 温度类变量用冷暖色
plt.title(f'{var_name} 空间分布 ({data_first.valid_time.values}) ')
plt.show()
```




```
[16]: # 图2：变量的时间序列（空间平均）
data_time_mean = data.mean(dim=['latitude', 'longitude'])
plt.figure(figsize=(12, 6))
data_time_mean.plot(linewidth=1.5, color='navy')
plt.title(f'{var_name} 时间序列（全球平均）')
plt.xlabel('时间')
plt.ylabel(f'{var_name} 数值')
plt.grid(alpha=0.3)
plt.show()
```

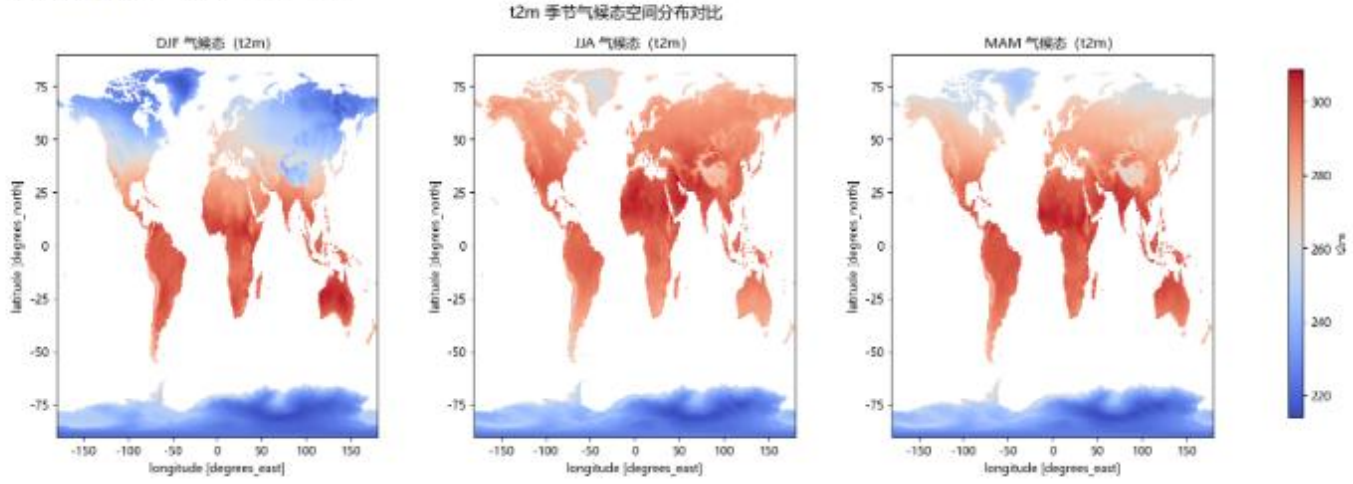


[18]:
```
# 图3：变量的数值分布直方图
data_flat = data.values.flatten()  # 展平数据为一维数组
plt.figure(figsize=(10, 6))
plt.hist(data_flat, bins=50, density=True, alpha=0.7, color='purple')
plt.title(f'{var_name} 数值分布直方图')
plt.xlabel(f'{var_name} 数值')
plt.ylabel('概率密度')
plt.grid(alpha=0.3)
plt.show()
```

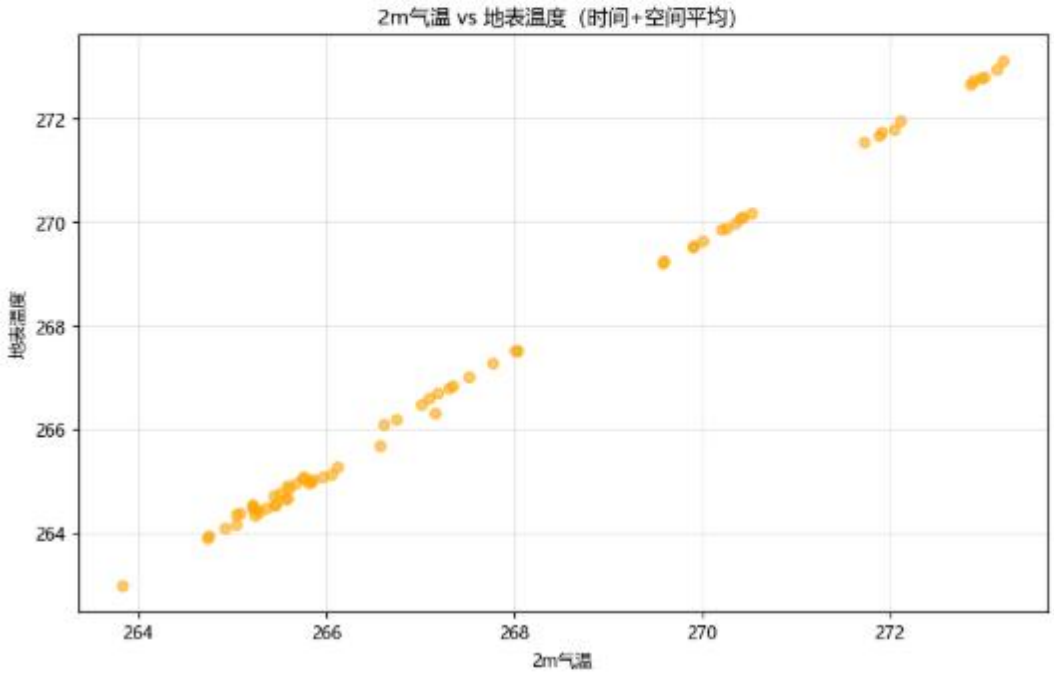


[27]:
```
# 图4：多季节气候态空间分布对比
climatology_season = ds[var_name].groupby(f'{time_dim}.season').mean(dim=time_dim)
print("数据集实际季节标签：", climatology_season.season.values)
seasons = list(climatology_season.season.values)

# 调整子图布局为1×3（因数据集仅含季节数量为3）
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(18, 6))
axes = axes.flatten()

for i, season in enumerate(seasons):
    clim = climatology_season.sel(season=season)
    im = clim.plot(ax=axes[i], cmap='coolwarm', add_colorbar=False)
    axes[i].set_title(f'{season} 气候态 ({var_name}) ')

# 统一颜色条
cbar_ax = fig.add_axes([0.95, 0.15, 0.01, 0.7])  # 对比一开始高的缩小了颜色条宽度和调整水平位置
fig.colorbar(im, cax=cbar_ax, label=var_name)
plt.subplots_adjust(right=0.9, wspace=0.3)  # 预留右侧空间给颜色条，同时调整子图间距
plt.suptitle(f'{var_name} 季节气候态空间分布对比', fontsize=14)  # 适当减小标题字号
plt.show()
```

数据集实际季节标签： ['DJF' 'JJA' 'MAM']



[20]:
```python
# 图5：t2m与skt的散点图
if 'skt' in ds.variables:
    skt = ds['skt']
    t2m_mean = data.mean(dim=['latitude', 'longitude'])
    skt_mean = skt.mean(dim=['latitude', 'longitude'])
    plt.figure(figsize=(10, 6))
    plt.scatter(t2m_mean, skt_mean, color='orange', alpha=0.6)
    plt.title('2m气温 vs 地表温度（时间+空间平均）')
    plt.xlabel('2m气温')
    plt.ylabel('地表温度')
    plt.grid(alpha=0.3)
    plt.show()
```



[ ]: