目录

1. 厄尔尼诺 3.4 指数

厄尔尼诺 3.4 指数是指太平洋厄尔尼诺现象的一个监测指标，通常用来评估厄尔尼诺-拉尼娜事件的发展和强度。它是通过测量太平洋赤道区域（通常是纬度 5 度北至 5 度南，经度 120 度西至 170 度西）的海温 anomalies（温度异常）来确定的。

在正常情况下，赤道太平洋地区的海表温度呈现周期性的变化，形成厄尔尼诺和拉尼娜两种状态。厄尔尼诺事件表现为赤道太平洋地区海水温度升高，而拉尼娜事件则表现为海水温度下降。

厄尔尼诺 3.4 指数是一个重要的指标，因为太平洋厄尔尼诺现象对全球气候产生广泛影响。它可以导致极端天气事件，如洪水、干旱和风暴，影响全球的降水和气温分布。因此，监测和理解厄尔尼诺事件对气象和气候学家以及决策者都具有重要意义。

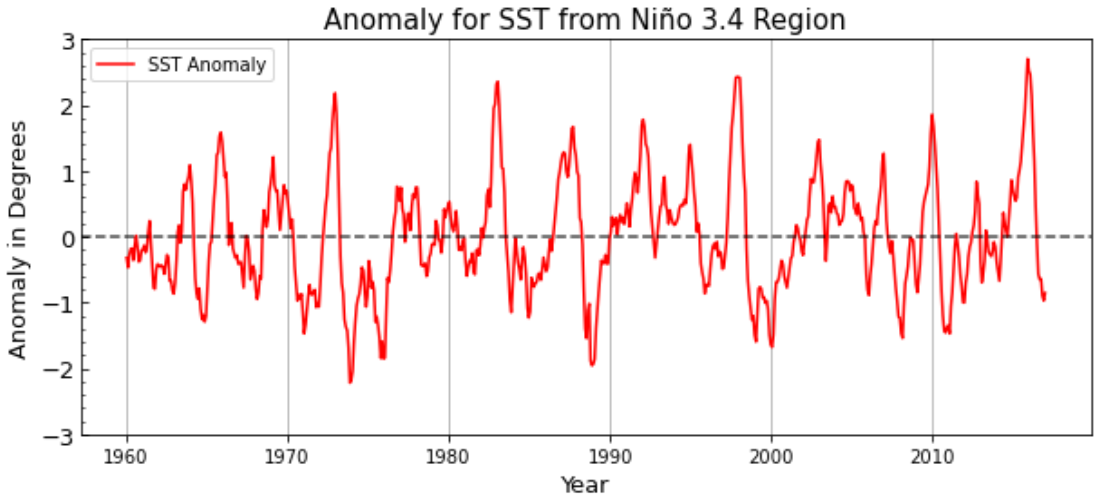
1.1 从厄尔尼诺 3.4 区域计算 SST 的月气候学，并从 SST 时间序列中减去气候学以获得异常。

关键代码：

```
# 创建图形对象
fig, ax = plt.subplots(figsize=(10, 4))
ax.plot(ano['time'], ano, color='r', label='SST Anomaly')

# 为图形图像创建网格
ax.grid(axis='x')
ax.tick_params(axis='y', labelsize=13, direction='in', length=4)
ax.yaxis.set_minor_locator(MultipleLocator(0.2))
ax.tick_params(axis="y", direction="in", which="minor", length=2)

# 创建坐标轴 y 和刻度线
ax.set_ylim(-3, 3)
ax.set_yticks(np.arange(-3, 3.2, 1))

# 在 y=0 处绘制基准线
ax.axhline(y=0, color='k', linestyle='dashed', alpha=0.7)

# 创建轴标签和标题
```

```
ax.set_xlabel('Year', fontsize=13)
ax.set_ylabel('Anomaly in Degrees', fontsize=13)
ax.set_title('Anomaly for SST from Niño 3.4 Region', fontsize=15)

# 添加图例
ax.legend()

plt.show()
```

运行结果：



在给定数据集中，首先，计算厄尔尼诺 3.4 区域的平均海温。变量 sst_anom 为异常值，group_data.mean(dim='time')为气候学值。之后从海温时间序列中减去气候学，以获得异常值序列。

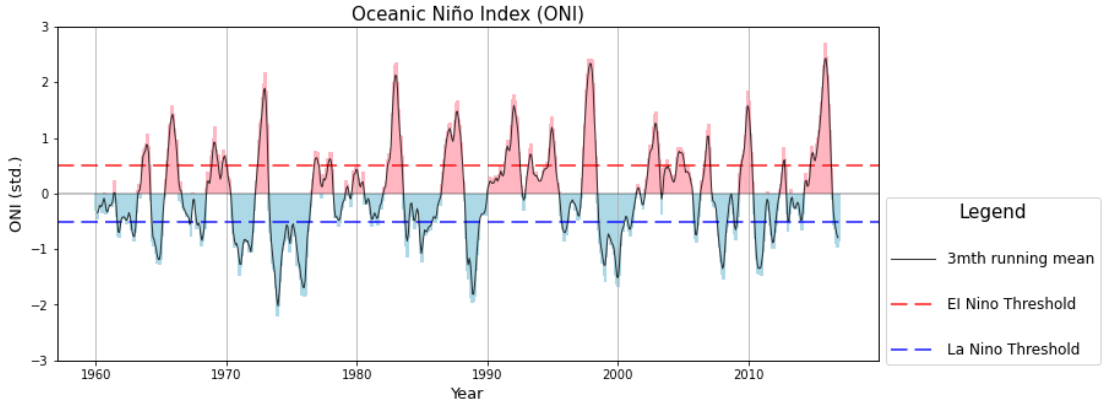
1.2 可视化展示厄尔尼诺 3.4

关键代码：

```
# 创建颜色列表
def color_mapping(value):
    return 'lightpink' if value >= 0 else 'lightblue'

# 创建图形和轴
fig, ax = plt.subplots(figsize=(12, 5))

# 绘制条形图
colors = [color_mapping(value) for value in y['sst']]
ax.bar(y.index, y['sst'], width=100, color=colors)

# 设置图形对象
ax.grid(axis='x')
ax.set_xlabel('Year', fontsize=13)
```

```
ax.set_ylabel('ONI (std.)', fontsize=13)
ax.set_title('Oceanic Niño Index (ONI)', fontsize=15)
ax.axhline(y=0, color='k', linestyle='-', alpha=0.3)
ax.set_ylim(-3, 3)
ax.set_yticks(np.arange(-3, 3.2, 1))

# 添加阈值线和图例
ax.axhline(y=0.5, color='r', linestyle=(0, (9, 4)), alpha=0.9, label='EI Nino Threshold')
ax.axhline(y=-0.5, color='b', linestyle=(0, (9, 4)), alpha=0.9, label='La Nino Threshold')


# 绘制条形图描边
ax2 = ax.twinx()
ax2.plot(y1.index, y1['sst'], color='k', label='3mth running mean', alpha=0.8, linewidth=1)
ax2.set_ylim(ax.get_ylim())
ax2.set_yticks([])

# 设置 Legend
lines, labels = ax2.get_legend_handles_labels()
lines2, labels2 = ax.get_legend_handles_labels()
legend = ax2.legend(lines + lines2, labels + labels2, title='Legend',
                    handlelength=3, bbox_to_anchor=(1, 0.51), loc=2, borderaxespad=0.5, fontsize=12, labelspacing=2)
legend.set_title("Legend", prop={"size": 15})

plt.show()
```

运行结果：



根据代码运行结果，可以找出大约 11 个 EI 厄尔尼诺事件和 8 个 La 厄尔尼诺事件。

2. 地球能量平衡

“地球能量平衡”是指地球与外部环境之间的能量交换过程。地球能量平衡是通过吸收、反射和辐射的方式来维持的。

地球接收来自太阳的能量，这包括可见光和其他形式的电磁辐射。这些能量主要在地球表面被吸收，使地球变得温暖。然而，地球也以辐射的形式向外部空间释放能量，主要是红外辐射。

地球能量平衡的关键概念是输入与输出之间的平衡。如果地球吸收的能量多于它辐射出去的能量，就会导致地球变暖，反之亦然。这个平衡是由大气、云层、海洋和陆地表面之间的复杂相互作用调节的。

总体而言，地球能量平衡是一个动态的过程，而气候系统的变化可能会对这个平衡产生影响。科学家使用各种观测和模型来研究和了解地球的能量平衡，以更好地理解气候变化和全球气候系统的运作。

本节将分析来自 NASA 的 CERES 项目的大气层顶部（TOA）辐射数据。

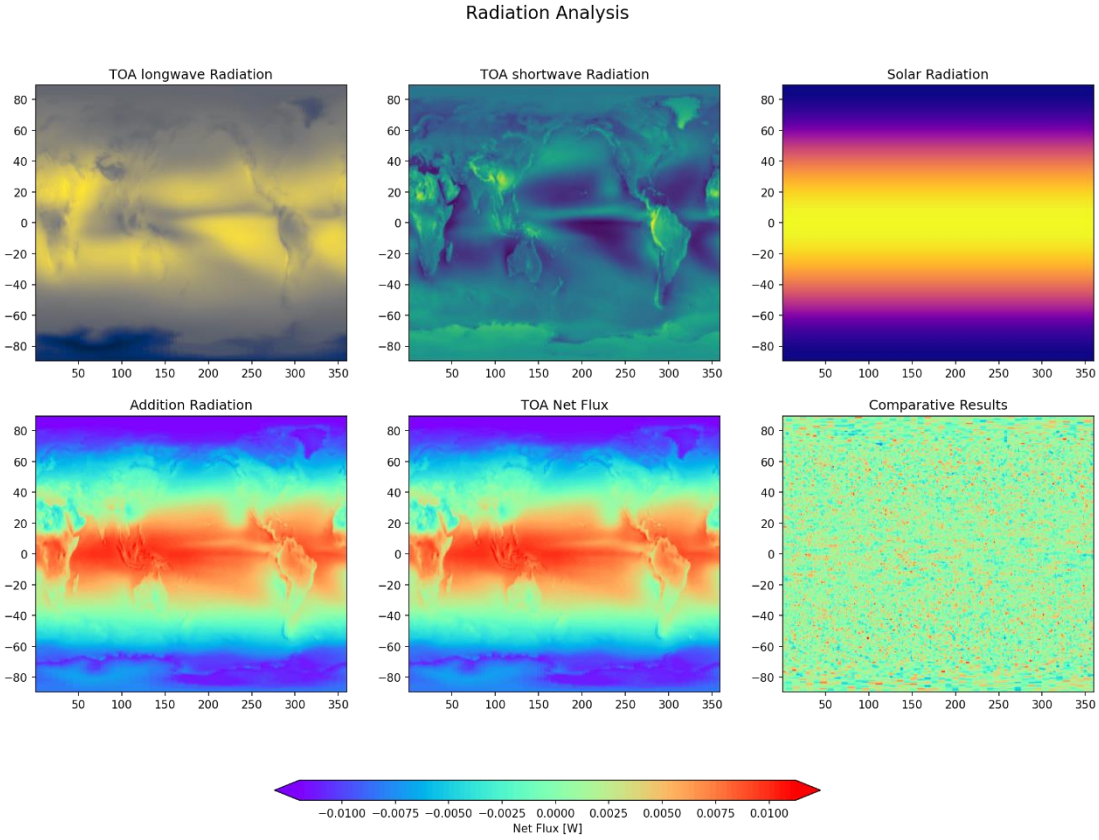
2.1 绘制全天条件下的时间平均 TOA 长波、短波和太阳辐射的 2D 图。将上述三个变量相加，并（直观地）验证它们是否等于 TOA 净通量。

关键代码：

```
# 数据读取
toa_lw_all_mon = ds.toa_lw_all_mon.mean(dim='time')
toa_sw_all_mon = ds.toa_sw_all_mon.mean(dim='time')
solar_mon = ds.solar_mon.mean(dim='time')
toa_net_all_mon = ds.toa_net_all_mon.mean(dim='time')

# 创建图形和轴
fig, axes = plt.subplots(2, 3, figsize=(17, 10), dpi=150)

# 画板基本设置
titles = ["TOA longwave Radiation", "TOA shortwave Radiation", "Sol
ar Radiation", "Addition Radiation", "TOA Net Flux", "Comparative R
esults"]
data = [toa_lw_all_mon, toa_sw_all_mon, solar_mon, solar_mon - toa_
sw_all_mon - toa_lw_all_mon, toa_net_all_mon, solar_mon - toa_sw_al
l_mon - toa_lw_all_mon - toa_net_all_mon]
cmaps = ['cividis', 'viridis', 'plasma', 'rainbow', 'rainbow','rain
bow']

# 循环绘制 6 个子图
```

```
for ax, title, data, cmap in zip(axes.flatten(), titles, data, cmaps):
    im = ax.contourf(lon, lat, data, cmap=cmap, levels=100, extend='both')
    ax.set_title(title, fontdict={'fontsize': 12})

# 绘制色条图例
cax = fig.add_axes([0.3, -0.01, 0.4, 0.025])
fig.colorbar(im, cax=cax, orientation='horizontal', extend='both').set_label('Net Flux [W]')

plt.suptitle("Radiation Analysis", fontsize=16)
plt.show()
```

运行结果：



在该段代码中，子图 1 为 TOA 长波辐射，子图 2 为 TOA 短波辐射，子图 3 为太阳辐射，子图 4 为附加辐射，子图 5 为 TOA 净通量，子图 6 为子图 4 和子图 5 比较结果。根据色条，子图 4 和子图 5 包含数据基本相等，验证了添加量等同于 TOA 净通量。

2.2 计算 TOA 入射太阳波、出射长波和出射短波，并验证其与给定漫画中数值大致相符

关键代码：

```
weights = np.cos(np.deg2rad(ds.lat))

# 对数据集进行权重平均，创建权重数组
toa_lw_all_mon_gr = ds.toa_lw_all_mon.weighted(weights).mean(dim=['
time','lon','lat'])
toa_sw_all_mon_gr = ds.toa_sw_all_mon.weighted(weights).mean(dim=['
time','lon','lat'])
solar_mon_gr = ds.solar_mon.weighted(weights).mean(dim=['time','lon
','lat'])

# 打印输出各个全球平均的辐射分量的数值
print("outgoing longwave:",toa_lw_all_mon_gr.values,'W·m^-2')
print("outgoing shortwave:",toa_sw_all_mon_gr.values,'W·m^-2')
print("TOA incoming solar:",solar_mon_gr.values,'W·m^-2')
```

运行结果：

```
outgoing longwave: 240.33322 W·m^-2
outgoing shortwave: 98.93702 W·m^-2
TOA incoming solar: 340.3247 W·m^-2
```

对比给定数据：

```
outgoing longwave: 240.26692 W·m^-2
outgoing shortwave: 99.13806 W·m^-2
TOA incoming solar: 340.28326 W·m^-2
```

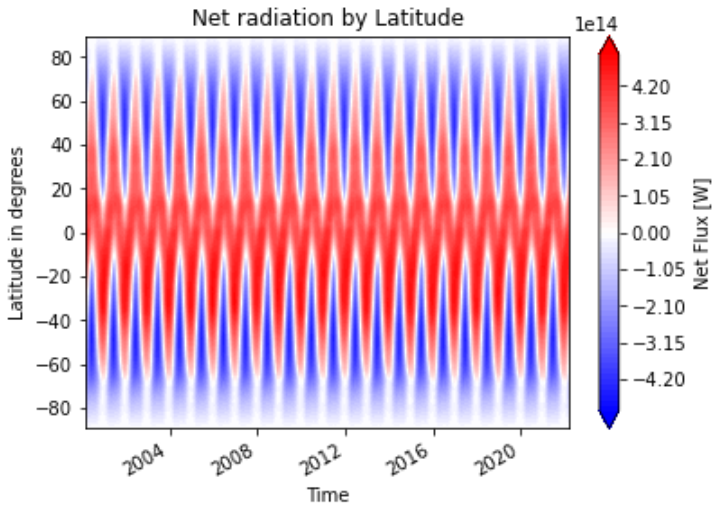
结论：TOA 太阳辐射入射角、长波出射角和短波出射角与给定的漫画大致吻合。

2.3 计算并绘制每个 1 度纬度带的净辐射总量。

关键代码：

```
# 'ds'是包含平均辐射的xarray 数据，计算净辐射总量
p = ds.toa_net_all_mon.weighted(weights).sum(dim='lon') * 111000**2

# 绘图
p.plot.contourf(x='time', levels=100, cmap='bwr',extend = 'both').c
olorbar.set_label('Net Flux [W]')
plt.ylabel('Latitude in degrees')
plt.xlabel('Time')
plt.title('Net radiation by Latitude')
plt.show()
```

运行结果：



使用 ds.toa_net_all_mon.mean(dim='lon')计算每个 1 度纬度带的净辐射总量，并使用 dataset.plot.contourf 进行绘图。

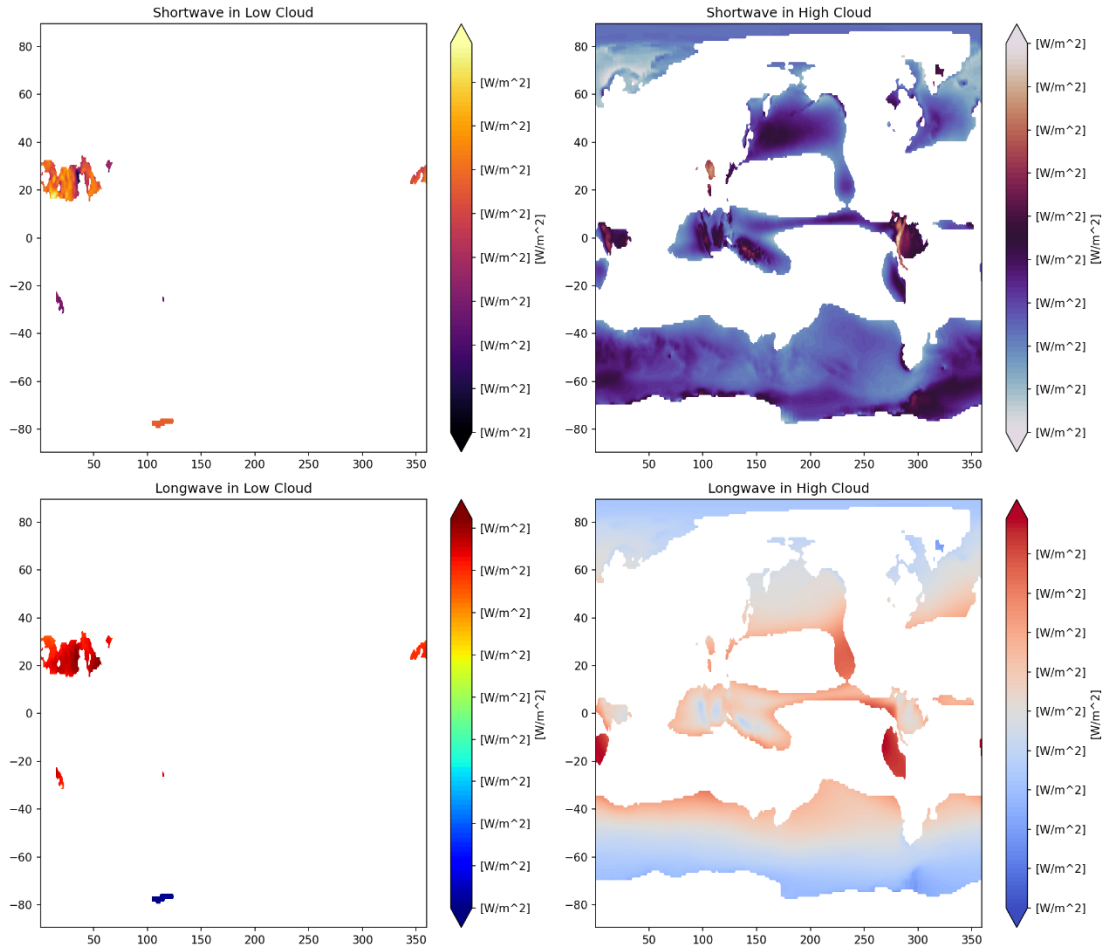
2.4 计算和绘制低云区和高云区的时间平均外发短波和长波辐射的合成图。

关键代码：

```
# 加载经纬度数据
lon = ds.lon
lat = ds.lat

# 根据云覆盖率划分低云和高云条件，云覆盖率低于等于25%为低云，大于等于75%为高云
low = ds.cldarea_total_daynight_mon.mean(dim='time') <= 25
high = ds.cldarea_total_daynight_mon.mean(dim='time')  >= 75

# 计算低云和高云条件下的全球平均短波辐射
sw_low = ds.toa_sw_all_mon.where(low).mean(dim='time')
sw_high = ds.toa_sw_all_mon.where(high).mean(dim='time')

# 计算低云和高云条件下的全球平均长波辐射
lw_low = ds.toa_lw_all_mon.where(low).mean(dim='time')
lw_high = ds.toa_lw_all_mon.where(high).mean(dim='time')

fig, axes = plt.subplots(2, 2, figsize=(14, 12), dpi=150)
fig.subplots_adjust(hspace=0.4)  # 调整子图之间的垂直间距

# 绘制子图
contourf_params = {'levels': 100, 'extend': 'both'}
im1 = axes[0, 0].contourf(lon, lat, sw_low, cmap='inferno', **conto
urf_params)
```

```
axes[0, 0].set_title("Shortwave in Low Cloud")

im2 = axes[0, 1].contourf(lon, lat, sw_high, cmap='twilight', **con
tourf_params)
axes[0, 1].set_title("Shortwave in High Cloud")

im3 = axes[1, 0].contourf(lon, lat, lw_low, cmap='jet', **contourf_
params)
axes[1, 0].set_title("Longwave in Low Cloud")

im4 = axes[1, 1].contourf(lon, lat, lw_high, cmap='coolwarm', **con
tourf_params)
axes[1, 1].set_title("Longwave in High Cloud")

# 统一色条
cbar_params = {'format': '[W/m^2]'}
for ax, im in zip(axes.flatten(), [im1, im2, im3, im4]):
    cbar = fig.colorbar(im, ax=ax, **cbar_params)
    cbar.set_label('[W/m^2]')

plt.tight_layout()
plt.show()
```

运行结果：



定义低云区为 ≤25% ,高云区为 ≥75%，本节中计算并绘制低云区和高云区的时间平均外发短波和长波辐射的合成图。

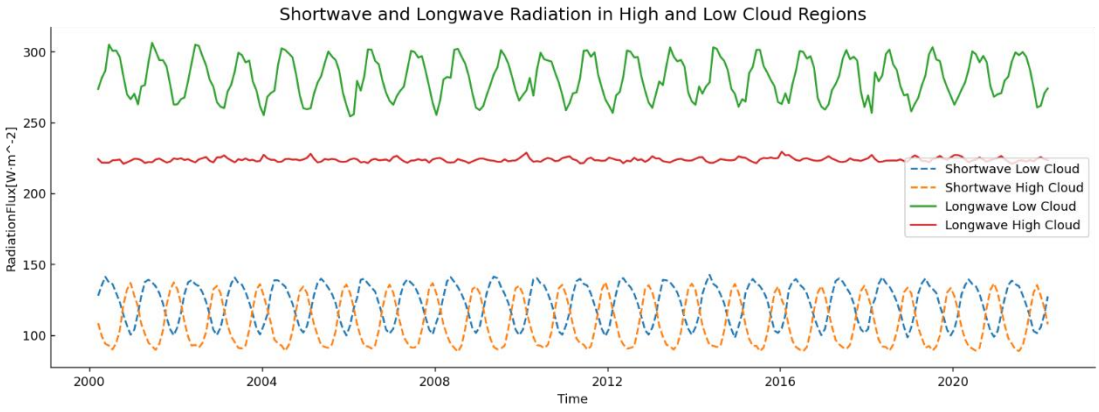
2.5 计算高低云区合成的短波和长波辐射的全球平均值。云对短波和长波辐射的总体影响是什么？

关键代码：

```
# 创建时间序列
time = lw_high_gm.time

# 创建图形和轴
fig, ax= plt.subplots(figsize=(15,5),dpi=130)

# 绘图
ax.plot(time,sw_low_gm,label='Shortwave Low Cloud',linestyle='--')
ax.plot(time,sw_high_gm,label='Shortwave High Cloud',linestyle='--
')
ax.plot(time,lw_low_gm,label='Longwave Low Cloud')
ax.plot(time,lw_high_gm,label='Longwave High Cloud')
```

```
# 设置图表属性
ax.set_xlabel('Time')
ax.set_ylabel('RadiationFlux[W·m^-2]')
ax.set_title('Shortwave and Longwave Radiation in High and Low Clou
d Regions',fontsize=14)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# 设置刻度线
ax.tick_params(axis='y', direction='in')

# 添加图例
ax.legend(loc='best')

plt.show()
```

运行结果：



根据低云高和高云高的条件创建掩码，然后与权重相乘，按照时间、月份、经度和纬度进行分组和平均，最终得到全球月平均的长波辐射。

对长波来说，云层会减少长波向外辐射的量。对短波来说，高云和低云的影响正好相反，高云区辐射的短波多，低云区辐射的短波少。在春季和冬季，低云区的短波辐射较多，而高云区的短波辐射较少。

3. 探索 netCDF 数据集

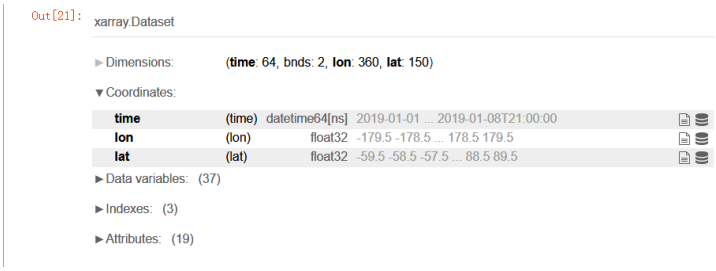
美国国家航空航天局戈达德地球科学数据和信息服务中心（GES DISC）网站上下载 netCDF 格式的数据集，本节选取了 GLDAS-CLSM 数据集（GLDAS_NOAH10_3H 2.1），下载了 64 份数据。

批量数据读取：

```
filepath = 'E:\CodeField\My_JupyterNotebook\PS3data\da'
filelist = list(Path(filepath).glob('*.nc4'))
ds = xr.open_mfdataset(filelist)
ds
```

数据直观展示：

Out[21]:



包含 37 个数据变量。

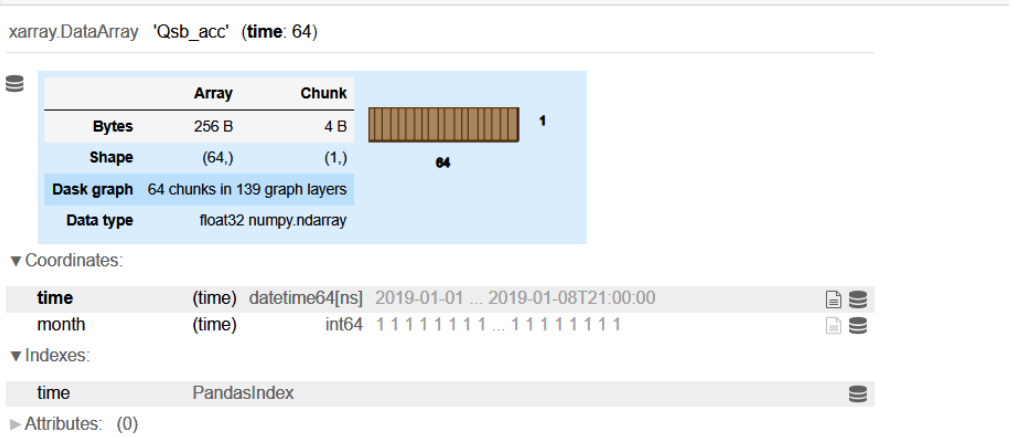
3.1 画出某个变量的时间序列

数据提取与展示关键代码：

```
dq_gro = ds.Qsb_acc.groupby('time.month')
dq_average = dq_gro - dq_gro.mean(dim='time')
dq_reav = dq_average.mean(dim=['lat','lon'])
dq_reav
```

数据集直观展示：

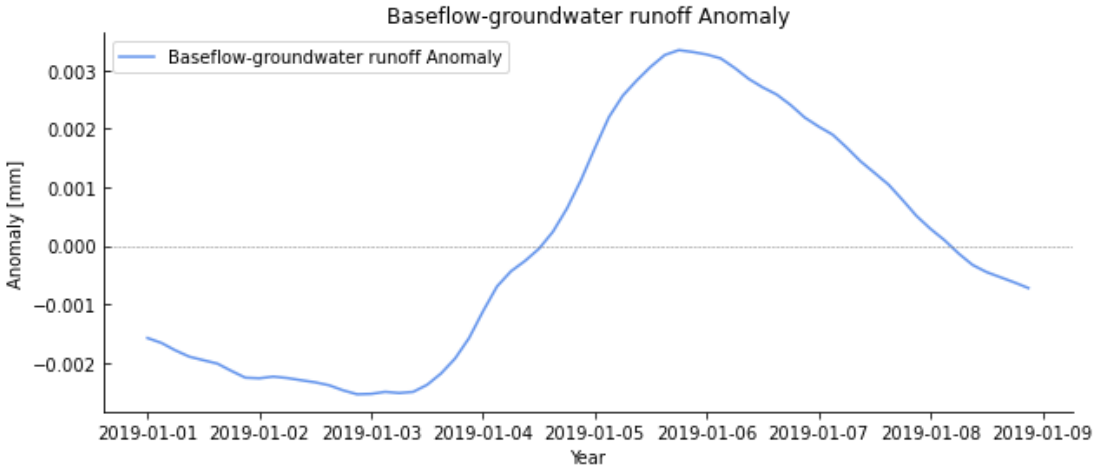
Out[22]:



关键代码：

```
g = dq_reav.copy()
fig, ax = plt.subplots(figsize=(10,4))

ax.plot(g['time'],g,color='cornflowerblue',label='Baseflow-
groundwater runoff Anomaly')
ax.axhline(0, color='gray', linestyle='--', linewidth=0.5)
ax.set_xlabel('Year')
ax.set_ylabel('Anomaly [mm]')
ax.set_title('Baseflow-groundwater runoff Anomaly')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.tick_params(axis='y', direction='in')
plt.legend()
plt.show()
```

运行结果：



3.2 使用数据集制作至少 5 个不同的图

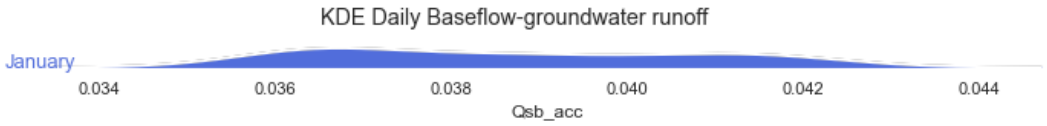
（1）Qsb_acc 数据的核密度估计

关键代码：

```
sns.set_theme(style="white", rc={"axes.facecolor": (0, 0, 0, 0), 'a
xes.linewidth':2})
palette = sns.color_palette('coolwarm', 12)
g = sns.FacetGrid(y, palette=palette, row="month", hue="month", asp
ect=9, height=1.2)
g.map_dataframe(sns.kdeplot, x="Qsb_acc", fill=True, alpha=1)
g.map_dataframe(sns.kdeplot, x="Qsb_acc", color='black')
g.map(sns.kdeplot, 'Qsb_acc',
      bw_adjust=1, clip_on=False,
      color="w", lw=2)
def label(x, color, label):
```

```
    ax = plt.gca()
    month_name = calendar.month_name[int(label)]
    ax.text(0, .2, month_name, color=color, fontsize=13,
            ha="left", va="center", transform=ax.transAxes)
g.map(label, "month")
g.fig.subplots_adjust(hspace=-.5)
g.set_titles("")
g.set(yticks=[], xlabel="Qsb_acc",ylabel='')
plt.setp(ax.get_xticklabels(), fontsize=15, fontweight='bold')
g.despine(bottom=True,left=True)
g.fig.suptitle('KDE Daily Baseflow-groundwater runoff', y=0.98)
```

运行结果：



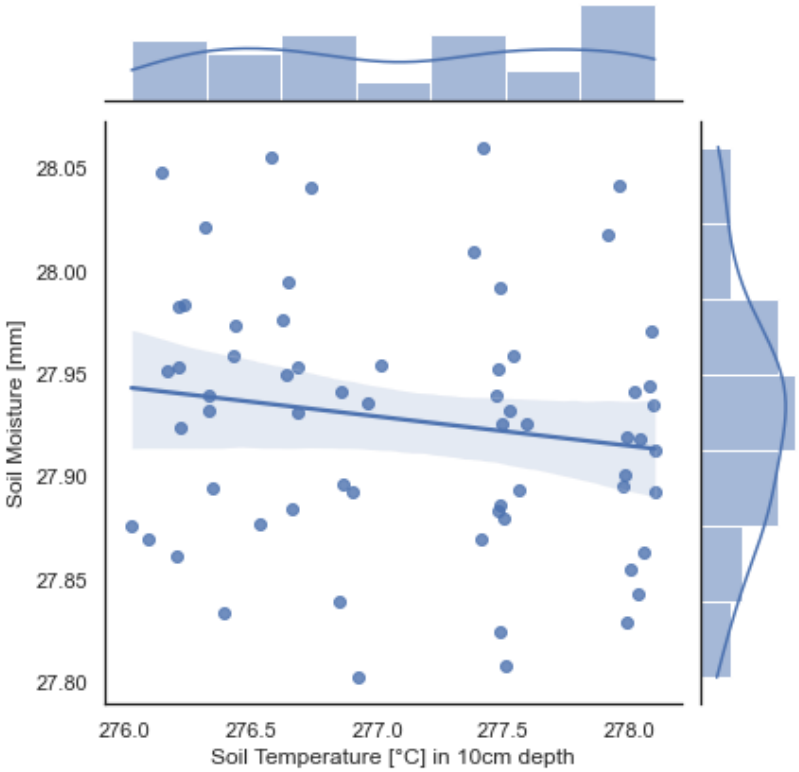
核密度估计 (KDE) 图用于显示每天的 Qsb_acc 值的分布情况。KDE 图显示 y 轴上变量的概率密度。

（2）10 厘米深度的土壤温度（°C）与土壤湿度（毫米）的相关性分析

关键代码：

```
cords_df = cords_df.dropna()
cords_df = cords_df.groupby('time').mean()
sns.set(style='white',palette='deep',color_codes=True)
sns.jointplot(x='SoilTMP0_10cm_inst', y='SoilMoi0_10cm_inst', data=
cords_df, kind='reg')
plt.xlabel('Soil Temperature [°C] in 10cm depth')
plt.ylabel('Soil Moisture [mm]')
plt.show()
```

运行结果：

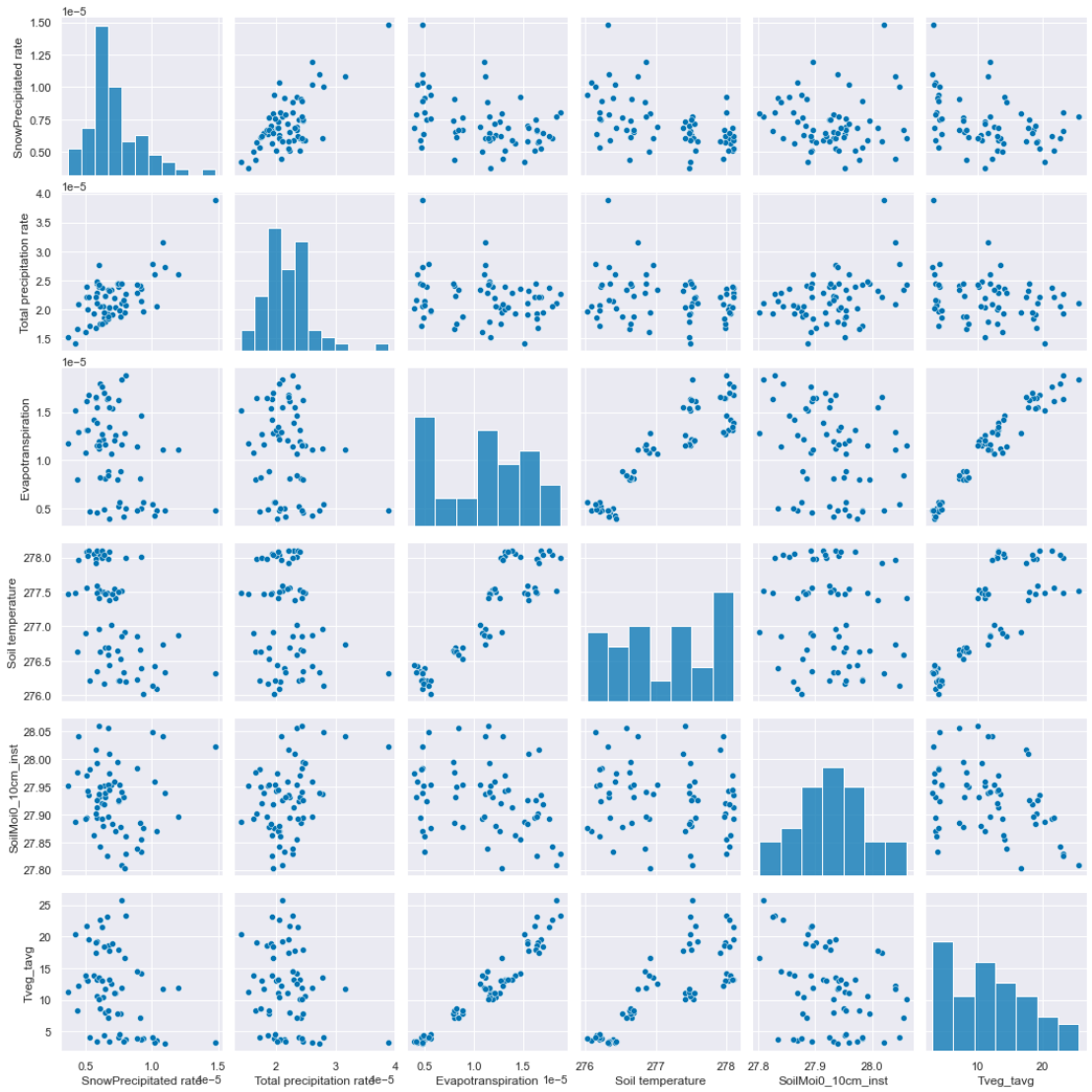


（3）6 变量相关性分析

关键代码：

```
new_column_names = {'Snowf_tavg': 'SnowPrecipitated rate', 'Rainf_f
_tavg': 'Total precipitation rate', 'Evap_tavg': 'Evapotranspiratio
n', 'SoilTMP0_10cm_inst': 'Soil temperature', 'SoilMoist_10cm_inst'
: 'Surface Soil moistur', 'TVeg_tavg': 'Transpiration'}

# 使用 rename()函数更改列名
cords_df = cords_df.rename(columns=new_column_names)

sns.set(style='darkgrid',palette='colorblind',color_codes=True)
sns.pairplot(cords_df)
```

运行结果：



Rainf_f_tavg : 总降水率

Evap_tavg : 蒸散量

SoilTMP0_10cm_inst : 地表土壤温度

SoilMoist_10cm_inst : 地表土壤湿度

TVeg_tavg : 蒸发量

通过多变量相关性分析，可以直观的观察到变量之间的关系。

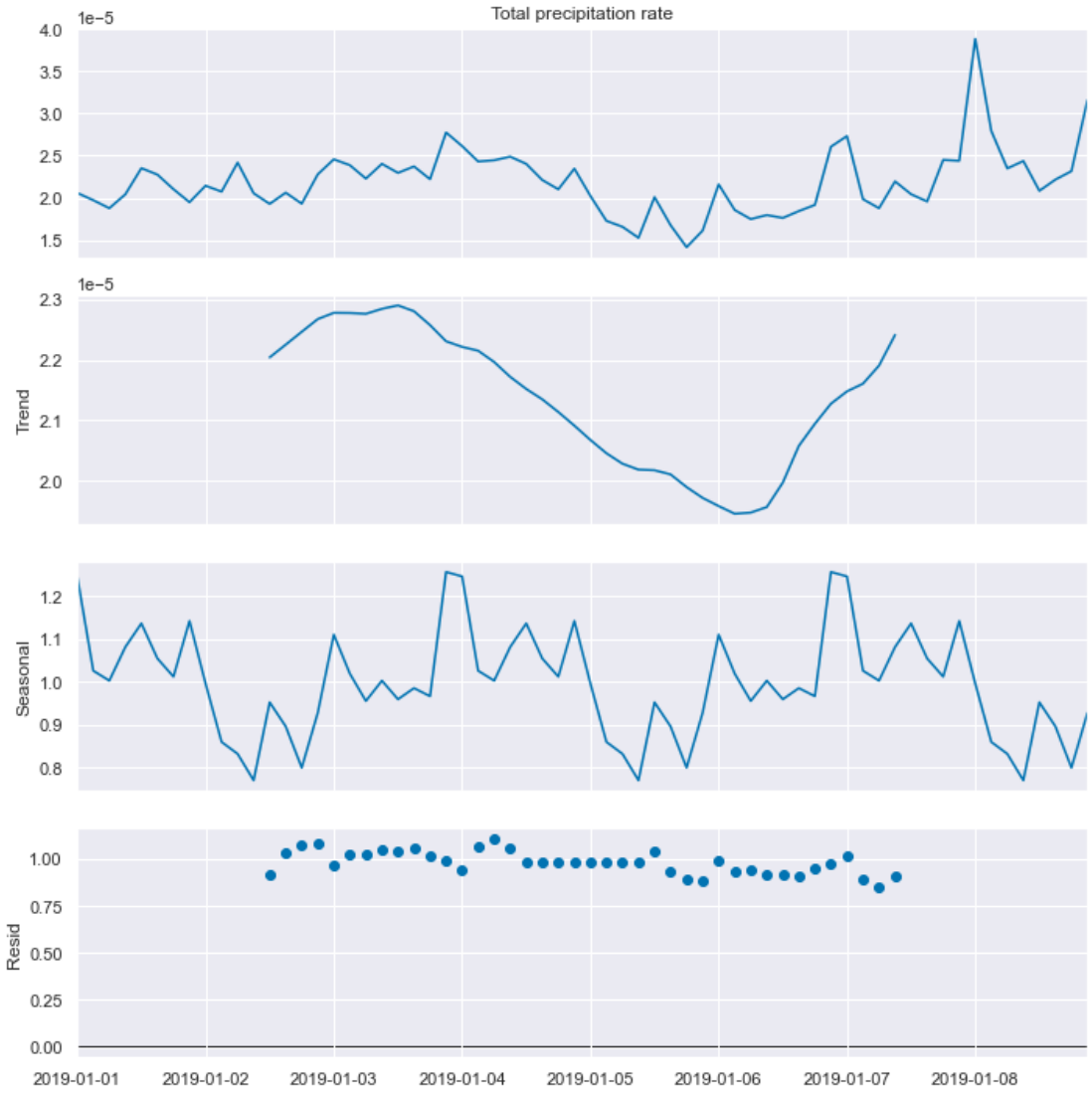
（4）时间序列数据分解

关键代码：

```
from statsmodels.tsa.seasonal import seasonal_decompose
from dateutil.parser import parse
```

```
# 导入数据
df2 = cords_df['Total precipitation rate'].copy()

# 序列分解
result = seasonal_decompose(df2, model='multiplicative')

# 绘图
plt.rcParams.update({'figure.figsize': (10,10)})
result.plot().suptitle('')
plt.show()
```

运行结果：

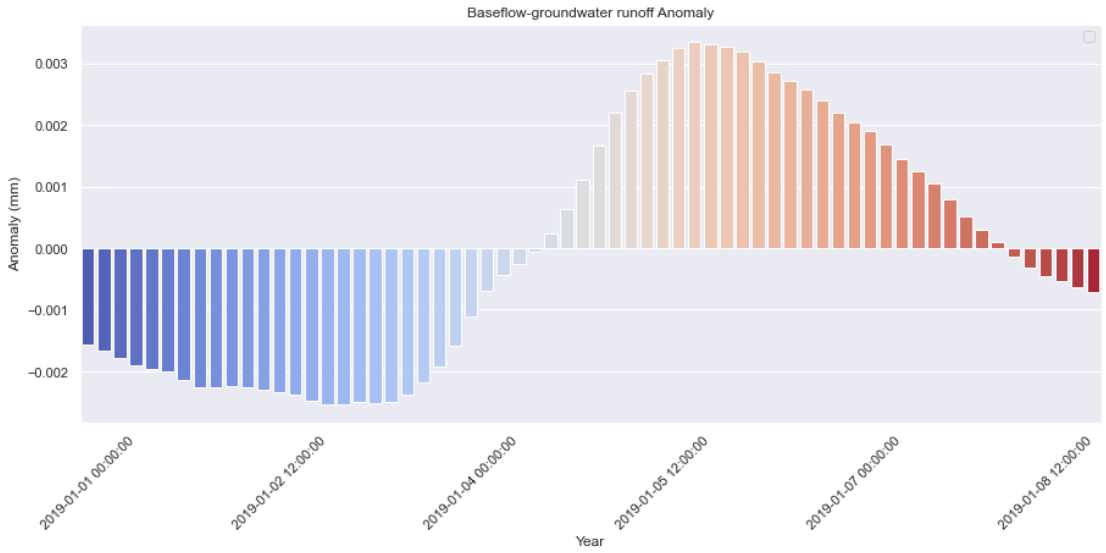


（5）Qsb_acc 绘制柱状图

关键代码：

```
import seaborn as sns
gs = dq_reav.copy().to_dataframe()
gs["date"] = gs.index
plt.figure(figsize=(15, 6))
sns.barplot(x='date',y='Qsb_acc',data=gs,palette = "coolwarm")
plt.xticks(ticks = np.arange(0, 64, 12),rotation=45)
plt.xlabel('Year')
plt.ylabel('Anomaly (mm)')
plt.title('Baseflow-groundwater runoff Anomaly')
plt.legend()
plt.show()
```

运行结果：



使用 sns.barplot 函数创建条形图，使用 plt.xticks(ticks=np.arange(0, 64, 12), rotation=45) 设置 x 轴刻度的位置和旋转角度。这里的刻度位置是每隔 12 个数据点设置一个刻度，角度设置为 45 度，可避免刻度标签重叠。

该图是 3.1 中结果的美化展示效果。