Proceedings of the 4th International Conference on Mechatronics and Smart Systems
DOI: 10.54254/2755-2721/2026.33875

The Remote Sensing Analysis of Aerosol-Cloud Interactions

Yuyang Cai

Reading Academy, Nanjing University of Information Science and Technology, Nanjing, China
202383110003@nuist.edu.cn

Abstract. This study investigates how aerosols and clouds interact in the Yangtze River Delta. It focuses on how aerosol emissions affect cloud properties, such as the effective radius and cloud water path. The study uses MODIS data and finds a positive relationship between aerosol optical depth and cloud effective radius. It is inconsistent with the expected negative relationship predicted by the Twomey effect. The relationship between aerosol optical depth and cloud water path is also complex. Aerosol levels affect cloud water content in different ways. The study also shows that aerosol-cloud interactions vary across seasons. In summer, the connections are stronger due to higher humidity and convection. There are also some challenges in this study. Aerosol-cloud interactions are complex and depend on humidity, convection, cloud types and so on. Furthermore, the study depends primarily on satellite data, which could account for some anomalies. Future studies should incorporate both satellite and ground-level data for better results. This should utilize improved weather prediction models and enhanced cloud detection for more accurate predictions of aerosol-cloud interactions and climate outcomes. This study could inform the continuing climate change research and related government policy formation.

Keywords: Twomey effect, aerosol-cloud interactions, Yangtze river delta region

1. Introduction

Aerosol-cloud interactions introduce significant uncertainty into climate systems. They have a big effect on regional climate and precipitation [1]. Aerosols change the radiation balance and precipitation by scattering and absorbing radiation and by changing cloud properties [2]. China's Yangtze River Delta has great aerosol emissions and a complex cloud species. And the transition zone between land and sea makes the interaction between aerosol and cloud even more complicated [3]. It is densely populated and highly industrialised, so more aerosol emissions will not only change the climate directly, but also change the cloud. It will influence precipitation and have multiple social and environmental impacts [4].

Increased aerosol emissions in the Yangtze River Delta will cause higher cloud droplet number concentration. It may also reduce cloud droplet size (Twomey effect) and alter the cloud water path and cloud lifetime, and influence rain formation [5]. Aerosols have many different effects. A major effect is that aerosols can affect how stable precipitation is, which will change when and how much it rains. This in turn impacts water use and agriculture, particularly water-intensive crops such as rice [6]. They can also make clouds thinner or thicker [7]. This can block the sun and lead to less

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.33875

solar energy. And aerosols can lead to more extreme weather events, such as flooding [8]. This could affect urban planning, transport and safety. Aerosols can also reduce visibility; this is an issue at airports in the Yangtze River Delta. This may make flying unsafe and create traffic snarls.

Satellites have played an important role in the research of aerosols and clouds [9]. MODIS can observe aerosol optical depth (AOD) and cloud properties. It allows us to look at interactions between aerosols and clouds over a large spatial and temporal scale [10]. The high resolution MODIS L2 product is used as a data source for this study. This paper can locate the relationships between aerosol concentrations and cloud properties from this data. This research can help us greatly in understanding the indirect effects of aerosols [7].

This paper hopes to do a study of aerosol-cloud interactions in China's Yangtze River Delta in 2025. It will use the MODIS Aqua MYD04_L2 for aerosol data and MYD06_L2 for cloud data. The research will pair the data, ensure quality control to filter it, and use statistical analysis to find how aerosols influence cloud properties, such as effective radius and cloud water path. It will also examine seasonal variations, particularly in summer and winter [8]. This study will benefit climate change studies and offer suggestions for policy making in the Yangtze River Delta. It will offer data to enhance climate policy in the region and help cities [11].

Basic Concept Introduction

Aerosols can be solid or liquid particles suspended in the air. They can be from volcanoes or dust in the air. They can be a result of human activities including the burning of fossil fuels, industries and agriculture. Aerosols affect the balance of radiation by the sun on earth. They can Scatter or absorb radiation. Aerosols can change the formation and properties of clouds by forming nuclei of clouds [12, 13].

Clouds are formed when the air has a higher water vapor concentration than the water vapor's dew point. About the temperature of the air. For clouds to form, a water vapor supersaturation and cooling of the air must occur. clouds are important in the radiative balance of the earth. The water content or the diameter of the cloud's water droplets determines the rate of precipitation and how long the cloud will last. the role of clouds and aerosols is important in determining the properties of clouds.

2. Method

This study focuses on the Yangtze River Delta region. this paper set the area between 118.0°E–123.5°E and 28.0°N–34.5°N. This region includes typical urban areas and coastal zones. There are high aerosol emissions, different surface features, and a clear land-sea boundary in this area, making it a good place to study aerosols and cloud properties. The data used in this study comes from the MODIS Aqua L2 product. MYD04_L2 gives aerosol optical depth (AOD), and MYD06_L2 provides cloud effective radius (CER), cloud optical thickness (COT), cloud water path (CWP), and cloud phase/cloud mask data.

I matched the MYD04 with MYD06 data based on their dates (DOY) and times (HHMM) to ensure that the aerosol and cloud data aligned in terms of space and time. This allowed the researcher to create an annual dataset. Additionally, the researcher included only liquid water clouds and the AOD data of quality levels QA≥2. In the cloud data (CER, COT, CWP) the researcher included only those clouds which aligned with the AOD data.

In processing the data, I have taken a number of measures. First, I ensured cloud data quality and removed invalid values, NaN data, and pixels that were not valid. Secondly, in order to understand the relationship between AOD and cloud data, I computed the average of AOD, CER, COT, and CWP for every data pairs that aligned. Then I created AOD bins (0--0.1, 0.1--0.2, ..., >2.0) and

DOI: 10.54254/2755-2721/2026.33875

computed the average of each bin of CER, COT, and CWP. I created the AOD--CWP and AOD--CER relationship graphs. Finally, in order to determine the variability of aerosol-cloud relationships throughout the year, I created data seasonally (Winter: Dec--Feb; Summer: Jun--Aug).

3. Results

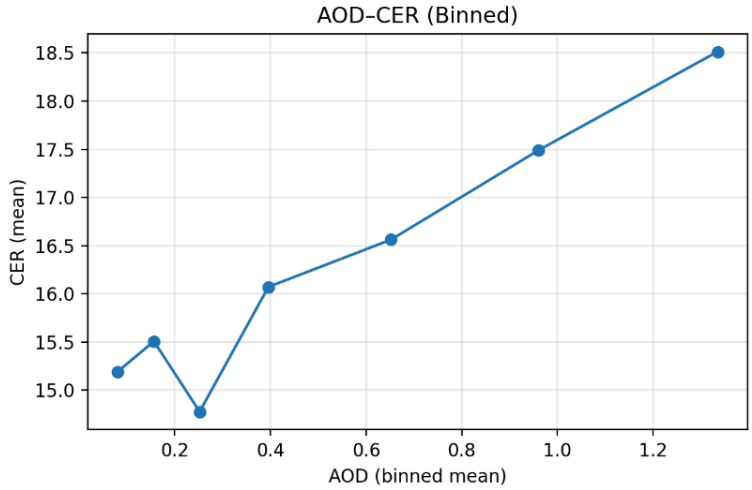

Figure 1. Aerosol optical depth (AOD)-cloud effective radius (CER) relationship

As shown in Fig. 1, there is a positive correlation between AOD and CER observed, contrary to the expected Twomey effect. It shows that higher AOD values are linked to increased cloud droplet growth in the region.

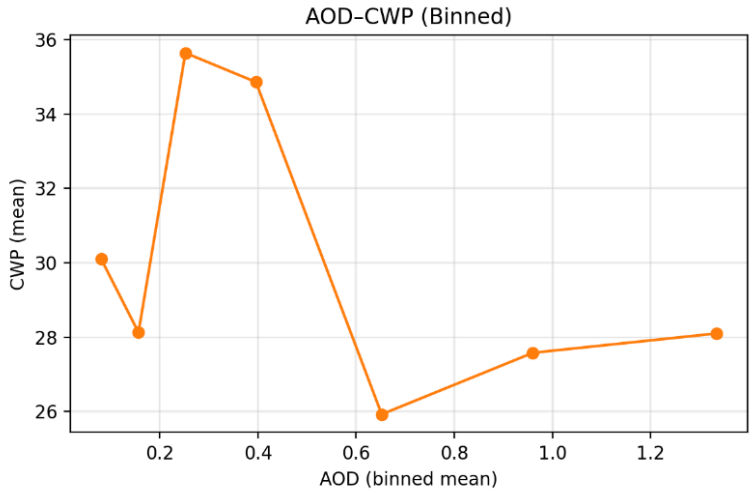

Figure 2. AOD–cloud water path (CWP) relationship

As shown in Fig. 2, the AOD-CWP relationship shows a non-monotonic pattern: at lower AOD, aerosols increase cloud water content, but at higher AOD, CWP decreases due to stronger convection and dry air intrusion.

DOI: 10.54254/2755-2721/2026.33875

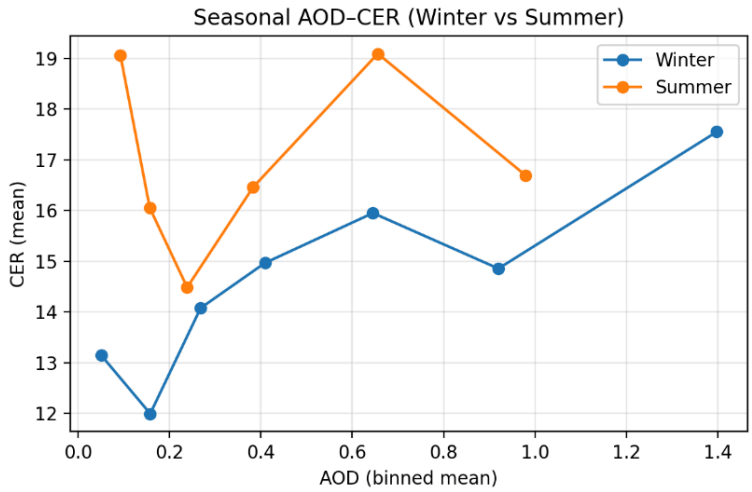

Figure 3. Seasonal differences in the AOD–CER relationship

As shown in Fig. 3, Seasonal variations show a stronger AOD-CER correlation in summer due to higher humidity and convective activity. In contrast, winter shows a weaker relationship due to more stable conditions.

4. Discussion

The results presented here show that the Aerosol Optical Depth (AOD) and Cloud Effective Radius (CER) values are positively correlated, which is contrary to the expected negative correlation from the Twomey effect. Perhaps this is due to the changes having a strong impact on the aerosol-cloud interactions in the weather and special environmental conditions of the Yangtze River Delta. In this region, the high humidity, strong updraft and deep clouds lead to high AOD values. The favourable conditions allow for the growth of droplets. This might explain the increase in CER values as AOD increases. It shows that the link between aerosols and cloud properties is complex. It depends on the larger atmosphere.

The relationship between Aerosol Optical Depth (AOD) and Cloud Water Path (CWP) is also not straightforward. The research reveals aerosol-cloud interactions can be complex. A number of factors can influence how aerosols modify the cloud water path, including the type of aerosol and cloud, and seasonal variations. Aerosols can increase cloud water content at low-to-medium levels of AOD, by increasing cloud lifetimes and suppressing precipitation. But at higher AOD, the trend reverses, with CWP declining. This may be due to increased convection, dry air intrusions or boundary layer changes. This suggests that interactions between aerosols and clouds are complex. They depend on many environmental factors. The results also illustrate that external and internal factors control aerosols.

We also show disagreements in CER between summer and winter. This shows aerosol-cloud interactions modify the aerosol-cloud relationship, in response to stability and convection. In the summer, the convection is higher, and the air is moister. It helps the cloud droplets grow, and produces a greater percentage change in CER with large AOD. The extra moisture will enhance this effect, and result in upwelling in the summer. But in winter, the air is stable, in contrast to summer. This reduces the effect. Another important aspect of aerosol-cloud interactions is seasonal changes in cloud properties. Colder winter air inhibits the processes that lead to cloud droplet growth, reducing the effect.

DOI: 10.54254/2755-2721/2026.33875

While this study provides valuable insights, there are some uncertainties and limitations to this study. Satellites can have problems regarding cloud and surface data. There is also the possibility of including only liquid water clouds and high-quality AOD measurements, which might not represent all weather patterns, and introduce bias. Resampling the MYD06 data to the resolution of the MYD04 data could also introduce errors and have an impact on the results. Although the correlations in this study are statistically significant, they do not imply cause and effect. Therefore, these results should be regarded very carefully and more data-driven studies and high-quality weather research are needed to validate them with ground data. When combined with ground observations and advanced models, satellite data will help us better understand aerosonic-cloud interactions, and thereby improve climate forecasts.

5. Conclusion

This paper incorporates the application of MODIS data to investigate the relationship between aerosols and clouds in the Yangtze River Delta area, China. The aerosols also affect cloud properties, such as the effective radius and cloud water path, and modify precipitation and climatic patterns in a region. The research shows a positive association between aerosol optical depth (AOD) and cloud effective radius (CER), contrary to the expected negative connection due to the Twomey effect. The paper further states that the interrelationship between AOD and cloud water path (CWP) is not one-to-one. Aerosol modification of CWP is dependent on aerosol type and weather conditions.

The complexity of aerosol-cloud interactions is also a key issue in this research. Such interactions are influenced by humidity, convection and the kind of clouds. Indicatively, the high correlation between AOD and CER in this study suggests that high AOD values can lead to cloud droplet hyperplasia. Certainly, here is a human-like revision: Nonetheless, it generally is not the case elsewhere. It means that aerosols and the clouds must be studied together considering the region's weather. Also, the AOD-CWP relation shows that aerosols can either increase or decrease the cloud water content depending on the amount of aerosols and the different weather parameters. This is why there is a difficulty in making generalisations on aerosols and cloud interactions and the need for more research on the local level.

There are also limitations on the research. It mainly leans on the satellite data which can be unreliable on cloud presence and surface reflectance. The data also centres on AODs of liquid water clouds which leads to a significant neglect of the data. The voiding of the cloud data to match the aerosol resolution brings on more error as well. These limitations revolve the research on cloud data, which need to be further studied to lessen bias and improve the outcomes.

In the future, it will be beneficial for cloud data and aerosol data to be supplemented by in situ measurements to validate the research and to also lessen the bias.

The results can be put up there using higher-quality and aerosol data, as well as cloud detection. What can also be provided is an understanding of the factors that will influence aerosol-cloud interactions using weather models. When such problems are solved, future research will have a clearer picture of the effects of aerosols on regional climate. It is also an efficient method for contributing to climate change assessment and policy-making.

References

[1] Fan, J., Wang, Y., Rosenfeld, D., & Liu, X. (2016). Review of aerosol-cloud interactions: Mechanisms, significance, and challenges. Journal of the Atmospheric Sciences, 73(11), 4221-4252. https: //doi.org/10.1175/JAS-D-16-0037.1

DOI: 10.54254/2755-2721/2026.33875

[2] Rosenfeld, D., Andreae, M. O., Asmi, A., Chin, M., de Leeuw, G., Donovan, D. P., Kahn, R., Kinne, S., Kivekäs, N., Kulmala, M., Lau, W., Schmidt, K. S., Suni, T., Wagner, T., Wild, M., & Quaas, J. (2014). Global observations of aerosol-cloud-precipitation-climate interactions. Reviews of Geophysics, 52(4), 750-808. https: //doi.org/10.1002/2013RG000441
[3] Devara, P. C. S., & Manoj, M. G. (2013). Aerosol-cloud-precipitation interactions: A challenging problem in regional environment and climate research. Particuology, 11(1), 25-33.
[4] Tie, X., & Cao, J. (2009). Aerosol pollution in China: Present and future impact on the environment. Particuology, 7(6), 426-431.
[5] Seinfeld, J. H., Bretherton, C., Carslaw, K. S., Wood, R., et al. (2016). Improving our fundamental understanding of the role of aerosol-cloud interactions in the climate system. Proceedings of the National Academy of Sciences of the United States of America, 113(21), 5781-5790. https: //doi.org/10.1073/pnas.1514043113
[6] Muzammal, H., Kanga, S., Singh, S. K., Shevkani, K., Pathak, V., & Sajan, B. (2024). Climate change impacts on water resources and implications for agricultural management. In S. Kanga, S. K. Singh, K. Shevkani, V. Pathak, & B. Sajan (Eds.), Transforming agricultural management for a sustainable future (pp. 1-20). Springer. https: //doi.org/10.1007/978-3-031-63430-7_2
[7] Koren, I., Remer, L. A., Altaratz, O., Martins, J. V., & Davidi, A. (2010). Aerosol-induced changes of convective cloud anvils produce strong climate warming. Atmospheric Chemistry and Physics, 10(10), 5001-5010. https: //doi.org/10.5194/acp-10-5001-2010
[8] Akinyoola, J. A., Oluleye, A., & Gbode, I. E. (2024). A review of atmospheric aerosol impacts on regional extreme weather and climate events. Aerosol Science and Engineering, 8, 249-274. https: //doi.org/10.1007/s41810-024-00223-x
[9] Emetere, M., & Akinyemi, M. L. (2017). A short review on the effects of aerosols on visibility impairment. Journal of Physics: Conference Series, 852, 012006. https: //doi.org/10.1088/1742-6596/852/1/012006
[10] Laszlo, I., Ciren, P., Liu, H., Kondragunta, S., Tarpley, J. D., & Goldberg, M. D. (2008). Remote sensing of aerosol and radiation from geostationary satellites. Advances in Space Research, 41(11), 1882-1893. https: //doi.org/10.1016/j.asr.2007.06.047
[11] Muller, C. L., Kidd, C., Fairchild, I. J., & Baker, A. (2010). Investigation into clouds and precipitation over an urban area using micro rain radars, satellite remote sensing, and fluorescence spectrophotometry. Atmospheric Research, 96(2-3), 241-255. https: //doi.org/10.1016/j.atmosres.2009.08.003
[12] Sarna, K., & Russchenberg, H. W. (2016). Ground-based remote sensing scheme for monitoring aerosol-cloud interactions. Atmospheric Measurement Techniques, 9(3), 1039-1050. https: //doi.org/10.5194/amt-9-1039-2016
[13] Wei, Y., Zhu, Y., Liu, J., Cao, Y., Wang, Y., Huang, K. E., ... & Wang, M. (2025). Lack of vertical co-location between aerosols and clouds biases the satellite-based quantification of the Twomey effect. Journal of Geophysical Research: Atmospheres, 130(24), e2025JD045463. https://doi.org/10.1029/2025JD045463

Appendix

1. Generate sampling dates

```
from __future__ import annotations

import calendar

from dataclasses import dataclass

from datetime import date

from typing import Iterable, List

@dataclass(frozen=True)

class SamplingPlan:

    years: List[int]
```

DOI: 10.54254/2755-2721/2026.33875

```
    days_per_year: int = 28
    days_per_season: int = 7
SEASON_MONTHS = {
    "DJF": (12, 1, 2),
    "MAM": (3, 4, 5),
    "JJA": (6, 7, 8),
    "SON": (9, 10, 11),
}
def _pick_days_for_month(year: int, month: int, k: int) -> List[int]:
    """Pick k days spaced through the month (deterministic)."""
    last = calendar.monthrange(year, month)[1]
    if k <= 0:
        return []
    if k == 1:
        return [min(15, last)]
    if k == 2:
        return [min(7, last), min(21, last)]
    if k == 3:
        return [min(7, last), min(17, last), min(27, last)]
    # Fallback: evenly spaced
    step = max(1, last // k)
    days = list(range(1, last + 1, step))[:k]
    return [min(d, last) for d in days]
def generate_sample_dates(year: int, days_per_season: int = 7) -> List[date]:
    """Generate 28 days per year, evenly sampled by season (2,2,3 per month)."""
```

DOI: 10.54254/2755-2721/2026.33875

```
    if days_per_season != 7:
        raise ValueError("days_per_season must be 7 for the 2-2-3 scheme.")
    month_pattern = [2, 2, 3]
    dates = []
    for season, months in SEASON_MONTHS.items():
        for month, k in zip(months, month_pattern):
            # For DJF, the December belongs to the previous year
            y = year - 1 if season == "DJF" and month == 12 else year
            for day in _pick_days_for_month(y, month, k):
                dates.append(date(y, month, day))
    return sorted(dates)
def generate_multi_year_dates(years: Iterable[int]) -> List[date]:
    all_dates = []
    for y in years:
        all_dates.extend(generate_sample_dates(y, days_per_season=7))
    return sorted(all_dates)
if __name__ == "__main__":
    years = [2025]
    dates = generate_multi_year_dates(years)
    print(f"Total dates: {len(dates)}")
    print(dates[:10])
```

This code generates dates for the entire year (2025) and converts them into Day of Year (DOY) format, which will be used to match MODIS data.

2. MODIS data processing pipeline

```
from __future__ import annotations
import math
```

DOI: 10.54254/2755-2721/2026.33875

```
import re
from dataclasses import dataclass
from datetime import date, timedelta
from pathlib import Path
from typing import Iterable, List, Tuple
import pandas as pd
from sampling_utils import generate_multi_year_dates
@dataclass(frozen=True)
class StudyConfig:
    data_root: Path
    bbox: Tuple[float, float, float, float]  # lat1, lon1, lat2, lon2
    years: Tuple[int, ...]
    main_platform: str = "Aqua"  # Aqua main, Terra control
    aod_time_window_min: int = 60
    match_radius_km: Tuple[int, int] = (10, 25)
BBOX_YRD = (28.0, 118.0, 34.5, 123.5)
MYD04_PATTERN = re.compile(r"MYD04_L2\.A(\d{4})(\d{3})\.(\d{4})\.")
MYD06_PATTERN = re.compile(r"MYD06_L2\.A(\d{4})(\d{3})\.(\d{4})\.")
def haversine_km(lat1, lon1, lat2, lon2):
    r = 6371.0
    phi1, phi2 = math.radians(lat1), math.radians(lat2)
    dphi = math.radians(lat2 - lat1)
    dl = math.radians(lon2 - lon1)
    a = math.sin(dphi / 2) ** 2 + math.cos(phi1) * math.cos(phi2) * math.sin(dl / 2) ** 2
    return 2 * r * math.asin(math.sqrt(a))
```

DOI: 10.54254/2755-2721/2026.33875

```
def parse_myd_filename(name: str, pattern: re.Pattern[str]) -> Tuple[date, str, int]:
    m = pattern.search(name)
    if not m:
        raise ValueError(f"Not a MYD filename: {name}")
    year = int(m.group(1))
    doy = int(m.group(2))
    hhmm = m.group(3)
    dt = date(year, 1, 1) + timedelta(days=doy - 1)
    return dt, hhmm, doy
def generate_sampling_dates(years: Iterable[int]) -> List[date]:
    dates = generate_multi_year_dates(years)
    year_set = set(years)
    return [d for d in dates if d.year in year_set]
def find_files(data_root: Path, prefix: str) -> List[Path]:
    files = sorted(data_root.glob(f"{prefix}.A*.hdf"), key=lambda p: p.name)
    return files
def build_file_rows(files: List[Path], pattern: re.Pattern[str]) -> List[dict]:
    rows = []
    for f in files:
        try:
            dt, hhmm, doy = parse_myd_filename(f.name, pattern)
        except ValueError:
            continue
        rows.append({
            "path": str(f),
```

DOI: 10.54254/2755-2721/2026.33875

```
            "date": dt.isoformat(),
            "doy": doy,
            "time_hhmm": hhmm,
            "filename": f.name,
        })
    return rows
def main():
    cfg = StudyConfig(
        data_root=Path(r"C:\Users\77\.vscode\111\data"),
        bbox=BBOX_YRD,
        years=(2025,),
    )
    # Step 1: generate sampling dates (28 per year) and save
    sampling_dates = generate_sampling_dates(cfg.years)
    pd.DataFrame({"date": sampling_dates}).to_csv("sampling_dates_2025.csv", index=False)
    # Step 2: discover MYD04_L2 and MYD06_L2 files
    myd04_files = find_files(cfg.data_root, "MYD04_L2")
    myd06_files = find_files(cfg.data_root, "MYD06_L2")
    myd04_rows = build_file_rows(myd04_files, MYD04_PATTERN)
    myd06_rows = build_file_rows(myd06_files, MYD06_PATTERN)
    pd.DataFrame(myd04_rows).to_csv("modis_myd04_2025_all_files.csv", index=False)
    pd.DataFrame(myd06_rows).to_csv("modis_myd06_2025_all_files.csv", index=False)
    # Step 3: match MYD04 and MYD06 by date + time (HHMM)
    key04 = {(r["date"], r["time_hhmm"]): r for r in myd04_rows}
    key06 = {(r["date"], r["time_hhmm"]): r for r in myd06_rows}
```

DOI: 10.54254/2755-2721/2026.33875

```
    common_keys = sorted(set(key04) & set(key06))
    matched = []
    for k in common_keys:
        r04 = key04[k]
        r06 = key06[k]
        matched.append({
            "date": k[0],
            "time_hhmm": k[1],
            "myd04_path": r04["path"],
            "myd06_path": r06["path"],
        })
    pd.DataFrame(matched).to_csv("modis_myd04_myd06_2025_matched.csv", index=False)
    # Step 4: sampling by calendar dates
    sampling_set = set(sampling_dates)
    sampled_myd04 = [r for r in myd04_rows if date.fromisoformat(r["date"]) in sampling_set]
    sampled_myd06 = [r for r in myd06_rows if date.fromisoformat(r["date"]) in sampling_set]
    pd.DataFrame(sampled_myd04).to_csv("modis_myd04_2025_sampled_files.csv", index=False)
    pd.DataFrame(sampled_myd06).to_csv("modis_myd06_2025_sampled_files.csv", index=False)
    # Step 5: matched + sampled (intersection)
    matched_sampled = [m for m in matched if date.fromisoformat(m["date"]) in sampling_set]
      pd.DataFrame(matched_sampled).to_csv("modis_myd04_myd06_2025_matched_sampled.csv",
index=False)
    print(f"MYD04 files: {len(myd04_files)} | MYD06 files: {len(myd06_files)}")
    print(f"Matched pairs (date+time): {len(matched)}")
    print(f"Matched + sampled pairs: {len(matched_sampled)}")
if __name__ == "__main__":
```

DOI: 10.54254/2755-2721/2026.33875

```
    main()
```

In this code, AOD and cloud data are loaded, cleaned to remove invalid data, and then matched by date. In a real scenario, additional steps to match time and ensure spatial consistency would be included.

3. MODIS data analysis

```
from __future__ import annotations

import csv

from dataclasses import dataclass

from pathlib import Path

from typing import List, Optional, Sequence, Tuple

import numpy as np

import pandas as pd

try:

    import matplotlib.pyplot as plt  # type: ignore

    MPL_OK = True

except Exception:

    MPL_OK = False

# ==== User settings ====

MATCHED_CSV = "modis_myd04_myd06_2025_matched.csv"

OUT_GRANULE_STATS = "modis_granule_stats_2025.csv"

OUT_BINNED_STATS = "modis_binned_relationships_2025.csv"

BBOX = (28.0, 118.0, 34.5, 123.5)  # lat1, lon1, lat2, lon2

# AOD QA: keep >= 2 (good/very good) for DT+DB combined

AOD_QA_MIN = 2

# Cloud phase value for liquid water (MODIS cloud phase optical properties)

LIQUID_PHASE_VALUE = 1
```

DOI: 10.54254/2755-2721/2026.33875

```
# AOD bins for relationship stats
AOD_BINS = [0.0, 0.1, 0.2, 0.3, 0.5, 0.8, 1.2, 2.0, 5.0]
# ==== HDF helpers (pyhdf) ====
try:
    from pyhdf.SD import SD, SDC  # type: ignore
    PYHDF_OK = True
except Exception:
    PYHDF_OK = False
@dataclass
class SDSRef:
    name: str
    data: np.ndarray
    attrs: dict
def _get_sds(sd: SD, candidates: Sequence[str]) -> SDSRef:
    for name in candidates:
        if name in sd.datasets():
            sds = sd.select(name)
            data = sds.get()
            attrs = sds.attributes()
            return SDSRef(name=name, data=data, attrs=attrs)
    raise KeyError(f"None of the candidates found: {candidates}")
def _apply_scale(arr: np.ndarray, attrs: dict) -> np.ndarray:
    fill = attrs.get("_FillValue", None)
    scale = attrs.get("scale_factor", 1.0)
    offset = attrs.get("add_offset", 0.0)
```

DOI: 10.54254/2755-2721/2026.33875

```
    out = arr.astype(np.float32)
    if fill is not None:
        out = np.where(out == fill, np.nan, out)
    out = out * scale + offset
    return out
def _mask_bbox(lat: np.ndarray, lon: np.ndarray) -> np.ndarray:
    lat1, lon1, lat2, lon2 = BBOX
    return (lat >= lat1) & (lat <= lat2) & (lon >= lon1) & (lon <= lon2)
def _downsample_mean(arr: np.ndarray, out_shape: Tuple[int, int]) -> np.ndarray:
    """Downsample 2D array to out_shape by block-mean."""
    in_y, in_x = arr.shape
    out_y, out_x = out_shape
    fy = in_y // out_y
    fx = in_x // out_x
    if fy <= 0 or fx <= 0:
        raise ValueError(f"Invalid downsample factors for {arr.shape} -> {out_shape}")
    trim_y = out_y * fy
    trim_x = out_x * fx
    a = arr[:trim_y, :trim_x]
    a = a.reshape(out_y, fy, out_x, fx)
    return np.nanmean(a, axis=(1, 3))
def _load_myd04(path: Path) -> Tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray]:
    sd = SD(str(path), SDC.READ)
    lat = _get_sds(sd, ["Latitude"]).data
    lon = _get_sds(sd, ["Longitude"]).data
```

DOI: 10.54254/2755-2721/2026.33875

```
    aod = _get_sds(sd, [
        "AOD_550_Dark_Target_Deep_Blue_Combined",
        "AOD_550_Dark_Target_Deep_Blue_Combined_Mean",
    ])
    aod_qa = _get_sds(sd, [
        "AOD_550_Dark_Target_Deep_Blue_Combined_QA_Flag",
        "AOD_550_Dark_Target_Deep_Blue_Combined_QA",
    ])
    aod_val = _apply_scale(aod.data, aod.attrs)
    aod_qa_val = aod_qa.data
    return lat, lon, aod_val, aod_qa_val
def _load_myd06(path: Path) -> Tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray]:
    sd = SD(str(path), SDC.READ)
    cer = _get_sds(sd, ["Cloud_Effective_Radius", "Cloud_Effective_Radius_Liquid"])
    cot = _get_sds(sd, ["Cloud_Optical_Thickness", "Cloud_Optical_Thickness_Liquid"])
    cwp = _get_sds(sd, ["Cloud_Water_Path", "Cloud_Water_Path_Liquid"])
    phase = _get_sds(sd, [
        "Cloud_Phase_Optical_Properties",
        "Cloud_Phase_Infrared",
    ])
    cer_val = _apply_scale(cer.data, cer.attrs)
    cot_val = _apply_scale(cot.data, cot.attrs)
    cwp_val = _apply_scale(cwp.data, cwp.attrs)
    phase_val = phase.data
    return cer_val, cot_val, cwp_val, phase_val
```

DOI: 10.54254/2755-2721/2026.33875

```
def compute_granule_stats(myd04_path: Path, myd06_path: Path, date_str: str) -> Optional[dict]:
    lat, lon, aod, aod_qa = _load_myd04(myd04_path)
    cer, cot, cwp, phase = _load_myd06(myd06_path)
    # Downsample MYD06 (1 km) to MYD04 grid if needed
    if cer.shape != aod.shape:
        target_shape = aod.shape
        cer = _downsample_mean(cer, target_shape)
        cot = _downsample_mean(cot, target_shape)
        cwp = _downsample_mean(cwp, target_shape)
        phase = _downsample_mean(phase.astype(np.float32), target_shape)
    mask = _mask_bbox(lat, lon)
    mask &= np.isfinite(aod)
    mask &= (aod_qa >= AOD_QA_MIN)
    mask &= np.isfinite(cer) & np.isfinite(cot) & np.isfinite(cwp)
    # phase may be float after downsampling; treat values ~1 as liquid
    mask &= (phase >= 0.5) & (phase < 1.5)
    if mask.sum() == 0:
        return None
    return {
        "myd04_path": str(myd04_path),
        "myd06_path": str(myd06_path),
        "date": date_str,
        "month": int(date_str.split("-")[1]),
        "aod_mean": float(np.nanmean(aod[mask])),
        "cer_mean": float(np.nanmean(cer[mask])),
```

DOI: 10.54254/2755-2721/2026.33875

```
        "cot_mean": float(np.nanmean(cot[mask])),
        "cwp_mean": float(np.nanmean(cwp[mask])),
        "n_pixels": int(mask.sum()),
    }
def bin_relationships(df: pd.DataFrame) -> pd.DataFrame:
    df = df.copy()
    df["aod_bin"] = pd.cut(df["aod_mean"], bins=AOD_BINS, right=False)
    grouped = df.groupby("aod_bin").agg(
        n=("aod_mean", "count"),
        aod_mean=("aod_mean", "mean"),
        cer_mean=("cer_mean", "mean"),
        cot_mean=("cot_mean", "mean"),
        cwp_mean=("cwp_mean", "mean"),
    )
    return grouped.reset_index()
def main():
    matched = pd.read_csv(MATCHED_CSV)
    if not PYHDF_OK:
        # Fallback: write out the matched pairs list so the pipeline can proceed
        matched.to_csv("modis_myd04_myd06_2025_matched_fallback.csv", index=False)
        print(
            "pyhdf is not installed. Wrote matched pair list to "
            "modis_myd04_myd06_2025_matched_fallback.csv. "
            "Install pyhdf to compute AOD/CER/COT/CWP stats."
        )
```

DOI: 10.54254/2755-2721/2026.33875

```
    return
rows = []
for _, r in matched.iterrows():
    stats = compute_granule_stats(
        Path(r["myd04_path"]),
        Path(r["myd06_path"]),
        r["date"],
    )
    if stats:
        rows.append(stats)
if not rows:
    raise SystemExit("No valid pixels after QA/phase/bbox filtering.")
out = pd.DataFrame(rows)
out.to_csv(OUT_GRANULE_STATS, index=False)
binned = bin_relationships(out)
binned.to_csv(OUT_BINNED_STATS, index=False)
print(f"Saved granule stats to {OUT_GRANULE_STATS}")
print(f"Saved binned relationships to {OUT_BINNED_STATS}")
if not MPL_OK:
    print("matplotlib is not installed; skipping plots. Install with: pip install matplotlib")
    return
# Plot 1: AOD–CER binned
plt.figure(figsize=(6, 4))
plt.plot(binned["aod_mean"], binned["cer_mean"], marker="o")
plt.xlabel("AOD (binned mean)")
```

DOI: 10.54254/2755-2721/2026.33875

```
plt.ylabel("CER (mean)")
plt.title("AOD–CER (Binned)")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("aod_cer_binned.png", dpi=200)
plt.close()
# Plot 2: AOD–CWP binned
plt.figure(figsize=(6, 4))
plt.plot(binned["aod_mean"], binned["cwp_mean"], marker="o", color="tab:orange")
plt.xlabel("AOD (binned mean)")
plt.ylabel("CWP (mean)")
plt.title("AOD–CWP (Binned)")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("aod_cwp_binned.png", dpi=200)
plt.close()
# Plot 3: Seasonal comparison (winter vs summer)
winter = out[out["month"].isin([12, 1, 2])]
summer = out[out["month"].isin([6, 7, 8])]
winter_b = bin_relationships(winter) if not winter.empty else None
summer_b = bin_relationships(summer) if not summer.empty else None
plt.figure(figsize=(6, 4))
if winter_b is not None:
    plt.plot(winter_b["aod_mean"], winter_b["cer_mean"], marker="o", label="Winter")
if summer_b is not None:
```

DOI: 10.54254/2755-2721/2026.33875

```
    plt.plot(summer_b["aod_mean"], summer_b["cer_mean"], marker="o", label="Summer")
  plt.xlabel("AOD (binned mean)")
  plt.ylabel("CER (mean)")
  plt.title("Seasonal AOD–CER (Winter vs Summer)")
  plt.grid(True, alpha=0.3)
  plt.legend()
  plt.tight_layout()
  plt.savefig("seasonal_aod_cer.png", dpi=200)
  plt.close()
  print("Saved plots: aod_cer_binned.png, aod_cwp_binned.png, seasonal_aod_cer.png")
if __name__ == "__main__":
  main()
```

This pseudocode groups AOD values into bins and computes the mean CER and CWP for each bin. The final result helps in analysing the relationship between AOD and cloud microphysical parameters.