ARC validation 2023

This notebook contains code and example runs to provide ARC estimates of a set of biophsycial variables over time, given a set of Sentinel-2 MSI observations. The approach is generalisable to other sensors, but that is not done here. Details of the approach are given in Feng et al. (2024).

The notebook also provides codes to show how to obtain and use the ground measurements collected as a part of this project. The measurements are available in the file SF_2023_samples_corrected_parsed.geojson. That in turn, is generated from raw data in SF_2023_ground_data.xlsx and SF_2023_ground_measurements_LAI_Cab.geojson by the script parse_2023_coords.py.

In section 1., we set (broad) parameters for the runs and define the driving file names. We also install and test codes.

In section 2., we explore the observational dataset, looking at the target field and surrounding area, visualising this as NDVI and rewal colour (RGB).

In section 3., we run the `arc` code to fit the archetype models to each pixel of the observations over the target field. The direct output of this is a matching of measured and (archtype-)modelled reflectance. Additionally, we have knowledge of the model biophysical parameters. We can derive per-pixel uncertainties for these, but here we look only at variation within the plot.

In section 4, we compare the modelled LAI values with fieldwork-measured values. In section 5, we do the same for Chlorophyll concentration. The datasets for modelled and measured biophysical parameters are stored in data files so we can later analyse all datasets together.

Section 6 provides a brief summary of the experiment.

1. Requirements

First, we need to install required codes and datafiles.

1.1 tags and parameters

first, we define some tags and parameters for this notebook. These will be used to define filenames associated with this run.

```
from src.utils import *
```

```
# filename tag
YEAR = 2023
TAG = f'SF_{YEAR}'

# buffer extent for visiualisation, in degrees
dx,dy = 0.009,0.009

# dates
start_date = f"{YEAR}-08-01"
end_date = f"{YEAR}-12-10"

# Constants
CROP_TYPE = "wheat"

NUM_SAMPLES = 1000000

# buffer control
buffer_n = 2
```

```
# plot layout
nx=5
# data every N days
repeat=5
```

These are ARC parameters.

The parameters are set to be appropriate for the crop under study (wheat). The time taken to process depends heavily on `NUM_SAMPLES` which controls the number of ensemble members used in the evaluation. Typically `100000` would be sufficient to estimate the mean behaviour. Estimating uncertainty typically requires more members. We use `1000000` here as a default to ensure we get a good solution. This will take around 5 minutes to process, so if you are impatient to proceed, reduce the value of `NUM_SAMPLES`.

Below, we suppose the **crop emergence** to be around 1 month after `start_date` and the **growth season length** (period to peak LAI) to be up to 2 months before the end date. These are just approximate vcalues to help- constrain the solution.

```
import datetime
yformat = '%Y-%m-%d'

t0 = int(datetime.datetime.strptime(start_date, yformat).strftime("%j"))
t1 = int(datetime.datetime.strptime(end_date, yformat).strftime("%j"))
year = int(datetime.datetime.strptime(start_date, yformat).strftime("%Y"))
dt = (t1 - t0)

# plot layout
ny=int(dt/repeat/nx)+1

# vary with time period set
#START_OF_SEASON = 240
#GROWTH_SEASON_LENGTH = 45
START_OF_SEASON = t0 + 30
GROWTH_SEASON_LENGTH = t1 - 50 - START_OF_SEASON

print(f'start of data: DOY {t0}, length: {t1-t0} days for {year}')
print(f'start of crop: DOY {START_OF_SEASON}, length: {GROWTH_SEASON_LENGTH} days')

# datetime locations
t0s = datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(t0))
t1s = datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(t1))
t0a = datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(START_OF_SEASON))
t1a = datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(START_OF_SEASON+GROWTH_SEASON_LENGTH))
```

```
start of data: DOY 213, length: 131 days for 2023
start of crop: DOY 243, length: 51 days
```

We require that two files:

```
files/{TAG}_samples_corrected_parsed.geojson
files/{TAG}.geojson
```

exists to drive the notebook.

The first defines sample locations and measurement data. The second defined the extent of the particular field of interest.

```
geojson_path= f'files/{TAG}.geojson'
# field
if not Path(geojson_path).exists():
    print(f"required geojson file {geojson_path} does not exists: please create it and re-run")
    exit()

# samples
```

```
samplefile= f'files/{TAG}_samples_corrected_parsed.geojson'
if not Path(samplefile).exists():
    print(f"required samples file {samplefile} does not exists: \
        please create it and re-run with src/parse_2023_coords.py in files")
    exit()
```

1.2 Install ARC and related code

The ARC code accesses the satellite data and fits the archetype models from Feng et al. (2024).

```
# install required code: In case of problems, check the log files generated
for p in pkg:
    if pkgutil.find_loader('arc') is None:
        !(pip install https://github.com/MarcYin/ARC/archive/refs/heads/main.zip > logs/arc_install.log) \
          &> logs/arc_install_stderr.log
```

1.3 Set Google EarthEngine credentials

Note that the **first** time you run this notebook, you need to run the `ee.Authenticate()` command and log in with google Earth Engine. This will divert you to a google login page. You may have to 'continue' through an unverified page (`Google hasn't verified this app`). When asked `Select what Earth Engine Notebook Client - dr.myname@gmail.com can access`, use `select all`. Then you should eventually arrive at an `Authorization code`, which will request you to `Please copy this code, switch to your application and paste it there:`. So, then copy the code and paste below.

```
# N.B. the first time you run this, it will run the ee.Authenticate() command
# after that, there shouyld be an authfile with -rw-------
authfile=Path('~/.config/earthengine/credentials').expanduser()
if not authfile.exists():
    ee.Authenticate()
```

1.4 Set json files for area required

We need to define two `geojson-format file <https://geojson.org>`__ to descibe the area of interest. The first defines a local set of coordinates (e.g. sample locations) to explore, and should be called `{TAG}.geojson`, where `TAG` defines the particular dataset of interest, e.g. `{TAG}_samples_corrected_parsed.geojson <files/%7BTAG%7D_samples_corrected_parsed.geojson>`__.

We need here to parse the location file `{TAG}_samples_corrected_parsed.geojson <files/%7BTAG%7D_samples_corrected_parsed.geojson>`__ and set a buffer around it (controlled by `dx,dy = 0.009,0.009` below) to extract a wider area for data visualisation. This produces a file `files/wider_area_{TAG}.geojson` with the wider area defined.

```
# parse the geojson file geojson_path
wider_geojson = f"files/wider_area_{TAG}.geojson"
print(f'extending {geojson_path} to {wider_geojson}')

# get extent
coords = []
with open(geojson_path, 'r') as f:
    x,y = np.array([feature['geometry']['coordinates'][:2] \
                    for feature in geojson.load(f)['features']]).T
    xmin,xmax,ymin,ymax = x.min()-dx,x.max()+dx,y.min()-dy,y.max()+dy

coords = np.array([x.ravel(),y.ravel()])
```

```
# form new geojson
s = '{"type":"FeatureCollection","features":[{"type":'+\
    '"Feature","properties":{},"geometry":{"coordinates":'+\
    f'[[[{xmin},{ymax}],[{xmin},{ymin}],'+\
    f'[{xmax},{ymin}],[{xmax},{ymax}],[{xmin},{ymax}]]],'+\
    '"type":"Polygon"}}]}'
print(s,file=Path(wider_geojson).open('w'))
print(f'wider extent [{xmin}:{xmax}] [{ymin}:{ymax}]')
```

```
extending files/SF_2023.geojson to files/wider_area_SF_2023.geojson
wider extent [28.182532:28.20872] [-28.199370000000002:-28.175466]
```

2 Data exploration

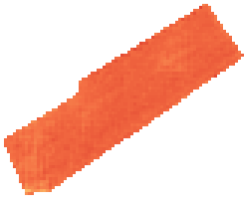
We first obtain and explore the observational dataset.

2.1 local area NDVI

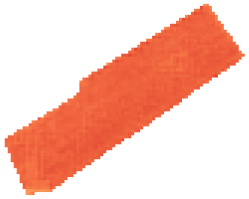
To explore the observational dataset, we can visualise the S2 data as NDVI over space and time in the target field.

```
from arc.s2_data_reader import get_s2_official_data

# plot geojson_path data
S2_data_folder = Path.home() / f"Downloads/{Path(geojson_path).stem}"
S2_data_folder.mkdir(parents=True, exist_ok=True)

s2_refs, s2_uncs, s2_angles, doys, mask, geotransform, crs = \
        get_s2_official_data(start_date, end_date, geojson_path, S2_data_folder=S2_data_folder)

ndvi = (s2_refs[:, 7] - s2_refs[:, 2]) / (s2_refs[:, 7] + s2_refs[:, 2])
fig, axs = plt.subplots(ny, nx , figsize=(ny*3, nx*5))
axs = axs.ravel()
for i in range(nx*ny): axs[i].imshow(ndvi[0]*np.nan)

for i in range(nx*ny):
    try:
        im = axs[i].imshow(ndvi[i], cmap='RdYlGn', vmin=0, vmax=1)
        # plot colorbar
        date = str(datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(doys[i]) - 1)).split()[0]
        axs[i].set_title(f'{date} : doy {doys[i]}')
    except:
        pass
    axs[i].axis('off')

cbar = fig.colorbar(im, ax=axs[nx*ny-1], orientation='horizontal', fraction=0.5)
cbar.set_label('NDVI')
```

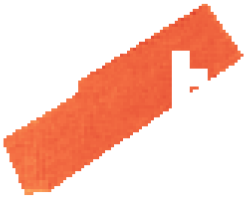
2023-08-04 : doy 216



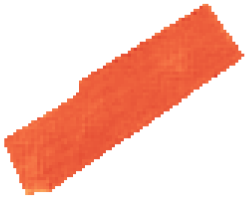
2023-08-14 : doy 226



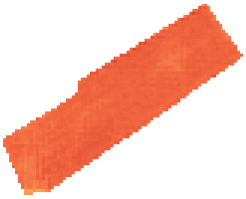
2023-08-19 : doy 231



2023-08-24 : doy 236



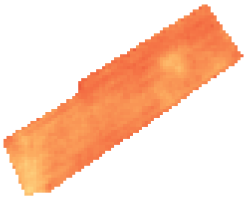
2023-08-29 : doy 241



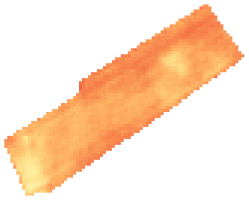
2023-09-03 : doy 246

2023-09-08 : doy 251

2023-09-13 : doy 256

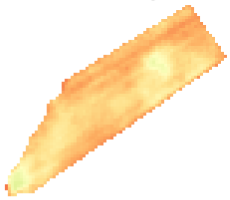


2023-09-18 : doy 261

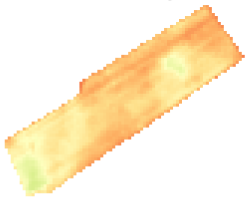


2023-09-23 : doy 266

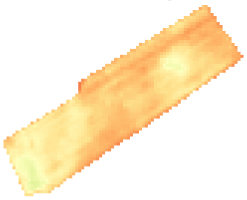
2023-09-28 : doy 271



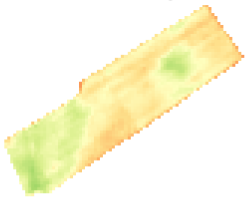
2023-10-03 : doy 276



2023-10-08 : doy 281

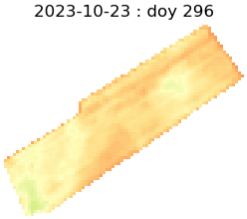


2023-10-13 : doy 286



2023-10-18 : doy 291

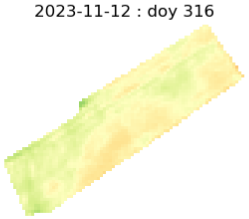
2023-10-23 : doy 296



2023-10-28 : doy 301

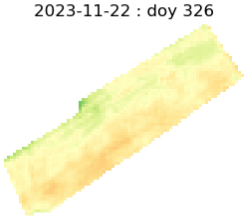
2023-11-02 : doy 306

2023-11-12 : doy 316

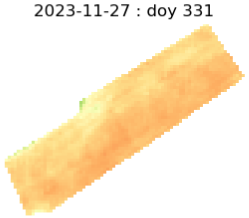


2023-11-17 : doy 321

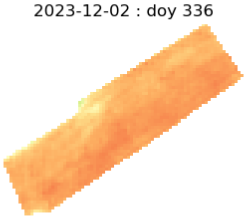
2023-11-22 : doy 326



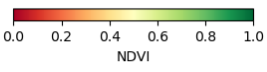
2023-11-27 : doy 331



2023-12-02 : doy 336



2023-12-07 : doy 341



The target field achieves a maximum NDVI of around 0.5, and is much lower in the main, which indicates low LAI. This is in keeping with field observations that the rain-fed field suffered from some water deficiency and only a low LAI was reached.

We see from this that there are 9 or 10 samples that cover the field over the season, with other observation dates masked for cloud and cloud shadow.

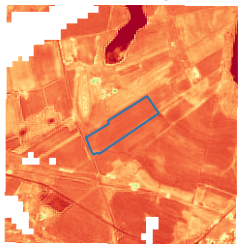
The data are obtained from Google Earth Engine and processed to surface reflectance, using the ARC function `get_s2_official_data()`.

2.2 Wider area NDVI

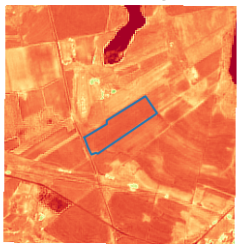
To see this field in context, we can visualise the surrounding area as well:

```
from pyproj import Transformer
# plot wider_geojson data
S2_data_folder = Path.home() / f"Downloads/{Path(wider_geojson).stem}"
S2_data_folder.mkdir(parents=True, exist_ok=True)

s2_refs, s2_uncs, s2_angles, doys, mask, geotransform, crs = \
    get_s2_official_data(start_date, end_date, wider_geojson, S2_data_folder=S2_data_folder)
ndvi = (s2_refs[:, 7] - s2_refs[:, 2]) / (s2_refs[:, 7] + s2_refs[:, 2])
fig, axs = plt.subplots(ny, nx , figsize=(ny*3, nx*5))
axs = axs.ravel()

# transform to local coords
geoms = []
for x,y in coords.T:
    geom = geometry.Point([x,y])
    geoms.append(geom)
#_=geometry.GeometryCollection(geoms)

Big_transformer = Transformer.from_crs("EPSG:4326", crs, always_xy=True)
Big_Tcoords = [i for i in Big_transformer.itransform(coords.T)]

Big_Acoords = np.array([[(i[0] - geotransform[0]) / geotransform[1], (i[1] - geotransform[3]) \
                    / geotransform[5]] for i in Big_Tcoords]).astype(int)

# blank the images
for i in range(nx*ny): axs[i].imshow(ndvi[0]*np.nan)
# data plot
for i in range(nx*ny):
    try:
        im = axs[i].imshow(ndvi[i], cmap='RdYlGn', vmin=0, vmax=1)
        date = str(datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(doys[i]) - 1)).split()[0]
        axs[i].set_title(f'{date} : doy {doys[i]}')
        if i < nx*ny-1:
            # not the last
            axs[i].plot(*Big_Acoords.T)
    except:
        pass
    axs[i].axis('off')

cbar = fig.colorbar(im, ax=axs[nx*ny-1], orientation='horizontal', fraction=0.5)
_=cbar.set_label('NDVI')
```

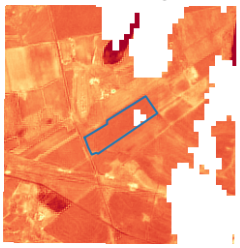
2023-08-04 : doy 216



2023-08-14 : doy 226

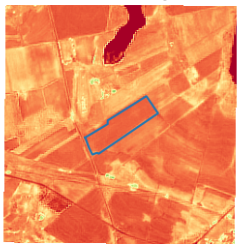


2023-08-19 : doy 231



2023-08-24 : doy 236

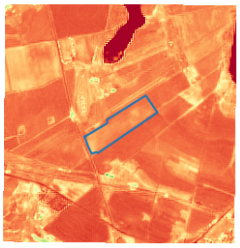
2023-08-29 : doy 241



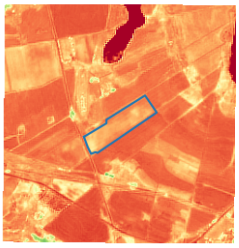
2023-09-03 : doy 246

2023-09-08 : doy 251

2023-09-13 : doy 256

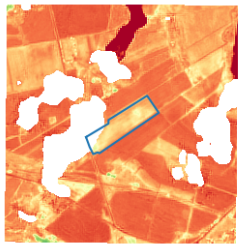


2023-09-18 : doy 261

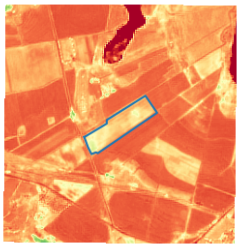


2023-09-23 : doy 266

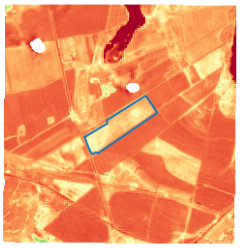
2023-09-28 : doy 271



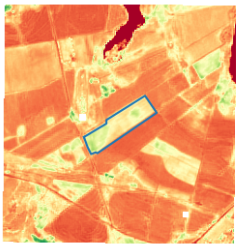
2023-10-03 : doy 276



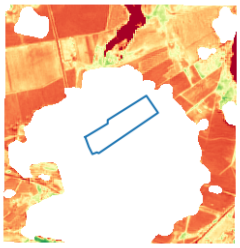
2023-10-08 : doy 281



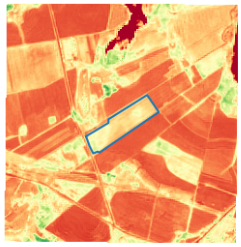
2023-10-13 : doy 286



2023-10-18 : doy 291



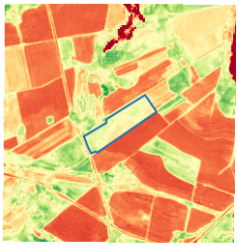
2023-10-23 : doy 296



2023-10-28 : doy 301

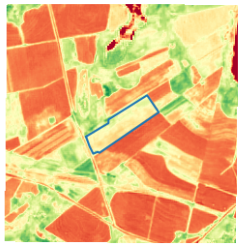
2023-11-02 : doy 306

2023-11-12 : doy 316

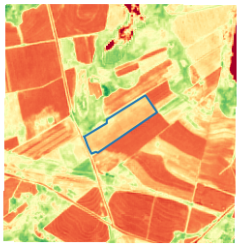


2023-11-17 : doy 321

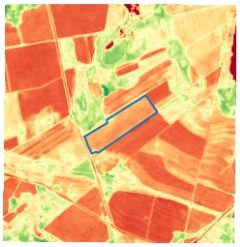
2023-11-22 : doy 326



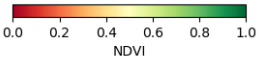
2023-11-27 : doy 331



2023-12-02 : doy 336



2023-12-07 : doy 341



2.2 Wider area RGB

and in True Colour (RGB):

```
# plot wider_geojson data
S2_data_folder = Path.home() / f"Downloads/{Path(wider_geojson).stem}"
S2_data_folder.mkdir(parents=True, exist_ok=True)

s2_refs, s2_uncs, s2_angles, doys, mask, geotransform, crs = \
        get_s2_official_data(start_date, end_date, wider_geojson, S2_data_folder=S2_data_folder)
ndvi = (s2_refs[:, 7] - s2_refs[:, 2]) / (s2_refs[:, 7] + s2_refs[:, 2])
fig, axs = plt.subplots(ny, nx , figsize=(ny*3, nx*5))
axs = axs.ravel()
# blank
for i in range(nx*ny): axs[i].imshow(s2_refs[0, [2, 1, 0], :, :].transpose((1, 2, 0))*np.nan)
# data plot
for i in range(nx*ny):
    try:
        rgb = 4 *s2_refs[i, [2, 1, 0], :, :].transpose((1, 2, 0))
        rgb[rgb>1]=1;
        im = axs[i].imshow(rgb)
        date = str(datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(doys[i]) - 1)).split()[0]
        axs[i].set_title(f'{date} : doy {doys[i]}')
        if i < nx*ny-1:
            # not the last
            axs[i].plot(*Big_Acoords.T)
    except:
        pass
    axs[i].axis('off')
```

2023-08-04 : doy 216

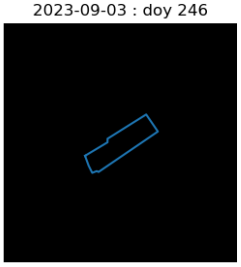
2023-08-14 : doy 226

2023-08-19 : doy 231

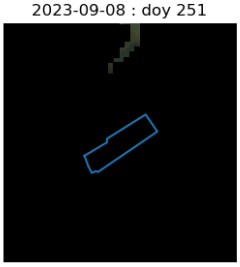
2023-08-24 : doy 236

2023-08-29 : doy 241

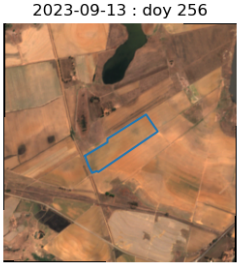
2023-09-03 : doy 246



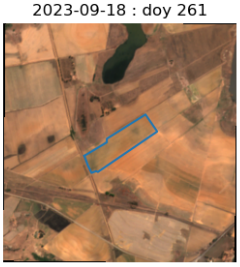
2023-09-08 : doy 251



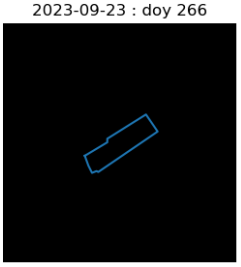
2023-09-13 : doy 256



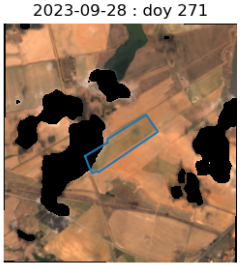
2023-09-18 : doy 261



2023-09-23 : doy 266

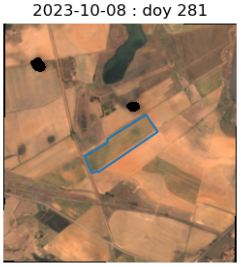


2023-09-28 : doy 271



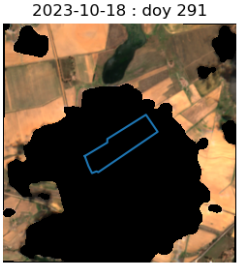
2023-10-03 : doy 276

2023-10-08 : doy 281



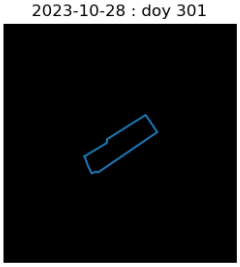
2023-10-13 : doy 286

2023-10-18 : doy 291

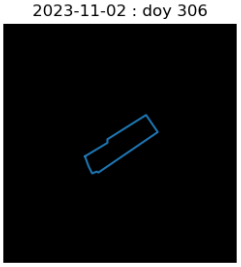


2023-10-23 : doy 296

2023-10-28 : doy 301

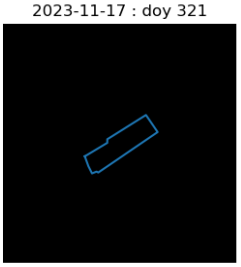


2023-11-02 : doy 306



2023-11-12 : doy 316

2023-11-17 : doy 321

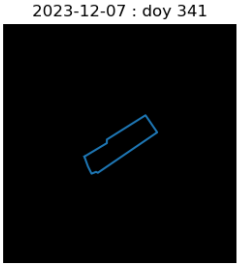


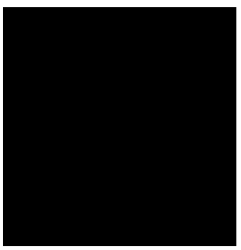
2023-11-22 : doy 326

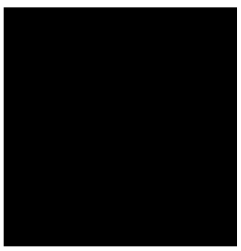
2023-11-27 : doy 331

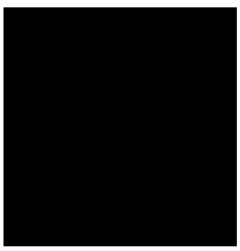
2023-12-02 : doy 336

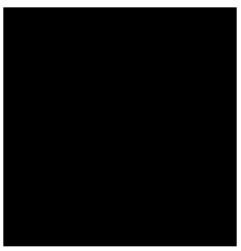
2023-12-07 : doy 341

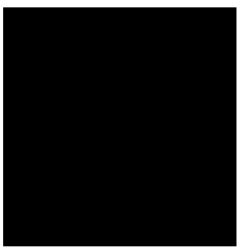


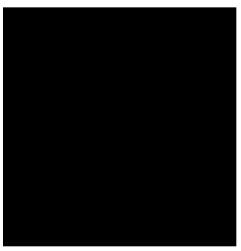








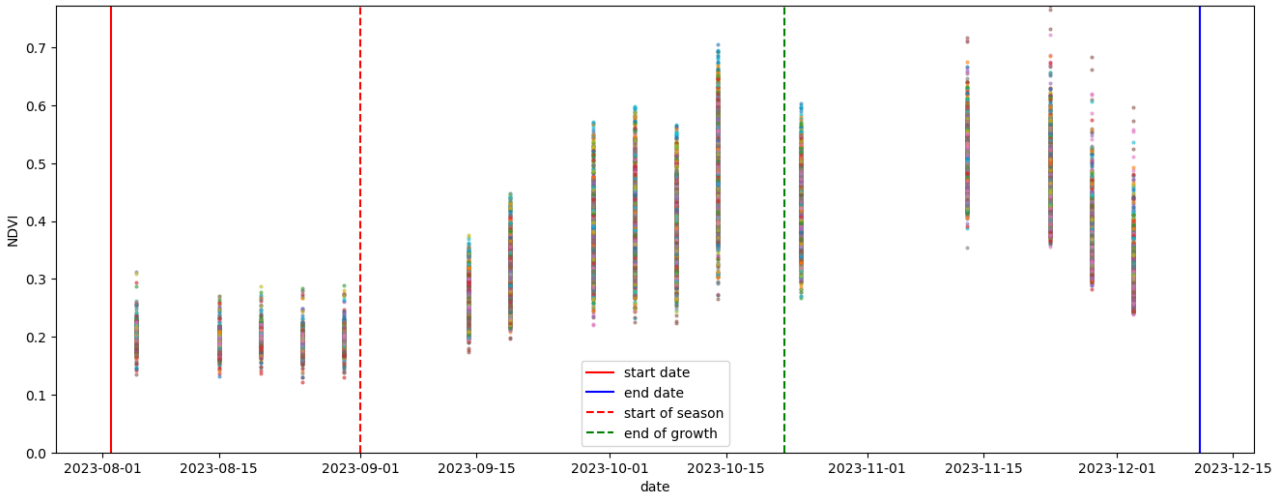




We can also visualise the dataset as RBG images. AGain, we can see that the target field is mostly quite brown, and certainly hads lower LAI than the field just to the North of the target.

If we visualise the NDVI for all field samples over time, we see a gradual increase in LAI (thence LAI)

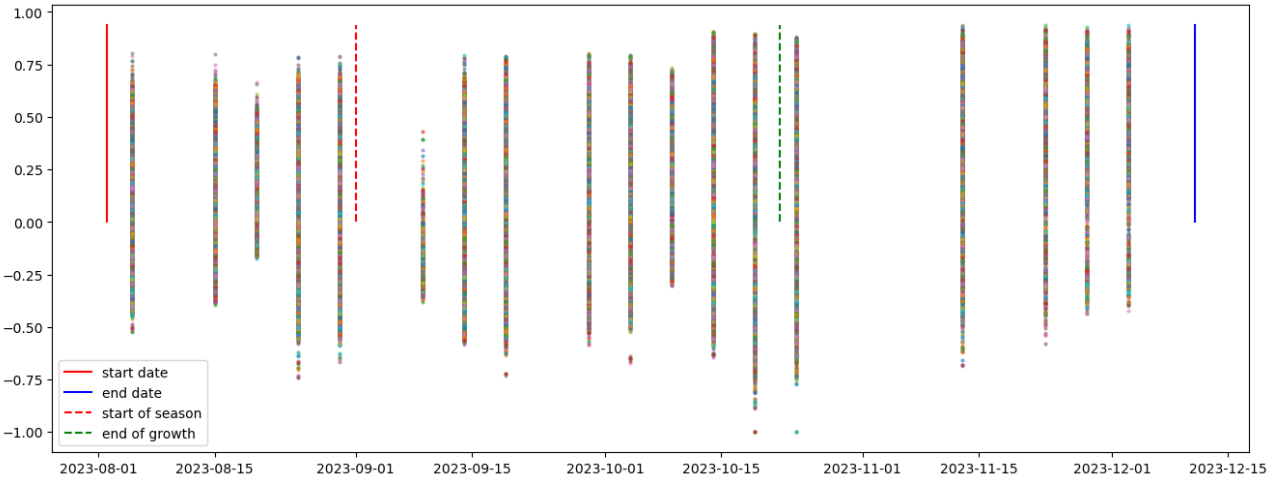
```
# plot geojson_path data
S2_data_folder = Path.home() / f"Downloads/{Path(geojson_path).stem}"
S2_data_folder.mkdir(parents=True, exist_ok=True)

s2_refs, s2_uncs, s2_angles, doys, mask, geotransform, crs = \
        get_s2_official_data(start_date, end_date, geojson_path, S2_data_folder=S2_data_folder)
ndvi = (s2_refs[:, 7] - s2_refs[:, 2]) / (s2_refs[:, 7] + s2_refs[:, 2])
ndvi_max = np.nanmax(ndvi)

transformer = Transformer.from_crs("EPSG:4326", crs, always_xy=True)

# plot all samples over time
plt.figure(figsize=(16, 6))
plt.plot([datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(doy)) \
              for doy in doys], ndvi.reshape(len(doys), -1)[:, ::], 'o', markersize=2, alpha=0.5)
plt.ylim(0,ndvi_max)
plt.ylabel('NDVI')
plt.xlabel('date')

plt.plot([t0s,t0s],[0,ndvi_max],'r',label='start date')
plt.plot([t1s,t1s],[0,ndvi_max],'b',label='end date')

plt.plot([t0a,t0a],[0,ndvi_max],'r--',label='start of season')
plt.plot([t1a,t1a],[0,ndvi_max],'g--',label='end of growth')
_=plt.legend()
```



The wider area is a more complex mixture:

```
# plot wider_geojson data
S2_data_folder = Path.home() / f"Downloads/{Path(wider_geojson).stem}"
S2_data_folder.mkdir(parents=True, exist_ok=True)


s2_refs, s2_uncs, s2_angles, doys, mask, geotransform, crs \
        = get_s2_official_data(start_date, end_date, wider_geojson, S2_data_folder=S2_data_folder)
ndvi = (s2_refs[:, 7] - s2_refs[:, 2]) / (s2_refs[:, 7] + s2_refs[:, 2])
ndvi_max = np.nanmax(ndvi)

plt.figure(figsize=(16, 6))
_ = plt.plot([datetime.datetime(year, 1, 1) + datetime.timedelta(days=int(doy)) \
              for doy in doys], ndvi.reshape(len(doys), -1)[:, ::], 'o', markersize=2, alpha=0.5)
```

```
plt.plot([t0s,t0s],[0,ndvi_max],'r',label='start date')
plt.plot([t1s,t1s],[0,ndvi_max],'b',label='end date')

plt.plot([t0a,t0a],[0,ndvi_max],'r--',label='start of season')
plt.plot([t1a,t1a],[0,ndvi_max],'g--',label='end of growth')
_=plt.legend()
```



3 Running ARC

ARC parameters

We now proceed to run the ARC code toi estimate the archeytpe model parameters from the S2 observations, for each pixel, fitting to the time series of S2 reflectances. The method is described in Feng et al. (2024).

The user can expore various parameters here in the notebook:

```
START_OF_SEASON = 243
CROP_TYPE = "wheat"
NUM_SAMPLES = 1000000
GROWTH_SEASON_LENGTH = 45
start_date = "2023-08-01"
end_date = "2023-12-10"
```

We have previously defined all of these above but you could modify those again here for convenience.

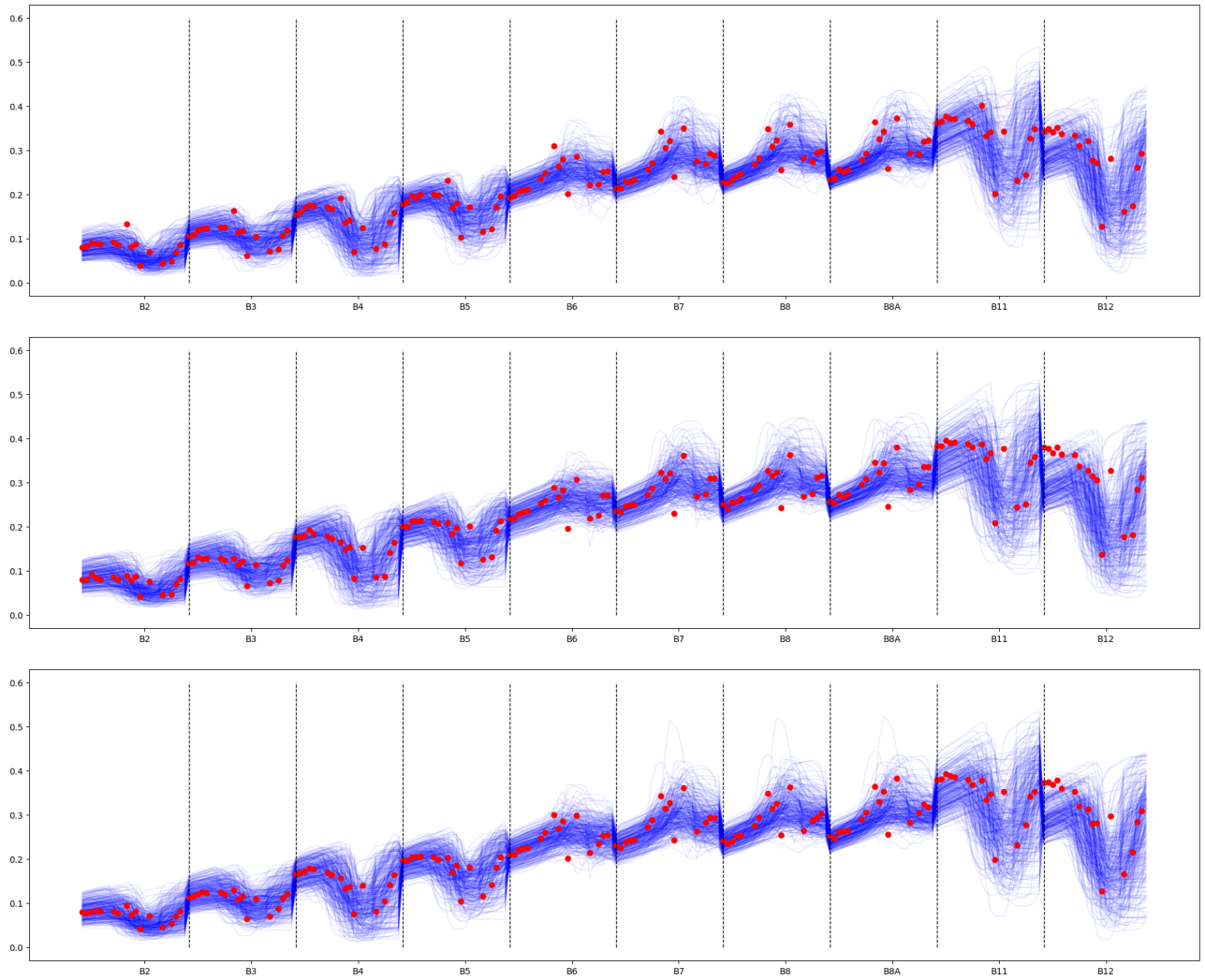
We process the local area, defined in `geojson_path`:

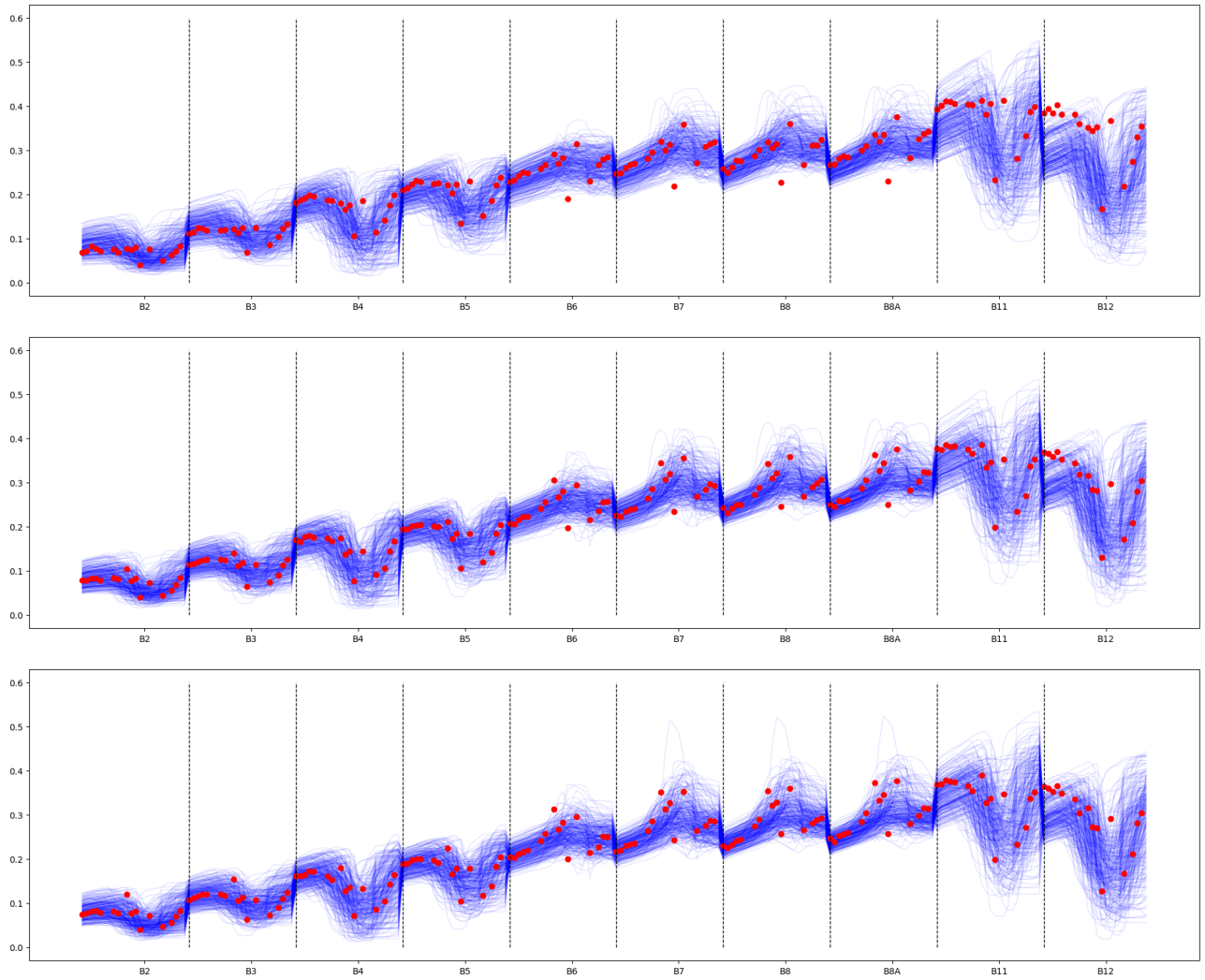
```
import arc

# Constants: set at top
# dates
#start_date = "2023-08-01"
#end_date = "2023-12-10"
print('START_OF_SEASON',START_OF_SEASON)
print('CROP_TYPE',CROP_TYPE)
print('NUM_SAMPLES',NUM_SAMPLES)
print('GROWTH_SEASON_LENGTH',GROWTH_SEASON_LENGTH)
print('start_date',start_date)
```

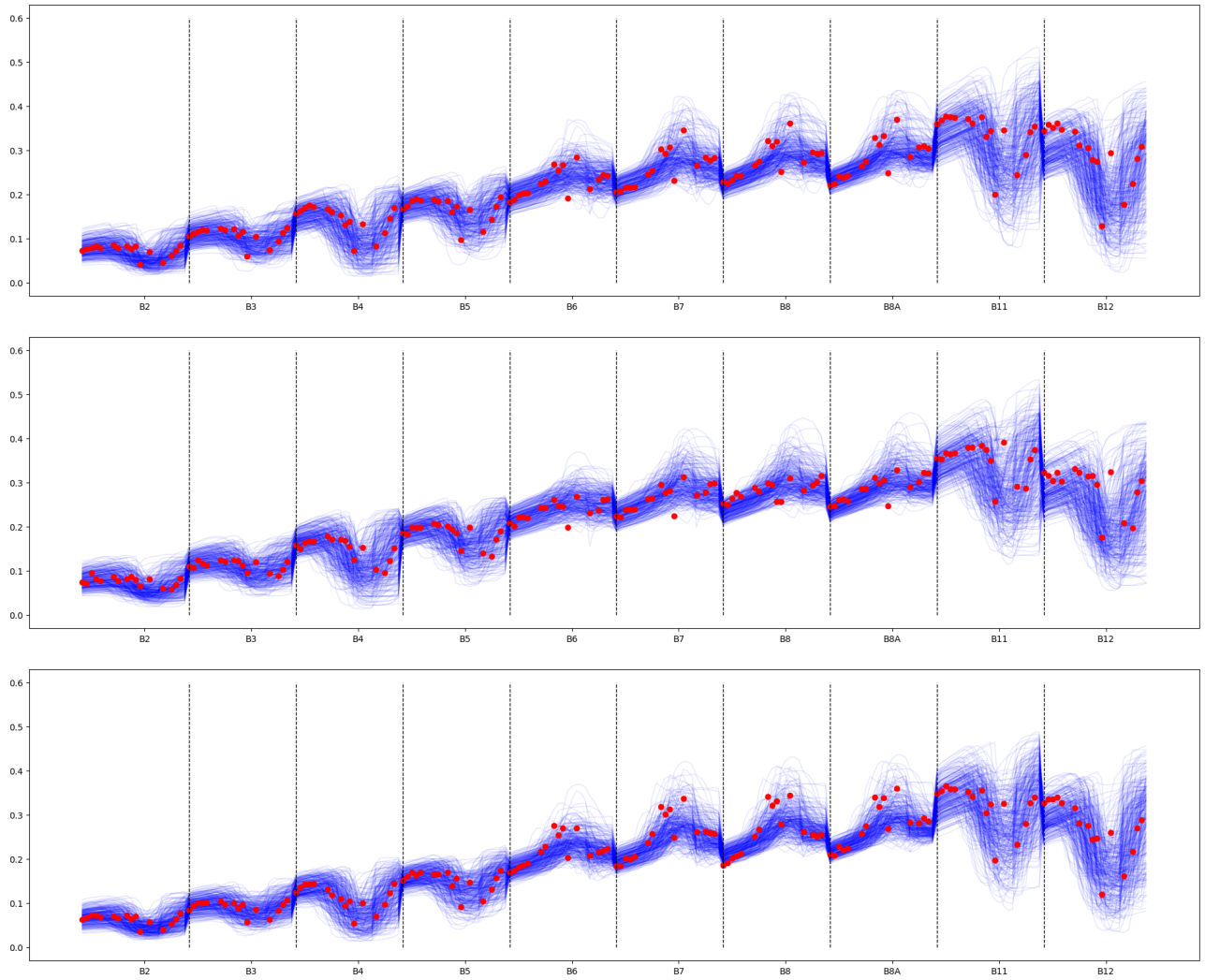
```python
print('end_date',end_date)

# Alternative Constants
#CROP_TYPE = "maize"
#NUM_SAMPLES = 2000000


def main(force=False):
    """Main function to execute the Arc field processing and plotting"""

    S2_data_folder = Path.home() / f"Downloads/{Path(geojson_path).stem}"
    S2_data_folder.mkdir(parents=True, exist_ok=True)
    ofile = f"{S2_data_folder}/{Path(geojson_path).stem}.npz"

    if not force:
        # load file if exists, unless set force
        # NB no plotting if in here and changing parameters
        # has no impact
        if Path(ofile).exists():
            with np.load(ofile) as x:
                return x['dat'],x['post_bio_tensor'],x['post_bio_unc_tensor'],\
                       x['mask'],x['doys']

    scale_data, post_bio_tensor, post_bio_unc_tensor, mask, doys = arc.arc_field(
        start_date,
        end_date,
        geojson_path,
        START_OF_SEASON,
        CROP_TYPE,
        ofile,
        NUM_SAMPLES,
        GROWTH_SEASON_LENGTH,
        str(S2_data_folder),
        plot=True,
    )
    return scale_data, post_bio_tensor, post_bio_unc_tensor, mask, doys

if __name__ == "__main__":
    scale_data, post_bio_tensor, post_bio_unc_tensor, mask, doys = main(force=True)
```

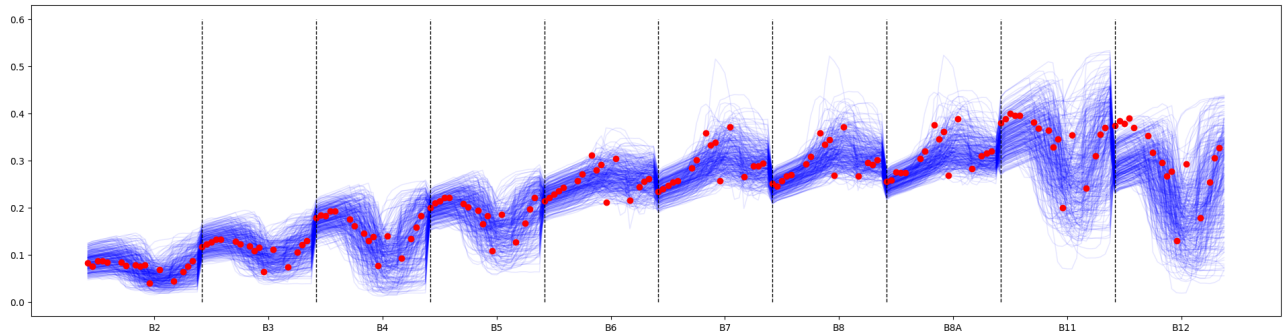
```
START_OF_SEASON 243
CROP_TYPE wheat
NUM_SAMPLES 1000000
GROWTH_SEASON_LENGTH 51
start_date 2023-08-01
end_date 2023-12-10
```

```
Predicting S2 reflectance: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████| 300/300 [01:26<00:00,  3.47slice/s]
```


0.6
0.5
0.4
0.3
0.2
0.1
0.0
B2
B3
B4
B5
B6
B7
B8
B8A
B11
B12


0.6
0.5
0.4
0.3
0.2
0.1
0.0
B2
B3
B4
B5
B6
B7
B8
B8A
B11
B12


0.6
0.5
0.4
0.3
0.2
0.1
0.0
B2
B3
B4
B5
B6
B7
B8
B8A
B11
B12



The ARC function we use here `arc.arc_field()` calculates the solution, and if `plot=True` will also illustrate the optimal set of ensemble reflectance members (blue lines) and observations (red dots). You should check that that ensemble set covers the observations. That means that there is a good match between the ensemble modelled reflectance and the measurements, so we should ber able to robustly estimate the canopy biophysical parameters.

The reflectance plots are laid out to show reflectance per S2 waveband over time for a number of pixel samples. You should be able to see whether or not the ensembles cover the reflectance at each waveband. If not, then consider changing the parameters, e.g. the season start or end or the number of ensemble samnples.

There is also a plot of sample LAI trajectories over time. We observe that maximum LAI seems to be reached around DOY 300 (October 27th). We can check if this is a sensible interpretation by looking at the exploratory data above.

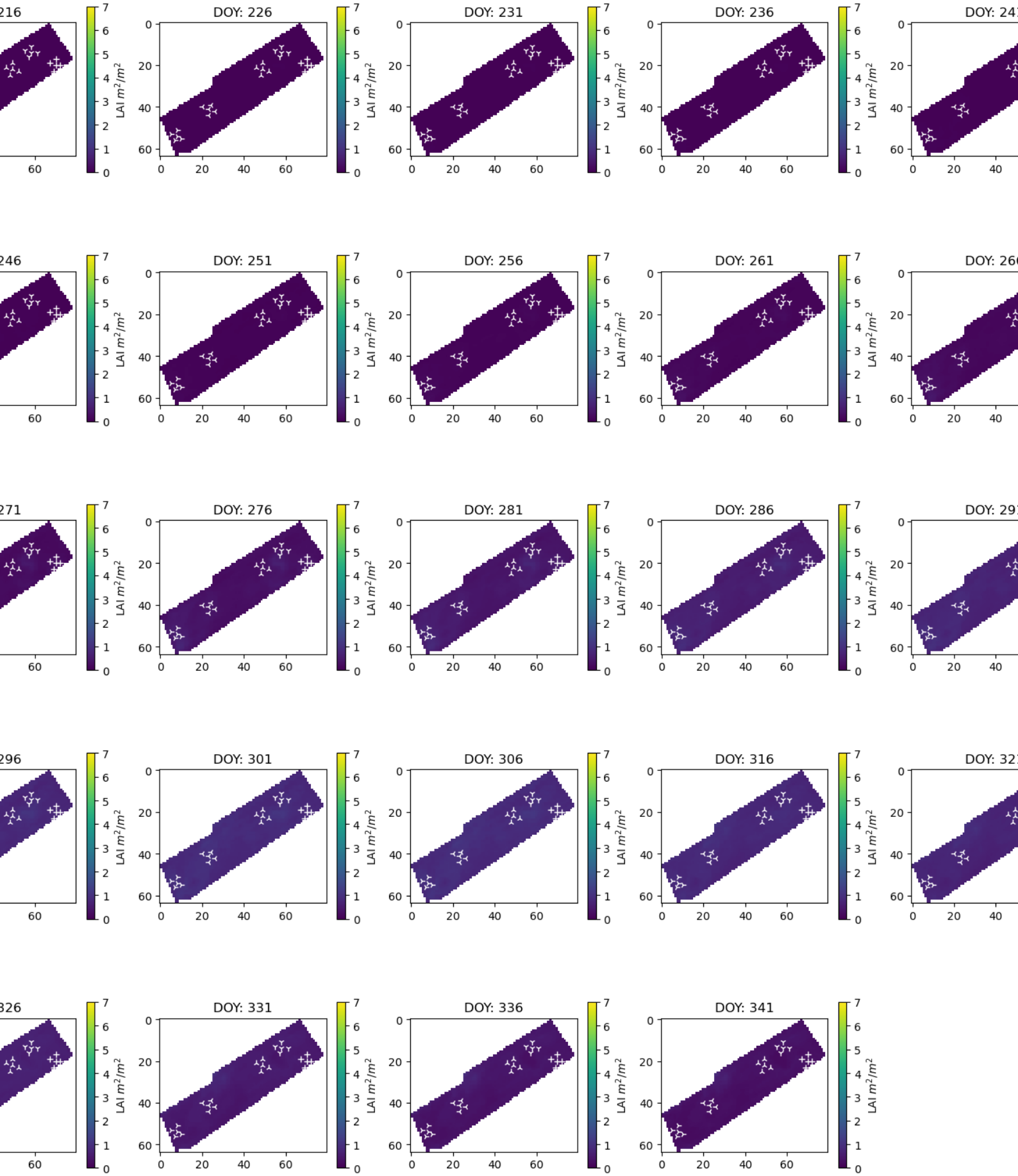
An image of the spatial LAI interpretation is also shown. Again, we can see the maximum LAI occurring around DOY 300. We notice quite a large variation in apparent LAI over the field, with the south-western portion of the field having maximum LAI of greater that 2, butr other parts of the field barely reaching an LAI of 1 or 1.5 perhaps.

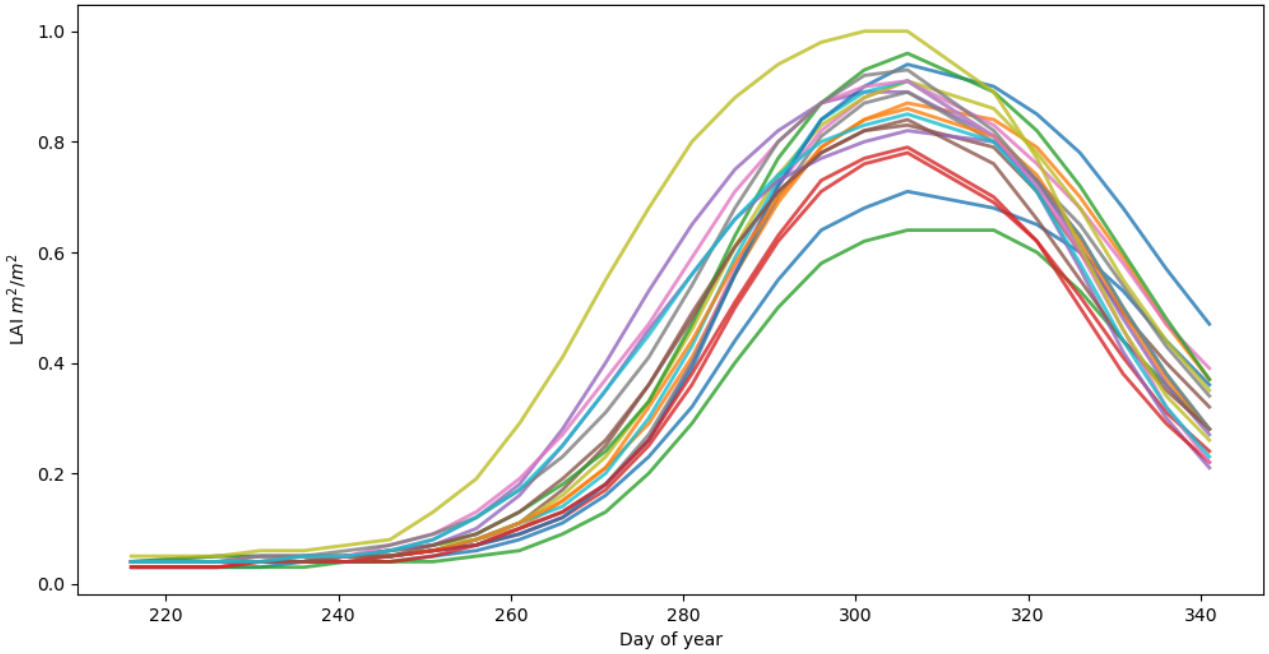
```
import geojson

# reconcile coordinate systems:
# get local image coordinates from sample geometry

features = geojson.load(open(samplefile, 'r'))['features']
geoms = []
coords = []
for feature in features:

    coord = feature['geometry']['coordinates']
    coords.append(coord[:2])

    geom = geometry.Point(coord)
    geoms.append(geom)
_=geometry.GeometryCollection(geoms)

#arc_dir = os.path.dirname(os.path.realpath(arc.__file__))
S2_data_folder = Path.home() / f"Downloads/{Path(geojson_path).stem}"
S2_data_folder.mkdir(parents=True, exist_ok=True)
output_file = f"{S2_data_folder}/{Path(geojson_path).stem}.npz"
f = np.load(output_file)

geotransform = tuple(f.f.geotransform.tolist())
crs = str(f.f.crs)

transformer = Transformer.from_crs("EPSG:4326", crs, always_xy=True)
Tcoords = [i for i in transformer.itransform(coords)]

Acoords = np.array([[(i[0] - geotransform[0]) / geotransform[1], (i[1] - geotransform[3]) \
                     / geotransform[5]] for i in Tcoords]).astype(int)
```

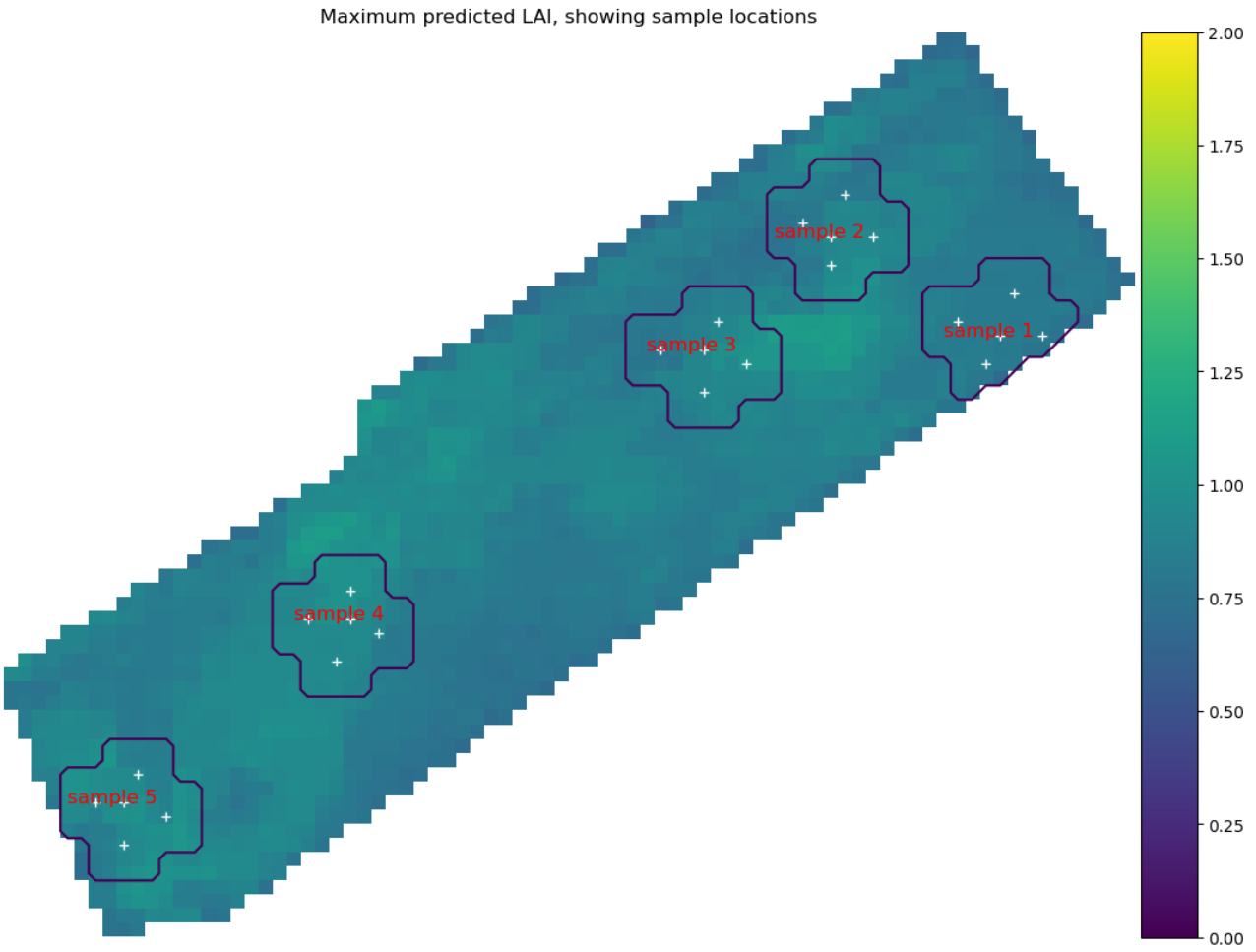
4 Leaf Area Index results

We now run some codes to explore the results for LAI. We extract these from the `samplefile`.

```
name = 'lai'

plot_maps(doys, post_bio_tensor, mask,name,Acoords=Acoords)
plot_over_time(doys, post_bio_tensor,name)
plot_area(post_bio_tensor,name,mask,samplefile=samplefile,\
          buffer_n=buffer_n,transformer=transformer,geotransform=geotransform)
```


216
DOY: 226
DOY: 231
DOY: 236
DOY: 241
246
DOY: 251
DOY: 256
DOY: 261
DOY: 266
271
DOY: 276
DOY: 281
DOY: 286
DOY: 291
296
DOY: 301
DOY: 306
DOY: 316
DOY: 321
326
DOY: 331
DOY: 336
DOY: 341
LAI m^2/m^2

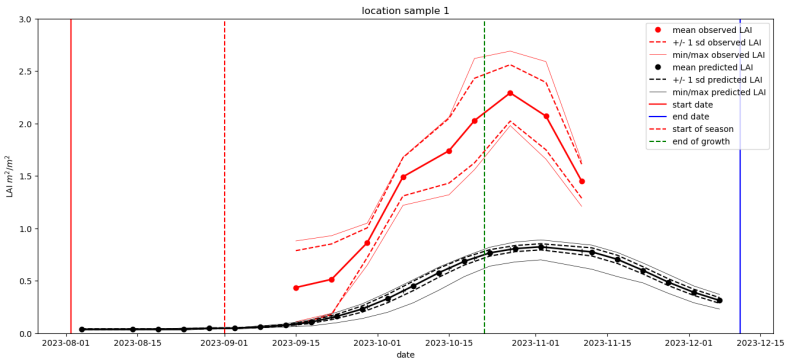

1.0
0.8
0.6
0.4
0.2
0.0
LAI m^2/m^2
220
240
260
280
300
320
340
Day of year

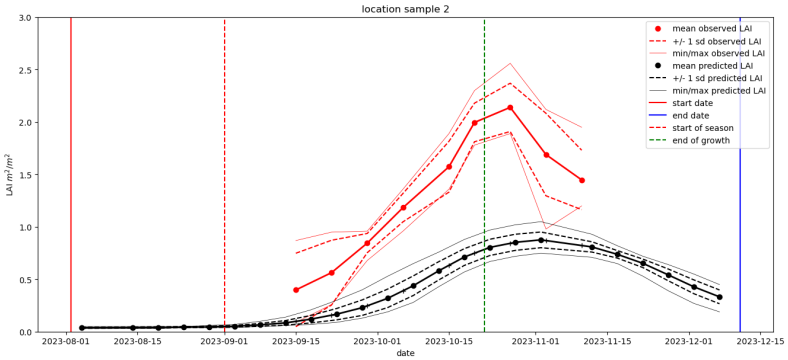


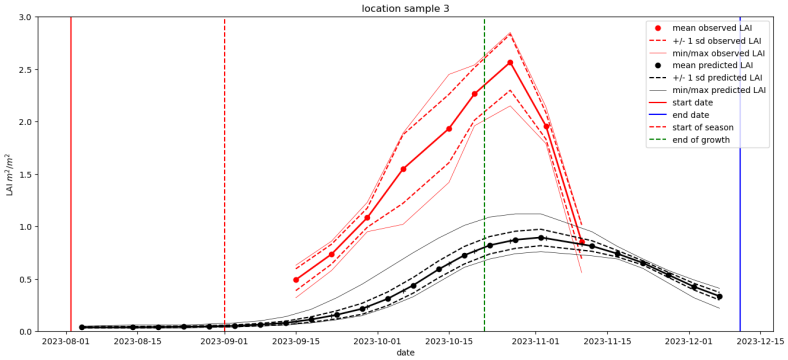
We can visualise the 5 sample locations, with 5 white crosses in the image below indicating the sub-sampling. For each of the 5 sample plots, we apply a buffer of e.g. `buffer_n = 2` i.e. a +/- 2 10m pixels around each sample location, so the samples cover around 50 m x 50 m.

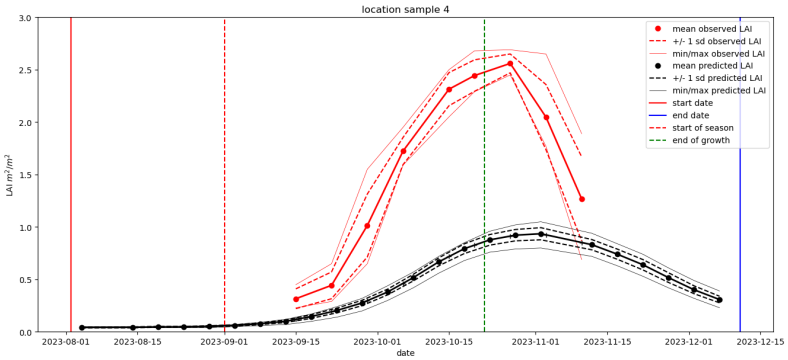
The base image shows the maximum LAI per pixel, confirming the observation above from NDVI that the LAI is higher in the lower left iof the field.

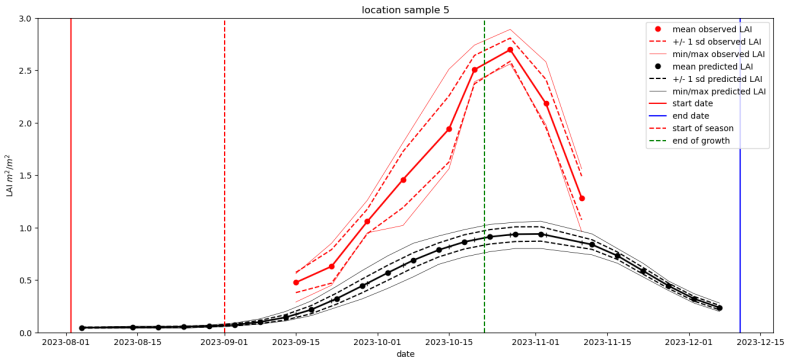
```
maxval, all_target_s2_pred, all_data = plot_biophys(post_bio_tensor,name,mask,\
           doys=doys,samplefile=samplefile,year=year,\
           t0s=t0s,t1s=t1s,t0a=t0a,t1a=t1a,\
           buffer_n=buffer_n,transformer=transformer,geotransform=geotransform)
```


location sample 1
mean observed LAI
+/- 1 sd observed LAI
min/max observed LAI
mean predicted LAI
+/- 1 sd predicted LAI
min/max predicted LAI
start date
end date
start of season
end of growth
LAI m^2/m^2
date
2023-08-01
2023-08-15
2023-09-01
2023-09-15
2023-10-01
2023-10-15
2023-11-01
2023-11-15
2023-12-01
2023-12-15
0.0
0.5
1.0
1.5
2.0
2.5
3.0


location sample 2
mean observed LAI
+/- 1 sd observed LAI
min/max observed LAI
mean predicted LAI
+/- 1 sd predicted LAI
min/max predicted LAI
start date
end date
start of season
end of growth
LAI m^2/m^2
date
2023-08-01
2023-08-15
2023-09-01
2023-09-15
2023-10-01
2023-10-15
2023-11-01
2023-11-15
2023-12-01
2023-12-15
0.0
0.5
1.0
1.5
2.0
2.5
3.0


location sample 3
mean observed LAI
+/- 1 sd observed LAI
min/max observed LAI
mean predicted LAI
+/- 1 sd predicted LAI
min/max predicted LAI
start date
end date
start of season
end of growth
LAI m^2/m^2
date
2023-08-01
2023-08-15
2023-09-01
2023-09-15
2023-10-01
2023-10-15
2023-11-01
2023-11-15
2023-12-01
2023-12-15
0.0
0.5
1.0
1.5
2.0
2.5
3.0


location sample 4
mean observed LAI
+/- 1 sd observed LAI
min/max observed LAI
mean predicted LAI
+/- 1 sd predicted LAI
min/max predicted LAI
start date
end date
start of season
end of growth
LAI m^2/m^2
date
2023-08-01
2023-08-15
2023-09-01
2023-09-15
2023-10-01
2023-10-15
2023-11-01
2023-11-15
2023-12-01
2023-12-15
0.0
0.5
1.0
1.5
2.0
2.5
3.0

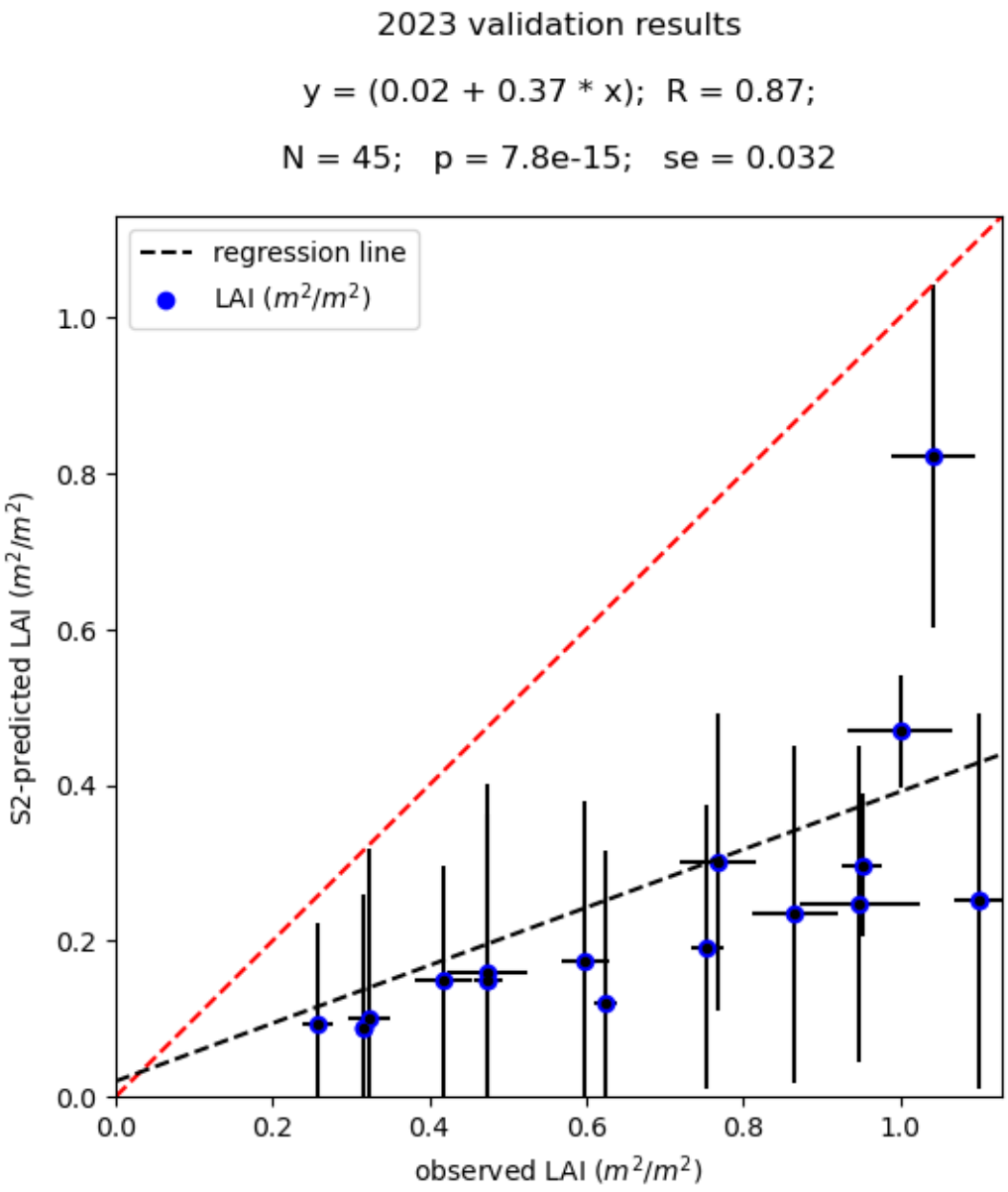

location sample 5
mean observed LAI
+/- 1 sd observed LAI
min/max observed LAI
mean predicted LAI
+/- 1 sd predicted LAI
min/max predicted LAI
start date
end date
start of season
end of growth
LAI m^2/m^2
date
2023-08-01
2023-08-15
2023-09-01
2023-09-15
2023-10-01
2023-10-15
2023-11-01
2023-11-15
2023-12-01
2023-12-15
0.0
0.5
1.0
1.5
2.0
2.5
3.0

The plots above visualise the observed (red) and predicted (black) patterns of LAI over time. The 9 field observations are shown as red dots. The LAI predicted from S2 using ARC is calculated at regular intervals (black dots).

The results show a clear under-estimation of LAI in the S2 ARC results that is quite consistent over time. Sample 5 seems generally clioser to the observations than the other samples.

If we plot the measured and S2-ARC-modelled data together for each of the 5 sample locations and 9 time periods, we can see how well the model predicts the (independent) LAI observations.

```
scatter_save(post_bio_tensor,name,mask,all_target_s2_pred, all_data,TAG,year=year)
```



The plot shows S2-ARC-predicted LAI (y-axis) against observed LAI (x-axis) for the 5 x 9 (N=45) samples. The LAI mean samples over the 5 sub-locations for observations and over the buffer areas shown above are used to calculate mean and standard deviation. The mean is shown as the blue dot, with +/- 1 standard deviation shown as the error bars.

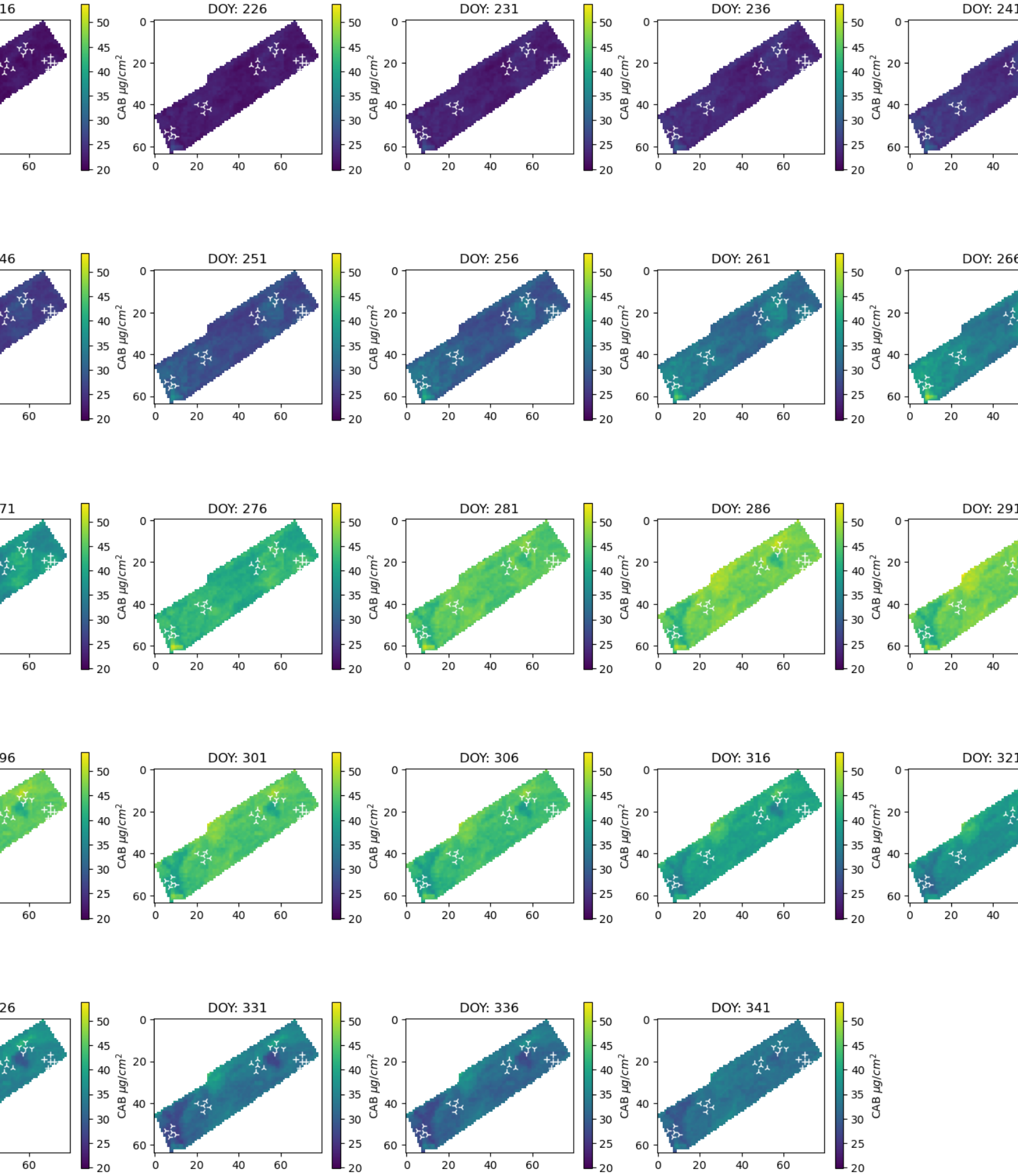
The regression shows a strong linear relationship (R=0.88) between LAI predicted from S2 compared to ground observations for 2023. But there is an apparent bias of -0.24 and a slope of 0.7, i.e. the S2-predictions under-predict the actual LAI. Most of the piints lie around the regression line, but there seems

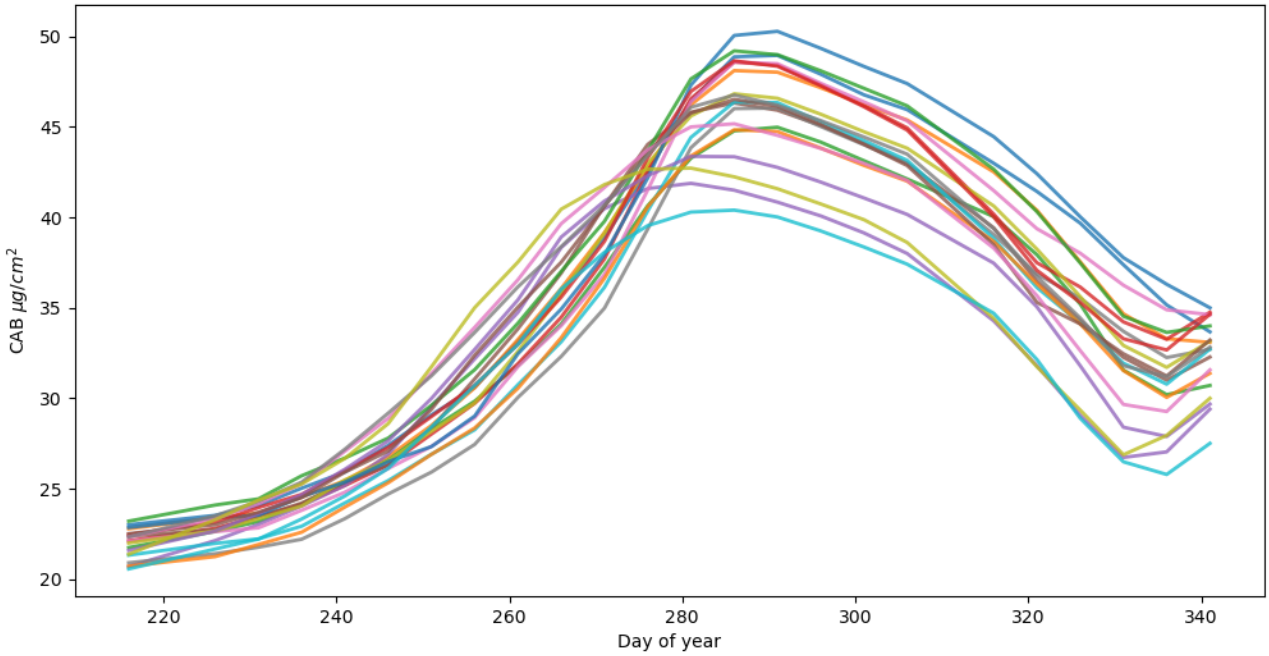
to be a cluster of outliers that are closer to the 1:1 line. Those points are mainly from the last time sample taken, which can be confirmed looking at the time plots above.

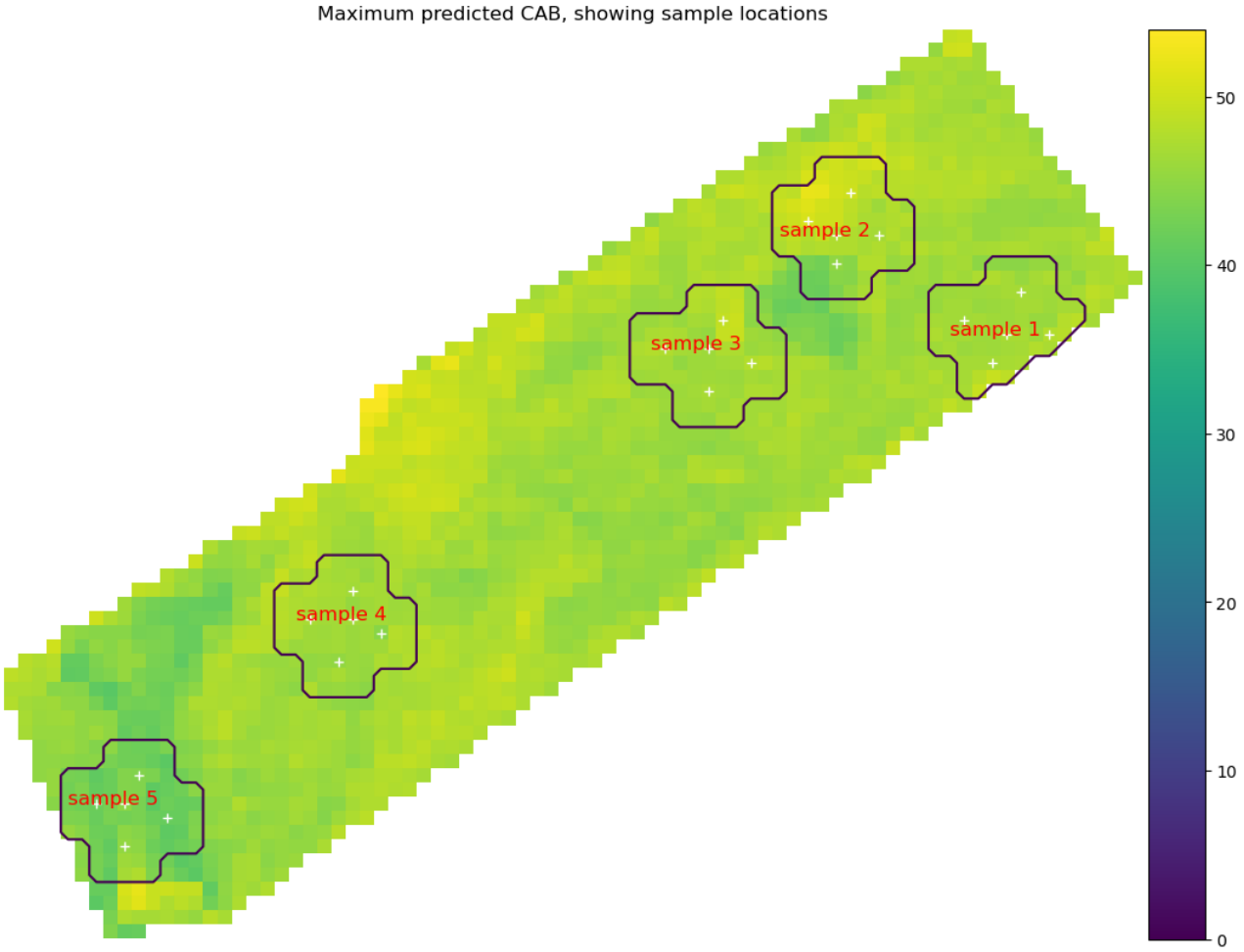
5 Chlorophyll results

We now examine the results for Chlorophyll (Cab), following the same approach as above:

```
name = 'cab'

plot_maps(doys, post_bio_tensor, mask,name,Acoords=Acoords)
plot_over_time(doys, post_bio_tensor,name)
plot_area(post_bio_tensor,name,mask,samplefile=samplefile,\
          buffer_n=buffer_n,transformer=transformer,geotransform=geotransform)
```

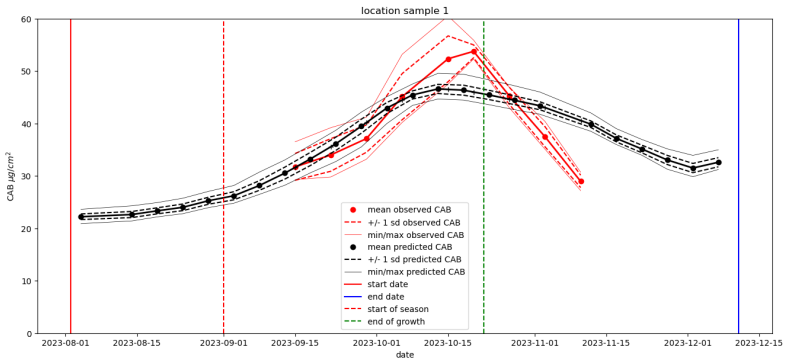

DOY: 226
DOY: 231
DOY: 236
DOY: 241
DOY: 251
DOY: 256
DOY: 261
DOY: 266
DOY: 276
DOY: 281
DOY: 286
DOY: 291
DOY: 301
DOY: 306
DOY: 316
DOY: 321
DOY: 331
DOY: 336
DOY: 341
CAB µg/cm²

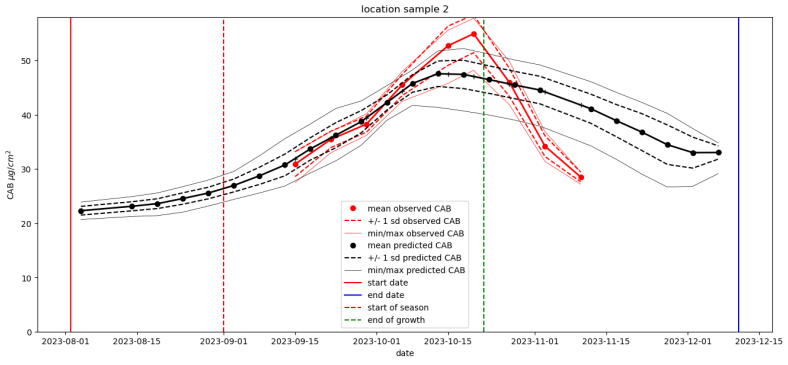

50
45
40
35
30
25
20
CAB $\mu g/cm^2$
220
240
260
280
300
320
340
Day of year

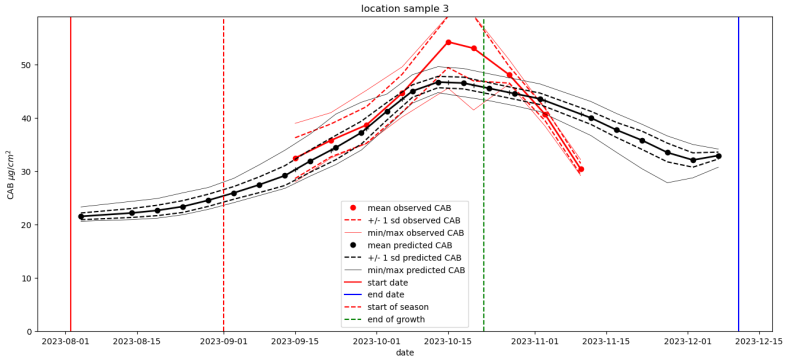


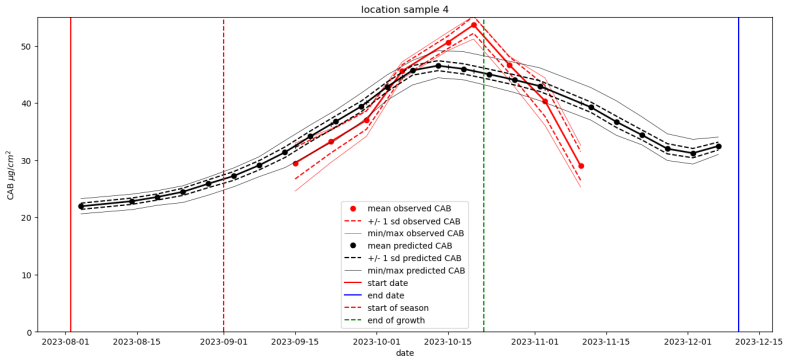
The maximum Chlorophyll concentration (Cab) predicted by ARC with S2 data is quite consistent over the field, with values around 70 $\mu g/cm^2$.

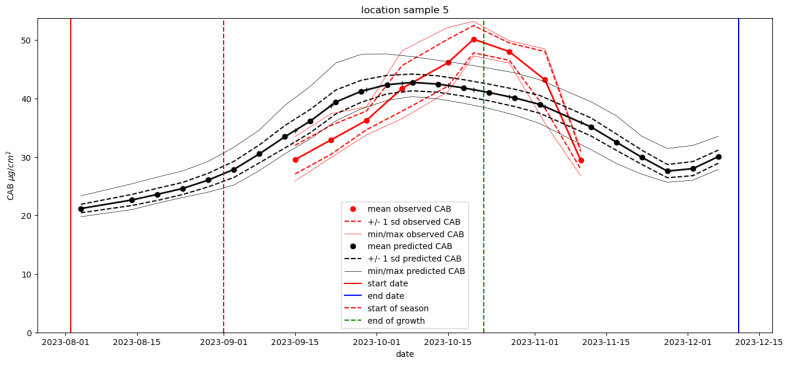
```
maxval, all_target_s2_pred, all_data = plot_biophys(post_bio_tensor,name,mask,\
            doys=doys,samplefile=samplefile,year=year,\
            t0s=t0s,t1s=t1s,t0a=t0a,t1a=t1a,\
            buffer_n=buffer_n,transformer=transformer,geotransform=geotransform)
```


location sample 1
CAB µg/cm²
date
mean observed CAB
+/- 1 sd observed CAB
min/max observed CAB
mean predicted CAB
+/- 1 sd predicted CAB
min/max predicted CAB
start date
end date
start of season
end of growth


location sample 2
CAB µg/cm²
date
mean observed CAB
+/- 1 sd observed CAB
min/max observed CAB
mean predicted CAB
+/- 1 sd predicted CAB
min/max predicted CAB
start date
end date
start of season
end of growth

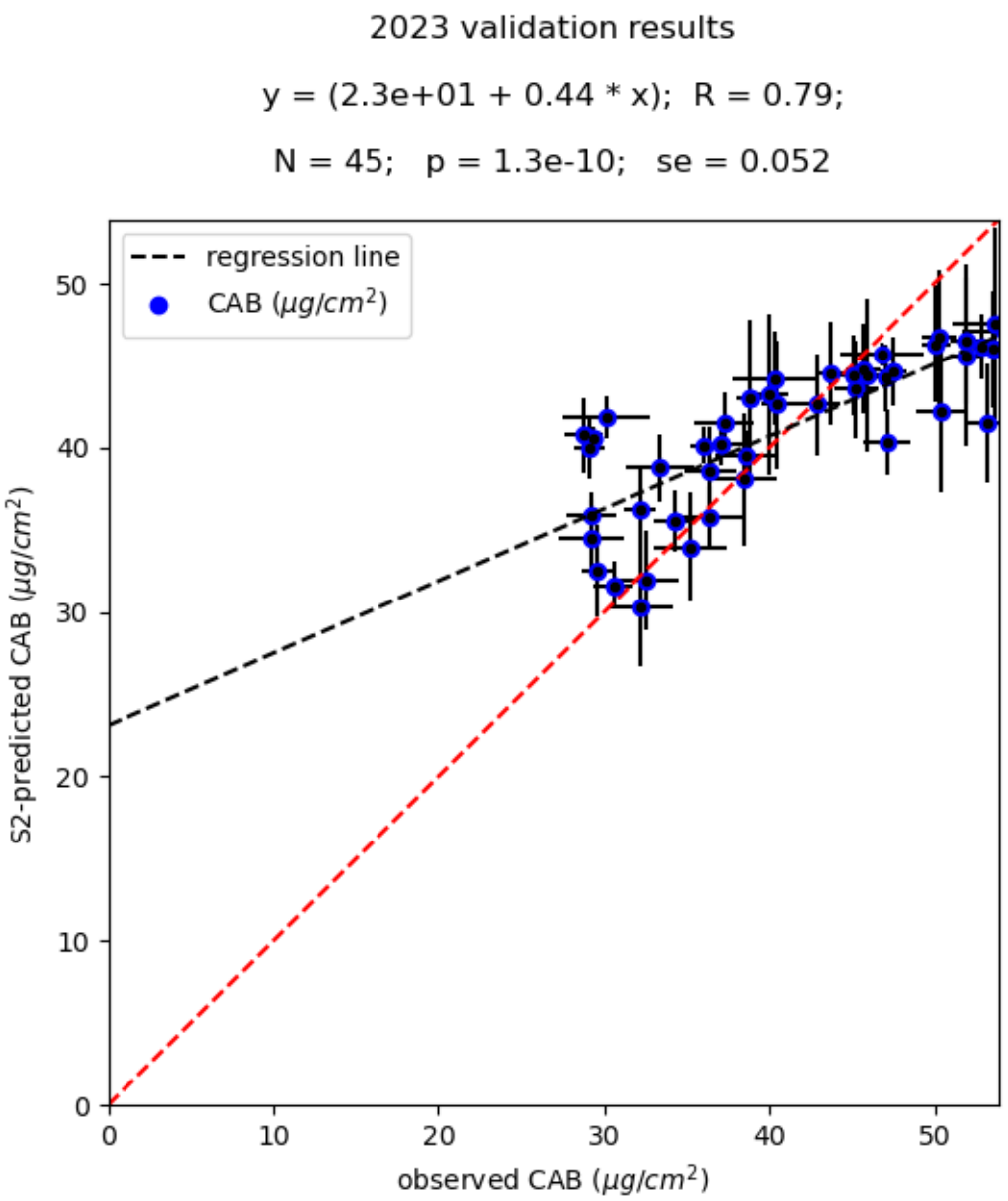

location sample 3
CAB µg/cm²
date
mean observed CAB
+/- 1 sd observed CAB
min/max observed CAB
mean predicted CAB
+/- 1 sd predicted CAB
min/max predicted CAB
start date
end date
start of season
end of growth


location sample 4
CAB µg/cm²
date
mean observed CAB
+/- 1 sd observed CAB
min/max observed CAB
mean predicted CAB
+/- 1 sd predicted CAB
min/max predicted CAB
start date
end date
start of season
end of growth


location sample 5
CAB µg/cm²
date
mean observed CAB
+/- 1 sd observed CAB
min/max observed CAB
mean predicted CAB
+/- 1 sd predicted CAB
min/max predicted CAB
start date
end date
start of season
end of growth

We now find that the Cab concentration is over-estimated by the ARC method here. The timimg of the peak of Cab is quite similar for all samples, although the peak from ARC seems slightly earlier than the peak for the measurements.

```
scatter_save(post_bio_tensor,name,mask,all_target_s2_pred, all_data,TAG,year=year)
```



The validation for 2023 indicates a bias of around 18 $\mu g/cm^2$ in the S2 predictions, and a slope of 0.5, but the temporal plots show that the issue is that the timing is slightly wrong..

6 Summary

In summary, we have perfomed a comparison of field-measured LAI and Cab over a rain-fed wheat field with estimsates of those biophysical parameters obtained from the ARC method. Observations were taken at 9 times over the season in 2023. There is a similar number of satellite observations that cover the field from cloud-free Sentinel-2 MSI data.

The approach is able to find a solution for ensemble estimates of reflectance in all S2 wavebands, and make consistent spatial and temporal estimates of LAI and Cab (and other biophysical variables, see Feng et al., 2024). When compared to field observations, the S2 ARC estimates appear to cover the same temporal

pattern as the ground observations, although the peak in values for the S2 estimates is slightly early compared to the field data. That could be an issue with not having sufficient samples of S2 at appropriate times. The LAI has a bias of 0.24, which is quite low, but a slope of around 0.7, showing a consistent under-0estimate with respect to the field LAI data. For Cab, there is a simple bias of around 16 $\mu g/cm^2$ in the S2 estimates.

All of the data and codes for users running their own experiments along the same lines as this are provided in this notebook. There is a similar mortebook and dataset for data from an irrigated wheat field measured in 2023.