Predicting Fall Foliage Color Change Using MODIS Imagery

By: Alex Herridge

Introduction

Every fall in the U.S., billions of trees lose their leaves as part of a complex, biochemical process. Before the leaves fall off, however, they transform from shades of green to the fiery colors of fall – reds, yellows, and oranges. The trees that lose their leaves every year are called deciduous, in contrast to trees that don't, which are called evergreen. This color change occurs over a period of roughly a month for a given location, although the start date varies across different latitudes and climates.

Many nature lovers, including myself, enjoy watching the forests ignite with colors. The most jaw-dropping views can be seen during peak color – the time when the leaf colors are the most vibrant. The goal of this project is to be able to evaluate the use of MODIS imagery to predict when a given deciduous forest will reach peak color.

Study Area

The area of study for this project is Willow Creek in Chequamegon-Nicolet National Forest, Wisconsin (Lat 45.8060, Long -90.0791). Willow Creek is a small region inside the 6,194.31 km^2 Chequamegon-Nicolet National Forest (Chequamegon–Nicolet National Forest). This site is mostly comprised of deciduous, broadleaf forests, which makes it ideal for this project.

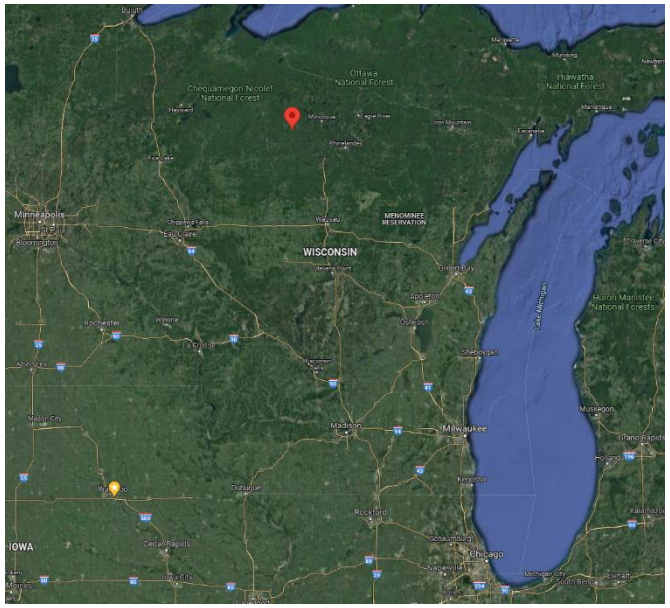


Figure 1 The location of Chequamegon–Nicolet National Forest

Figure 2 MODIS Landcover IGBP Classification Scheme (Type_1 MCD12Q1 2019)

Data Description

PhenoCam Images

The PhenoCam Network is a network of tower-mounted cameras which are designed to track phenological, or seasonal, changes in vegetation. There is a tower located at Willow Creek that has been in operation since 2011, with data available in a CSV format through 2018. This data includes daily means for the Green Chromatic Coordinate (GCC) and Red Chromatic Coordinate (RCC) among other values, which are derived from the roughly 20 images that are taken each day at the site.

Figure 3 PhenoCam image taken at Willow Creek on 9/30/21 at 11 AM

MODIS Subset Data

NASA provides Moderate Resolution Imaging Spectroradiometer (MODIS) data for fixed locations, including Willow Creek. There are multiple products available. The MOD09A1 MODIS/Terra Surface Reflectance 8-Day L3 Global 500m SIN Grid V006 dataset (Vermote) was selected for this project, as it allows for the calculation of vegetation indices. This dataset includes digital values for the 289 pixels which surround the site. There are digital values provided for each of the 7 visible light bands measured by MODIS. Only the 9 central pixels (127, 128, 129, 144, 145, 146, 161, 162, 163) were used to derive vegetation index values, as those pixels are in the closest proximity to the PhenoCam tower.

Blue box is the site pixel

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102
103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136
137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187
188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204
205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221
222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238
239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272
273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289

Data Parsing

Both sets of data were parsed and analyzed using a custom-made Python script. For details, see the Appendix.

Methods

To achieve the goal of this project, the idea is to determine which, if any, MODIS-derived vegetation indices correlate to the GCC value measured by the PhenoCam. Then, using the vegetation index values, predict what the GCC value would be, and use a regression line to determine when peak color will occur.

Vegetation Indices Evaluated

$$\text{NDVI} = \frac{\text{NIR} - \text{RED}}{\text{NIR} + \text{RED}}$$

$$\text{WDRVI}(\alpha) = \frac{\alpha(\text{NIR}) - \text{RED}}{\alpha(\text{NIR}) + \text{RED}}$$

$$\text{EVI3} = \frac{\text{G}(\text{NIR} - \text{RED})}{\text{NIR} + \text{C}_1\text{RED} + \text{C}_2 BLUE + L}, G = 2.5, \text{C}_1 = 6, \text{C}_2 = 7.5, L = 1$$

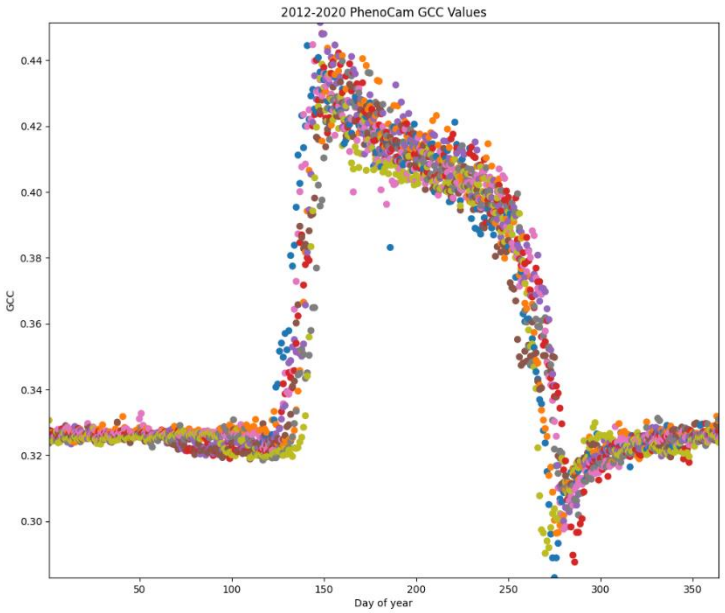
$$\text{EVI2} = \frac{\text{G}(\text{NIR} - \text{RED})}{\text{NIR} + \text{C}_1\text{RED} + L}, G = 2.5, \text{C}_1 = 2.4, L = 1$$

$$\text{Tasseled Cap Greeness} *= -0.4064(\text{B1}) + 0.5129(\text{B2}) + -0.2744(\text{B3}) + -0.2893(\text{B4}) + 0.4882(\text{B5}) + -0.0036(\text{B6}) + -0.4169(\text{B7})$$

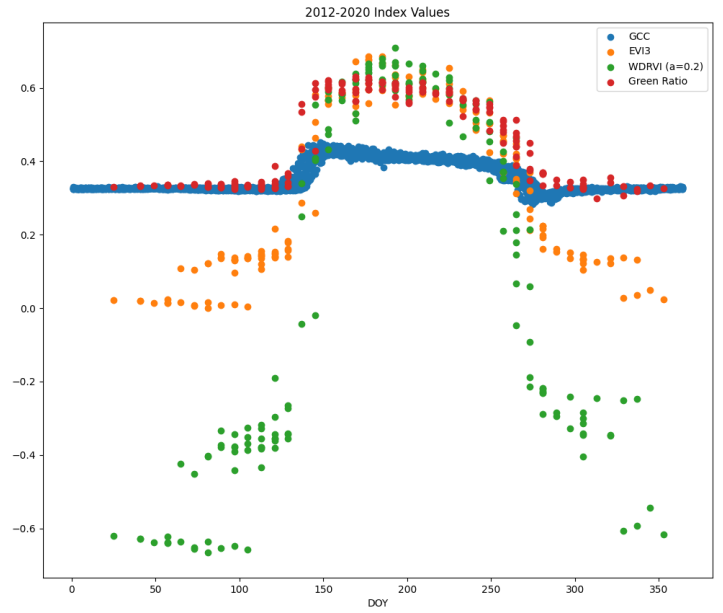
* B1 = Band 1, B2 = Band 2, etc.

$$\text{Green Ratio} = \text{GREEN}/(\text{RED} + \text{BLUE} + \text{GREEN})$$

$$\text{Complement of Red Ratio} = 1 - \text{RED}/(\text{RED} + \text{BLUE} + \text{GREEN})$$



2012-2020 PhenoCam GCC values displays the values for each year in a different color. We can see that the curve looks very similar regardless of the year.

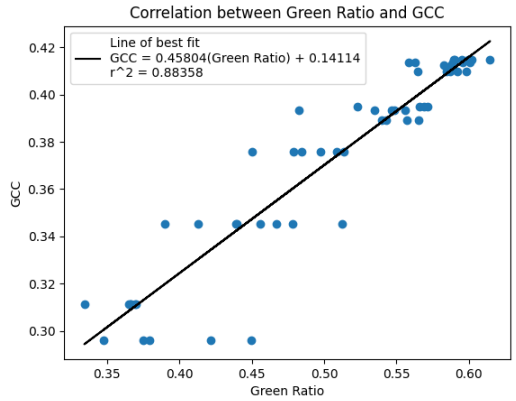


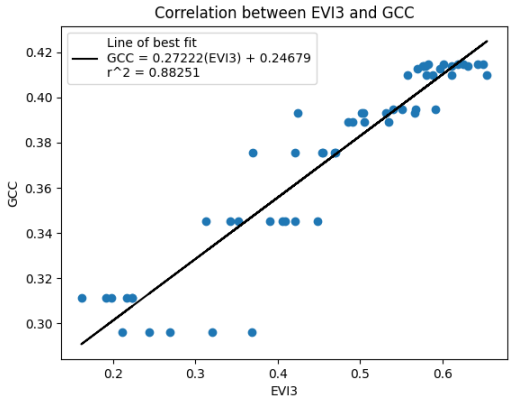
2012-2020 Index Values shows a comparison between the top vegetation indices and the PhenoCam GCC values.

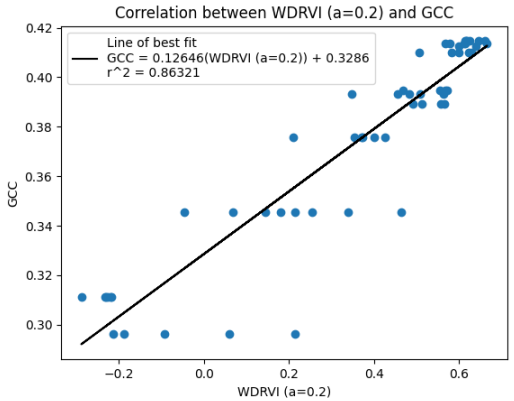
Results

GCC Estimation Models for each Vegetation Index

Index	Estimation Model	R^2
Green Ratio	GCC = 0.45804(Green Ratio) + 0.14114	0.88358
EVI3	GCC = 0.27222(EVI3) + 0.24679	0.88251
EVI2	GCC = 0.27434(EVI2) + 0.22753	0.86651
WDRVI (a=0.2)	GCC = 0.12646(WDRVI (a=0.2)) + 0.3286	0.86321
Compl. of Red Ratio	GCC = 0.508(Red Ratio) + 0.03562	0.85825
WDRVI (a=0.25)	GCC = 0.1333(WDRVI (a=0.25)) + 0.31491	0.85596
WDRVI (a=0.3)	GCC = 0.14105(WDRVI (a=0.3)) + 0.30248	0.84963
NDVI	GCC = 0.26854(NDVI) + 0.15745	0.81069
Tasseled Cap	GCC = 0.53465(T_CAP) + 0.22385	0.80376

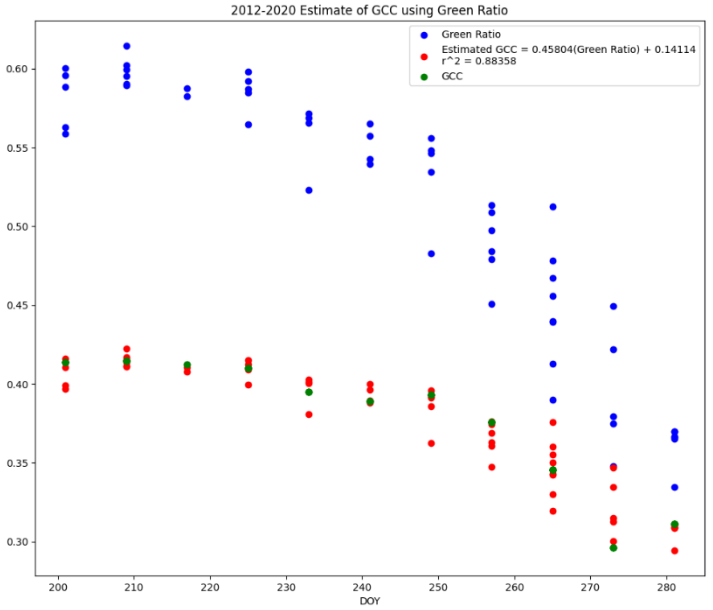


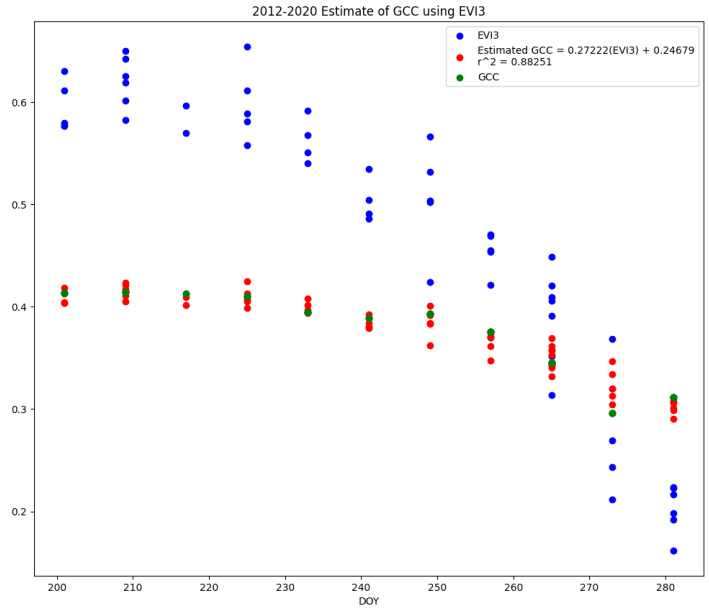


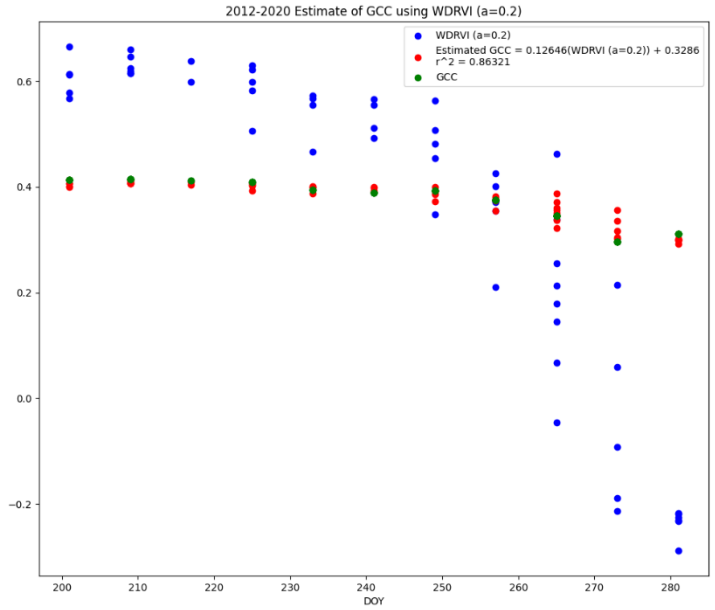


From the graphs above, we can see that there is a strong linear relationship between the vegetation index values and the PhenoCam GCC values. The Green Ratio had the best r squared value. Only three indices are displayed in these results for the sake of brevity.

Of note, there also seems to be increased variation in the vegetation index values as the GCC decreases, which could negatively impact the estimation accuracy. More investigation is required to determine the cause of this additional variation. The homogeneity of the vegetation index values at the upper end can likely be attributed to the indices saturating.







From the previous graphs, we see that all three indices yield a reasonable estimate for the PhenoCam GCC value.

Discussion

Now we have identified a vegetation index to use for GCC estimation. For the next step of this project, data should be collected from other years (e.g., 2021) to validate the model.

Ideally, this analysis would have been done using more sites within the PhenoCam network. There are many other locations that cover broadleaf, deciduous forests, whose inclusion in the analysis would broaden the scope of inference.

To use the models in this paper in a real-world scenario, the steps would be as follows:

1. Determine a GCC threshold for the period in which you are interested, such as peak color.
2. Calculate the vegetation index, whose model you would like to use (e.g., Green Ratio), for the most recent MODIS images.
3. Apply the respective model to the vegetation index values.
4. Find a line of best fit for the estimated GCC values calculated in step 3.
5. Determine when the threshold will be hit using the regression function from step 4.
6. Repeat as new images are released to increase accuracy of the predictions.

Another extension of this work could be to create a heatmap of estimated GCC values for each MODIS image in a timeseries and combine them to create an animation to show the change is GCC over time.

References

Chequamegon–Nicolet National Forest. 30 11 2021. <https://en.wikipedia.org/wiki/Chequamegon%E2%80%93Nicolet_National_Forest>.

Vermote, E. "MOD09A1 MODIS/Terra Surface Reflectance 8-Day L3 Global 500m SIN Grid V006." NASA EOSDIS Land Processes DAAC, 2015. https://doi.org/10.5067/MODIS/MOD09A1.006.

Appendix

Python Parser Code

```
import csv
from typing import Callable, Tuple

import matplotlib.pyplot as plt
import numpy as np

INVALID_VALUE = -1
VALID_YEARS = [2012, 2013, 2015, 2016, 2017, 2018, 2019, 2020]
VALID_DAYS = range(200, 282)  #range(0, 365)  #

Pixel = list[float]
Band = list[float]

RED = 0
NIR = 1
BLUE = 2
GREEN = 3


class Sample:
    acquisition_day: int
    acquisition_year: int
    product: str
    bands: list[Band]
    pixels: list[Pixel]

    def __init__(self, acquisition_day: int, acquisition_year: int, product: str, bands: list[Band]):
        self.acquisition_day = acquisition_day
        self.acquisition_year = acquisition_year
        self.product = product
        self.bands = bands

        self.pixels = []
```

```
        for i in range(len(bands[0])):
            pixel = []

            for digital_values in self.bands:
                pixel.append(digital_values[i])

            self.pixels.append(pixel)

    def valid(self) -> bool:
        for digital_values in self.bands:
            if any(digital_value == INVALID_VALUE for digital_value in digital_values):
                return False

        return self.acquisition_year in VALID_YEARS and self.acquisition_day in VALID_DAYS

    def __str__(self) -> str:
        desc = f"{self.acquisition_day}\n" \
            f"{self.acquisition_year}\n" \
            f"{self.product}\n"

        for i, digital_values in enumerate(self.bands):
            desc += f"{i}: {digital_values}\n\n"

        return desc


def read_samples_from_file(csv_file_path: str, number_of_bands: int, pixels_of_interest: list[int],
pixel_data_offset: int, rows_to_skip: int, parse_acquisition_day: Callable[[list[str]], int],
parse_acquisition_year: Callable[[list[str]], int], parse_product: Callable[[list[str]], str]) ->
list[Sample]:
    samples = []

    with open(csv_file_path, newline='') as csvfile:
        csv_reader = csv.reader(csvfile)

        for _ in range(rows_to_skip + 1):
            row = next(csv_reader)

        keep_parsing = True
```

```
        while keep_parsing:
            bands: list[Band] = []

            acquisition_day = parse_acquisition_day(row)
            acquisition_year = parse_acquisition_year(row)
            product = parse_product(row)

            for _ in range(number_of_bands):
                digital_values: Band = []

                for pixel in pixels_of_interest:
                    cell = row[pixel + pixel_data_offset]

                    if cell in ['F', 'NA']:
                        digital_values.append(INVALID_VALUE)
                    else:
                        digital_values.append(float(cell))

                bands.append(digital_values)

                try:
                    row = next(csv_reader)
                except StopIteration:
                    keep_parsing = False

            sample = Sample(acquisition_day, acquisition_year,
                            product, bands.copy())

            if sample.valid():
                samples.append(sample)

    return samples


def sample_avg(sample: Sample, callable: Callable[[Pixel], float], params, complement: bool = False):
    if complement:
        return sum(1 - callable(pixel, **params) for pixel in sample.pixels) / len(sample.pixels)
    else:
```

```
        return sum(callable(pixel, **params) for pixel in sample.pixels) / len(sample.pixels)


def ndvi(pixel: Pixel) -> float:
    return (pixel[NIR] - pixel[RED])/(pixel[NIR] + pixel[RED])


def wdrvi(pixel: Pixel, alpha: float = 0.25) -> float:
    return (alpha*pixel[NIR] - pixel[RED])/(alpha*pixel[NIR] + pixel[RED])


def evi3(pixel: Pixel) -> float:
    G = 2.5
    C1 = 6
    C2 = 7.5
    L = 1
    return G * (pixel[NIR] - pixel[RED])/(pixel[NIR] + C1*pixel[RED] + C2*pixel[BLUE] + L)


def evi2(pixel: Pixel) -> float:
    G = 2.5
    C1 = 2.4
    L = 1
    return G * (pixel[NIR] - pixel[RED])/(pixel[NIR] + C1*pixel[RED] + L)


def red_ratio(pixel: Pixel) -> float:
    return pixel[RED]/(pixel[RED] + pixel[BLUE] + pixel[GREEN])


def green_ratio(pixel: Pixel) -> float:
    return pixel[GREEN]/(pixel[RED] + pixel[BLUE] + pixel[GREEN])


def tasseled_cap_greenness(pixel: Pixel) -> float:
    # Coef from https://gis.stackexchange.com/questions/351077/how-to-calculate-and-export-tasseled-
caps-from-modis-collection-in-gee
    return -0.4064*pixel[0] + 0.5129*pixel[1] + -0.2744*pixel[2] + -0.2893*pixel[3] + 0.4882*pixel[4]
+ -0.0036*pixel[5] + -0.4169*pixel[6]
```

```
def polynomial_to_str(poly_coef: list[float], label_x: str, label_y: str) -> str:
    poly_coef = [round(coef, 5) for coef in poly_coef]

    string = f'{label_y} = '

    for index, coef in enumerate(poly_coef[:-2]):
        string += f'{coef}({label_x})^{len(poly_coef) - index - 1} '

    string += f'{poly_coef[-2]}({label_x}) + {poly_coef[-1]}'

    return string


def r_squared(data_x, data_y) -> float:
    correlation_matrix = np.corrcoef(data_x, data_y, rowvar=True)
    return round(correlation_matrix[0][1]**2, 5)


def correlation_graph(label_x: str, data_x: list[float], label_y: str, data_y: list[float]) -> None:
    _, ax = plt.subplots()

    handles = []
    labels = []

    ax.scatter(data_x, data_y)

    reg_coef = np.polyfit(data_x, data_y, 1)
    reg_fn = np.poly1d(reg_coef)
    estimated_y = [reg_fn(x) for x in data_x]
    handles.append(plt.plot(data_x, estimated_y, 'k')[0])
    labels.append(f'Line of best fit\n{polynomial_to_str(reg_coef, label_x, label_y)}\nr^2 =
{r_squared(data_y, estimated_y)}')

    print(polynomial_to_str(reg_coef, label_x, label_y))

    ax.set_title(f'Correlation between {label_x} and {label_y}')
    ax.set_xlabel(label_x)
```

```
    ax.set_ylabel(label_y)
    ax.legend(handles, labels)


def estimation_graph(label_x: str, data_x: list[float], label_y1: str, data_y1: list[float], label_y2:
str, data_y2: list[float]) -> None:
    _, ax = plt.subplots(figsize=(12,10))

    print(len(data_y2))

    handles = []
    labels = []

    handles.append(plt.scatter(data_x, data_y1, c='b'))
    labels.append(label_y1)

    y2_y1_reg_coef = np.polyfit(data_y1, data_y2, 1)
    y2_y1_reg_fn = np.poly1d(y2_y1_reg_coef)

    estimated_y2 = [y2_y1_reg_fn(y1) for y1 in data_y1]
    handles.append(plt.scatter(data_x, estimated_y2, c='r'))
    labels.append(f'Estimated {polynomial_to_str(y2_y1_reg_coef, label_y1, label_y2)}\nr^2 =
{r_squared(data_y2, estimated_y2)}')

    handles.append(plt.scatter(data_x, data_y2, c='g'))
    labels.append(label_y2)

    ax.set_title(f'{VALID_YEARS[0]}-{VALID_YEARS[-1]} Estimate of {label_y2} using {label_y1}')
    ax.set_xlabel(label_x)
    ax.legend(handles, labels)


def standard_graph(title: str, label_x: str, datasets: dict[str, Tuple[list[float], list[float]]]) ->
None:
    _, ax = plt.subplots(figsize=(12,10))

    handles = []
    labels = []
```

```
    # datasets_list = [dataset for dataset in datasets.items()]
    # label_y1, (data_x1, data_y1) = datasets_list.pop()
    # handles.append(plt.scatter(data_x1, data_y1))
    # labels.append(label_y1)

    for label_y, data in datasets.items():
        data_x, data_y = data
        handles.append(plt.scatter(data_x, data_y))
        labels.append(label_y)

    ax.set_title(title)
    ax.set_xlabel(label_x)
    if len(datasets) == 1:
        ax.set_ylabel([k for k in datasets.keys()][0])
    else:
        ax.legend(handles, labels)


if __name__ == '__main__':
    surface_reflectance_csv_file_path = 'filtered_scaled_MOD09A1.csv'
    phenocam_csv_file_path =
'S:\school\geog5225\data_processing\provisional_data\data_record_4\willowcreek_DB_1000_1day.csv'
    pixels_of_interest = [127, 128, 129, 144, 145, 146, 161, 162, 163]

    surface_reflectance_samples = read_samples_from_file(
        surface_reflectance_csv_file_path, 7, pixels_of_interest, 4, 0, parse_acquisition_day=lambda
row: int(row[2][5:]), parse_acquisition_year=lambda row: int(row[2][1:5]), parse_product=lambda row:
row[1])

    # surface_reflectance_samples.sort(key=lambda sample: sample.acquisition_day)

    phenocam_samples = read_samples_from_file(
        phenocam_csv_file_path, 1, [11, 16], 5, 23, parse_acquisition_day=lambda row: int(row[2]),
parse_acquisition_year=lambda row: int(row[1]), parse_product=lambda _: 'PhenoCam')

    # phenocam_samples.sort(key=lambda sample: sample.acquisition_day)

    data_to_graph = {}
```

```
""" Phenocam """

pheno_x = []
pheno_gcc = []

for sample in phenocam_samples:
    pheno_x.append(sample.acquisition_day)
    pheno_gcc.append(sample.pixels[0][0])

data_to_graph['GCC'] = (pheno_x, pheno_gcc)

""" Surface Reflectance """

sr_x = []

for sample in surface_reflectance_samples:
    sr_x.append(sample.acquisition_day)

synced_pheno_gcc = []
for x in sr_x:
    synced_pheno_gcc.append(pheno_gcc[pheno_x.index(x)])

red_based = False

veg_indicies = {
    'Green Ratio': (green_ratio, {}, red_based),
    'EVI3': (evi3, {}, red_based),
    'EVI2': (evi2, {}, red_based),
    'WDRVI (a=0.2)': (wdrvi, {'alpha': 0.2}, red_based),
    'Red Ratio': (red_ratio, {}, not red_based),
    'WDRVI (a=0.25)': (wdrvi, {'alpha': 0.25}, red_based),
    'WDRVI (a=0.3)': (wdrvi, {'alpha': 0.3}, red_based),
    'NDVI': (ndvi, {}, red_based),
    'T_CAP': (tasseled_cap_greenness, {}, red_based)
}

for i, veg_index in enumerate(veg_indicies.items()):
    index_name, sample_avg_params = veg_index
```

```
        print(index_name)

        vi_values = []
        for sample in surface_reflectance_samples:
            vi_values.append(sample_avg(sample, *sample_avg_params))

        print('r^2 =', r_squared(synced_pheno_gcc, vi_values))

        estimation_graph('DOY', sr_x, index_name, vi_values, 'GCC', synced_pheno_gcc)
        correlation_graph(index_name, vi_values, 'GCC', synced_pheno_gcc)
        data_to_graph[index_name] = (sr_x, vi_values)

    standard_graph(f'{VALID_YEARS[0]}-{VALID_YEARS[-1]} Index Values', 'DOY', data_to_graph)

    standard_graph(f'{VALID_YEARS[0]}-{VALID_YEARS[-1]} PhenoCam GCC Values', 'DOY', {'GCC': (pheno_x,
pheno_gcc)})

    plt.show()
```