# AgPV Index

Exploratory data analysis for creation of an AG PV index: a composite scoring metric to identify USA counties with high cobenefits potential for agrivoltaics

- Data inputs: solar supply, weather hazards, energy burden, minority owned cropland

## Data Processing

Things to do:

- aggregate NREL's solar supply data by county
- get relevant weather hazards data from csv file (hail and drought are positives, tornado is negative.)
  - might start ith overall agriculture burden for positives first
- load energy burden data
- get minoirty owned data from R2R indices to start

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
```

### Solar Supply Data NREL

Commonly cited NREL data for solar supply. I'm aggregating this by county.

https://www.nrel.gov/gis/solar-resource-maps.html

In [2]:
```
# got solar supply data
solar_dir = 'solar-pv-reference-access-2023'
solar_file = 'reference_access_2030_moderate_supply-curve.csv'

solar_df = pd.read_csv(os.path.join(solar_dir, solar_file), dtype={'cnty_fips':str}
solar_df['cnty_fips'] = solar_df['cnty_fips'].apply(lambda x: x.zfill(5))

# get total for each county
location_cols = ['cnty_fips', 'county', 'state']
solar_sums_df = solar_df.groupby(location_cols, as_index=False).sum()

# I think we just need capacity_mw_ac
solar_sums_df = solar_sums_df[location_cols + ['capacity_mw_ac', 'capacity_mw_dc']]

# rename fips column
solar_sums_df = solar_sums_df.rename(columns={'cnty_fips':'FIPS'})
```

```
solar_sums_df.head()
```

Out[2]:

| | FIPS | county | state | capacity_mw_ac | capacity_mw_dc |
|---|---|---|---|---|---|
| **0** | 01001 | Autauga | Alabama | 22349.202978 | 29947.935193 |
| **1** | 01003 | Baldwin | Alabama | 46948.973155 | 62911.632997 |
| **2** | 01005 | Barbour | Alabama | 43344.657028 | 58081.848135 |
| **3** | 01007 | Bibb | Alabama | 28368.690972 | 38014.050777 |
| **4** | 01009 | Blount | Alabama | 18483.385380 | 24767.738851 |

## NRI Weather hazards data

FEMA has created weather hazard index scores for various weather hazards. Right now, I'm only focusing on how these hazards affect agricultural losses. For AgPV, tornadoes are negative, since they destroy solar panels. Drought, hail, and heatwave are positives, since AgPV can help protect crops or diversify farmer income hagainst these hazards.

Relevant columns:

- TRND_ALRA: expected tornado ag loss rate
- HWAV_ALRA: heatwave expected ag loss rate
- HAIL_ALRA: hail expected ag loss rate
- DRGT_ALRA: drought expected loss
- RESL_SCORE: community resilience score
- SOVI_SCORE: sovial vulnerability score

can use different suffixes for different metrics. Other metrics of interest:

- RISKV: risk value
- RISKS: risk index score
- RISKR: risk index rating

https://hazards.fema.gov/nri

In [3]:
```
data_dir = 'AgPV_data'
nri_dir = 'NRI_Table_Counties'
nri_file = 'NRI_Table_Counties.csv'

nri_df = pd.read_csv(os.path.join(data_dir, nri_dir, nri_file), dtype={'STCOFIPS':s

# keep relevant columns
prefixes = ['TRND', 'HWAV', 'HAIL', 'DRGT']
suffixes = ['ALRA', 'RISKV', 'RISKS']
nri_keep_cols = ['_'.join([pf, sf]) for pf in prefixes for sf in suffixes]

nri_df = nri_df[['STCOFIPS', 'RESL_SCORE', 'SOVI_SCORE'] + nri_keep_cols]
```

```
# rename relevant columns
nri_df = nri_df.rename(columns={'STCOFIPS':'FIPS'})
nri_df.head()
```

Out[3]:

| | FIPS | RESL_SCORE | SOVI_SCORE | TRND_ALRA | TRND_RISKV | TRND_RISKS | HWAV_ALRA |
|---|---|---|---|---|---|---|---|
| **0** | 01001 | 51.810001 | 51.299999 | 0.000029 | 2.831745e+06 | 73.846643 | 2.747418e-06 |
| **1** | 01003 | 86.120003 | 31.030001 | 0.000004 | 7.982720e+06 | 91.377665 | 9.080529e-07 |
| **2** | 01005 | 6.240000 | 99.269997 | 0.000009 | 2.845056e+06 | 73.942094 | 2.747418e-06 |
| **3** | 01007 | 19.730000 | 80.779999 | 0.000067 | 2.609927e+06 | 71.969456 | 4.200027e-06 |
| **4** | 01009 | 22.820000 | 51.369999 | 0.000077 | 6.930682e+06 | 89.723194 | 4.200027e-06 |

## Energy Burden

Percent of household income spent on energy. AgPV can help high burden counties lower their energy burden.

https://www.energy.gov/scep/slsc/lead-tool

In [4]:
```
eburden_file = 'LEAD Tool Data Counties.csv'

eburden_df = pd.read_csv(os.path.join(data_dir, eburden_file), skiprows=range(0,8),
eburden_df = eburden_df.rename(columns={'Geography ID':'FIPS'})
eburden_df = eburden_df[['FIPS', 'Energy Burden (% income)']]
eburden_df.head()
```

Out[4]:

| | FIPS | Energy Burden (% income) |
|---|---|---|
| **0** | 01001 | 3 |
| **1** | 01003 | 2 |
| **2** | 01005 | 4 |
| **3** | 01007 | 4 |
| **4** | 01009 | 3 |

## R2R Data

Farmer income and % of minority owned cropland analyzed for Roads to Removal. Don't get too sad at the minority owned farm numbers.

In [5]:
```
farm_income_file = 'avg_farm_income.csv'

# 'Avg Farm Net Income ($)' has a bunch of wierd str values. '(D)' is undisclosed,
farm_income_df = pd.read_csv(os.path.join(data_dir, farm_income_file), dtype={'Coun
farm_income_df['FIPS'] = farm_income_df.apply(lambda x: x['State ANSI'].zfill(2) +
farm_income_df = farm_income_df[['FIPS', 'Avg Farm Net Income ($)']]
```

```
farm_income_df.head()
```

Out[5]:

| | FIPS | Avg Farm Net Income ($) |
|---|---|---|
| **0** | 01001 | 18,279 |
| **1** | 01011 | 71,850 |
| **2** | 01047 | 35,071 |
| **3** | 01051 | 9,847 |
| **4** | 01063 | 19,870 |

In [6]:

```
# minority owned cropland
minority_crop_file = 'minority_owned_cropland_counties.csv'

min_crop_df = pd.read_csv(os.path.join(data_dir, minority_crop_file), dtype={'fips_
min_crop_df = min_crop_df.rename(columns={'fips_code':'FIPS'})
min_crop_df = min_crop_df[['FIPS', 'Percent Minority Owned']]
min_crop_df.head()
```

Out[6]:

| | FIPS | Percent Minority Owned |
|---|---|---|
| **0** | 01005 | 0.000000 |
| **1** | 01011 | 1.984635 |
| **2** | 01013 | 0.000000 |
| **3** | 01025 | 0.000000 |
| **4** | 01043 | 0.415887 |

# Merge all DFs together

In [7]:

```
all_dfs = [solar_sums_df, nri_df, eburden_df, farm_income_df, min_crop_df]
all_dfs = [df.set_index('FIPS') for df in all_dfs]
merged_df = pd.concat(all_dfs, axis=1)
merged_df = merged_df.dropna(subset=['county'])

# percent minority owned has lots of missing data (due to a lack of minority owned
merged_df['Percent Minority Owned'] = merged_df['Percent Minority Owned'].fillna(0)

# clean farm income data. This column is the messiest
def clean_farm_income(x):
    if isinstance(x, float):
        return x

    if x.strip() == '(D)':
        return np.nan

    else:
        return float(x.replace(',', ''))
```

```
def fill_na_max(ser: pd.Series):
    return ser.fillna(ser.max())

merged_df['Avg Farm Net Income ($)'] = merged_df['Avg Farm Net Income ($)'].apply(c

# fill undefined rows (nan entries) with max value. This is most conservative estim
merged_df['Avg Farm Net Income ($)'] = fill_na_max(merged_df['Avg Farm Net Income (

# identify counties with negative net income
merged_df['Negative Net Farm Income'] = merged_df['Avg Farm Net Income ($)'].apply(

merged_df
```

Out[7]:

| | county | state | capacity_mw_ac | capacity_mw_dc | RESL_SCORE | SOVI_SCORE |
|---|---|---|---|---|---|---|
| **FIPS** | | | | | | |
| **01001** | Autauga | Alabama | 22349.202978 | 29947.935193 | 51.810001 | 51.299999 |
| **01003** | Baldwin | Alabama | 46948.973155 | 62911.632997 | 86.120003 | 31.030001 |
| **01005** | Barbour | Alabama | 43344.657028 | 58081.848135 | 6.240000 | 99.269997 |
| **01007** | Bibb | Alabama | 28368.690972 | 38014.050777 | 19.730000 | 80.779999 |
| **01009** | Blount | Alabama | 18483.385380 | 24767.738851 | 22.820000 | 51.369999 |
| **...** | ... | ... | ... | ... | ... | ... |
| **56037** | Sweetwater | Wyoming | 401041.398994 | 537395.558058 | 30.709999 | 37.400002 |
| **56039** | Teton | Wyoming | 4915.661374 | 6586.986832 | 39.529999 | 19.190001 |
| **56041** | Uinta | Wyoming | 83816.558073 | 112314.202527 | 27.820000 | 40.639999 |
| **56043** | Washakie | Wyoming | 72277.627225 | 96852.034274 | 62.029999 | 26.610001 |
| **56045** | Weston | Wyoming | 155044.608219 | 207759.812413 | 9.740000 | 23.200001 |

3082 rows × 22 columns

## Data Exploration

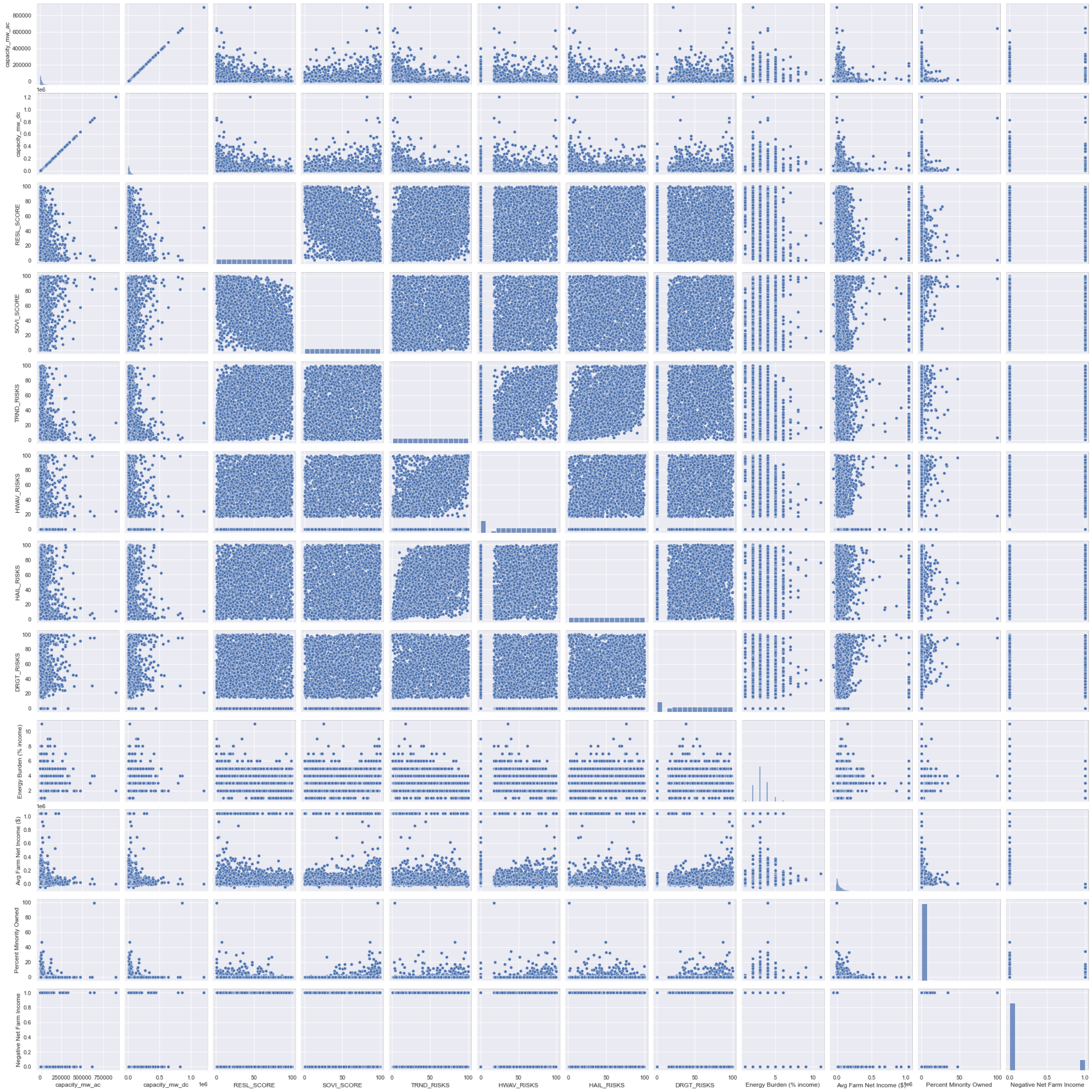
Are there strong correlations in the undrelying variables> That might skew results.

Ac and DC solar potential are nearly identical and strongly correlated, so we can just use one of them.

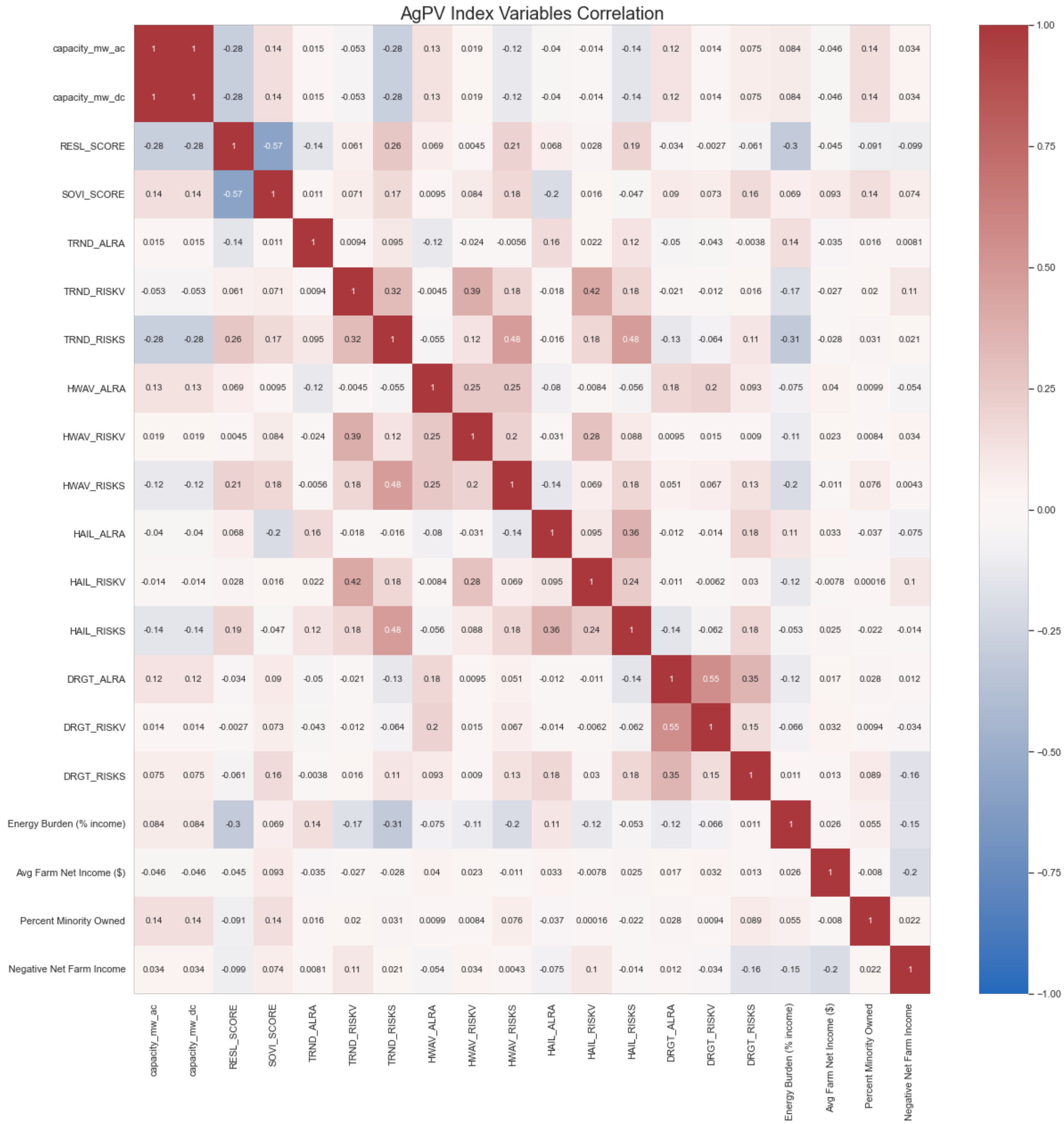
Other than that, I didn't see any strong correlations between the variables. There is some positive correlation (~0.5) between tornado risk and the other weather hazards.

In [8]:

```
# only use one set of nri metric to make the graph comprehensible
sns.set_theme(style='darkgrid')
plot_cols = [col for col in merged_df.columns if '_RISKV' not in col and '_ALRA' no
sns.pairplot(merged_df[plot_cols])

# save plot
graph_dir = 'AgPV_graphs'
graph_file = 'agpv_pairplot.png'
plt.savefig(os.path.join(graph_dir, graph_file))
```




In [9]:
```
fig, ax = plt.subplots(figsize=(20,20))
corr_df = merged_df.iloc[:, 2:].corr()
sns.heatmap(corr_df, annot=True, cmap='vlag', vmin=-1, vmax=1)
plt.title('AgPV Index Variables Correlation', size=20)

# save file
corr_file = 'agpv_corr.png'
plt.savefig(os.path.join(graph_dir, corr_file))
```




# Index Preprocessing

NRI score values seem appropriate over using the value or expected loss metric since SVI and community resiliency also use score.

For roads to removal, I used a Box-Cox transform to try and fit to a normal distribution. This time, I'm trying quantile normalization, which really forces the data to a normal distribution. The results look much more gaussian. In a sense, it's mangling the data, but since we're already doing so much to it anyway with the scaling, it's probably OK. In most cases, we get very pretty bell curves.

Process:

- Transform underlying variables to fit normal dist'n

- Minmax scale data [0-1]
- Invert 'negative' variables (tornadoes and farmer income)
- Average values of underlying variables
- Minmax scal results [0-1]

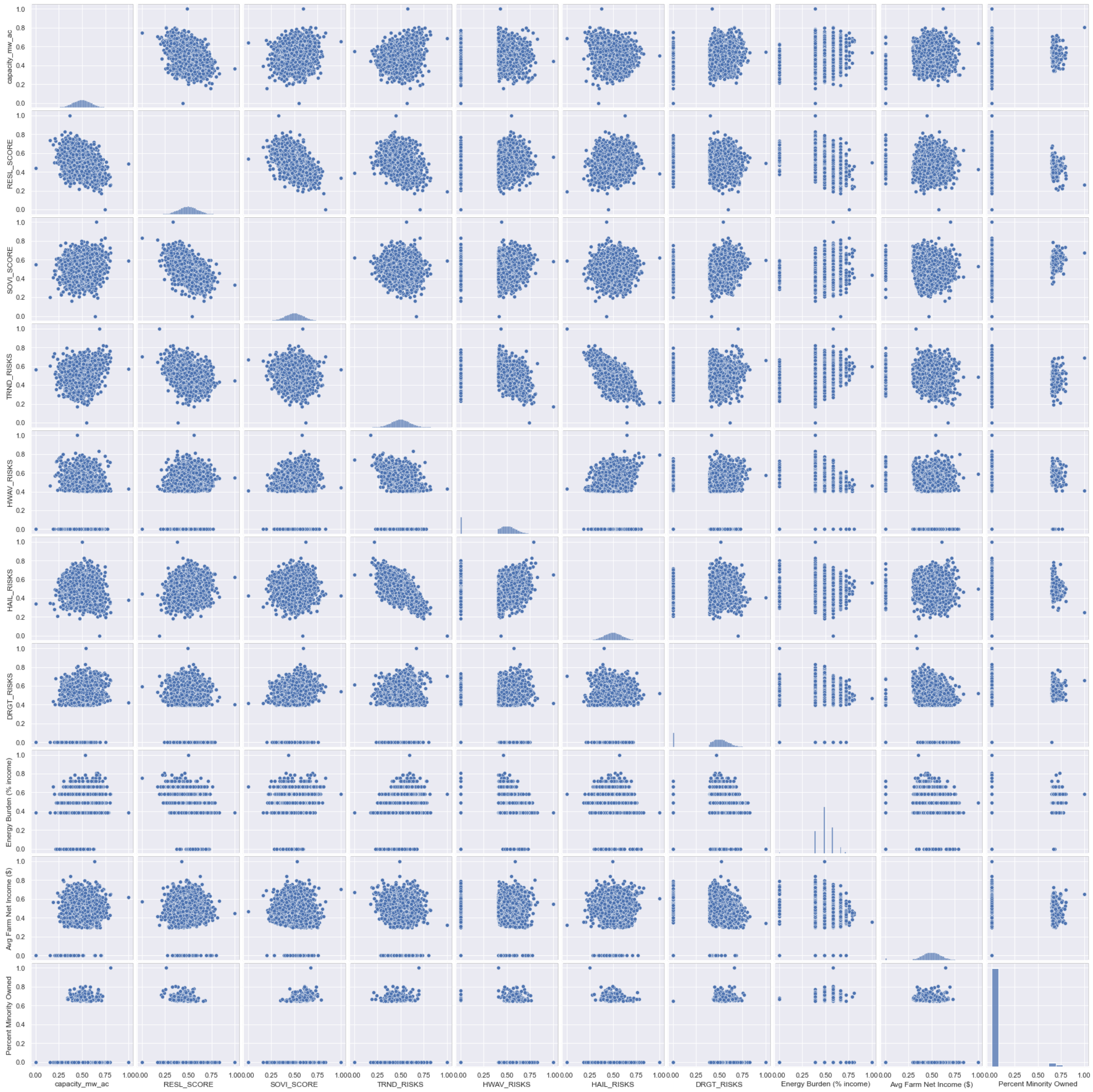
In [10]:
```
# use quantile normalization to make everything have a normal distribution
from sklearn.preprocessing import quantile_transform, MinMaxScaler
X = quantile_transform(merged_df[plot_cols].iloc[:, 2:], output_distribution='norma

# scale all values 0-1
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

# create df from transformed data
normalized_df = pd.DataFrame(X, columns=merged_df[plot_cols].iloc[:, 2:].columns, i
normalized_df = normalized_df.drop(columns=['capacity_mw_dc', 'Negative Net Farm In

# need to invert the neagtives in the index: tornado risk and farm income
normalized_df['Avg Farm Net Income ($)'] = 1 - normalized_df['Avg Farm Net Income (
normalized_df['TRND_RISKS'] = 1 - normalized_df['TRND_RISKS']
normalized_df.head()
```

Out[10]:

| | capacity_mw_ac | RESL_SCORE | SOVI_SCORE | TRND_RISKS | HWAV_RISKS | HAIL_RISKS |
|---|---|---|---|---|---|---|
| **FIPS** | | | | | | |
| **01001** | 0.547606 | 0.504096 | 0.503589 | 0.440039 | 0.607810 | 0.515519 |
| **01003** | 0.612906 | 0.603788 | 0.453189 | 0.369846 | 0.628607 | 0.505205 |
| **01005** | 0.606443 | 0.352327 | 0.735681 | 0.439753 | 0.548137 | 0.473010 |
| **01007** | 0.570571 | 0.418034 | 0.584009 | 0.445549 | 0.541421 | 0.482337 |
| **01009** | 0.530352 | 0.428265 | 0.503724 | 0.379363 | 0.588573 | 0.482416 |

In [11]:
```
sns.pairplot(normalized_df)
```

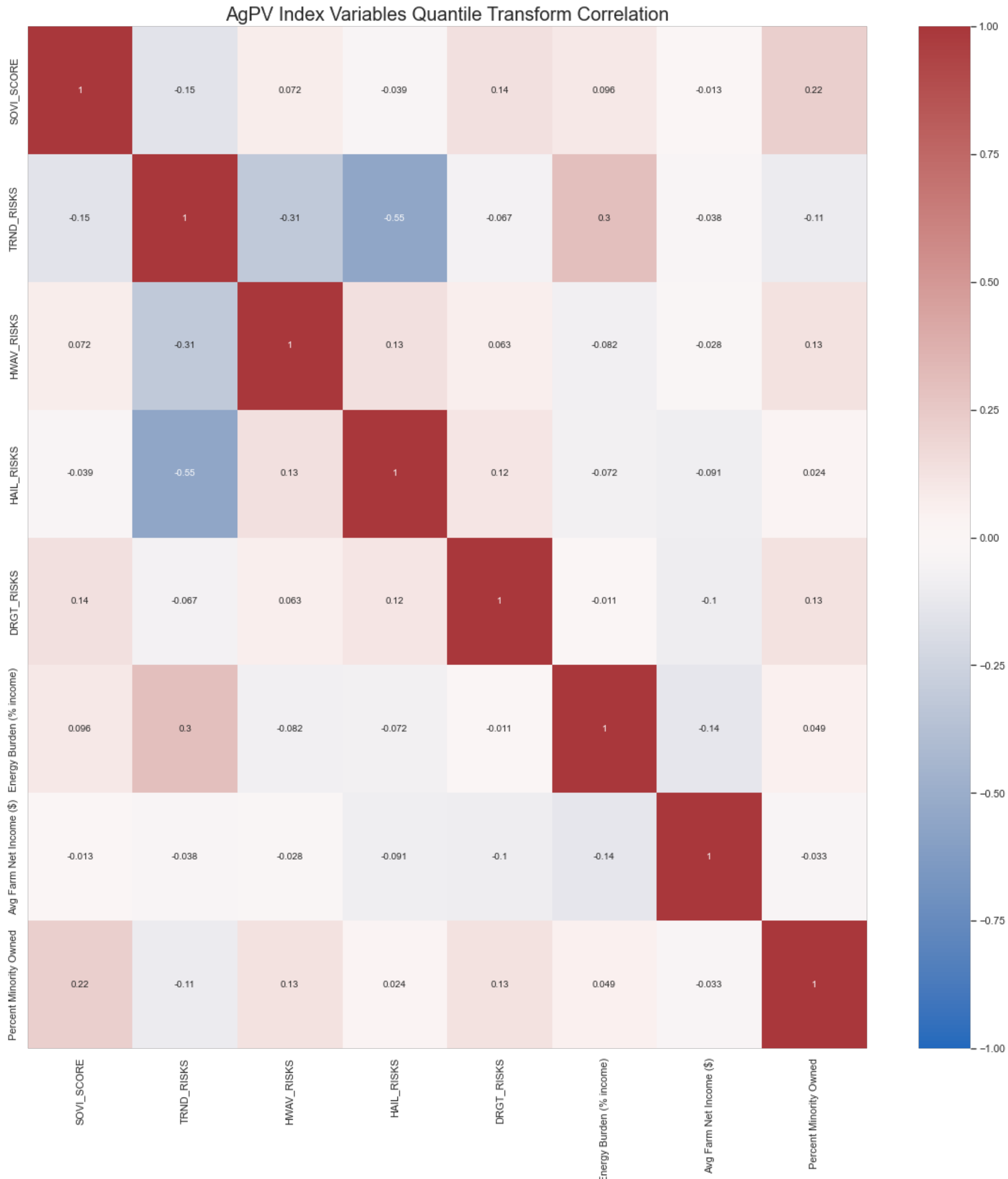
Out[11]: <seaborn.axisgrid.PairGrid at 0x28b963424c0>



The transformations seemed to slightly exxagerate correlation between toranado and hail risk. Tornado and energy burden also have some correlation (0.3). This could cause some bias in the overall index.

In [12]:
```
# correlation map with simplified variables and transformed
fig, ax = plt.subplots(figsize=(20,20))
corr_df = normalized_df.iloc[:, 2:].corr()
sns.heatmap(corr_df, annot=True, cmap='vlag', vmin=-1, vmax=1)
plt.title('AgPV Index Variables Quantile Transform Correlation', size=20)
```

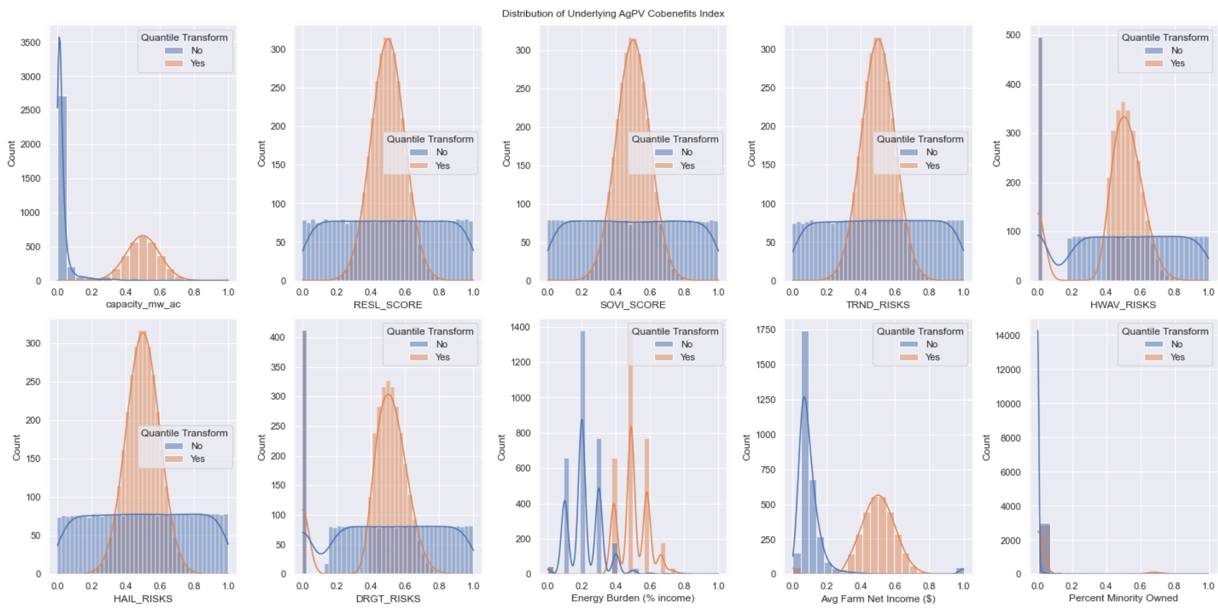
Out[12]:
```
Text(0.5, 1.0, 'AgPV Index Variables Quantile Transform Correlation')
```




In [13]:
```
# Get same columns from normalized df
plot_cols_df = merged_df[normalized_df.columns]
# scale un tranformed daata
scaled_original_data = scaler.fit_transform(plot_cols_df)
scaled_original_df = pd.DataFrame(scaled_original_data, index=plot_cols_df.index, c
# add column to signify if this data has been quantile trandformed
scaled_original_df['Quantile Transform'] = 'No'
# join dfs
xform_comp_df = pd.concat([scaled_original_df, normalized_df])
xform_comp_df['Quantile Transform'] = xform_comp_df['Quantile Transform'].fillna('Y
```

Look at how wonderful these bell curves look!

In [14]:
```
# look at histograms of transformed data compared to just minmax scaling. Look how .
fig, axes = plt.subplots(2, 5, figsize=(20,10))

for col, ax in zip(xform_comp_df.iloc[:, :-1].columns, axes.flatten()):
    sns.histplot(xform_comp_df, x=col, kde=True, ax=ax, hue='Quantile Transform')

fig.suptitle('Distribution of Underlying AgPV Cobenefits Index')
plt.tight_layout()
plt.savefig(os.path.join(graph_dir, 'agpv_variable_distribution.png'))
```




## Ag PV Index Calculation

Apache, AZ stands out due to its near 100% minority cropland ownership, reasonably high energy burden (80th percentile), high drought risk (95th percentile), low torndo risk (2nd percentile), and high solar supply (99th percentile).

Most of this land is owned by the Navajo nation, so the high minoirty ownership is not suprising.

Also note that most of the top counties in overall index seem to have high social vulnerability as well. This makes sense, since climate hazards make up much of the underlying variables.

In [15]:
```
# get average of relevant variables
index_vars = ['capacity_mw_ac', 'TRND_RISKS', 'HWAV_RISKS', 'HAIL_RISKS', 'DRGT_RIS
              'Avg Farm Net Income ($)', 'Percent Minority Owned']

# minmax scale final result
agpv_benefits_score = normalized_df[index_vars].mean(axis=1)
agpv_benefits_score = (agpv_benefits_score - agpv_benefits_score.min()) / (agpv_ben
```

In [16]:
```
# merge calculated score with underlying data
score_df = pd.concat([merged_df, agpv_benefits_score], axis=1)
score_df = score_df.rename(columns={0:'AgPV_cobenefits_score'})
```

```
# filter relevant columns
score_df = score_df[plot_cols + ['AgPV_cobenefits_score']]
score_df = score_df.drop(columns=['capacity_mw_ac'])

# show top n counties
score_df.sort_values(['AgPV_cobenefits_score'], ascending=False).head(10)
```

Out[16]:

| FIPS | county | state | capacity_mw_dc | RESL_SCORE | SOVI_SCORE | TRND_RISKS | HW |
|---|---|---|---|---|---|---|---|
| 04001 | Apache | Arizona | 861473.520644 | 0.640000 | 96.500000 | 3.372574 | |
| 46071 | Jackson | South Dakota | 151744.473926 | 0.380000 | 97.930000 | 16.671969 | |
| 40135 | Sequoyah | Oklahoma | 26711.406422 | 7.100000 | 86.220001 | 82.278078 | |
| 40107 | Okfuskee | Oklahoma | 29631.438316 | 9.930000 | 86.599998 | 41.552657 | |
| 06107 | Tulare | California | 38451.738669 | 12.830000 | 93.440002 | 33.439389 | |
| 48029 | Bexar | Texas | 37654.127497 | 38.639999 | 92.489998 | 99.872733 | |
| 46121 | Todd | South Dakota | 119771.505357 | 0.190000 | 96.279999 | 41.170856 | |
| 41045 | Malheur | Oregon | 332408.348564 | 6.080000 | 99.430000 | 10.881324 | |
| 51083 | Halifax | Virginia | 43117.128771 | 33.669998 | 70.150002 | 28.125994 | |
| 06047 | Merced | California | 64317.236668 | 15.470000 | 96.980003 | 28.635062 | |

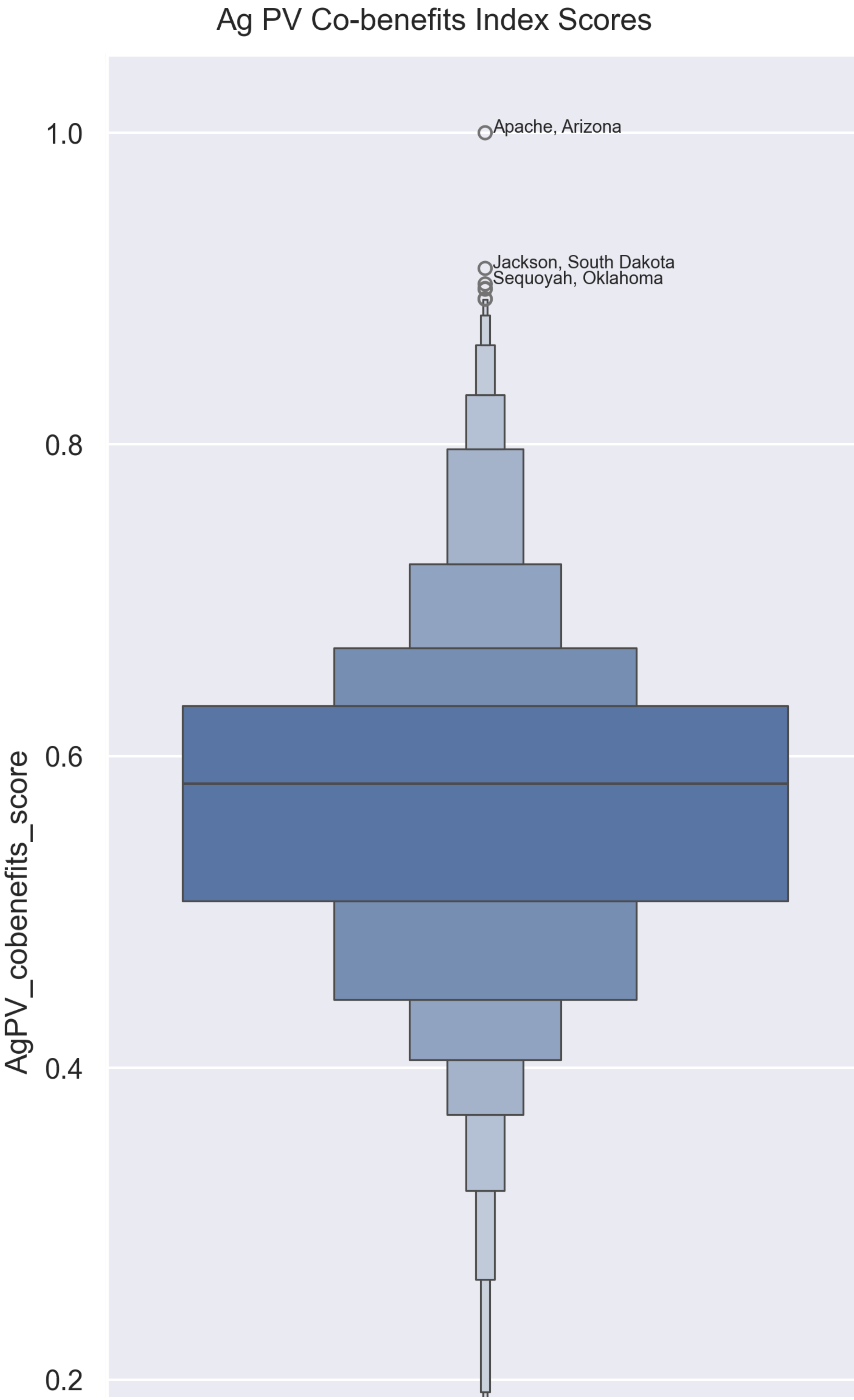
In [17]:

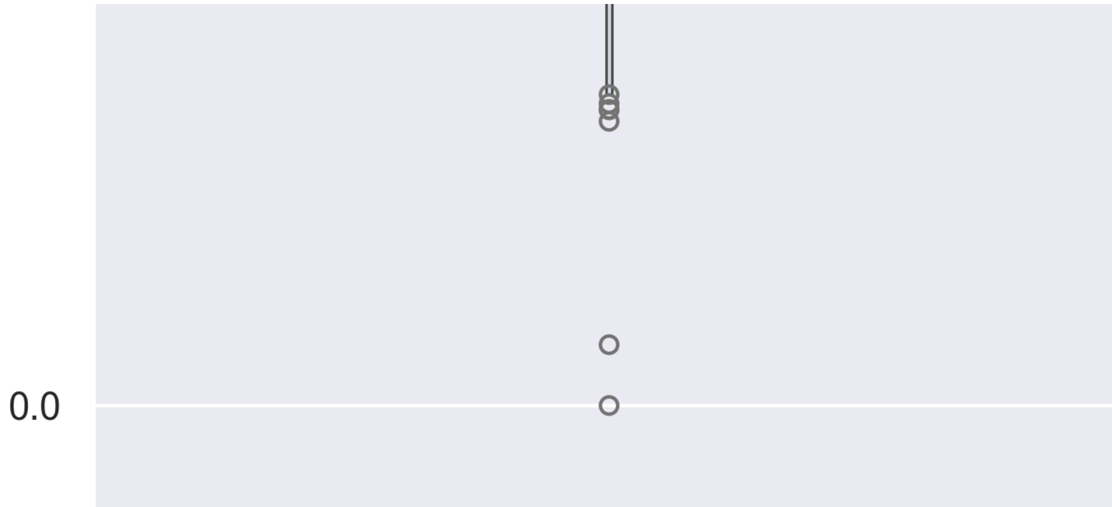
```
score_df.sort_values(['AgPV_cobenefits_score'], ascending=False).rank(pct=True).hea
```

Out[17]:

| FIPS | county | state | capacity_mw_dc | RESL_SCORE | SOVI_SCORE | TRND_RISKS | HWA |
|---|---|---|---|---|---|---|---|
| 04001 | 0.022713 | 0.024335 | 0.999676 | 0.006814 | 0.964958 | 0.024010 | 0 |
| 46071 | 0.437216 | 0.764925 | 0.957171 | 0.004218 | 0.979559 | 0.156067 | 0 |
| 40135 | 0.818624 | 0.691759 | 0.648605 | 0.071382 | 0.863076 | 0.819598 | 0 |
| 40107 | 0.674886 | 0.691759 | 0.686243 | 0.098962 | 0.866321 | 0.404932 | 0 |
| 06107 | 0.901687 | 0.060513 | 0.773848 | 0.127515 | 0.934783 | 0.322842 | 0 |

In [18]:
```
fig, ax = plt.subplots(1,1, figsize=(5,10), dpi=300)
g = sns.boxenplot(score_df, y='AgPV_cobenefits_score')

# annotate top n outliers
def annotate_outliers(row, g):
    text = ', '.join([row['county'], row['state']])
    y = row['AgPV_cobenefits_score']
    g.annotate(text, xy=(0.01, y), ha='left', size=7)

score_df.sort_values('AgPV_cobenefits_score', ascending=False).iloc[:3].apply(lambd

fig.suptitle('Ag PV Co-benefits Index Scores')
plt.tight_layout()
plt.savefig(os.path.join(graph_dir, 'agpv_score_boxen.png'))
```


Ag PV Co-benefits Index Scores
Apache, Arizona
Jackson, South Dakota
Sequoyah, Oklahoma
1.0
0.8
0.6
0.4
0.2
AgPV_cobenefits_score



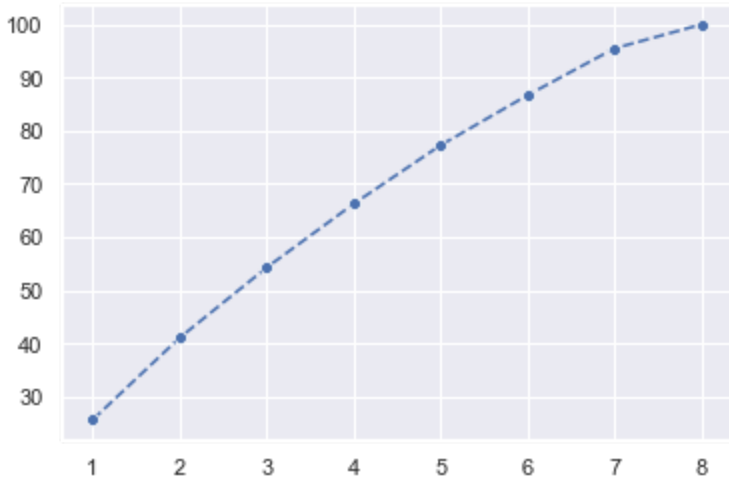

## Clustering

Let's do some simple unsupervised learning on this data set. Maybe using clusters can provide more nuaced themes and analysis for groups that a single number might overlook.

This post has some cool visualization ideas:

In [40]:
```
from sklearn.preprocessing import Normalizer, StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

# normalize data
X_norm = StandardScaler().fit_transform(merged_df[index_vars])

# principal component analysis
pca = PCA()
pca.fit(X_norm)
# get percent variance explained
percent_var = np.round(pca.explained_variance_ratio_ * 100, 3)

sns.lineplot(x=range(1, len(percent_var) + 1), y=percent_var.cumsum(), marker='o',
```

Out[40]: <Axes: >




Without quantile normaization, 6 principle components explain ~85% of the variance. This seems like a good number to choose.

In [52]:
```
n_comps=6
pca = PCA(n_components=n_comps)
pca.fit(X_norm)
scores_pca = pca.transform(X_norm)

k_values = range(2,21)
# clustering
WCSS = [] # holds within cluster sum of squares for eack value of k

for k in k_values:
    kmeans_pca = KMeans(n_clusters=k, init='k-means++', random_state=123)
    kmeans_pca.fit(scores_pca)
    WCSS.append(kmeans_pca.inertia_)
```

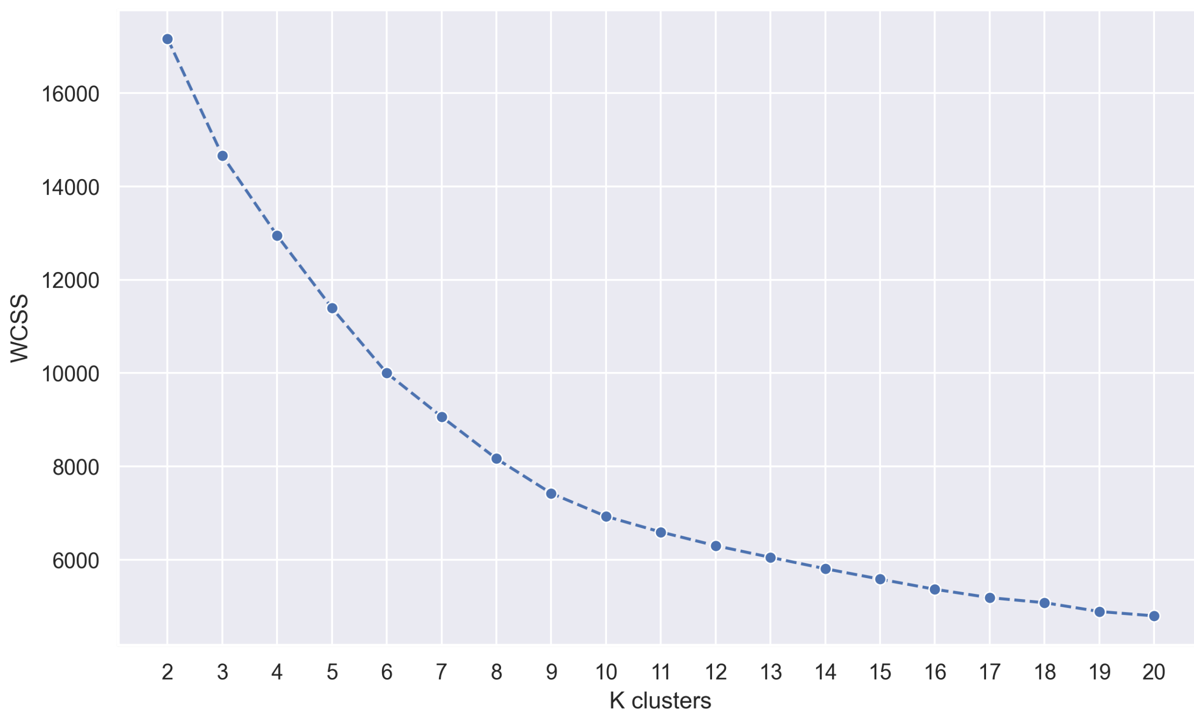
```
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

In [53]:

```python
plt.figure(figsize=(10,6), dpi=300)
g = sns.lineplot(x=k_values, y=WCSS, marker='o', linestyle='--')
g.set_xticks(k_values)
g.set_xlabel('K clusters')
g.set_ylabel('WCSS')
```

Out[53]: `Text(0, 0.5, 'WCSS')`




There's no clear elbow point. K=8 or 9 seems like there's a slight change in slope of the line. Using silhoutte score might give more detail. Let's use 8 for now.

In [54]:

```python
k = 8
kmeans_pca = KMeans(n_clusters=k, init='k-means++', random_state=123)
```

```
kmeans_pca.fit(scores_pca)
```

```
C:\Users\stanley27\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
he value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

Out[54]:

```
▾          KMeans
KMeans(random_state=123)
```

In [55]:

```
# Add pricipal components to original df
pc_col_names = ['PC' + str(k + 1) for k in range(n_comps)]
cluster_df = pd.concat([merged_df[index_vars], pd.DataFrame(scores_pca, columns=pc_

# add labels from kmeans
cluster_df['Cluster'] = kmeans_pca.labels_
cluster_df['Cluster'] = cluster_df['Cluster'].astype(str)
cluster_df
```

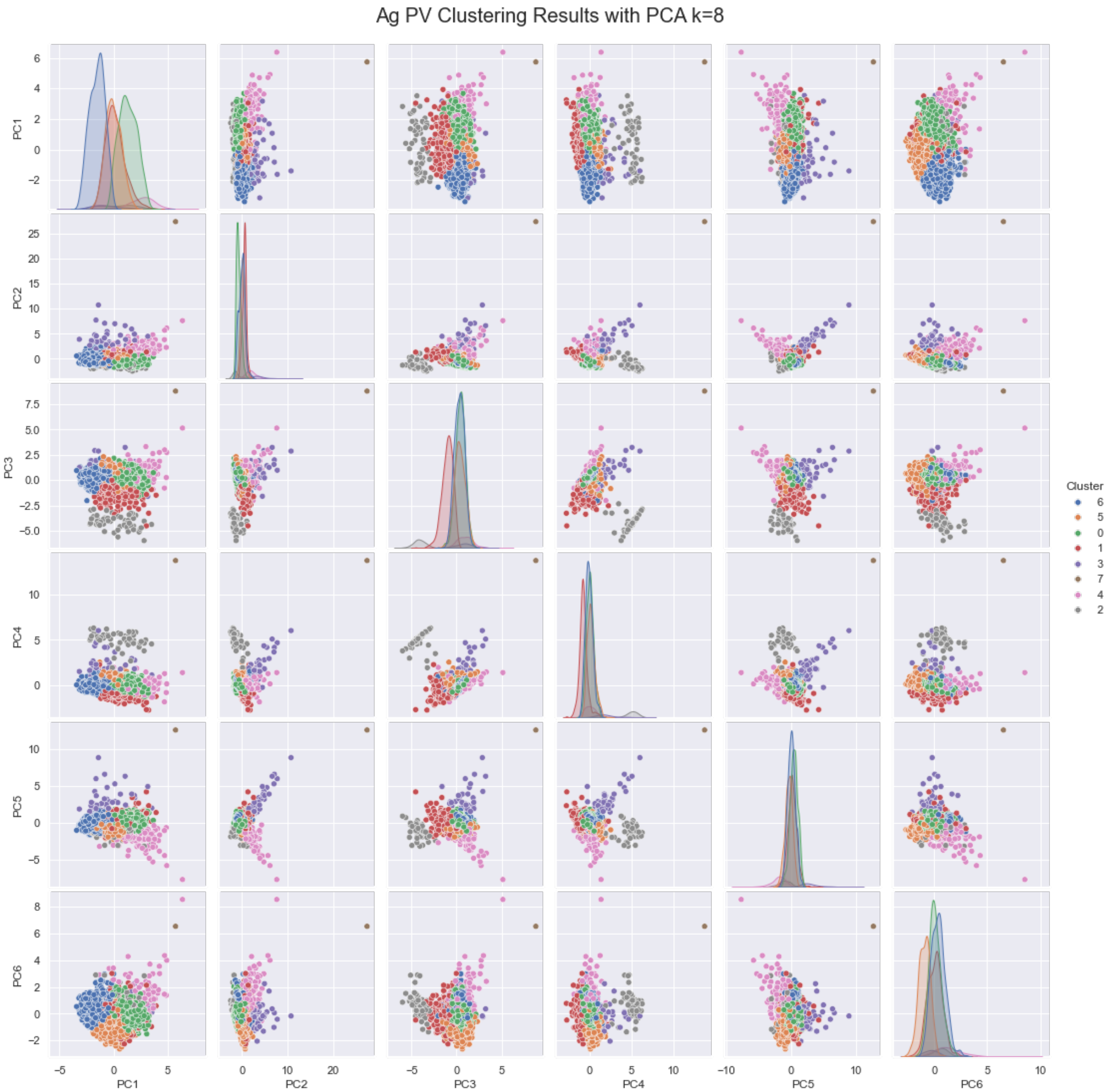
Out[55]:

| | capacity_mw_ac | TRND_RISKS | HWAV_RISKS | HAIL_RISKS | DRGT_RISKS | Energy Burden (% income) | In |
|---|---|---|---|---|---|---|---|
| **FIPS** | | | | | | | |
| **01001** | 22349.202978 | 73.846643 | 87.082405 | 57.111040 | 50.652243 | 3.0 | 18 |
| **01003** | 46948.973155 | 91.377665 | 91.091314 | 52.943048 | 86.796055 | 2.0 | 35 |
| **01005** | 43344.657028 | 73.942094 | 69.710468 | 39.993637 | 80.687241 | 4.0 | 63 |
| **01007** | 28368.690972 | 71.969456 | 67.228762 | 43.684378 | 32.421254 | 4.0 | -1 |
| **01009** | 18483.385380 | 89.723194 | 82.437162 | 43.716195 | 52.147630 | 3.0 | 66 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **56037** | 401041.398994 | 10.531339 | 0.000000 | 3.340757 | 29.812281 | 2.0 | 21 |
| **56039** | 4915.661374 | 21.189946 | 0.000000 | 39.102768 | 21.539930 | 2.0 | 6 |
| **56041** | 83816.558073 | 12.567611 | 0.000000 | 17.817372 | 25.676106 | 3.0 | 10 |
| **56043** | 72277.627225 | 13.013045 | 0.000000 | 5.058861 | 43.143493 | 4.0 | 28 |
| **56045** | 155044.608219 | 24.657970 | 0.000000 | 76.073815 | 19.153675 | 7.0 | 15 |

3082 rows × 15 columns

In [71]:

```
sns.pairplot(cluster_df.iloc[:,-n_comps-1:], hue='Cluster')
plt.suptitle('Ag PV Clustering Results with PCA k=8', size=20, y=1.02)
plt.savefig(os.path.join(graph_dir, 'agpv_pca_clustering.png'))
```




# Clustering Analysis

Most of the results seem jumbled together, but there's a few groups that stand out.

- Cluster 2 has clear separation in a few of the views.
- Cluster 4 also gets some separation
- Cluster 7 only has 1 point, but it is a clear outlier from everything else. I bet this is Apache, AZ
- The purple cluster (cluster 3) seems to be closer to cluster 7 than the other clusters

Let's have a look at the clusters:

In [67]:
```
cluster_summary = cluster_df.groupby(['Cluster']).mean().round(2)
# drop pc values
cluster_summary = cluster_summary.loc[:, :'Percent Minority Owned']
cluster_summary
```

Out[67]:

| | capacity_mw_ac | TRND_RISKS | HWAV_RISKS | HAIL_RISKS | DRGT_RISKS | Energy Burden (% income) | Ir |
|---|---|---|---|---|---|---|---|
| **Cluster** | | | | | | | |
| **0** | 20948.68 | 29.27 | 27.08 | 30.19 | 21.01 | 3.35 | |
| **1** | 23573.55 | 45.71 | 36.26 | 70.49 | 69.86 | 4.04 | |
| **2** | 16339.48 | 44.07 | 44.14 | 40.17 | 24.39 | 2.75 | 1( |
| **3** | 28635.92 | 62.10 | 72.03 | 48.65 | 68.00 | 3.74 | |
| **4** | 233861.88 | 14.58 | 30.13 | 26.53 | 61.93 | 3.31 | |
| **5** | 19520.99 | 47.86 | 62.51 | 29.67 | 66.85 | 2.91 | |
| **6** | 16671.52 | 80.31 | 70.85 | 73.52 | 48.11 | 2.59 | |
| **7** | 642890.58 | 3.37 | 17.82 | 1.34 | 95.26 | 4.00 | |

Cluster 7 is Apache AZ. High minority cropland ownership!

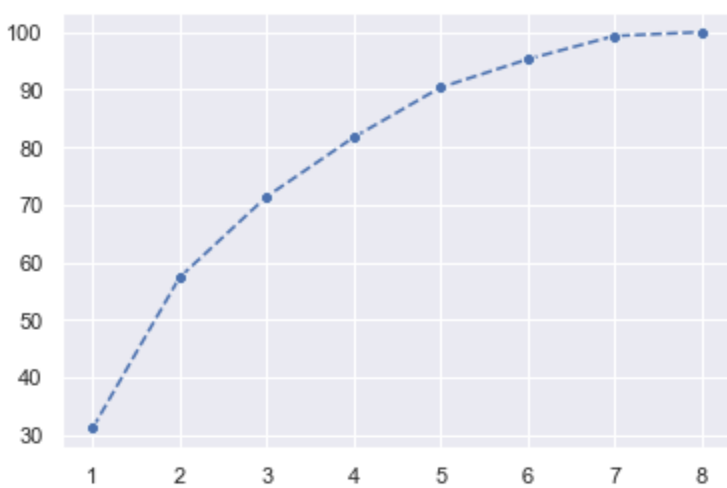
Cluster 4 is defined by it's high solar potential Cluster 2 has high Avg Farm Net income Cluster 6: high tornado risk

In [392…

```
# normalize data
X_norm = Normalizer().fit_transform(normalized_df[index_vars])

# principal component analysis
pca = PCA()
pca.fit(X_norm)
# get percent variance explained
percent_var = np.round(pca.explained_variance_ratio_ * 100, 3)

sns.lineplot(x=range(1, len(percent_var) + 1), y=percent_var.cumsum(), marker='o',
```

Out[392…

```
<Axes: >
```



```
In [ ]:
```