Transform

Ingest the raw data from the Bureau of Labor Statistics and transform it into simplified files prepared for analysis.

In [1]:
```
import os
import cpi
import pandas as pd
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
pd.set_option("display.max_columns", None)
```

Set all the years of data to transform

In [4]:
```
years = range(1990, 2016)
```

The shortlist of industries to extract from the data

In [5]:
```
whitelist = pd.DataFrame([
    ('10', 'Total, all industries', 'total'),
    ('111', 'Crop production', 'crops'),
    ('1151', 'Support activities for crop production', 'crops'),
], columns=['industry_code', 'industry_name', 'industry_group'])
```

Where to find the CSV files

In [6]:
```
path_template = './data/{}.annual.singlefile.csv'
```

Area titles crosswalk to decode the raw data files

In [30]:
```
area_titles = pd.read_csv("./data/area_titles.csv")
```

Loop through all years and transform the state and county level data for each

In [8]:
```
for year in years:
    print "Transforming {}".format(year)

    # Read in the csv
    df = pd.read_csv(path_template.format(year), dtype={"area_fips": str})

    # Decode the area titles
    df = df.merge(area_titles, on="area_fips", how="inner")

    # Filter it down to desired industries using whitelist
    filtered_df = df.merge(whitelist, on='industry_code', how="inner")

    # Filter it down to the statewide aggregation level for each industry
    state_df = filtered_df[
        # Statewide totals for all industries
        ((filtered_df.agglvl_code == 50) & (filtered_df.industry_group == 'tot
al')) |
        # Statewide totals for our selected industries
        (
            (filtered_df.agglvl_code.isin([55, 56])) &
            (filtered_df.own_code == 5) &
            (filtered_df.industry_group == 'crops')
        )
    ]

    # Filter it down to the county aggregation level for each industry
    county_df = filtered_df[
        # County totals for all industries
        ((filtered_df.agglvl_code == 70) & (filtered_df.industry_group == 'tot
al')) |
        # County totals for our selected industries
        (
            (filtered_df.agglvl_code.isin([75, 76])) &
            (filtered_df.own_code == 5) &
            (filtered_df.industry_group == 'crops')
        )
    ]

    # Trim to only the columns we want
    trimmed_columns = [
        'area_fips',
        'area_title',
        'industry_code',
        'industry_name',
        'industry_group',
        'agglvl_code',
        'year',
        'own_code',
        'avg_annual_pay',
        'annual_avg_emplvl',
        'total_annual_wages',
    ]
    trimmed_state_df = state_df[trimmed_columns]
    trimmed_county_df = county_df[trimmed_columns]

    # Adjust wages for inflation
```

```
    trimmed_state_df['total_annual_wages_2015'] = trimmed_state_df.apply(
        lambda x: cpi.to_2015_dollars(x.total_annual_wages, x.year),
        axis=1
    )
    trimmed_county_df['total_annual_wages_2015'] = trimmed_county_df.apply(
        lambda x: cpi.to_2015_dollars(x.total_annual_wages, x.year),
        axis=1
    )

    # Group totals by industry group
    groupby = [
        'year',
        'area_fips',
        'area_title',
        'industry_group'
    ]
    aggregation = {
        'annual_avg_emplvl': 'sum',
        'total_annual_wages_2015': 'sum'
    }
    grouped_state_df = trimmed_state_df.groupby(groupby).agg(aggregation).rese
t_index()
    grouped_county_df = trimmed_county_df.groupby(groupby).agg(aggregation).re
set_index()

    # Recalculate average pay for the new group
    grouped_state_df['avg_annual_pay_2015'] = (
        grouped_state_df.total_annual_wages_2015 / grouped_state_df.annual_avg
_emplvl
    )
    grouped_county_df['avg_annual_pay_2015'] = (
        grouped_county_df.total_annual_wages_2015 / grouped_county_df.annual_a
vg_emplvl
    )

    # Write out each annual file separately
    grouped_state_df.to_csv("./data/transformed_state_{}.csv".format(year), in
dex=False)
    grouped_county_df.to_csv("./data/transformed_county_{}.csv".format(year),
index=False)
```

```
Transforming 1990
Transforming 1991
Transforming 1992
Transforming 1993
Transforming 1994
Transforming 1995
Transforming 1996
Transforming 1997
Transforming 1998
Transforming 1999
Transforming 2000
Transforming 2001
Transforming 2002
Transforming 2003
Transforming 2004
Transforming 2005
Transforming 2006
Transforming 2007
Transforming 2008
Transforming 2009
Transforming 2010
Transforming 2011
Transforming 2012
Transforming 2013
Transforming 2014
Transforming 2015
```

Combine all the annual files

In [9]:
```
combined_state_df = pd.concat(
    [pd.read_csv("./data/transformed_state_{}.csv".format(year), dtype={"area_fips": str}) for year in years],
    ignore_index=True
)
```

In [10]:
```
combined_county_df = pd.concat(
    [pd.read_csv("./data/transformed_county_{}.csv".format(year), dtype={"area_fips": str}) for year in years],
    ignore_index=True
)
```

Write them out

In [11]:
```
combined_state_df.to_csv("./data/transformed_state.csv", index=False)
```

In [12]:
```
combined_county_df.to_csv("./data/transformed_county.csv", index=False)
```