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

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',
            1
            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'
   1
   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

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)
```