Accessing ARCOS data in R and Python

Authors: arcos and arcospy developers

Last updated: August 7, 2020

Installation

The original `arcos` software was written in R by members of the data reporting team (Steven Rich, Andrew Ba Tran, Aaron Williams, Jason Holt) at *The Washington Post*. Interested researchers at the University of Maryland (Jeffery Sauer, Dr. Taylor M. Oshan) translated the software to python and called it `arcospy`. Both software offer the same functionality as an API wrapper to query the ARCOS data (hosted here).

R

The latest release of `arcos` is available via CRAN, while the development version is available from GitHub.

```r
# R

# CRAN

install.packages("arcos")

# Github

# install.packages("devtools")
devtools::install_github('wpinvestigative/arcos')
```

The R dependencies are `string`, `stringr`, `magrittr`, `jsonlite`, `dplyr`, `urltools`, and `vroom`.

Python

Python is available on PyPI and can be installed via `pip`.

```
# Python

pip install arcospy
```

The Python dependencies are `pandas` and `requests`.

Quick start

All commands share the name name between `arcos` and `arcospy`. Users can choose between commands that return raw data, commands that return summarized data, and commands that return potentially relevant supplemental data. Note that commands that return raw data may take significantly longer to run due to large size of the underlying datasets.

Once users have chosen a command that suits their needs, the general format of the API calls for the geographic areas of interest (such as county and state) as well as an API key. The default key is “WaPo” (additional keys available here). Outputs from all of the functions are delivered in popular formats, specifically `data.frames` in R and `pandas.DataFrame` in Python. We now demonstrate generic functionality with the `state_population` command to get annual population values for states between 2006 and 2012.

R API

```r
# R

library(arcos)
statepop <- state_population(state = "WV", key = "WaPo")
head(statepop)
```

```
##   BUYER_STATE year population
## 1          WV 2009    1811403
## 2          WV 2010    1840802
## 3          WV 2011    1846372
## 4          WV 2012    1850481
## 5          WV 2006    1827912
## 6          WV 2007    1834052
```

Help for the R `arcos` package can be retrieved via `?arcos`. Additional documentation on the R API is available via CRAN.

Python API

```python
# Python

import arcospy
statepop = arcospy.state_population(state = "WV", key = "WaPo")
statepop.head()
```

```
##   BUYER_STATE  year  population
## 0          WV  2009     1811403
## 1          WV  2010     1840802
## 2          WV  2011     1846372
## 3          WV  2012     1850481
## 4          WV  2006     1827912
```

Help for the Python `arcospy` package can be retrieved via `help(arcospy)`. Additional documentation on the Python API is available via Github.

Example use: county-level analysis over time

Gathering data

One of the most straightforward analyses of the ARCOS data is examining trends in opioid orders at the county level over time. This analysis can be used to highlight relative increases and decreases of orders during the ramp-up phase of the Opioid Crisis (mid-2000s) towards its peak (early 2010s). To do so, we make use of the `summarized_county_annual` function to get the total number of opioid dosage units provided to each county of a given state from 2006 to 2014. For the purposes of this tutorial, we limit the analysis to Massachusetts.

First, gather the total number of dosage units for each county of Massachusetts.

```
# R

MA_orders <- summarized_county_annual(state = "MA", key = "WaPo")

head(MA_orders)
```

```
##   BUYER_COUNTY BUYER_STATE year count DOSAGE_UNIT countyfips
## 1   BARNSTABLE          MA 2006 17387     6526795      25001
## 2   BARNSTABLE          MA 2007 17904     7078630      25001
## 3   BARNSTABLE          MA 2008 17769     7291600      25001
## 4   BARNSTABLE          MA 2009 17493     7777070      25001
## 5   BARNSTABLE          MA 2010 17673     7815550      25001
## 6   BARNSTABLE          MA 2011 17035     7630040      25001
```

```
# Python

MA_orders = arcospy.summarized_county_annual(state = "MA", key = "WaPo")

MA_orders.head()
```

```
##   BUYER_COUNTY BUYER_STATE  year  count  DOSAGE_UNIT countyfips
## 0   BARNSTABLE          MA  2006  17387      6526795      25001
## 1   BARNSTABLE          MA  2007  17904      7078630      25001
## 2   BARNSTABLE          MA  2008  17769      7291600      25001
## 3   BARNSTABLE          MA  2009  17493      7777070      25001
## 4   BARNSTABLE          MA  2010  17673      7815550      25001
```

Adjusting by population

However, the above orders are unadjusted That is, they do not account for the underlying population wherein counties with more people are more likely to order more opioids. `arcos` has a built-in function to gather the population data called `county_population`. We now gather the population data, summarize the population data at the county level, join it to the opioid orders data, and create an average pills per person per year metric.

```
# R

# Gather the population data

MA_pop <- county_population(state = "MA", key = "WaPo")

# Summarize the data at the county level

library(dplyr)
MA_pop <- MA_pop %>%
  group_by(BUYER_COUNTY, BUYER_STATE, countyfips) %>%
  summarize(avg_pop=mean(population, na.rm=T)) %>%
  rename(buyer_county=BUYER_COUNTY, buyer_state=BUYER_STATE)

# Join to the opioid orders data
```

```
MA_merge <- left_join(MA_orders, MA_pop)

# Create an adjusted average pills per person per year metric

MA_merge <- MA_merge %>%
  mutate(pills_per_person_avg=DOSAGE_UNIT/avg_pop/9)

head(MA_merge)
```

```
##   BUYER_COUNTY BUYER_STATE year count DOSAGE_UNIT countyfips buyer_county
## 1   BARNSTABLE          MA 2006 17387     6526795      25001   BARNSTABLE
## 2   BARNSTABLE          MA 2007 17904     7078630      25001   BARNSTABLE
## 3   BARNSTABLE          MA 2008 17769     7291600      25001   BARNSTABLE
## 4   BARNSTABLE          MA 2009 17493     7777070      25001   BARNSTABLE
## 5   BARNSTABLE          MA 2010 17673     7815550      25001   BARNSTABLE
## 6   BARNSTABLE          MA 2011 17035     7630040      25001   BARNSTABLE
##   buyer_state  avg_pop pills_per_person_avg
## 1          MA 217652.3             3.331917
## 2          MA 217652.3             3.613627
## 3          MA 217652.3             3.722348
## 4          MA 217652.3             3.970180
## 5          MA 217652.3             3.989824
## 6          MA 217652.3             3.895121
```

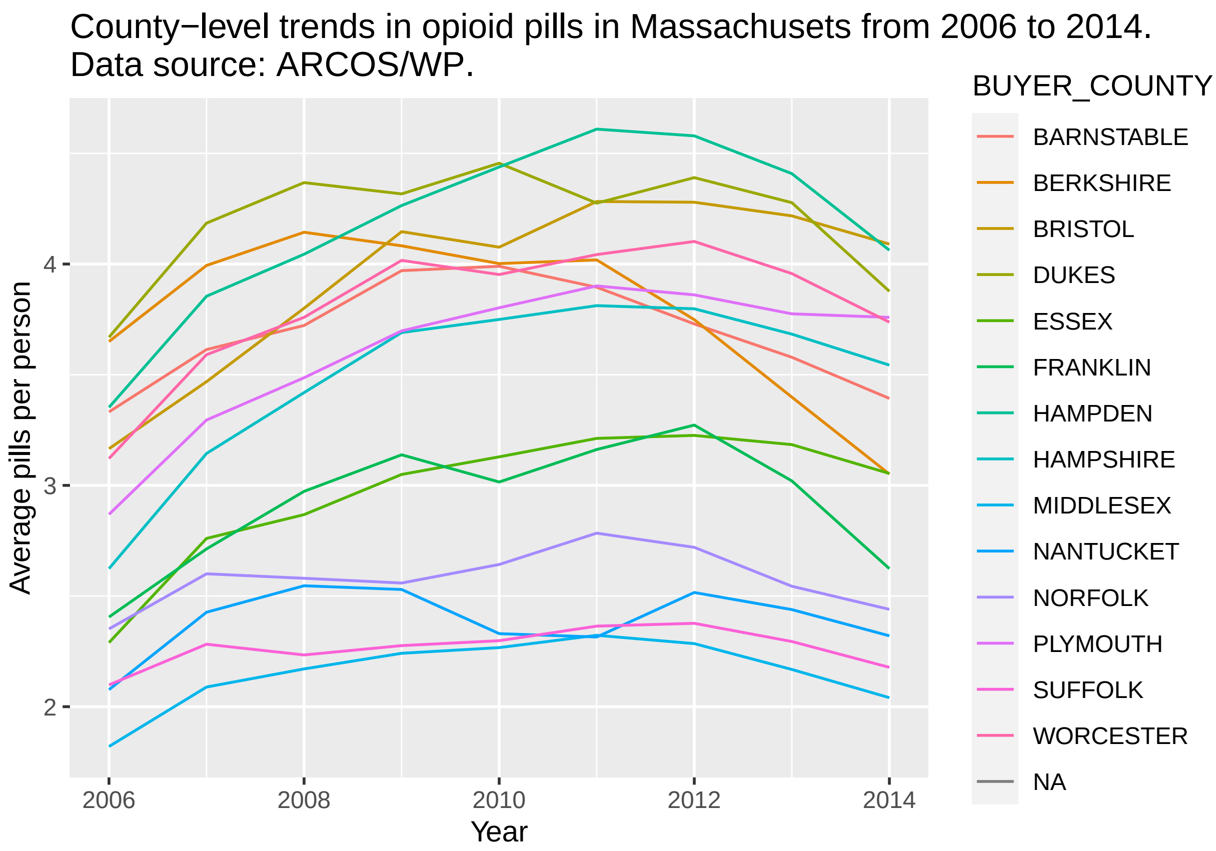
```
# Python

# Gather the population data

MA_pop = arcospy.county_population(state="MA", key="WaPo")

# Summarize the data at the county level

MA_pop = MA_pop.groupby(['BUYER_COUNTY', 'BUYER_STATE', 'countyfips']).mean().reset_index()

# Join to the opioid orders data

import pandas as pd
MA_merge = pd.merge(MA_orders, MA_pop, on=['BUYER_COUNTY', 'BUYER_STATE', 'countyfips'])

# Clean up duplicated year column
del MA_merge['year_y']
MA_merge = MA_merge.rename(columns={"year_x": "year"})

# Create an adjusted average pills per person per year metric

MA_merge['pills_per_person'] = MA_merge['DOSAGE_UNIT']/MA_merge['population']/9

# Sort padnas.df to match R output from above
MA_merge = MA_merge.sort_values('BUYER_COUNTY')
MA_merge.head()
```

```
##   BUYER_COUNTY BUYER_STATE  year  ...  COUNTY     population pills_per_person
## 0   BARNSTABLE          MA  2006  ...     1.0  217652.333333         3.331917
## 1   BARNSTABLE          MA  2007  ...     1.0  217652.333333          3.613627
## 2   BARNSTABLE          MA  2008  ...     1.0  217652.333333          3.722348
## 3   BARNSTABLE          MA  2009  ...     1.0  217652.333333          3.970180
## 4   BARNSTABLE          MA  2010  ...     1.0  217652.333333          3.989824
##
## [5 rows x 10 columns]
```

Rendering graphs

Now that we have an adjusted measure of the number of pills ordered to a given county in a given year, we can create some simple visualizations to see county-level trends. For the purposes of this tutorial, we will render some straightforward line graphs, although additional documentation is available for R users and Python users on how to make maps using the ARCOS data.

```
# R

library(ggplot2)

# Render plot

MA_merge %>%
  ggplot(aes(x=year,
             y=pills_per_person_avg,
             group=BUYER_COUNTY,
             color=BUYER_COUNTY)) +
    geom_line() +
  ylab('Average pills per person')+
  xlab('Year') +
  ggtitle('County-level trends in opioid pills in Massachusets from 2006 to 2014. \nData source: ARCOS/W
```

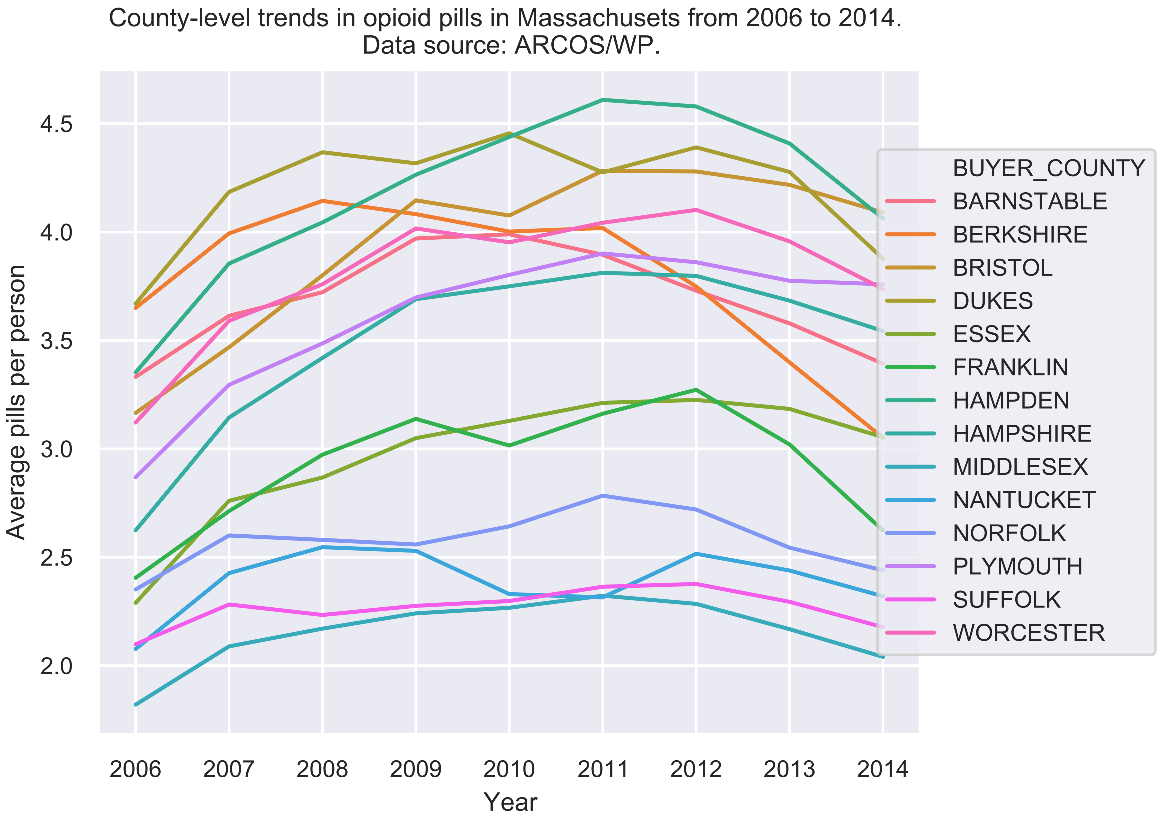


```
# Python

import seaborn as sns
sns.set(font_scale=0.8)
import matplotlib.pyplot as plt

# Render plot
# Note: we need to include a few extra lines to render the
# legend outside the plot. See:
# https://stackoverflow.com/questions/30490740/move-legend-outside-figure-in-seaborn-tsplot

fig, ax1 = plt.subplots(1,1)

g = sns.lineplot(x="year", y="pills_per_person", hue="BUYER_COUNTY", data=MA_merge, ax=ax1)
g.set(xlabel="Year", ylabel="Average pills per person")
```

```
## [Text(0.5, 0, 'Year'), Text(0, 0.5, 'Average pills per person')]
```

```
g.set_title("County-level trends in opioid pills in Massachusets from 2006 to 2014. \nData source: ARCOS

# Automatically adjust bounding box of figure

box = g.get_position()
g.set_position([box.x0, box.y0, box.width * 0.85, box.height])

# Place legend
```

```
g.legend(loc='center right', bbox_to_anchor=(1.30, 0.5), ncol=1)

plt.show()
```



The above spaghetti graphs help to identify the range of opioid availability in a given county in Massachusetts. While the numbers may seem relatively low (e.g. only 1-5 pills), remember that this is a **population-level** measure. There is also clear variation among counties in a given state, even after adjusting for population.

Users should note that the above visualization methods may not be suitable for all states. Massachusetts has only 10 counties, whereas most states have more than 20 counties. So many lines would likely render the above plot uninterpretable, so we encourage users to pursue additional options.

Important considerations for use

As preivously hinted at with the adjustment of population, there are a few important considerations users should make when using the ARCOS data.

First and foremost, adjusting by population is almost always a safe adjustment to be made across raw, summarized, and pharmacy-level data. While we provide one method of adjustment above, users may also adjust by yearly-specific measures of population as well.

Secondly, users might consider looking at specific subsets of drug distributors. Specifically, users should examine the variable `BUYER_BUS_ACT` returned by several functions. This variable describes the type of distributor (chain, retail pharmacy, private, medical, etc). Depending on the type of interest of the user, certain types of distributors may or may not need to be excluded.

Lastly, as discussed elsewhere in the `arcos` and `arcospy` documentation there are numerous ways to conceptualize the unit of analysis for the distribution of opioids. In the data currently available via `arcos` and `arcospy`, these units are primarily:

- total number of records for a geographic unit (adjusted or unadjusted)
- total number of all opioid pills (adjusted or unadjusted)
- total number of specific opioid pills (adjusted or unadjusted)
- total amount (in weight) of all or specific opioid pills (adjusted or unadjusted)

However, there are other common units of analysis that appear in the literature, such as morphine milligram equivalents (MMEs) or prescription counts, although these are not directly observable in the present data. It is the responsibility of the user to select an appropriate unit of analysis that has precedent in their field or take appropriate steps to standardize the data.

Additional resources

- The Opioid Files: a series of investigative articles by *The Washington Post*
- Drug Enforcement Agency presentation on the structure and purpose of ARCOS
- Example academic publication using earlier versions of ARCOS data