Transforming External Legacy Metadata for ArchivesSpace

A labor of love

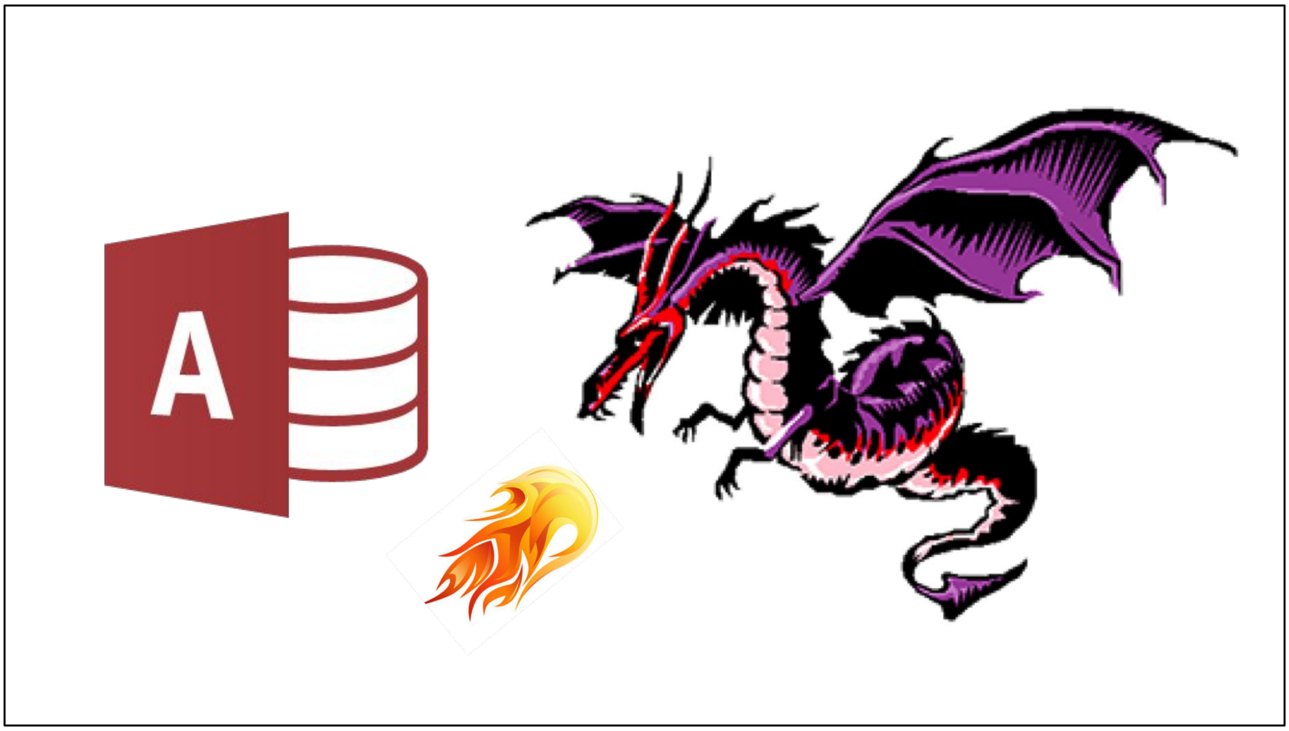
Bria Parker, Discovery and Metadata Services

When the AFL-CIO collection was acquired in 2013, University of Maryland Libraries received a database export of all the metadata associated with the materials. While we already had a proven process for transforming our local metadata from The Beast into ArchivesSpace, this process would not apply. In this presentation I will outline the challenges of transforming the AFL-CIO metadata and describe the transformation workflows. Specifically, I will show the tool (Stead) that was modified and used for the majority of the transformation work, as well as review the other steps, tips, and tools I utilized to format the metadata for optimal import into ArchivesSpace.

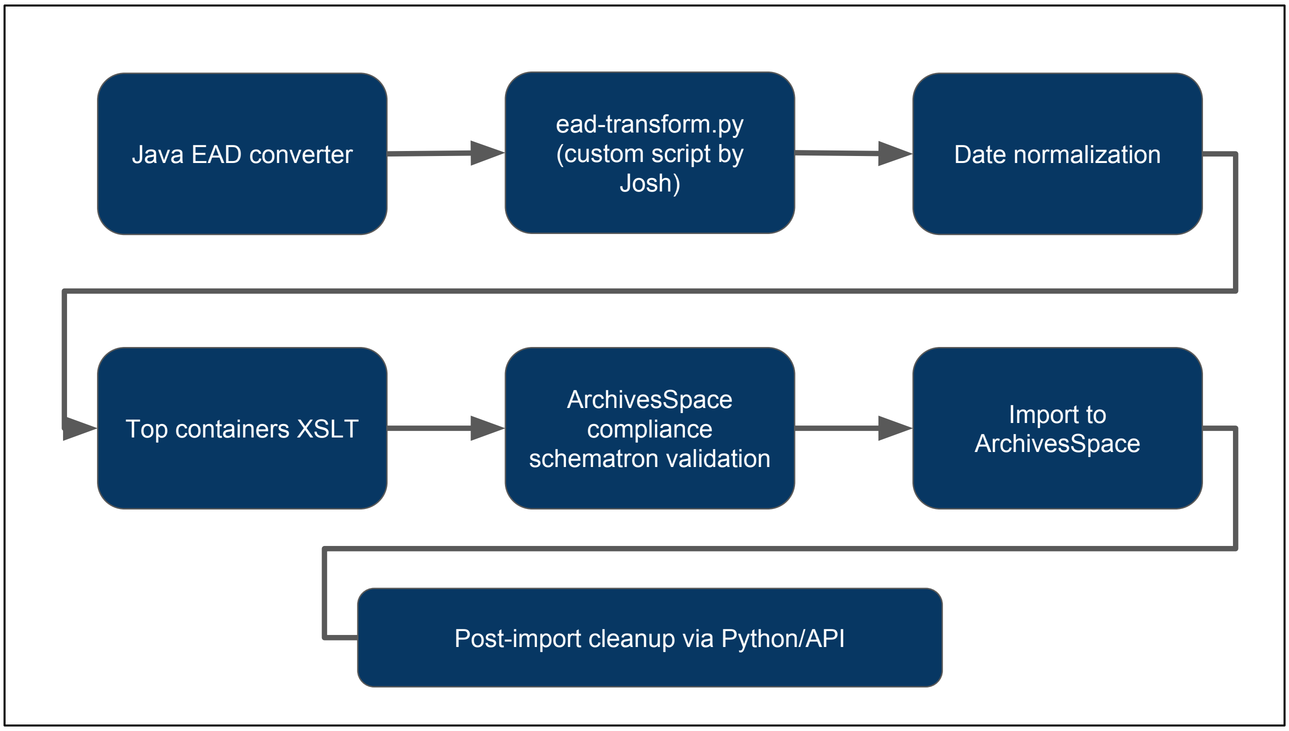
ArchivesSpace

a community served by LYRASIS

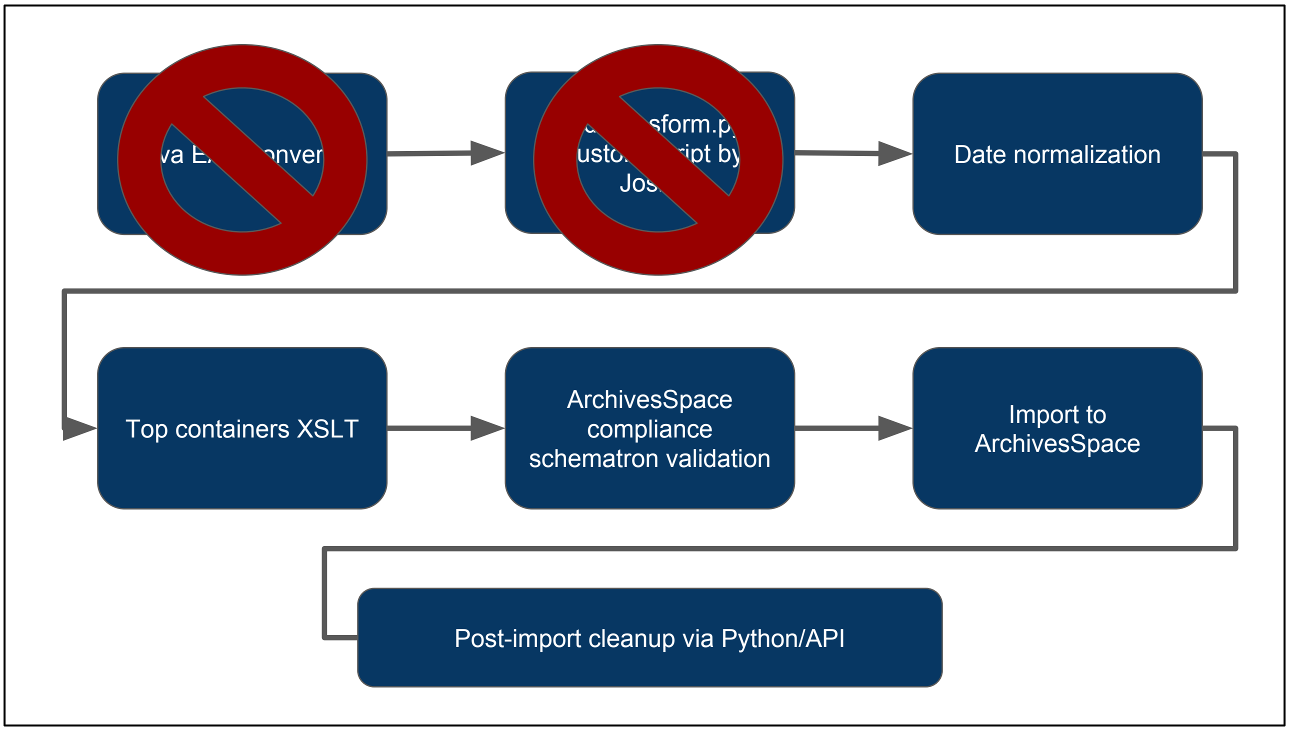
First, a little bit about ArchivesSpace: ArchivesSpace is an open source content management system for archival metadata and materials that manages the entire lifecycle of archival collections, similar to how an Integrated Library System (ILS), manages library resources.



SCUA and SCPA have spent the past few years migrating our archival metadata into ArchivesSpace from an Access database lovingly and aptly named The Beast. We developed a process for cleaning up accessions in spreadsheets to be loaded via an importer within the system. We developed a workflow and tools for getting finding aid metadata from The Beast into ArchivesSpace. But this presentation isn't about that. Some of those processes were not going to work for our next challenge, since we didn't even have EAD XML to start with for AFL-CIO. We had to get to that point before we could reuse any processes and tools.


Java EAD converter
ead-transform.py (custom script by Josh)
Date normalization
Top containers XSLT
ArchivesSpace compliance schematron validation
Import to ArchivesSpace
Post-import cleanup via Python/API

Here's a quick view of the workflow we used for the finding aid metadata in the Beast - first there wasJava converter that got the metadata from the Beast into EAD XML (it's a black box, though, and no one remains that really knows what's up with it. Then Josh Westgard created a python script to do some batch processing on the EAD XML to correct some common problems that were going to cause import errors. Next, a date normalization process, followed by a stylesheet to add top container barcodes, and a schematron to do error checking. Then it was on to import, and some post-import processing.

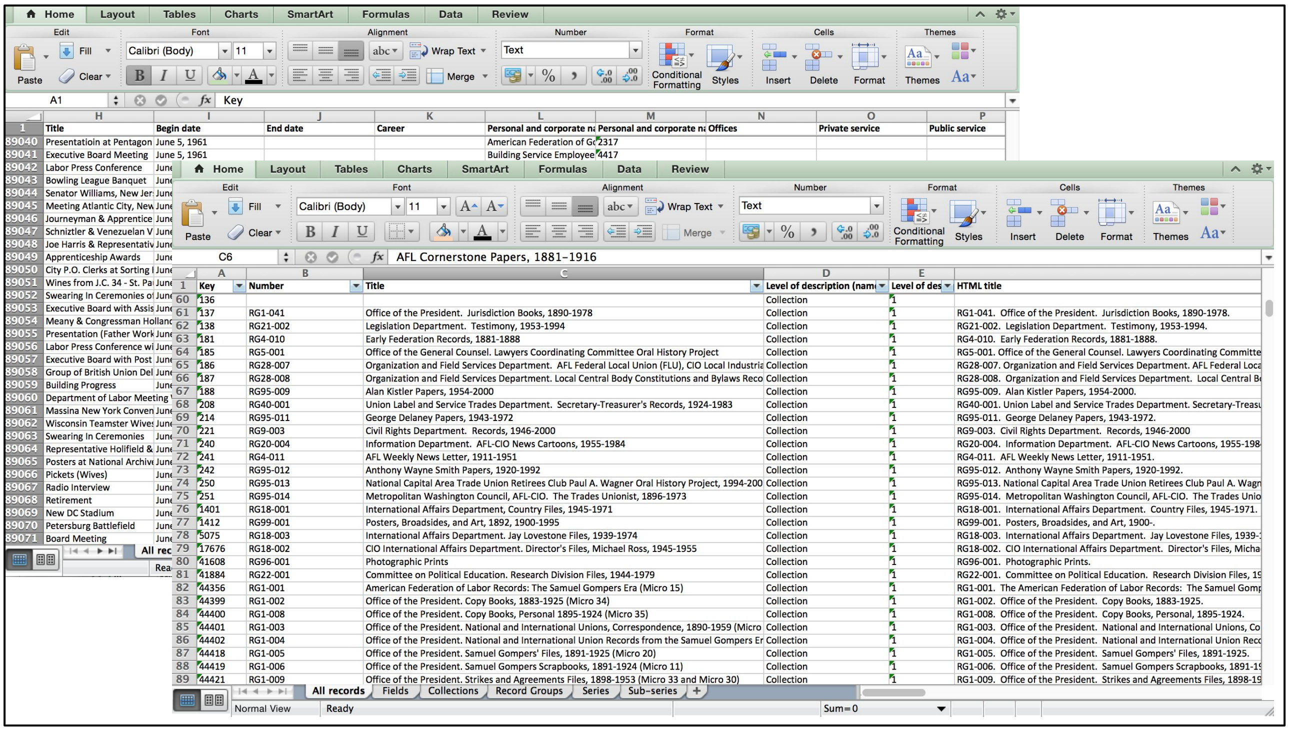


But, for the AFL-CIO metadata, those first two steps were not applicable.

Woman at International Ladies Garment Workers Union march - http://hdl.handle.net/1903.1/32553

As I mentioned in the beginning, when we acquired the AFL-CIO collection, we received a database export of all the metadata associated with the materials. This metadata had been stored in an Eloquent database, Eloquent being the name of an actual database system, not necessarily the adjective that would be used to describe the database…

And while our finding aid data was stored in a database, it was not in the same format, and could not use the same conversion tools to get it into ArchivesSpace, so we needed to find a different approach.



A1 Key

	Title	Begin date	End date	Career	Personal and corporate na	Personal and corporate na	Offices	Private service	Public service
89040	Presentatioin at Pentagon	June 5, 1961			American Federation of Go	2317			
89041	Executive Board Meeting	June 5, 1961			Building Service Employee	4417			
89042	Labor Press Conference	June							
89043	Bowling League Banquet	June							
89044	Senator Williams, New Jer	June							
89045	Meeting Atlantic City, New	June							
89046	Journeyman & Apprentice	June							
89047	Schniztler & Venezuelan V	June							
89048	Joe Harris & Representativ	June							
89049	Apprenticeship Awards	June							
89050	City P.O. Clerks at Sorting I	June							
89051	Wines from J.C. 34 - St. Pa	June							
89052	Swearing In Ceremonies of	June							
89053	Executive Board with Assis	June							
89054	Meany & Congressman Hollanc								
89055	Presentation (Father Work	June							
89056	Labor Press Conference wi	June							
89057	Executive Board with Post	June							
89058	Group of British Union Del	June							
89059	Building Progress	June							
89060	Department of Labor Meeting								
89061	Massina New York Conven	June							
89062	Wisconsin Teamster Wives	June							
89063	Swearing In Ceremonies	June							
89064	Representative Holifield &	June							
89065	Posters at National Archive	June							
89066	Pickets (Wives)	June							
89067	Radio Interview	June							
89068	Retirement	June							
89069	New DC Stadium	June							
89070	Petersburg Battlefield	June							
89071	Board Meeting	June							

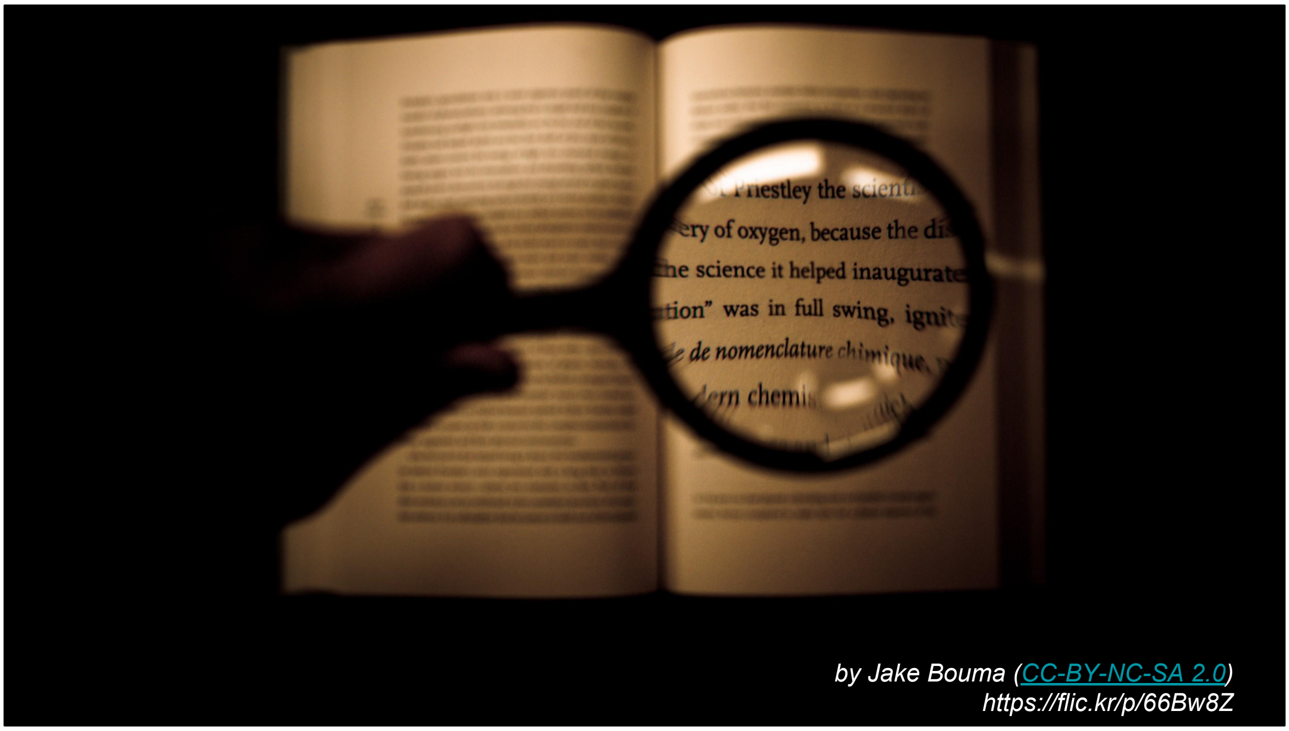
C6 AFL Cornerstone Papers, 1881–1916

	Key	Number	Title	Level of description (nam	Level of des	HTML title
60	136			Collection	1	
61	137	RG1-041	Office of the President. Jurisdiction Books, 1890-1978	Collection	1	RG1-041. Office of the President. Jurisdiction Books, 1890-1978.
62	138	RG21-002	Legislation Department. Testimony, 1953-1994	Collection	1	RG21-002. Legislation Department. Testimony, 1953-1994.
63	181	RG4-010	Early Federation Records, 1881-1888	Collection	1	RG4-010. Early Federation Records, 1881-1888.
64	185	RG5-001	Office of the General Counsel. Lawyers Coordinating Committee Oral History Project	Collection	1	RG5-001. Office of the General Counsel. Lawyers Coordinating Committe
65	186	RG28-007	Organization and Field Services Department. AFL Federal Local Union (FLU), CIO Local Industria	Collection	1	RG28-007. Organization and Field Services Department. AFL Federal Loca
66	187	RG28-008	Organization and Field Services Department. Local Central Body Constitutions and Bylaws Reco	Collection	1	RG28-008. Organization and Field Services Department. Local Central Bo
67	188	RG95-009	Alan Kistler Papers, 1954-2000	Collection	1	RG95-009. Alan Kistler Papers, 1954-2000.
68	208	RG40-001	Union Label and Service Trades Department. Secretary-Treasurer's Records, 1924-1983	Collection	1	RG40-001. Union Label and Service Trades Department. Secretary-Treasu
69	214	RG95-011	George Delaney Papers, 1943-1972	Collection	1	RG95-011. George Delaney Papers, 1943-1972.
70	221	RG9-003	Civil Rights Department. Records, 1946-2000	Collection	1	RG9-003. Civil Rights Department. Records, 1946-2000
71	240	RG20-004	Information Department. AFL-CIO News Cartoons, 1955-1984	Collection	1	RG20-004. Information Department. AFL-CIO News Cartoons, 1955-198
72	241	RG4-011	AFL Weekly News Letter, 1911-1951	Collection	1	RG4-011. AFL Weekly News Letter, 1911-1951.
73	242	RG95-012	Anthony Wayne Smith Papers, 1920-1992	Collection	1	RG95-012. Anthony Wayne Smith Papers, 1920-1992.
74	250	RG95-013	National Capital Area Trade Union Retirees Club Paul A. Wagner Oral History Project, 1994-200	Collection	1	RG95-013. National Capital Area Trade Union Retirees Club Paul A. Wagn
75	251	RG95-014	Metropolitan Washington Council, AFL-CIO. The Trades Unionist, 1896-1973	Collection	1	RG95-014. Metropolitan Washington Council, AFL-CIO. The Trades Unio
76	1401	RG18-001	International Affairs Department, Country Files, 1945-1971	Collection	1	RG18-001. International Affairs Department. Country Files, 1945-1971.
77	1412	RG99-001	Posters, Broadsides, and Art, 1892, 1900-1995	Collection	1	RG99-001. Posters, Broadsides, and Art, 1900-.
78	5075	RG18-003	International Affairs Department. Jay Lovestone Files, 1939-1974	Collection	1	RG18-003. International Affairs Department. Jay Lovestone Files, 1939-1
79	17676	RG18-002	CIO International Affairs Department. Director's Files, Michael Ross, 1945-1955	Collection	1	RG18-002. CIO International Affairs Department. Director's Files, Micha
80	41608	RG96-001	Photographic Prints	Collection	1	RG96-001. Photographic Prints.
81	41884	RG22-001	Committee on Political Education. Research Division Files, 1944-1979	Collection	1	RG22-001. Committee on Political Education. Research Division Files, 19
82	44356	RG1-001	American Federation of Labor Records: The Samuel Gompers Era (Micro 15)	Collection	1	RG1-001. The American Federation of Labor Records: The Samuel Gomp
83	44399	RG1-002	Office of the President. Copy Books, 1883-1925 (Micro 34)	Collection	1	RG1-002. Office of the President. Copy Books, 1883-1925.
84	44400	RG1-008	Office of the President. Copy Books, Personal 1895-1924 (Micro 35)	Collection	1	RG1-008. Office of the President. Copy Books, Personal, 1895-1924.
85	44401	RG1-003	Office of the President. National and International Unions, Correspondence, 1890-1959 (Micro	Collection	1	RG1-003. Office of the President. National and International Unions, Co
86	44402	RG1-004	Office of the President. National and International Union Records from the Samuel Gompers Er	Collection	1	RG1-004. Office of the President. National and International Union Reco
87	44418	RG1-005	Office of the President. Samuel Gompers' Files, 1891-1925 (Micro 20)	Collection	1	RG1-005. Office of the President. Samuel Gompers' Files, 1891-1925.
88	44419	RG1-006	Office of the President. Samuel Gompers Scrapbooks, 1891-1924 (Micro 11)	Collection	1	RG1-006. Office of the President. Samuel Gompers Scrapbooks, 1891-19
89	44421	RG1-009	Office of the President. Strikes and Agreements Files, 1898-1953 (Micro 33 and Micro 30)	Collection	1	RG1-009. Office of the President. Strikes and Agreements Files, 1898-19

All records | Fields | Collections | Record Groups | Series | Sub-series

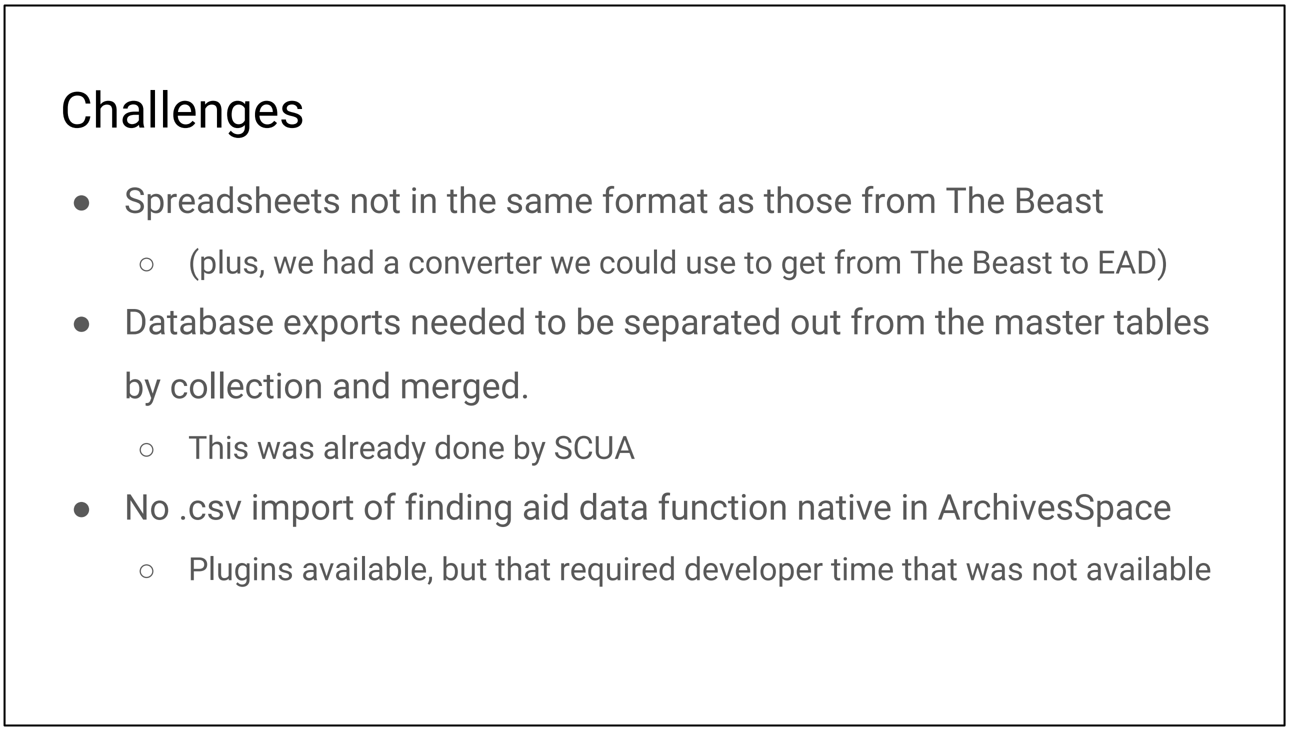
Normal View | Ready | Sum=0

So here's a snippet view of what received: spreadsheet dumps of the data - one spreadsheet for each table in the database. We had one big spreadsheet of the collection, series, and subseries metadata, another with box, folder, and item level metadata, as well as others for subjects and names.


Priestley the scienti
ery of oxygen, because the di
the science it helped inaugurate
tion" was in full swing, ignit
de nomenclature chimique,
ern chemis

by Jake Bouma (CC-BY-NC-SA 2.0)
https://flic.kr/p/66Bw8Z

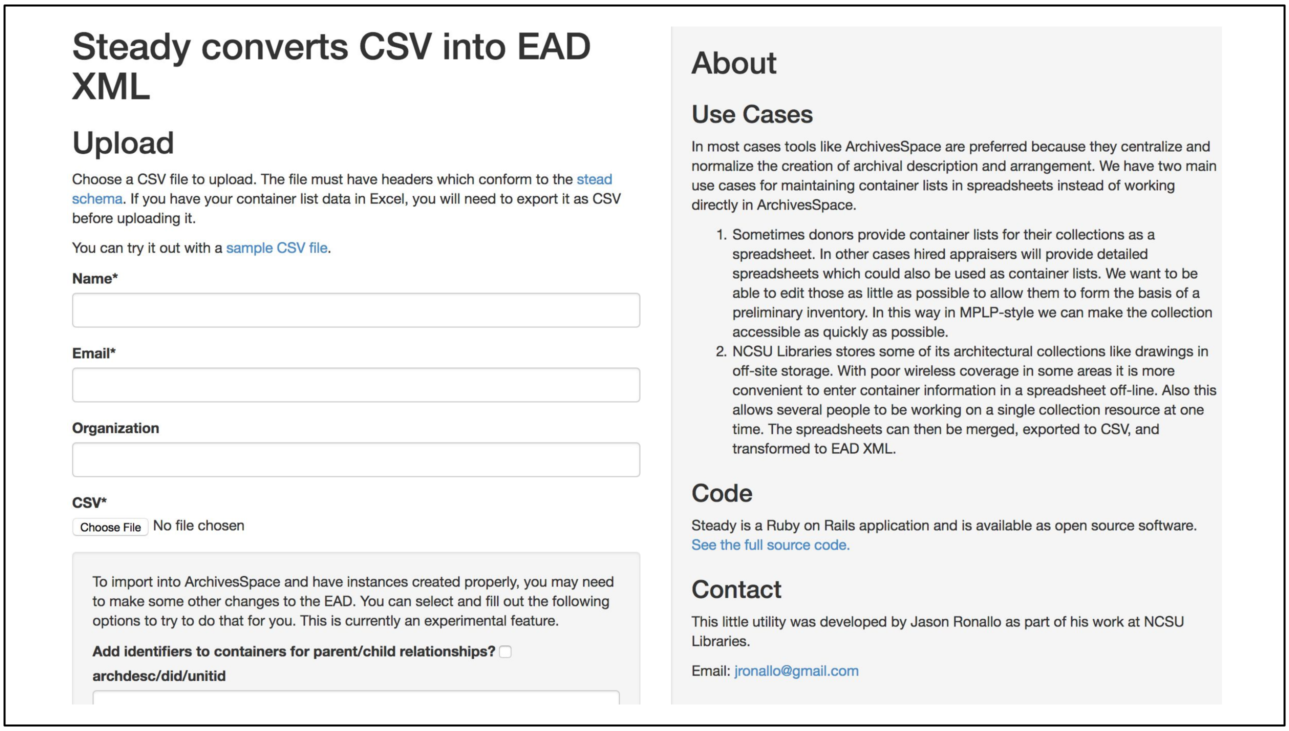
We began exploring our options - how would we get this in? AFL-CIO is a big, important collection, and we had very little discoverability for it. There was a collection guide online that gave brief descriptions of the collections, but we really needed to make these more broadly discoverable for researchers.



Challenges

- Spreadsheets not in the same format as those from The Beast
  - (plus, we had a converter we could use to get from The Beast to EAD)
- Database exports needed to be separated out from the master tables by collection and merged.
  - This was already done by SCUA
- No .csv import of finding aid data function native in ArchivesSpace
  - Plugins available, but that required developer time that was not available

Challenges:
-We had no way to get from spreadsheet to EAD XML.
-The spreadsheets we did have - one with collection/series/subseries level metadata, one with box/folder/item level metadata, These needed to be separated out by collection and merged, no matter what our process was. (Thankfully Jen and her GA had already done this0
-also had biographical tables - which were left out of this process bc it got too complicated. - some of this can be done in bulk afterwards using the API.
-no csv import of resource/finding aid data functionality available in ArchivesSpace
-plugins existed, but did not account for as much detail as we had and thus would require lots of development work, and there were other development priorities across the Libraries.



Steady converts CSV into EAD XML

Upload

Choose a CSV file to upload. The file must have headers which conform to the stead schema. If you have your container list data in Excel, you will need to export it as CSV before uploading it.

You can try it out with a sample CSV file.

Name*

Email*

Organization

CSV*

Choose File No file chosen

To import into ArchivesSpace and have instances created properly, you may need to make some other changes to the EAD. You can select and fill out the following options to try to do that for you. This is currently an experimental feature.

Add identifiers to containers for parent/child relationships? ☐

archdesc/did/unitid

About

Use Cases

In most cases tools like ArchivesSpace are preferred because they centralize and normalize the creation of archival description and arrangement. We have two main use cases for maintaining container lists in spreadsheets instead of working directly in ArchivesSpace.

1. Sometimes donors provide container lists for their collections as a spreadsheet. In other cases hired appraisers will provide detailed spreadsheets which could also be used as container lists. We want to be able to edit those as little as possible to allow them to form the basis of a preliminary inventory. In this way in MPLP-style we can make the collection accessible as quickly as possible.
2. NCSU Libraries stores some of its architectural collections like drawings in off-site storage. With poor wireless coverage in some areas it is more convenient to enter container information in a spreadsheet off-line. Also this allows several people to be working on a single collection resource at one time. The spreadsheets can then be merged, exported to CSV, and transformed to EAD XML.

Code

Steady is a Ruby on Rails application and is available as open source software. See the full source code.

Contact

This little utility was developed by Jason Ronallo as part of his work at NCSU Libraries.

Email: jronallo@gmail.com

We had previously heard about and played around with a tool called Steady, created by Jason Ronallo at NCSU. This tool would take a csv and convert it to EAD XML. Which we *could* import into ArchivesSpace. But! Again, it didn't capture as much detail as we liked (if we had the metadata, I wanted to use it, rather than have a lot of manual work post import!), PLUS, you had to convert one csv at a time.

by owlpacino (CC-BY-ND 2.0)
https://flic.kr/p/6vwycW

I liked Steady a lot, though, so I started digging. I wondered if I could alter Steady to accept and map more columns. THEN, in looking at the github repository for it, I saw that Steady was actually a wrapper that worked with a Ruby gem he had created called Stead (which stood for Spreadsheets to Encoded Archival Description). Stead is what was doing the heavy lifting of the converting. Steady was the mechanism for passing files back and forth between the gem and the user.

So, I began looking into Steead more. since it required a pretty specific .csv template, I had my GA at the time, Sydney Vaile, work on mapping collection inventories that Jen Eidsen's GA had created from the Eloquent data into the Stead csv template

Additional arguments

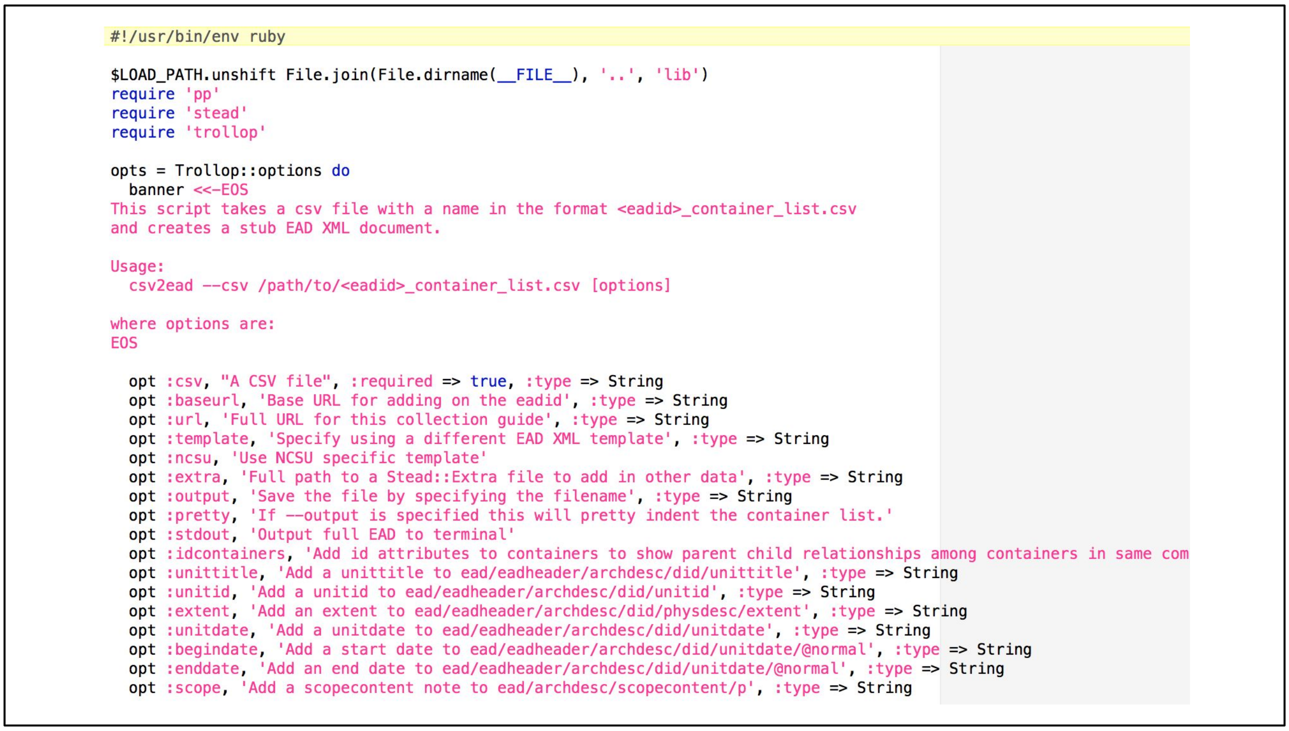
- Collection level
    - Resource id
    - Extent
    - Scope
    - Normalized dates

In the meantime I needed to roll up my sleeves and get comfy with Ruby.

I copied the github repository to my machine and began exploring what changes could be made.

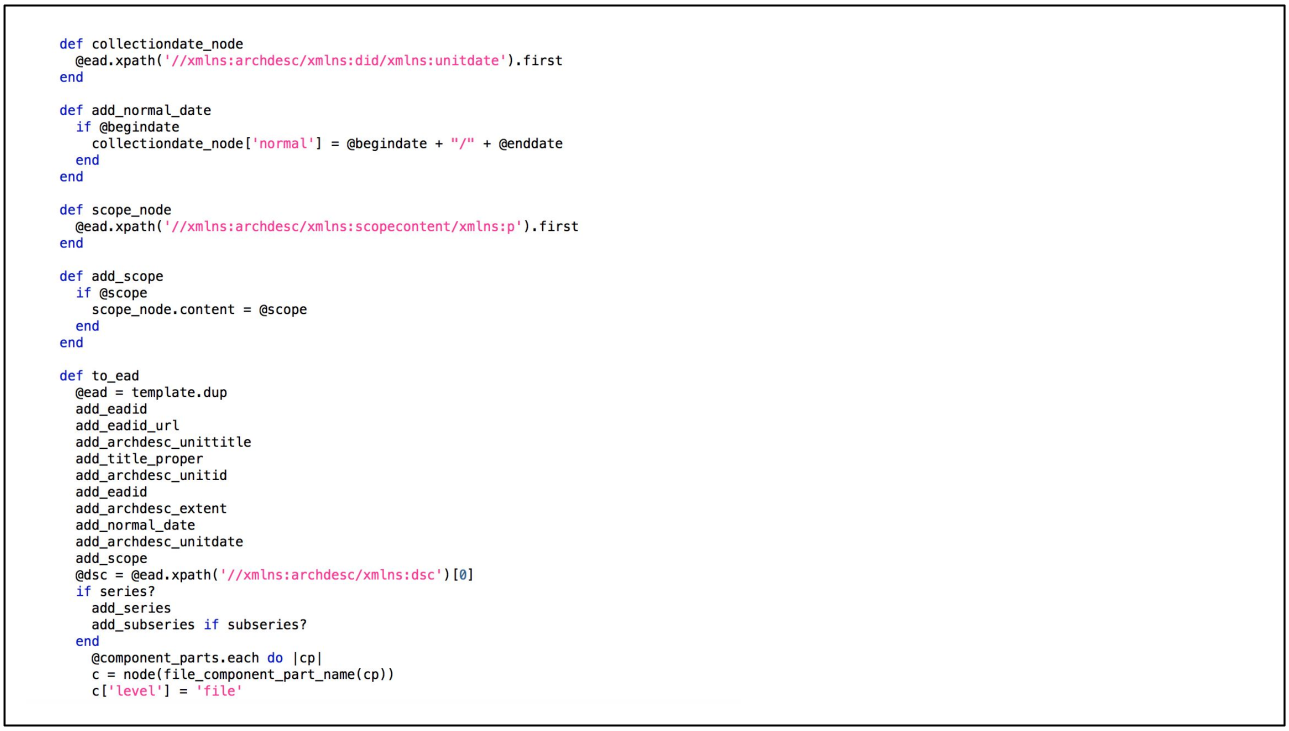
The first thing I did was add additional arguments to set collection-level metadata (and by arguments I'm referring to the actual input provided in the command line when running the program.

I added arguments for including resource id, extent, scope and content notes metadata at the collection level, while also parsing out dates for normalization.



```
#!/usr/bin/env ruby

$LOAD_PATH.unshift File.join(File.dirname(__FILE__), '..', 'lib')
require 'pp'
require 'stead'
require 'trollop'

opts = Trollop::options do
  banner <<-EOS
This script takes a csv file with a name in the format <eadid>_container_list.csv
and creates a stub EAD XML document.

Usage:
  csv2ead --csv /path/to/<eadid>_container_list.csv [options]

where options are:
EOS

  opt :csv, "A CSV file", :required => true, :type => String
  opt :baseurl, 'Base URL for adding on the eadid', :type => String
  opt :url, 'Full URL for this collection guide', :type => String
  opt :template, 'Specify using a different EAD XML template', :type => String
  opt :ncsu, 'Use NCSU specific template'
  opt :extra, 'Full path to a Stead::Extra file to add in other data', :type => String
  opt :output, 'Save the file by specifying the filename', :type => String
  opt :pretty, 'If --output is specified this will pretty indent the container list.'
  opt :stdout, 'Output full EAD to terminal'
  opt :idcontainers, 'Add id attributes to containers to show parent child relationships among containers in same com
  opt :unittitle, 'Add a unittitle to ead/eadheader/archdesc/did/unittitle', :type => String
  opt :unitid, 'Add a unitid to ead/eadheader/archdesc/did/unitid', :type => String
  opt :extent, 'Add an extent to ead/eadheader/archdesc/did/physdesc/extent', :type => String
  opt :unitdate, 'Add a unitdate to ead/eadheader/archdesc/did/unitdate', :type => String
  opt :begindate, 'Add a start date to ead/eadheader/archdesc/did/unitdate/@normal', :type => String
  opt :enddate, 'Add an end date to ead/eadheader/archdesc/did/unitdate/@normal', :type => String
  opt :scope, 'Add a scopecontent note to ead/archdesc/scopecontent/p', :type => String
```

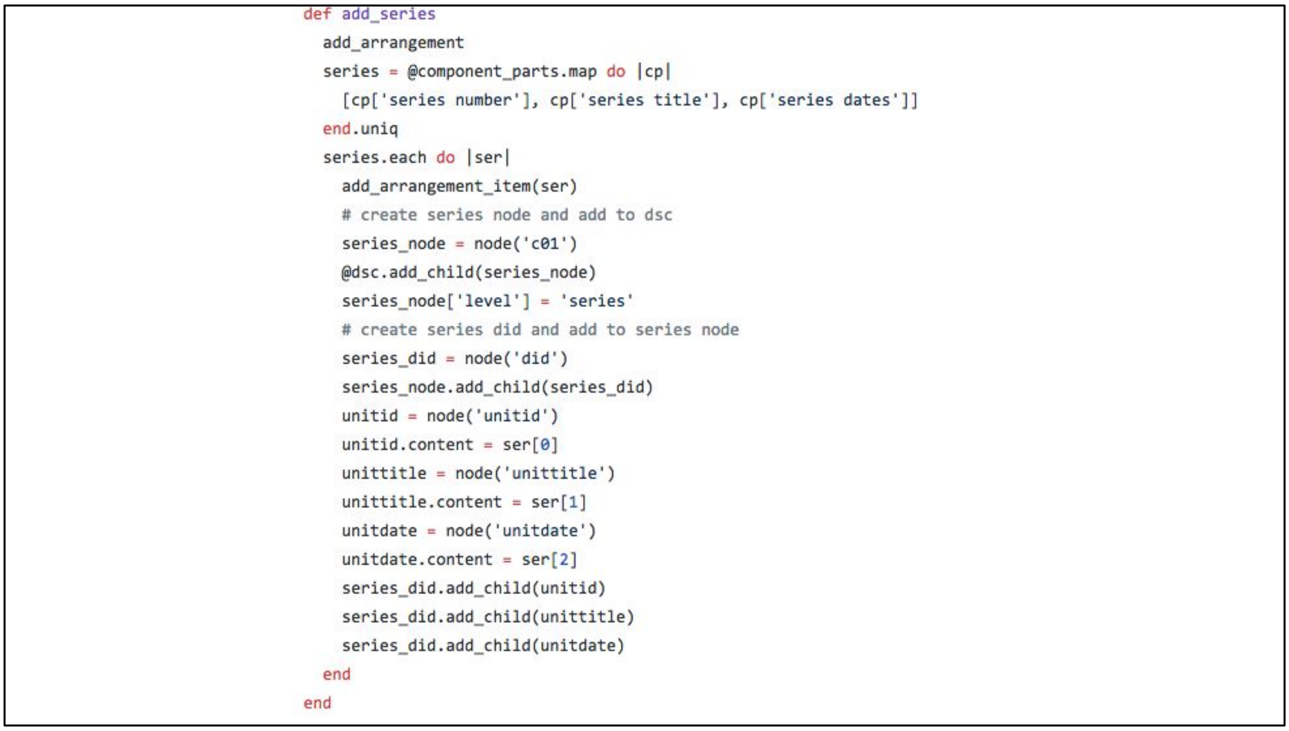
Here is a snippet of the file that controls the input arguments...



```
def collectiondate_node
  @ead.xpath('//xmlns:archdesc/xmlns:did/xmlns:unitdate').first
end

def add_normal_date
  if @begindate
    collectiondate_node['normal'] = @begindate + "/" + @enddate
  end
end

def scope_node
  @ead.xpath('//xmlns:archdesc/xmlns:scopecontent/xmlns:p').first
end

def add_scope
  if @scope
    scope_node.content = @scope
  end
end

def to_ead
  @ead = template.dup
  add_eadid
  add_eadid_url
  add_archdesc_unittitle
  add_title_proper
  add_archdesc_unitid
  add_eadid
  add_archdesc_extent
  add_normal_date
  add_archdesc_unitdate
  add_scope
  @dsc = @ead.xpath('//xmlns:archdesc/xmlns:dsc')[0]
  if series?
    add_series
    add_subseries if subseries?
  end
    @component_parts.each do |cp|
    c = node(file_component_part_name(cp))
    c['level'] = 'file'
```

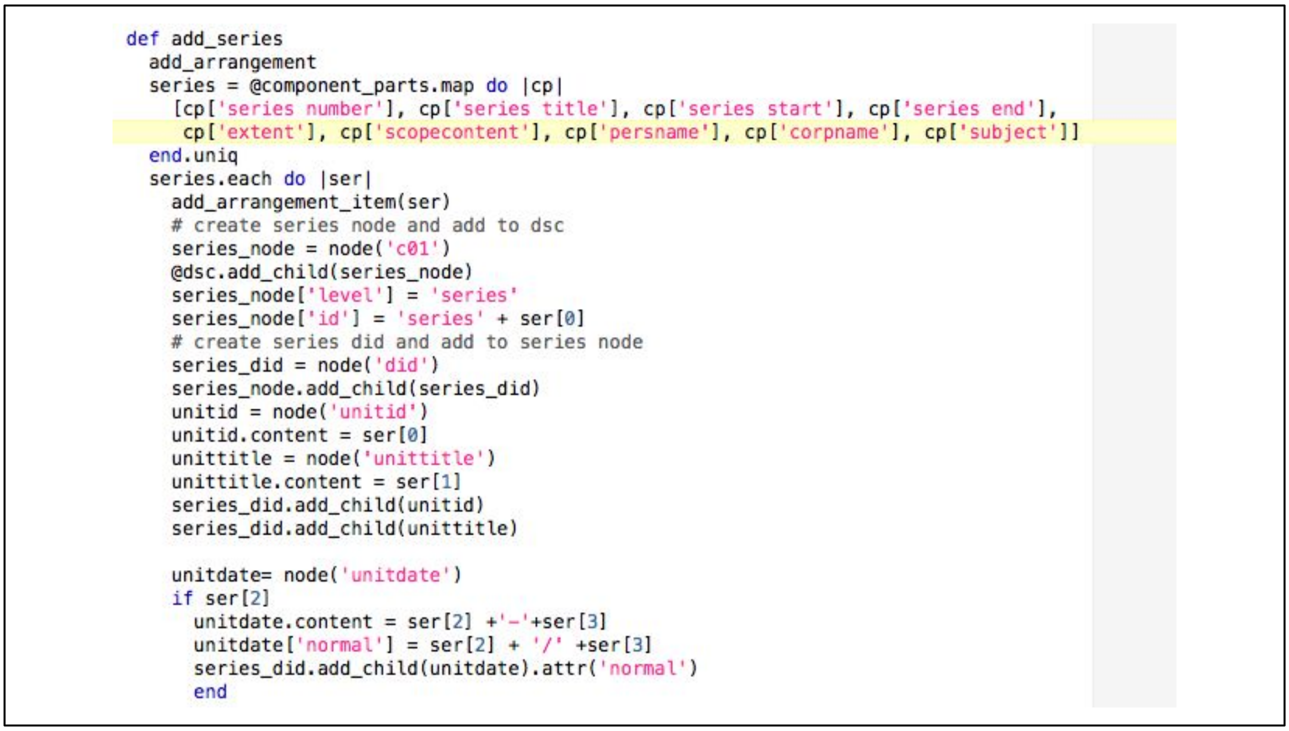
And here is a snippet showing these arguments (or input) getting mapped to an element within the EAD, using XPath.

(Xpath is a method for navigating XML elements and attributes - so it's like a map with directions to say "take this and follow the XML tree to *here* and put the data there).



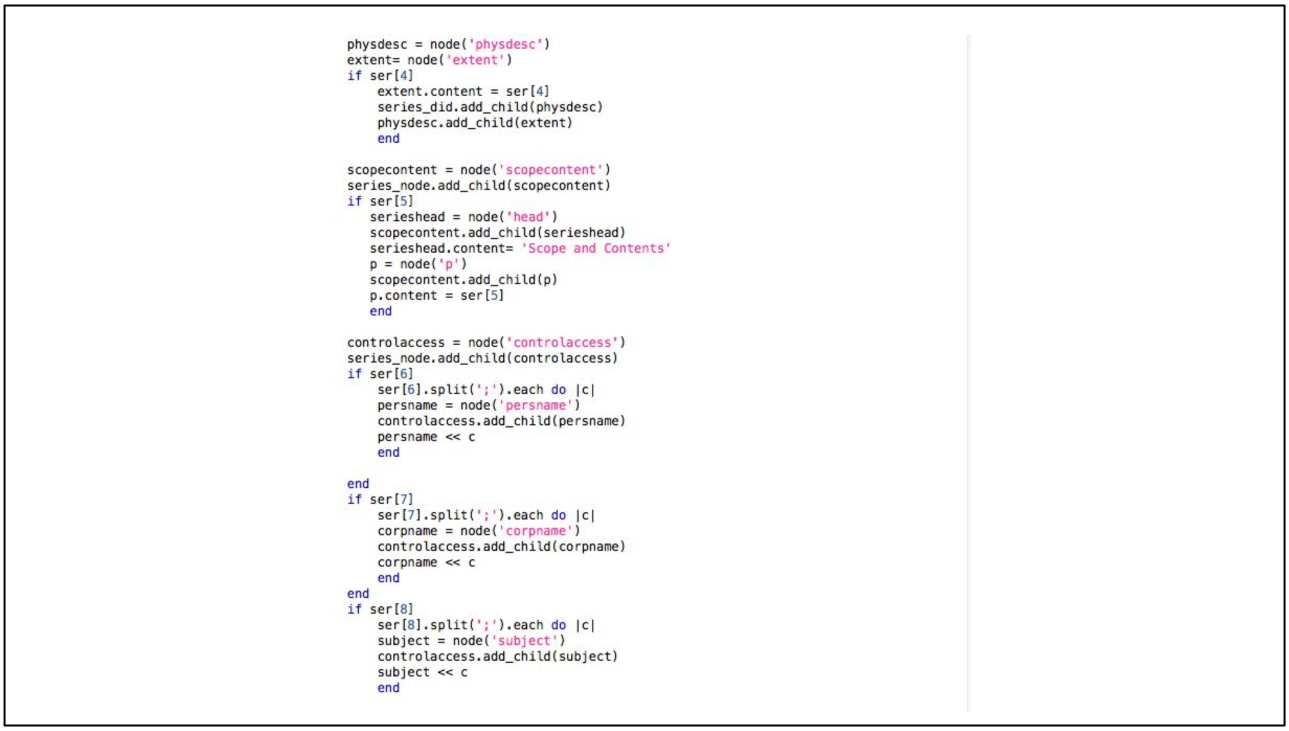
```
def add_series
  add_arrangement
  series = @component_parts.map do |cp|
    [cp['series number'], cp['series title'], cp['series dates']]
  end.uniq
  series.each do |ser|
    add_arrangement_item(ser)
    # create series node and add to dsc
    series_node = node('c01')
    @dsc.add_child(series_node)
    series_node['level'] = 'series'
    # create series did and add to series node
    series_did = node('did')
    series_node.add_child(series_did)
    unitid = node('unitid')
    unitid.content = ser[0]
    unittitle = node('unittitle')
    unittitle.content = ser[1]
    unitdate = node('unitdate')
    unitdate.content = ser[2]
    series_did.add_child(unitid)
    series_did.add_child(unittitle)
    series_did.add_child(unitdate)
  end
end
```

In addition to accommodating additional collection metadata at the command line, I was able to adjust the script to handle the additional metadata we had at the series level, such as scope and contents notes, series extent, related subjects and names, as well as date parsing for normalization, so the function for creating a series went from taking 25 lines of code and fitting on one slide….



```
def add_series
  add_arrangement
  series = @component_parts.map do |cp|
    [cp['series number'], cp['series title'], cp['series start'], cp['series end'],
     cp['extent'], cp['scopecontent'], cp['persname'], cp['corpname'], cp['subject']]
  end.uniq
  series.each do |ser|
    add_arrangement_item(ser)
    # create series node and add to dsc
    series_node = node('c01')
    @dsc.add_child(series_node)
    series_node['level'] = 'series'
    series_node['id'] = 'series' + ser[0]
    # create series did and add to series node
    series_did = node('did')
    series_node.add_child(series_did)
    unitid = node('unitid')
    unitid.content = ser[0]
    unittitle = node('unittitle')
    unittitle.content = ser[1]
    series_did.add_child(unitid)
    series_did.add_child(unittitle)

    unitdate= node('unitdate')
    if ser[2]
      unitdate.content = ser[2] +'-'+ser[3]
      unitdate['normal'] = ser[2] + '/' +ser[3]
      series_did.add_child(unitdate).attr('normal')
      end
```

To taking nearly 70 lines of code and barely fitting on two slides.

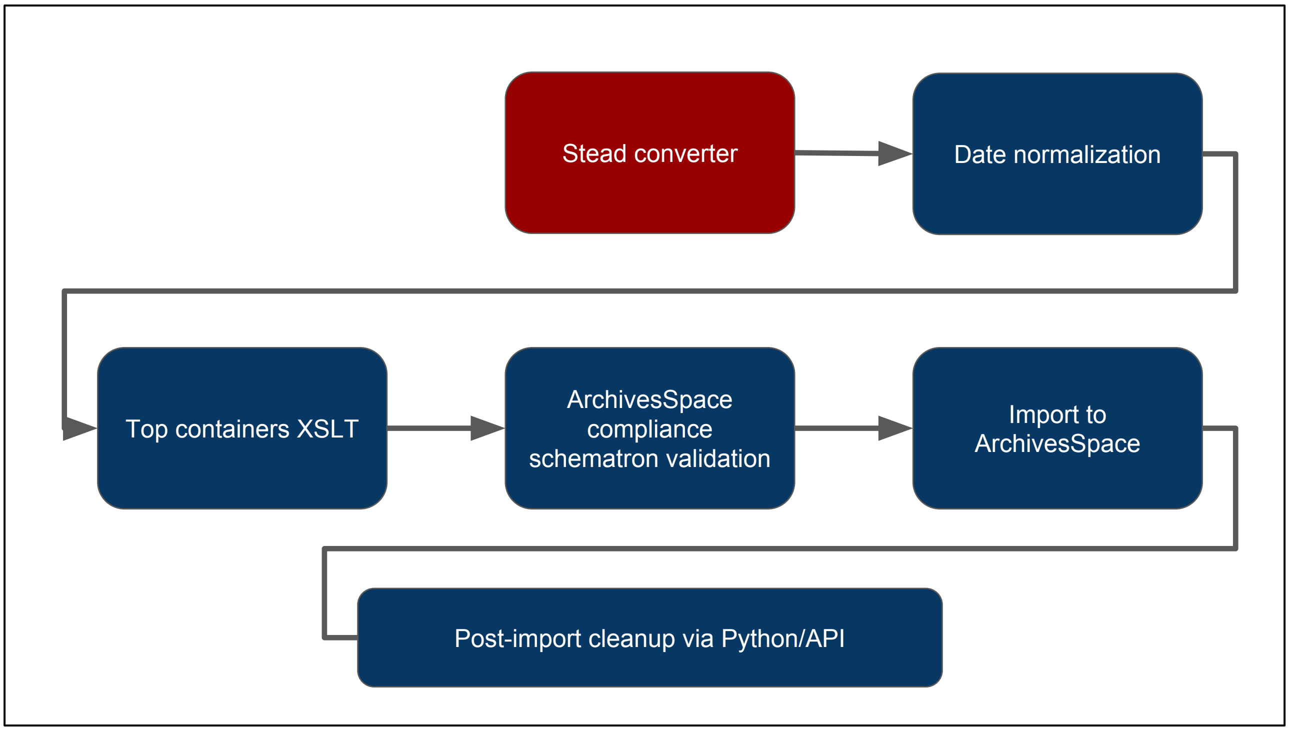


```
physdesc = node('physdesc')
extent= node('extent')
if ser[4]
    extent.content = ser[4]
    series_did.add_child(physdesc)
    physdesc.add_child(extent)
    end

scopecontent = node('scopecontent')
series_node.add_child(scopecontent)
if ser[5]
   serieshead = node('head')
   scopecontent.add_child(serieshead)
   serieshead.content= 'Scope and Contents'
   p = node('p')
   scopecontent.add_child(p)
   p.content = ser[5]
   end

controlaccess = node('controlaccess')
series_node.add_child(controlaccess)
if ser[6]
    ser[6].split(';').each do |c|
    persname = node('persname')
    controlaccess.add_child(persname)
    persname << c
    end

end
if ser[7]
    ser[7].split(';').each do |c|
    corpname = node('corpname')
    controlaccess.add_child(corpname)
    corpname << c
    end
end
if ser[8]
    ser[8].split(';').each do |c|
    subject = node('subject')
    controlaccess.add_child(subject)
    subject << c
    end
```

I made the script more precise and accommodated more metadata in the transformation, which meant less manual work for curators and GAs down the road. Everyone wins!

by gotcredit (CC-BY 2.0)
https://flic.kr/p/SDDTLb

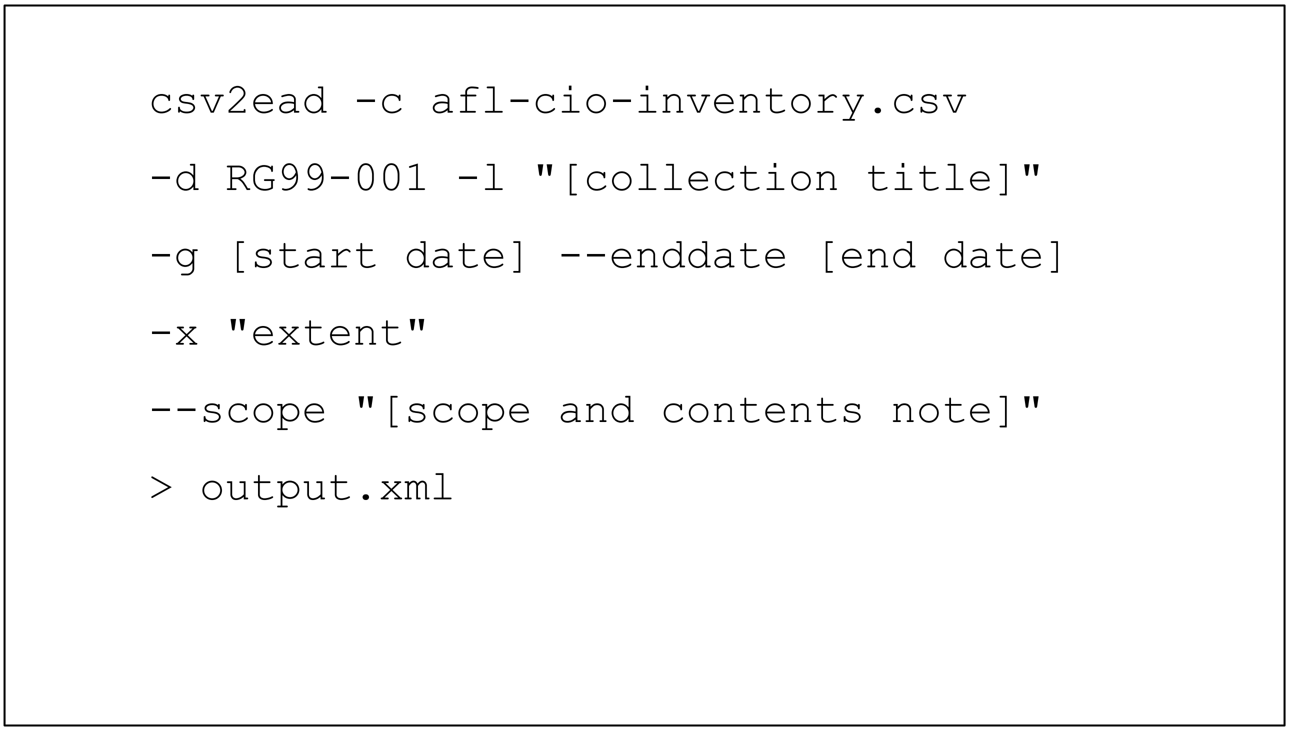
So, I had conquered series level metadata, and container lists for archival objects within those series, but was still fumbling with the subseries.
Enter Caitlin. In the summer of 2017 I had a field study student, Caitlin Rizzo who was interested in learning more about metadata transformation work, specifically through code. We developed a project for her to work with me to further modify the Stead code to accept and properly map subseries metadata. She was ultimately successful and was able to incorporate the additional subseries level metadata we had that was not in the original code - scope and contents notes, extents, subjects, people and organizations!! Now, we had a tool for transforming nearly all of the AFL-CIO metadata.


Stead converter
Date normalization
Top containers XSLT
ArchivesSpace compliance schematron validation
Import to ArchivesSpace
Post-import cleanup via Python/API

And I could replace the first two steps of the previous process with just the Stead converter. Now the transformation could begin!

Real talk

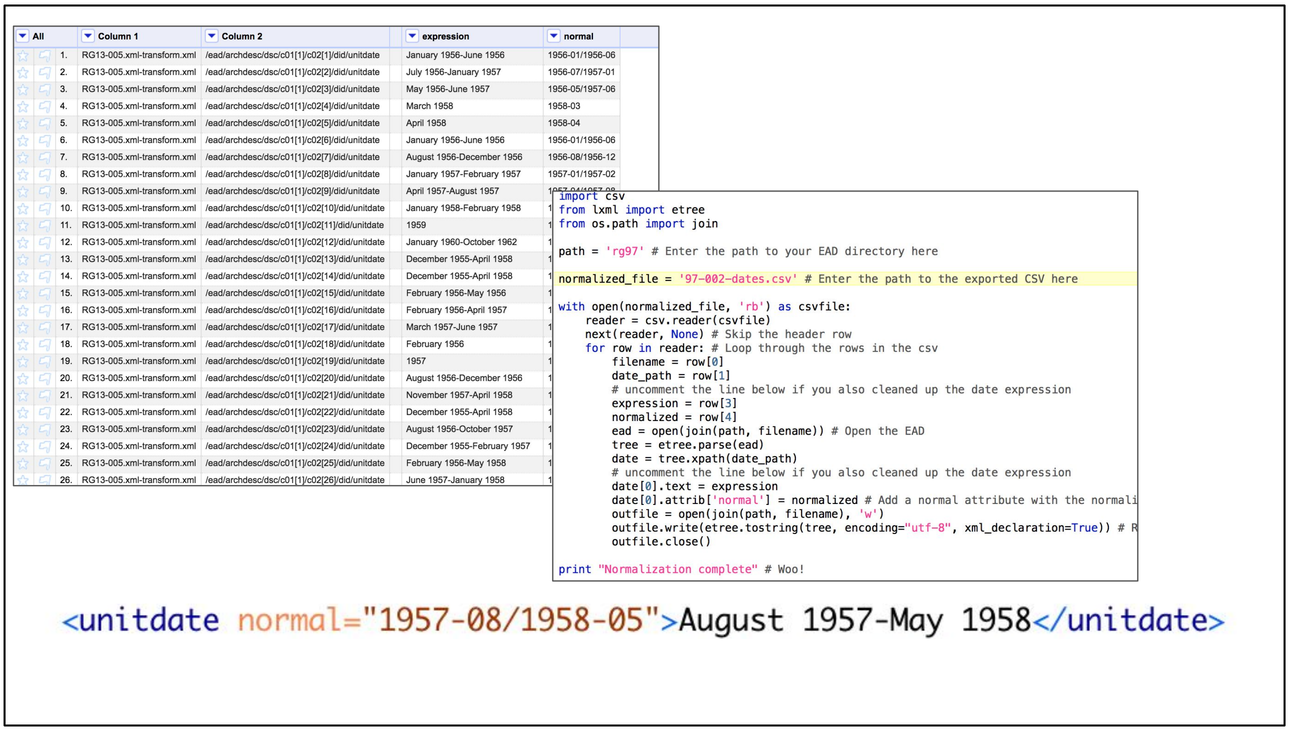
Now for some real talk. Not everything worked the way I wanted it to. I converted the bulk of the collection level metadata spreadsheet into a tab delimited text that incorporated the argument tags, with the thought of developing a bash scrip that I could run at the command line that would go line by line through the file and convert them en masse. I spent a lot of time trying to make that work. It did not. I don't know if it was the nature of the ruby gem, or limits to my ability (PROBABLY), but I abandoned that work in favor of it just getting done. So I just copied and pasted each line into the command line. I tell you this to illustrate that sometimes something just doesn't work. And that it's okay.



```
csv2ead -c afl-cio-inventory.csv
-d RG99-001 -l "[collection title]"
-g [start date] --enddate [end date]
-x "extent"
--scope "[scope and contents note]"
> output.xml
```

So, here is what it looked like in the command line when I'd run it
Csv2ead was the name of the function within the gem - the -c argument gives the filename to convert, ….etc.

When it finished processing, the output file was sitting in the same directory as the source csv.



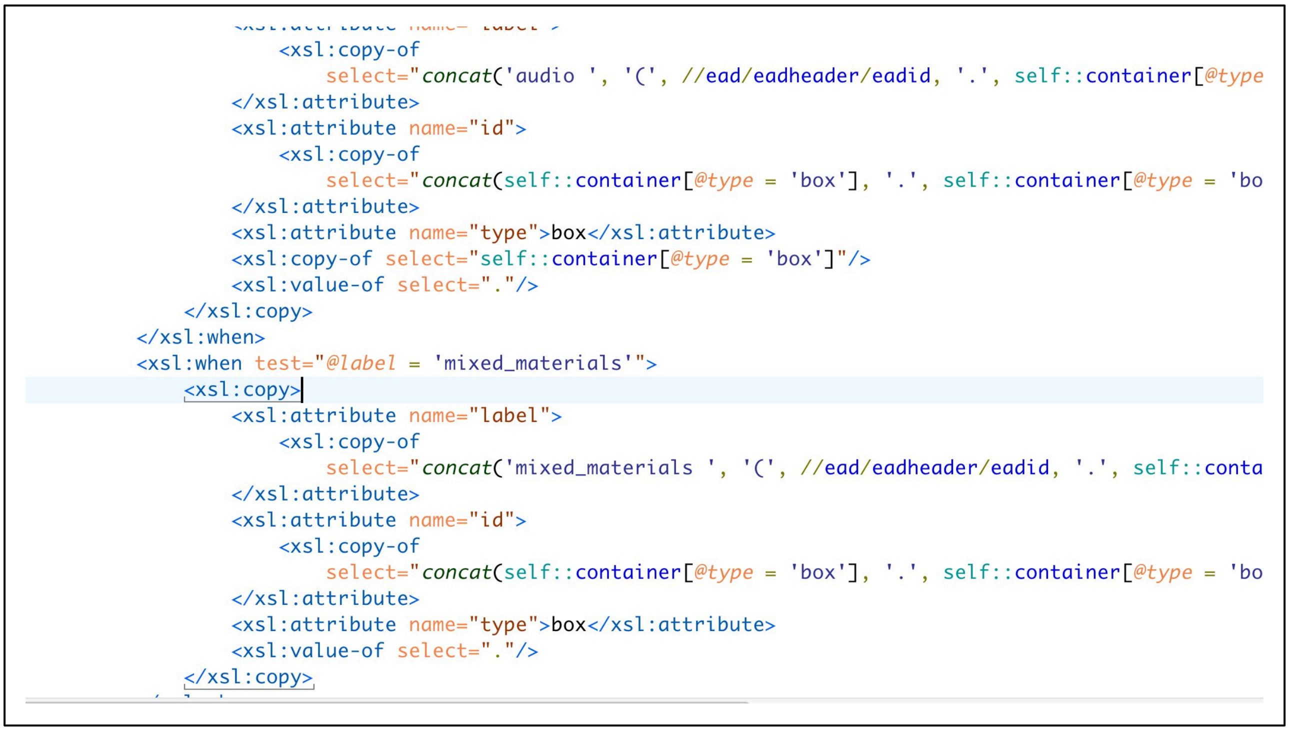
All			Column 1	Column 2	expression	normal
☆		1.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[1]/did/unitdate	January 1956-June 1956	1956-01/1956-06
☆		2.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[2]/did/unitdate	July 1956-January 1957	1956-07/1957-01
☆		3.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[3]/did/unitdate	May 1956-June 1957	1956-05/1957-06
☆		4.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[4]/did/unitdate	March 1958	1958-03
☆		5.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[5]/did/unitdate	April 1958	1958-04
☆		6.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[6]/did/unitdate	January 1956-June 1956	1956-01/1956-06
☆		7.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[7]/did/unitdate	August 1956-December 1956	1956-08/1956-12
☆		8.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[8]/did/unitdate	January 1957-February 1957	1957-01/1957-02
☆		9.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[9]/did/unitdate	April 1957-August 1957	1957-04/1957-08
☆		10.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[10]/did/unitdate	January 1958-February 1958	1
☆		11.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[11]/did/unitdate	1959	1
☆		12.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[12]/did/unitdate	January 1960-October 1962	1
☆		13.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[13]/did/unitdate	December 1955-April 1958	1
☆		14.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[14]/did/unitdate	December 1955-April 1958	1
☆		15.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[15]/did/unitdate	February 1956-May 1956	1
☆		16.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[16]/did/unitdate	February 1956-April 1957	1
☆		17.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[17]/did/unitdate	March 1957-June 1957	1
☆		18.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[18]/did/unitdate	February 1956	1
☆		19.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[19]/did/unitdate	1957	1
☆		20.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[20]/did/unitdate	August 1956-December 1956	1
☆		21.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[21]/did/unitdate	November 1957-April 1958	1
☆		22.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[22]/did/unitdate	December 1955-April 1958	1
☆		23.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[23]/did/unitdate	August 1956-October 1957	1
☆		24.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[24]/did/unitdate	December 1955-February 1957	1
☆		25.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[25]/did/unitdate	February 1956-May 1958	1
☆		26.	RG13-005.xml-transform.xml	/ead/archdesc/dsc/c01[1]/c02[26]/did/unitdate	June 1957-January 1958	1

```python
import csv
from lxml import etree
from os.path import join

path = 'rg97' # Enter the path to your EAD directory here

normalized_file = '97-002-dates.csv' # Enter the path to the exported CSV here

with open(normalized_file, 'rb') as csvfile:
    reader = csv.reader(csvfile)
    next(reader, None) # Skip the header row
    for row in reader: # Loop through the rows in the csv
        filename = row[0]
        date_path = row[1]
        # uncomment the line below if you also cleaned up the date expression
        expression = row[3]
        normalized = row[4]
        ead = open(join(path, filename)) # Open the EAD
        tree = etree.parse(ead)
        date = tree.xpath(date_path)
        # uncomment the line below if you also cleaned up the date expression
        date[0].text = expression
        date[0].attrib['normal'] = normalized # Add a normal attribute with the normali
        outfile = open(join(path, filename), 'w')
        outfile.write(etree.tostring(tree, encoding="utf-8", xml_declaration=True)) # R
        outfile.close()

print "Normalization complete" # Woo!
```

```xml
<unitdate normal="1957-08/1958-05">August 1957-May 1958</unitdate>
```

So, now I had EAD XML, but it still wasn't quite ready to be imported to ArchivesSpace. At this point, I could actually re-use and repurpose processes and tools used for The Beast data. I re-used the process for normalizing dates (thanks Bentley Library for documenting their process and sharing their code!). Why are normalized dates important? Normalization makes the dates machine readable, which leads to better date searching/faceting for our users

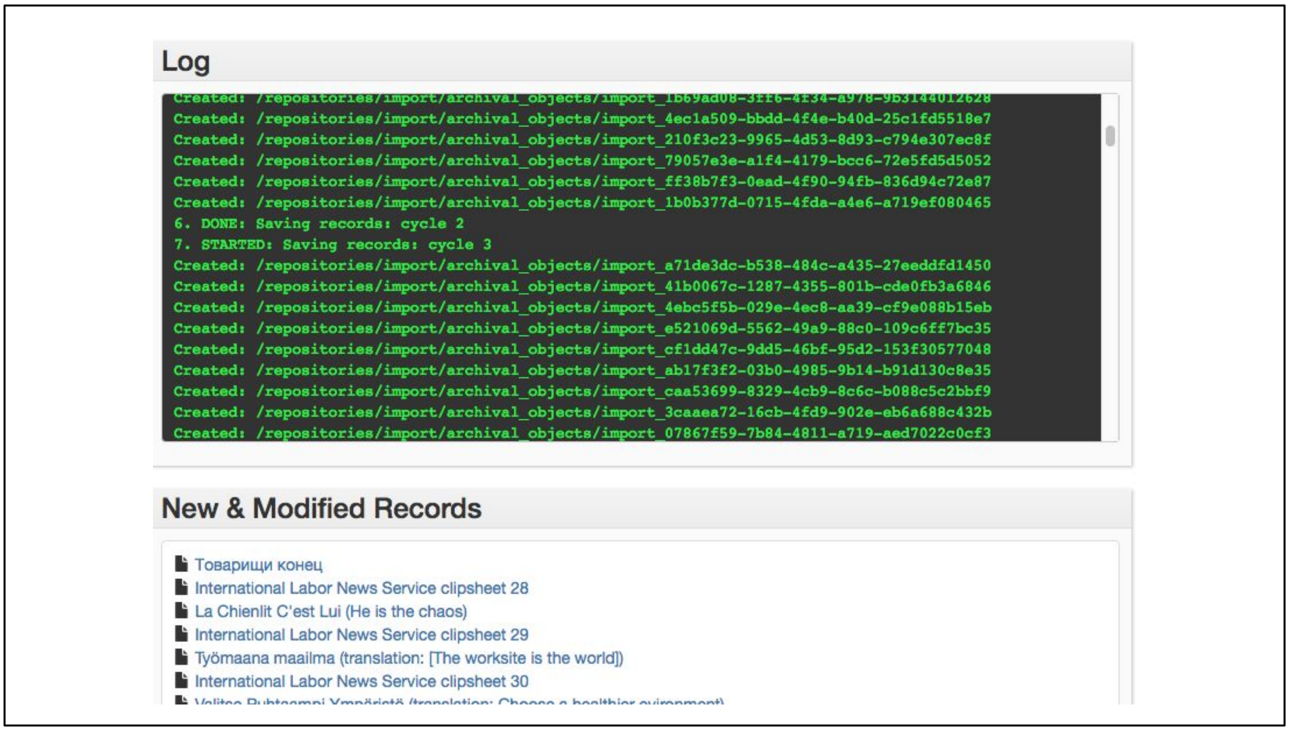
This date normalization process involved a python script to fix simple date format cases, then a python script to extract dates the first script couldn't fix, then open refine work to parse and normalize the dates, and a python script to load those normalized dates back into the XML.



```
                    <xsl:copy-of
                        select="concat('audio ', '(', //ead/eadheader/eadid, '.', self::container[@type
                </xsl:attribute>
                <xsl:attribute name="id">
                    <xsl:copy-of
                        select="concat(self::container[@type = 'box'], '.', self::container[@type = 'bo
                </xsl:attribute>
                <xsl:attribute name="type">box</xsl:attribute>
                <xsl:copy-of select="self::container[@type = 'box']"/>
                <xsl:value-of select="."/>
            </xsl:copy>
        </xsl:when>
        <xsl:when test="@label = 'mixed_materials'">
            <xsl:copy>
                <xsl:attribute name="label">
                    <xsl:copy-of
                        select="concat('mixed_materials ', '(', //ead/eadheader/eadid, '.', self::conta
                </xsl:attribute>
                <xsl:attribute name="id">
                    <xsl:copy-of
                        select="concat(self::container[@type = 'box'], '.', self::container[@type = 'bo
                </xsl:attribute>
                <xsl:attribute name="type">box</xsl:attribute>
                <xsl:value-of select="."/>
            </xsl:copy>
```

I was also able to tweak XSLT I had developed for creating ArchivesSpace-compliant top containers out of the container lists. The top container part was pretty crucial, as it gives us better control over the boxes in a collection - each box gets a barcode (and since our boxes aren't barcoded yet, I generated dummy barcodes) and that box is now Box 1 in a certain collection, but also is this exact box that lives on this floor and is not the same as the Box 1 in this other collection, and operates much like how we manage books and items in Aleph.

Once all that was complete, I also had a schematron I could run to check that everything would pass ArchivesSpace import.



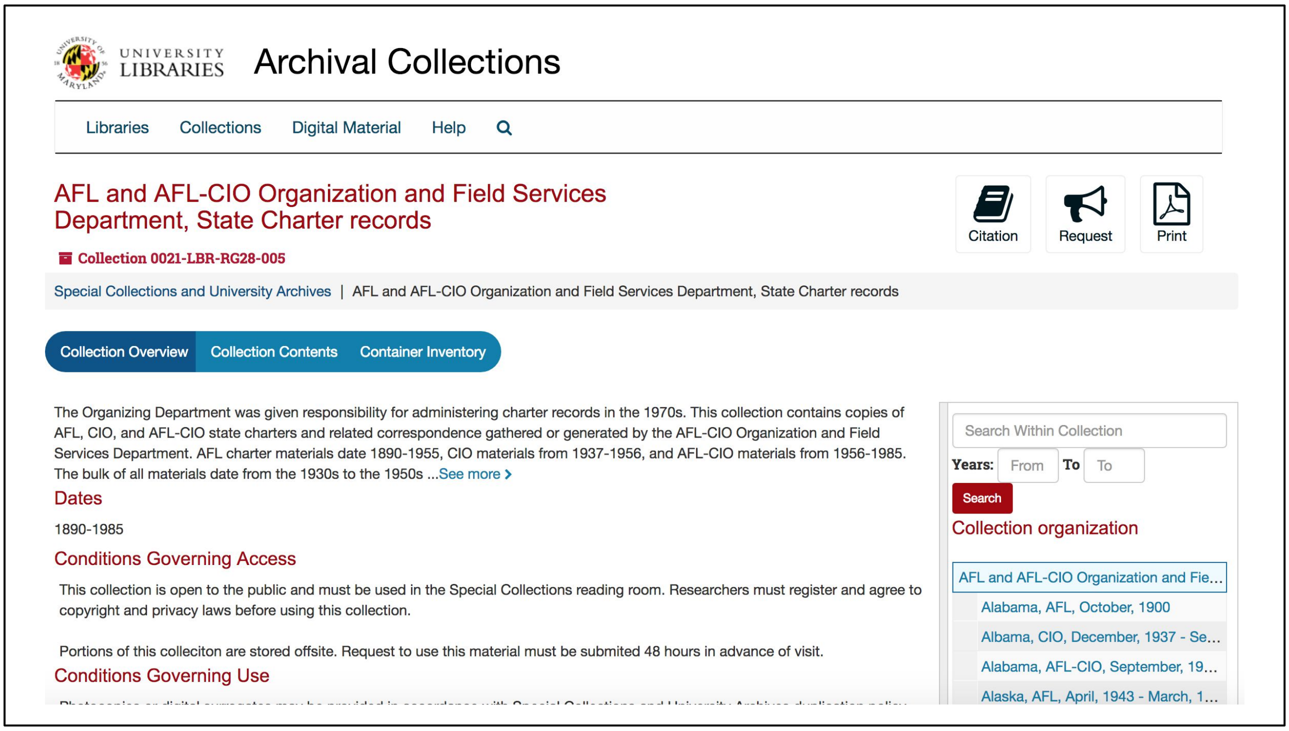
Log

```
Created: /repositories/import/archival_objects/import_1b69ad08-3ff6-4f34-a978-9b3144012628
Created: /repositories/import/archival_objects/import_4ec1a509-bbdd-4f4e-b40d-25c1fd5518e7
Created: /repositories/import/archival_objects/import_210f3c23-9965-4d53-8d93-c794e307ec8f
Created: /repositories/import/archival_objects/import_79057e3e-a1f4-4179-bcc6-72e5fd5d5052
Created: /repositories/import/archival_objects/import_ff38b7f3-0ead-4f90-94fb-836d94c72e87
Created: /repositories/import/archival_objects/import_1b0b377d-0715-4fda-a4e6-a719ef080465
6. DONE: Saving records: cycle 2
7. STARTED: Saving records: cycle 3
Created: /repositories/import/archival_objects/import_a71de3dc-b538-484c-a435-27eeddfd1450
Created: /repositories/import/archival_objects/import_41b0067c-1287-4355-801b-cde0fb3a6846
Created: /repositories/import/archival_objects/import_4ebc5f5b-029e-4ec8-aa39-cf9e088b15eb
Created: /repositories/import/archival_objects/import_e521069d-5562-49a9-88c0-109c6ff7bc35
Created: /repositories/import/archival_objects/import_cf1dd47c-9dd5-46bf-95d2-153f30577048
Created: /repositories/import/archival_objects/import_ab17f3f2-03b0-4985-9b14-b91d130c8e35
Created: /repositories/import/archival_objects/import_caa53699-8329-4cb9-8c6c-b088c5c2bbf9
Created: /repositories/import/archival_objects/import_3caaea72-16cb-4fd9-902e-eb6a688c432b
Created: /repositories/import/archival_objects/import_07867f59-7b84-4811-a719-aed7022c0cf3
```

New & Modified Records

- Товарищи конец
- International Labor News Service clipsheet 28
- La Chienlit C'est Lui (He is the chaos)
- International Labor News Service clipsheet 29
- Työmaana maailma (translation: [The worksite is the world])
- International Labor News Service clipsheet 30
- Valitse Puhtaampi Ympäristö (translation: Choose a healthier evironment)

Now for import! At this point, with this process, I have imported over 85 resources/finding aids into ArchivesSpace production, accounting for approximatel 2400 linear feet of materials..


UNIVERSITY LIBRARIES
Archival Collections
Libraries Collections Digital Material Help
Citation Request Print

AFL and AFL-CIO Organization and Field Services Department, State Charter records

Collection 0021-LBR-RG28-005

Special Collections and University Archives | AFL and AFL-CIO Organization and Field Services Department, State Charter records

Collection Overview Collection Contents Container Inventory

The Organizing Department was given responsibility for administering charter records in the 1970s. This collection contains copies of AFL, CIO, and AFL-CIO state charters and related correspondence gathered or generated by the AFL-CIO Organization and Field Services Department. AFL charter materials date 1890-1955, CIO materials from 1937-1956, and AFL-CIO materials from 1956-1985. The bulk of all materials date from the 1930s to the 1950s ...See more

Dates

1890-1985

Conditions Governing Access

This collection is open to the public and must be used in the Special Collections reading room. Researchers must register and agree to copyright and privacy laws before using this collection.

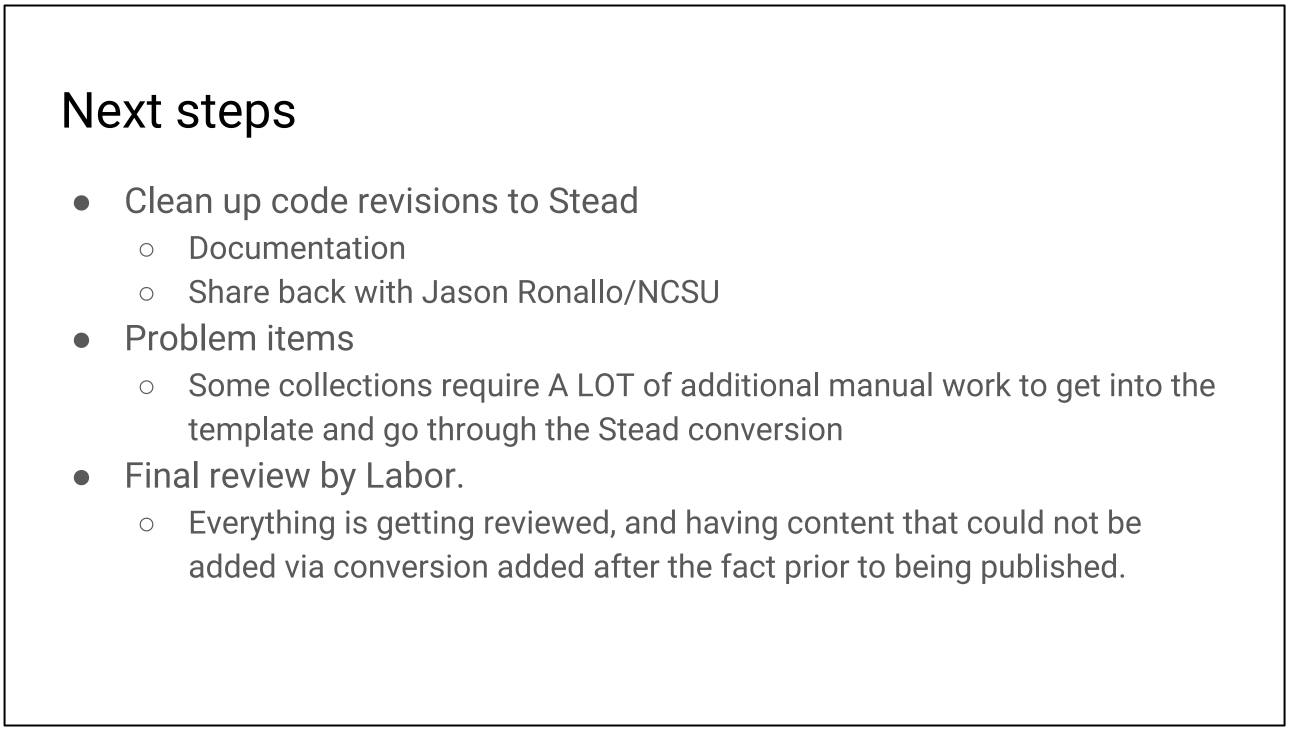
Portions of this colleciton are stored offsite. Request to use this material must be submited 48 hours in advance of visit.

Conditions Governing Use

Collection organization

- AFL and AFL-CIO Organization and Fie…
  - Alabama, AFL, October, 1900
  - Albama, CIO, December, 1937 - Se…
  - Alabama, AFL-CIO, September, 19…
  - Alaska, AFL, April, 1943 - March, 1…

They are still undergoing review by Jen, et al in SCUA, but many have already been approved! So here's a quick view of the new ArchivesSpace public user interface, and you can a little along the side about series...



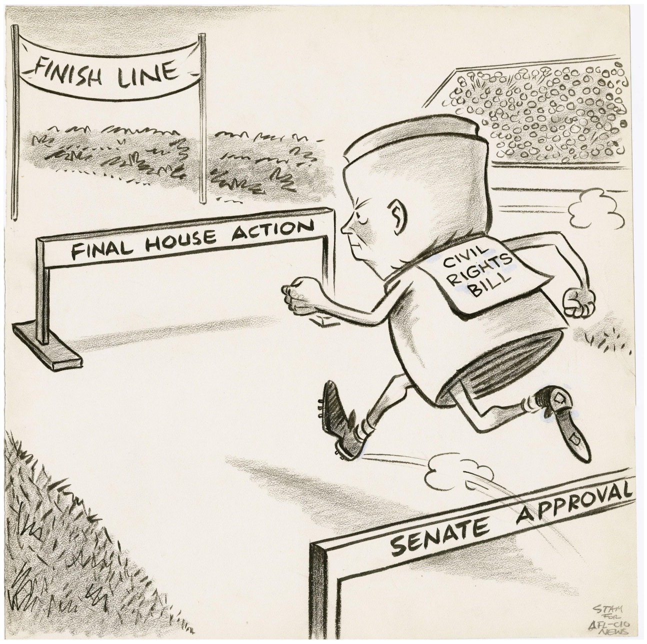
Next steps

- Clean up code revisions to Stead
  - Documentation
  - Share back with Jason Ronallo/NCSU
- Problem items
  - Some collections require A LOT of additional manual work to get into the template and go through the Stead conversion
- Final review by Labor.
  - Everything is getting reviewed, and having content that could not be added via conversion added after the fact prior to being published.

Next steps - finish cleaning up the code, and documenting it, and sharing back with Jason Ronallo for possible inclusion into his original codebase, or to maybe at least have him link to our expanded STEAD code.

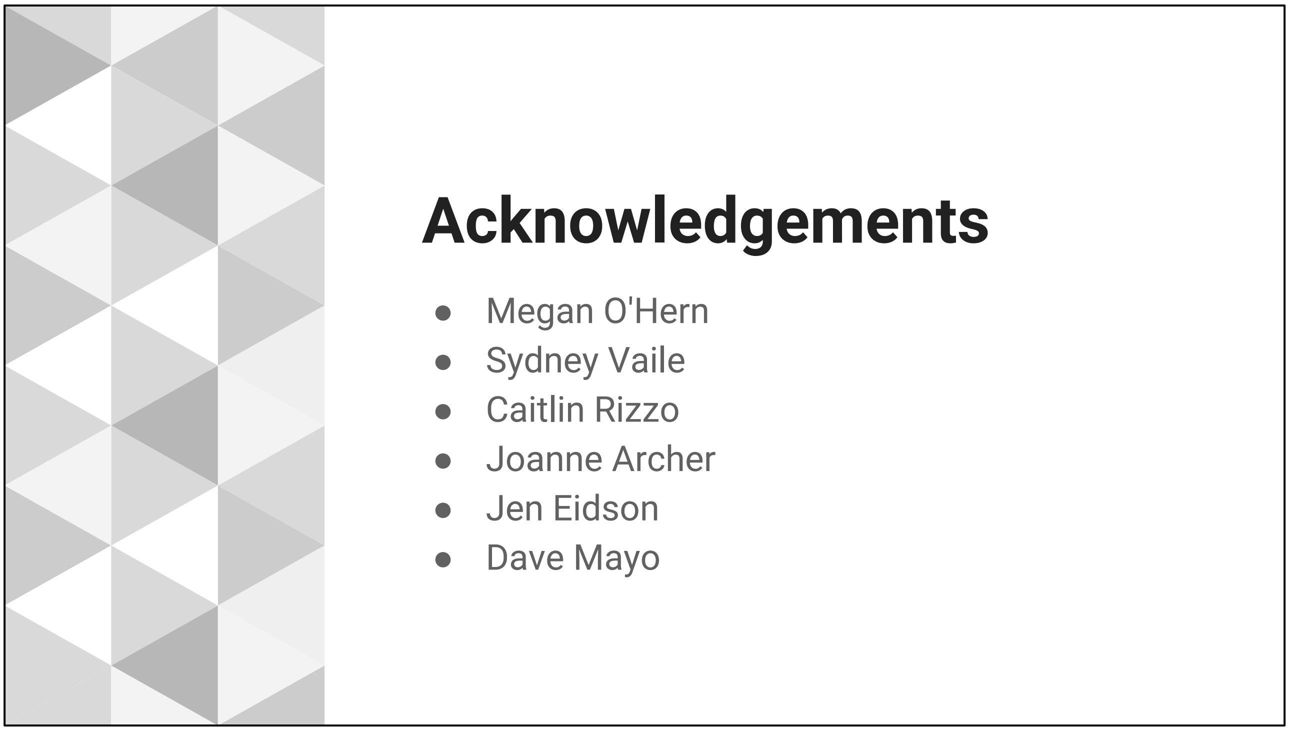
There are still a handful of problem collections that have required additional manual work and research to get into the stead template, and there are some outstanding questions remaining on those.

And, as I mentioned, the final review by Labor, as well as some work that can be done using the API to add parts that we just weren't able to accommodate using Stead.



Cartoon of Civil Rights Bill...ALF-CIO News, https://hdl.handle.net/1903.1/40014

So, we're pretty close to the finish line



Acknowledgements

- Megan O'Hern
- Sydney Vaile
- Caitlin Rizzo
- Joanne Archer
- Jen Eidson
- Dave Mayo

I wanted to acknowledge a few folks who have personally helped me with this project along the way, or contributed significant work on this, specifically, Megan O'Hern, Sydney Vaile, Caitlin Rizzo, Joanne Archer, Jen Eidson, and Dave Mayo.

Questions?

Thanks!

blparker@umd.edu