Ohio Data

A computational analysis of library circulation data.

Attributions

This report contains information from OhioLINK Circulation Data (https://www.oclc.org/research/areas/systemwide-library/ohiolink/circulation.html) which is made available by OCLC Online Computer Library Center, Inc. and OhioLINK under the ODC Attribution License (https://www.oclc.org/research/areas/systemwide-library/ohiolink/odcby.html).

Imports

Libraries used in the analysis

- *pymarc* handles the bibliographic data stored in .marc files

In [2]:
```
import sys
#!{sys.executable} -m pip install matplotlib
from pymarc import MARCReader
from tabulate import tabulate
import pickle
from matplotlib import pyplot
import random
```

Circulation Data

Loading Circulation Data

In [58]:
```
filePath = 'OhioCirculationData/OhioLinkCirc.fil'
with open(filePath, 'r') as f:
    fileDat = f.read().splitlines()
f.close()
circDat = [item.split('\t') for item in fileDat]
```

Example data from "OhioLinkCirc.fil"

Each line of data contains all or some of the following information seen below:

[unique Id, OCLC no., work no., institution supplying data, campus, administrative unit, subunit, date entered into system, date of last use, circulation status, total times circulated, anual circulation]

In [60]:
```
index = random.randint(0, len(circDat))
print(circDat[0], '\n')
print(circDat[index])
```

```
['1973102', '1', '1', 'Youngstown', 'Youngstown State University', 'Maag Libr
ary', '', '08-28-2003', '', '1', '0', '0']

['1926105', '878425', '101081', 'KentState', 'Kent State University', 'Main L
ibrary', 'General Collection', '06-20-1994', '12-03-1999', '1', '2', '0']
```

In [ ]:
```

```

Functions

Functions used to parse the circulation data and gather circulation statistics.

In [86]:
```
# Counts the number of unique items in a list
def uniqItems(itemList):
    oclcNum = [item[1] for item in itemList]
    return len(set(oclcNum))

# Counts the total number of institutions supplying data
def numInstitutes(itemList):
    institutions = [item[3] for item in itemList]
    return len(set(institutions))

# Counts the number of items that are currently circulating
def itemsInCirc(itemList):
    inCirc = [int(item[9])for item in itemList]
    return sum(inCirc)

# Stores each resources circulation data in a dictionary.
# Each key corresponds to a unique OCLC number.
def gatherCircStats(itemList):
    #{oclcNum:[numItems, numItemsInCirc, totalCirculation, anualCirc]}
    itemDic = {}
    for item in itemList:
        if item[1] not in itemDic:
            itemDic[int(item[1])] = [1, int(item[9]), int(item[10]), int(item[
11]), [item[3]]]
        else:
            itemDic[item[1]][0] += 1
            itemDic[item[1]][1] += int(item[9])
            itemDic[item[1]][2] += int(item[10])
            itemDic[item[1]][3] += int(item[11])
            itemDic[item[1]][4].append(item[3])
    return itemDic

# The average number of times a resource corresponding to a unique
# OCLC no. has circulated.
def getAvgCirc(item):
    return item[2] / item[0]

# Computes the average number of times a resource corresponding
# to a unique OCLC no. circulated in a year.
def getAvgAnualCirc(item):
    return item[3] / item[0]

circStats = gatherCircStats(circDat)
```

Basic Statistics

In [66]:
```
print('There are ' + str(len(circDat)) + ' items in the data set and ' + str(i
temsInCirc(circDat)) + " are in circulation.")
print(str(uniqItems(circDat)) + ' of the items are unique')
print(str(numInstitutes(circDat)) + ' institutions are supplying data.')
```

```
There are 29002327 items in the data set and 24637368 are in circulation.
6779969 of the items are unique
74 institutions are supplying data.
```

Bibliographic Data

Bibliograhic data is stored in .marc files that follow the MARC 21 standard for bibliographic data.

Parse Error

From reading the pymarc documentation (https://pymarc.readthedocs.io/en/latest/#reading) and the MARCReader docstrings in pymarc.reader source code (http://localhost:8888/?token=1d99f31e99827dd33afb03006153048202fb3b4711687efc), it was determined that the parameters used for MARCReader allow the data in the file to be correctly parsed.

- The parse error comes from the leader (LDR) not having the appropriate encoding value to indicate how the record should be decoded.
- The issue was not related to the LCCN field tag ('050')

Missing LCCN Data

Items that did not have a LCCN did not contain a field with the tag '050'.

- Some records contain a field with tag '090' for locally assigned LCCNs.
- Records without '090' or '050' tag appear to not contain LCCN data within the marc file.
  - On Classify (http://classify.oclc.org/classify2/ClassifyDemo?&startRec=0) these items contained LCCN numbers for some holdings while some where unclassified.
  - For example, see the entry for an item with OCLC # 6728 (http://classify.oclc.org/classify2/ClassifyDemo?search-standnum-txt=6728&startRec=0).
- Potential tag(s) of interest:
  - '055': Library and Archives Canada (LAC)
    - uses LCCN with sections specifically created for Canadian history/lit.
  - '060' and '070' are LCCN-like call numbers used by the National Library of Medicine and the National Library of Agriculture respectively

Loading Bibliographic Data

In [ ]:
```
filePath1 = 'OhioCirculationData/OhioLINK_1.marc'
filePath2 = 'OhioCirculationData/OhioLINK_2.marc'

def getMarcData(filepath, biblist, noLCC):
    i = 0
    with open(filepath, 'rb') as f:
        reader = MARCReader(f, to_unicode=True, force_utf8=True,utf8_handling=
'ignore')
        for record in reader:
            if i == 0:
                example = record
            lcc = [field.value() for field in record.get_fields('050')]
            if lcc == []:
                lcc = [field.value() for field in record.get_fields('090')]
                if lcc == []:
                    num = random.randint(1, 1000)
                    if num == 1:
                        noLCC.append(record)
            biblist.append(((int(record['001'].value()[3:])), lcc))
            i+=1
    return (biblist, noLCC, example)

try:
    with open ('bibData.pk', 'rb') as f:
        biblist = pickle.load(f)
        example = pickle.load(f)
        noLCC = pickle.load(f)

except FileNotFoundError:
    biblist = []
    noLCC = []
    biblist, noLCC, example1 = getMarcData(filePath1, biblist, noLCC)
    biblist, noLCC, _ = getMarcData(filePath2, biblist, noLCC)

    with open('bibData.pk', 'wb') as f:
        pickle.dump(biblist, f)
        pickle.dump(example1, f)
        pickle.dump(noLCC, f)
```

Example of bibliographic data

- The first is an example of an entry in a .marc file.
- The second is a tuple containg the relevant information extracted from the .marc file. The first entry is the OCLC number and the second is the resources Libray of Congress Classification.

In [6]:
```
print(example1)
print(biblist[0])
```

```
=LDR  01075cam  2200289 a 4500
=001  ocm00000001\
=003  OCoLC
=005  20061229000001.0
=008  690526s1963\\\\ilua\\\j\\\\\\000\1\eng\\
=010  \\$a   63011276
=040  \\$aDLC$cDLC$dIUL$dOCL$dOCLCQ$dTML$dOCL$dOCLCQ$dBTCTA
=019  \\$a6567842$a9987701$a53095235
=042  \\$alcac
=050  00$aPZ5$b.R1924
=082  00$a[Fic]
=096  \\$aQV 4 An78 v.46 2006
=245  04$aThe Rand McNally book of favorite pastimes /$cillustrated by Doroth
y Grider.
=246  30$aFavorite pastimes
=260  \\$aChicago :$bRand McNally,$c[1963]
=300  \\$a110 p. :$bcol. ill. ;$c33 cm.
=520  \\$aBoys and girls in these four stories work hard to master ballet dan
cing, riding, baton twirling, and swimming.
=505  0\$aLittle ballerina / by D. Grider -- Little horseman / by M. Watts --
Little majorette / by D. Grider -- Little swimmers / by V. Hunter.
=650  \1$aShort stories.
=700  1\$aGrider, Dorothy.
=700  1\$aHunter, Virginia.
=938  \\$aBaker and Taylor$bBTCP$n63011276
=994  \\$a11$bOCL$i00466

(1, ['PZ5 .R1924'])
```

Examples of Records missing LCCNs

In [7]:
```
for i in range(2):
    index = random.randint(0, len(noLCC))
    print(noLCC[index])
```

```
=LDR  01324cam  2200301Ii 4500
=001  ocm03087722\
=003  OCoLC
=005  20040407224153.0
=008  770701s1977\\\\dcu\\\\\\\\\\f000\0\eng\d
=040  \\$aGPO$cGPO$dOCLCQ
=037  \\$a044-000-90966-9
=037  \\$b20402
=041  1\$aengfreporspa
=043  \\$an-----$as-----
=074  \\$a899
=086  0\$aS 9.10:8413
=130  0\$aConvention to Prevent and Punish the Acts of Terrorism Taking the F
orm of Crimes Against Persons and Related Extortion That are of International
Significance.
=245  10$aOrganization of American States Convention on Terrorism between the
United States of America and other governments, done at Washington February
2, 1971.
=260  \\$a[Washington] :$bDept. of State :$bFor sale by the Supt. of Docs.,
U.S. Govt. Print. Off.,$c1977.
=300  \\$a[2], 34 p. ;$c24 cm.
=490  1\$aTreaties and other international acts series ;$v8413
=500  \\$aEnglish, French, Portuguese, and Spanish.
=500  \\$a"Entered into force with respect to the United States of America Oc
tober 20, 1976."
=610  20$aOrganization of American States.
=650  \0$aTerrorism.
=710  1\$aUnited States.$bDept. of State.
=810  1\$aUnited States.$tTreaties, etc. (Treaties and other international ac
ts series) ;$v8413.
=994  \\$a11$bOCL$i00660

=LDR  00532nam  2200193Ia 4500
=001  ocm21516025\
=003  OCoLC
=005  19931118140955.0
=008  900511s1988\\\\caua\\\\\\\\\\000\0\eng\d
=040  \\$aIFA$cIFA
=020  \\$a0887346073 :$c$8.95
=092  \\$a791.33
=100  1\$aRoberts, James E.
=245  10$aStrutter's guide to clown makeup /$cby Jim Roberts.
=260  \\$a[Studio City, CA :$bEmpire Pub. Service],$c1988.
=300  \\$a79 p. :$bill. ;$c22 cm.
=650  \0$aClowns.
=650  \0$aTheatrical makeup.
=994  \\$a11$bOCL$i00228
```

Basic Statistics

In [9]:
```
total = len(biblist)
LCCN = [item[1] for item in biblist]
noLCCDat = [item[0] for item in biblist if item[1] == []]
withNoLCC = len(noLCCDat)
onePlusLCCN = len([item for item in LCCN if len(item) > 1])
```

In [10]:
```
table = [["Library of Congress", (total-withNoLCC),  withNoLCC, onePlusLCCN],\
                 ]
print(tabulate(table, headers=["", "# with a\nclassification",\
                               "# without a\nclassification", \
                               "# with multiple\nclassifications "]))
```

```
                          # with a       # without a     # with multiple
                        classification    classification    classifications
------------------  ----------------  ----------------  ------------------
Library of Congress           5797733            982236               48014
```

In [11]:
```
LoCDat = [[item[0], item[1]] for item in biblist if len(item[1]) != 0]
```

Library of Congress Analysis

Check against circulation stats

In [96]:
```
oclcNums = random.sample(biblist, 20)
for item in oclcNums:
    print(item[0], item[1], circStats[item[0]])
```

```
31480026 [] [1, 1, 0, 0, ['Cincinnati']]
1178120 ['PZ3.N4294 Pi PR6027.E855'] [1, 1, 0, 0, ['Ursuline']]
955814 ['PN6149.P5 T5'] [1, 1, 2, 0, ['OhioUniv']]
3880773 ['PZ3.S5439 C'] [1, 0, 0, 0, ['OhioState']]
14139134 ['DS644 .T64 1986'] [1, 1, 1, 0, ['OhioUniv']]
56672136 ['HD8686.5 .R69 2005'] [1, 1, 1, 1, ['Toledo']]
1974914 ['TK7878.7 .A4'] [1, 1, 0, 0, ['OhioUniv']]
38537257 ['QH81 .W56 1996'] [1, 1, 1, 0, ['BaldwinWallace']]
52974437 ['BX350.A6 C45 D76 1985'] [1, 0, 0, 0, ['OhioState']]
2029364 ['PZ3.C123 Mas PR4404'] [1, 1, 0, 0, ['Youngstown']]
7918267 [] [1, 0, 0, 0, ['WrightState']]
682803 ['PZ8.1.C8 T48 1970'] [1, 1, 2, 0, ['Akron']]
11634384 ['F1476.G92 P7'] [1, 1, 2, 0, ['Miami']]
17336884 ['PS3511.A86 R6 1986'] [1, 1, 4, 0, ['CuyahogaCC']]
36666373 ['DR27.G8 T75 1995'] [1, 1, 0, 0, ['Cincinnati']]
29549192 ['LC3993.9 .M35 1994'] [1, 1, 4, 0, ['WrightState']]
8767003 [] [1, 0, 0, 0, ['OhioState']]
546691 ['QE515 .M3 1958'] [1, 1, 0, 0, ['Walsh']]
56460911 ['QA76.73.P224 Z35 2004'] [1, 1, 3, 2, ['Dayton']]
8958813 ['HD2766.U51 P64'] [1, 1, 2, 0, ['OhioState']]
```

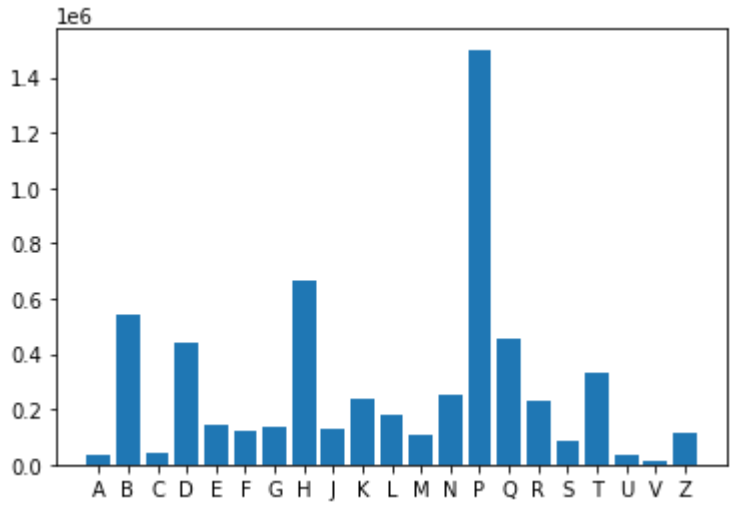
In [97]:
```
def sortByClass(LoCDat):
    categories = "ABCDEFGHJKLMNPQRSTUVZ"
    classDict = {c: [] for c in categories}
    classDict["Other"] = []
    for item in LoCDat:
        label = item[1][0][0]
        if label in categories and len(item[1][0]) > 2:
            classDict[label].append((item[0], item[1][0]))
        elif label == "[":
            label = item[1][0][1]
            if label in categories and len(item[1][0]) > 2:
                classDict[label].append((item[0], item[1][0]))
        else:
            classDict["Other"].append((item[0], item[1][0]))
    return (classDict)

# some LCCNS still have weird formats
# need to combine numeric second chars - constitutes 2nd level sub category
def sum2ndSubCats(classDict):
    for item in classDict:
        if item != "Other":
            secondChars = [entry[1][1] for entry in classDict[item] if entry[1
][1].isalnum()]
            numSubCats = len(set(secondChars))
            classDict[item].append(numSubCats)
    return classDict

classification = sortByClass(LoCDat)
classification = sum2ndSubCats(classification)
```

Items Per Category

In [28]:
```
for cat in classification:
    print(cat, len(classification[cat]))
```

```
A 31700
B 545093
C 43738
D 440232
E 143300
F 123737
G 137402
H 667557
J 130887
K 238424
L 176524
M 107144
N 251058
P 1500546
Q 455726
R 228454
S 82205
T 330249
U 35079
V 11575
Z 116041
Other 1076
```

In [29]:
```
x = list(classification.keys())[:-1]
height = [len(classification[cat]) for cat in classification if cat != "Other"
]
graph = pyplot.bar(x, height, width=0.8)
```



Discussion of LCCNs entries that were placed in "Other"

The main kinds of entries placed in other were those that begain with the sequence "YA", numbers, or lower case letters. Lower case letters will be placed in other for the time being because although some were a part of strings that looked like actual LCCNs, some were not. The amount of data in other is sufficently small that excluding the lower case class labels doesn't significantly detract from the amount of usable data that is available. "YA" is used to classify young adult finction but only accounts for about 500 books in the entire data set. Other accounts for ~0.01% of the data.

Number of Secondary Subcategories per Category

TBD

Notes

Reminders

- 082 = Dewey
  - Dewey sometimes just [Fic] or [E])
- 050 = LCCN
  - ~.01% of data with a non empty LCCN entry was excluded due to format
- non-circ = can't be checked out
  - i.e. google says non-circulating items are often journals, magazines, newspapers, refernece books etc.
- started to try and look at subcategories, but there are still issue with the formating of some LCCNs. Need to further investigate how to best parse the LCCNs.

Questions

- Does it matter if the LoC classification was assigned by another institution besides the LoC?
- What about books classed in more than one LoC category
- Should locally assigned LCCNs be trusted.
  - quick comparison of local numbers in '090' field to numbers in worldcat lead to either identical LCCNs or LCCN that were identical for the first few characters.