**Notes Started 3/20/24 . . . Added notes - 6/10/14**

**Notes on Adjustments for SNF Surveys:**

Errors in pulling “Survey Type” this is meant to correct for -

* Life Safety Surveys are throwing off the review, so need to be noted after all
  + New Note (6/10/24) -We’ve had some surveys start to appear with E-tags. These are emergency preparedness codes. Those are likely not separate surveys so I don’t think they affect the general categorization. We also don’t care about the details of \*which\* E-tag is cited. Normally there wouldn’t be any, since it’s unusual to be out of compliance - but there was recently a survey that had a long list because the surveyor objected to how they stored the document so marked the facility as have zero compliance - <https://dlp.vermont.gov/sites/dlp/files/documents/green-mountain-2024-03-28.pdf>
  + Is there a way to tell the program to disregard E-tags? Or simply note if an E 000 appears on the survey under a column like “Emergency Preparedness Y / N”?
* The run of logic currently used is getting tripped up by the plan of corrections part of the model, which isn’t really necessary because we can use the SS level to know if the corrections were needed. This removes that component.
* Adds logic related to what type of complaint for complaint surveys
* Expands key words (found more variations when looking at the results)

Other items being cleaned up:

* Accuracy in the Violations Y/N
* Adjustment to F-tag list for more accurate counts of violations
* Use of the Key Words search
* Options for Facility Name
* Options for more accurate SS= pull

Code from shared\_functions.py

[As a note on the code so that we can remember why we did things a certain way, I believe that this categorization function cues off of the cover letter, not the survey type listed in the ‘000’ comments because:

1. Those comments are in a column format that is difficult to read in a left-to-right full page text conversion.
2. If there are multiple surveys in one citation statement ([like the recent E- and F- combo one](https://dlp.vermont.gov/sites/dlp/files/documents/green-mountain-2024-03-28.pdf)) it could be difficult to pull the correct text block (although I believe that it is only in the F 000 text blocks that we would find sub-categories so that is less of a problem]

Is this a correct memory?

def type\_of\_survey(c,violations\_bool,sentence=None):

key\_words = ['revisit','health',,,'annual','re-licensure','complaint',’investigation’, ‘recertification’, ‘life safety’]

#get first sentence of letter

if sentence == None:

sentence = get\_letter\_sentence(c)#does period at end come with last word?

#type='?'

#L = sentence.split(' ')

S = sentence

words1 = ['annual survey','re-licensure','re -licensure', 're- licensure','health','annual survey','re licensure', ’recertification’ ‘re-certification’]

words2 = ['investigation','complaint']

#def condition2():

I think that the reason why this is divided into violations or no violations is to run the plan of correction logic, which is no longer necessary(?)

if not violations\_bool:

for word in words1:

if word in S: return 'Health'

for word in words2:

if word in S: return 'Investigation'

if 'revisit' in S: return 'Revisit'

if ‘life safety’ in S: return ‘Life Safety’

else: return '?' #send to helen

else:

for word in words1:

if word in S: return 'Health'

for word in words2:

if word in S: return 'Investigation'

if 'revisit' in S: return 'Revisit'

if ‘life safety’ in S: return ‘Life Safety’

else: return '?'

#Can this return a series if there is more than one of the words present? For example, the cell can say “health, investigation”?

#Can there be an additional function with this run of logic:

Define Self Report

Perform the function only if there was a word in Words2

Get the text between F000 or F 000 and the words “42 CFR” (I believe this text block always has that phrase near the end because it’s the first half of citing the part of the code referred to in the complaint)

Look for any of these words “self-report” “self report” “self-reported” “self reported” “facility report” “facility reported”

If any of the words appear, return 1, else 0

I think this will then give us a ‘Self Report’ column in the final spreadsheet where an investigation survey initiated by a self-report is 1, by another complaint type is 0, and if it’s not applicable a blank or ?

return '?'

def get\_severity(c):

result = set()

L = c.split()

for elem in L:

if 'SS=' in elem: result.add(elem[3:])

return ', '.join(list(result))

[I don’t know why the spreadsheet is missing so many of these, is it possible that adding ‘SS =’ and ‘SS= ‘ variations would catch more? [See also notes at end on possible solution for SNFs](#1drfwlxulji1)]

def pull\_all\_data(P,facility\_type):

if facility\_type == 'SNF':

pull\_data = snf\_code.pull\_data

else: pull\_data = rch\_alr\_code.pull\_data

#print(pull\_data)

txt1 = os.path.join(P,'txt1')

txt2 = os.path.join(P,'txt2')

txt3 = os.path.join(P,'txt3')

txt1\_paths = list\_txt\_files(txt1)

#print('PATHS',paths)

all\_data = dict()

#print(len(paths))

data\_pulled = dict()

for txt\_name in txt1\_paths:

#print(data\_pulled)

pdf\_path = os.path.join(P,'PDF',txt\_name[:-3]+'pdf')

try:

path\_to\_txt1 = os.path.join(txt1,txt\_name)

contents1 = readFile(path\_to\_txt1)

data\_pulled1 = pull\_data(contents1,full\_path=pdf\_path)

data\_pulled = data\_pulled1

# if data\_pulled1['Type of Survey'] == '\*': print('XXX',pdf\_path,data\_pulled1['R tags'],data\_pulled1['Type of Survey'])

# if data\_pulled1['Type of Survey'] == '\*\*': print('000',pdf\_path,data\_pulled1['R tags'],data\_pulled1['Type of Survey'])

except Exception as e:

print(facility\_type,'EXCEPTION',e,type(e),type(e).\_\_name\_\_)

try:

path\_to\_txt2 = os.path.join(txt2,txt\_name)

contents2 = readFile(path\_to\_txt2)

data\_pulled2 = pull\_data(contents2)

data\_pulled = merge\_dicts(data\_pulled,data\_pulled2,priority=1)

except Exception as e:

print(facility\_type,'EXCEPTION',e,type(e),type(e).\_\_name\_\_)

try:

path\_to\_txt3 = os.path.join(txt3,txt\_name)

contents3 = readFile(path\_to\_txt3)

data\_pulled3 = pull\_data(contents3)

data\_pulled = merge\_dicts(data\_pulled,data\_pulled3)

except Exception as e:

print(facility\_type,'EXCEPTION',e,type(e),type(e).\_\_name\_\_)

traceback.print\_exc()

try:

#print(data\_pulled)

data\_pulled['Type of Facility'] = facility\_type

#data\_pulled['Time Between (days)'] = rch\_alr\_code.time\_elapsed(data\_pulled['Date of Original Survey'],data\_pulled['Date of Results'])

all\_data[txt\_name] = data\_pulled

except Exception as e:

print('zero data pulled',txt\_name)

print(facility\_type,'EXCEPTION',e, type(e),type(e).\_\_name\_\_)

#print(len(all\_data),'ALL DATA')

#line below hopefully not needed

#all\_data = update\_pin(all\_data,data\_pulled3)

#print('HERE')

return all\_data

Code from snf\_code.py

import os

import json

import pandas as pd

import shared\_functions

from PIN\_code import \*

VIOLATIONS = ['483.10','483.12','483.15','483.20','483.21','483.24','483.25','483.30','483.35','483.40','483.45','483.50','483.55','483.60','483.65','483.70','483.75','483.80','483.85','483.90','483.95']

MONTHS = ['january ','february ','march ','april ','may ','june ','july ','august ','september ','october ','november ','december ']

def list\_txt\_files(p):

l = []

for filename in os.listdir(p):

if filename.endswith('txt'): l.append(filename)

return l

def readFile(path):

with open(path, "r",errors='ignore') as f:

return f.read()

def get\_all\_F\_tags(c):

F\_tags = set()

for line in c.splitlines():

try:

if line.startswith('F ') and line[2:2+3].isdigit() and line[5].isspace():

F\_tags.add(line[0:5])

except: pass

try:

if line.startswith('{F ') and line[3:3+3].isdigit() and line[6] == '}':

F\_tags.add(line[1:6])

except: pass

if len(F\_tags)==0: return '?'

return ','.join(list(F\_tags))

def get\_violations(c):

result = []

#result = []

for v in VIOLATIONS:

if v in c:

result.append(v)

return ','.join(result)

The code used to prompt if the user wanted a key word check (I think, but maybe I was just typing them into the underlying code)?

words = ["contracted","agency associate","allegation","altercation","shortages","accident hazard", "accident hazards" ,"falls prevention","psychotropic","physical restraint","lap restraint","GDR"]

def SNF\_key\_words(c):

found = dict()

for w in words: found[w] = 0

for key in found:

if '+' in key: (obsolete)

s1,s2 = key.split('+')

n1 = c.count(s1)

n2 = c.count(s2)

found[key] = (n1,n2)

else:

n = c.count(key)

found[key] = n

return found

#asdfasdfa

#asdfasdfas

"""asdfasdfasdf

asdfasdf

asdfasd"""

def length\_of\_report(c):

s = 'Page 1 of '

n = '?'

for line in c.splitlines():

if s in line:

i = line.find(s)

n = line[i+len(s):]

break

return n

def get\_viol\_bool(s):

l = s.split(',')

non\_000\_tag = 0

for elem in l:

c = elem.strip()

c = elem[1:]

c = c.strip()

if c.isdigit() and int(c)>0: non\_000\_tag = 1

return non\_000\_tag

[The F-000 tag is still the best way to know if there are violations - we could account for surveys where the F-tag isn’t pulled by either having a ‘?’ in the F-tag lead to a ? in the violations, and/or checking the “F-tag doesn’t = 000 group for any surveys with a value in SS= and having those go to a Y for violations (that piece is in the shared functions code). If that is too tricky, there were only 3 miscoded cells and they’re easy to clean up directly from the spreadsheet]

def pull\_data(c,full\_path='None'):

result = dict()

result['Length of Report'] = length\_of\_report(c) + ' pages'

result['Violations'] = get\_violations(c)

result['F tags'] = get\_all\_F\_tags(c)

[Can you use either the non\_000\_tag variable or the presence of a value in violations to only add F 000 to a result if it is the only tag? Alternatively, you can just take out F 000 altogether from the results, because we don’t really need to know its presence in the F-tag list, its main purpose in this context is to know if there were any violations]

result['Facility Name'] = find\_PIN\_snf(c)

[Another option for facilities, which I don’t feel great about but may work out okay, is that the file naming convention appears to be consistent, if we took the first 8 letters I believe that would identify the facility and from there it can be correlated to the facility ID. It’s possible that’s easier to do from the Excel sheet than in the code itself but I don’t know?]

result['Date of Original Survey'] = shared\_functions.date\_of\_survey(c)

result['Date of Results'] = shared\_functions.date\_of\_results(c)

result['Severity Levels'] = shared\_functions.get\_severity(c)

violations\_bool = get\_viol\_bool(result['F tags'])

result['Violations Y/N'] = violations\_bool #'Y' if (violations\_bool or len(result['Violations'].split(','))>0) else 'N'

if result['Violations']!= '': result['Violations Y/N'] = 1

result['Type of Survey'] = shared\_functions.type\_of\_survey(c,violations\_bool)

key\_word\_search = SNF\_key\_words(c)

for key in key\_word\_search:

result[key] = key\_word\_search[key]

return result

----

Possible solutions for SS= issues:

One thing I was thinking for the SS= translation problems in the SNF surveys, what if it were set up to find that block and then mark if any of the letters in the different severity levels appear? Since "S" isn't one of those letters, it doesn't matter if it grabs the full block not just the final letter after "=".

In that run of logic you'd be looking to mark if there are letters in any of the following groupings / how many times they appear:

A / B / C

D / E

F

G

H / I

J / K / L

So, a survey statement with two citations marked F and one marked G would show up in the grid:

A / B / C = 0

D / E = 0

F = 2

G = 1

H / I = 0

J / K / L = 0

And then some way to indicate if the program is unable to read some of the letters - a "?" - so we know if the results are reliable. (Since there should be a letter with every finding except F 000, we could have a double check mechanism to see if the number in the SS= tally matches the number of citations?)

FYI - in the grid, this translates to:

A / B / C = Substantial Compliance

D / E = Not substantial compliance, but not yet substandard care

F = Start of Substandard Care

G = Start of Actual Harm

H / I = Actual Harm plus Substandard Care

J / K / L = Immediate Jeopardy

That level of detail doesn't need to go into the spreadsheet, but it's for your background on why we'd divide it up that way.