1.1a_create_transfused_cohort

May 26, 2021

1 1.1a_create_transfused_cohort

- python 2.7.x
- from the mimic iii PostgreSQL database
- label admissions as **transfused** or **control** based on the ICD9 codes and inputs (labeled with our custom dictionary).
- all the notes for each admission (hadm_id) get ordered by time and concatenated into one note per admission.
- create tables for transfused `transfused_notes_sink` and control `ctrl_notes_sink`

1.1 import libraries, connect to mimic database

[15]:
```
conn.commit();
cur.close();
conn.close();
```

[1]:
```
import sys

import time
from datetime import datetime
import datetime

import pandas as pd
import random

from tqdm import tnrange, tqdm_notebook
from time import sleep

from importlib_metadata import version

# things to connect to the posgres database
import psycopg2
from sqlalchemy import create_engine, update, event

POSTGRES_CONNECT = os.environ.get("POSTGRES_CONNECT")
POSTGRES_ENGINE = os.environ.get("POSTGRES_ENGINE")
conn = psycopg2.connect(POSTGRES_CONNECT)
cur = conn.cursor();
```

```
cur.execute("""SET search_path = mimiciii;""")
engine = create_engine(POSTGRES_ENGINE)

libraries = ['pandas','sqlalchemy','psycopg2','tqdm']
print('last ran: ',datetime.now() )
print("Python Version:", sys.version[0:7])
print( "operating system:", sys.platform)

for lib in libraries:
    print(lib + ' version: ' + version(lib))
```

```
last ran:  2019-12-24 23:53:36.145380
Python Version: 3.7.3 (
operating system: darwin
pandas version: 0.24.2
sqlalchemy version: 1.3.3
psycopg2 version: 2.7.6.1
tqdm version: 4.32.1
```

1.2 Create ICD-9 based groups of admissions

- use the identified ICD-9 codes
- create transfused group
- create grey group
- create control group (everything that's not transfused or grey)

1.2.1 1.1.1 create tranfused group from icd-9 codes

table that lists procedures by admission + pull out rows from `procedures_icd` that + have one of these icd9 codes [9901, 9903,9904, 9905, 9907] + exist in the `patients_adult` table + admissions (hadm_id) = 7514

[ ]:
```
cur.execute("""
DROP TABLE IF EXISTS mimiciii.transfusion_icd9;

SELECT p.*, i.icd9_code, i.hadm_id
    INTO transfusion_icd9
FROM mimiciii.procedures_icd i
INNER JOIN mimiciii.patients_adult p
    ON i.subject_id=p.subject_id
    WHERE i.icd9_code IN ('9901','9903','9904','9905','9907');""")
```

Print counts of transfusion admissions using only icd9 selection criteria.

[4]:
```
cur.execute("""
SELECT COUNT(DISTINCT hadm_id) AS transf_admissions_count,
COUNT(distinct icd9_code) AS code_count
FROM mimiciii.transfusion_icd9
```

```
;""")

print(pd.DataFrame(cur.fetchall(), columns=[␣
↪'transf_admissions_count','code_count']).to_string(index=False))
```

```
 transf_admissions_count  code_count
                    7514           5
```

1.2.2 1.1.2 create grey group from icd-9 codes

create grey group from table that lists procedures (icd-9 codes) by admission

- pull out rows from `procedures_icd` that + have one of these icd9 codes [9900, 9902] + exist in the `patients_adult` table
- unique admissions (hadm_id) = 64

[ ]:
```
cur.execute("""
DROP TABLE IF EXISTS mimiciii.grey_icd9;

SELECT p.*, i.icd9_code, i.hadm_id
    INTO mimiciii.grey_icd9
FROM mimiciii.procedures_icd i
INNER JOIN mimiciii.patients_adult p
    ON i.subject_id=p.subject_id
    WHERE i.icd9_code IN ('9900','9902');""")
```

Print counts of grey admissions using only icd9 selection criteria.

[5]:
```
cur.execute("""
SELECT COUNT(DISTINCT hadm_id) AS grey_admissions_count,
    COUNT(DISTINCT icd9_code) AS code_count
FROM mimiciii.grey_icd9;""")

print(pd.DataFrame(cur.fetchall(), columns=[␣
↪'grey_admissions_count','code_count']).to_string(index=False))
```

```
 grey_admissions_count  code_count
                    64           2
```

1.2.3 1.1.3 create control icd9 group `ctrl_icd9`

from table that lists procedures (icd-9 codes) by admission + keep all admissions that are not in the `transfusion_icd9` or the `grey_icd9` tables + are in the `patients_adult` table + this way we end up with only admissions that have never been assigned one of our transfusion or grey icd9 procedure codes + the 'IS NOT TRUE' is there because of Null values, otherwise we would use 'NOT IN' + unique admissions = 34269

[ ]:
```
cur.execute("""
DROP TABLE IF EXISTS mimiciii.ctrl_icd9;
```

```
SELECT p.*, c.icd9_code, c.hadm_id
    INTO mimiciii.ctrl_icd9
FROM mimiciii.procedures_icd c

INNER JOIN mimiciii.patients_adult p
    ON c.subject_id=p.subject_id

    WHERE (c.subject_id IN (
            SELECT  x.subject_id
            FROM mimiciii.grey_icd9 x))
        IS NOT TRUE

    AND (c.subject_id IN (
            SELECT  t.subject_id
            FROM mimiciii.transfusion_icd9 t))
        IS NOT TRUE
    ;""")
```

Print counts of ctrl admissions using only icd9 selection criteria.

[6]:
```
cur.execute("""
SELECT COUNT(DISTINCT hadm_id),
COUNT(DISTINCT icd9_code) AS code_count
FROM mimiciii.ctrl_icd9;""")

print(pd.DataFrame(cur.fetchall(), columns=[␣
↪'ctrl_admissions_count','code_count']).to_string(index=False))
```

```
 ctrl_admissions_count  code_count
                 34269        1871
```

1.3 Label each input event as transfused, grey, or control

- load in the D_items identified as transufe group and Grey group from xlsx sheet
- use all adult input events to find

1. transfused inputs (T) = inputs ever been assigned a transfuse label
2. grey inputs (G) = inputs that have been assigned a grey label
3. control inputs (N) = inputs that have only been assigned labels that are NOT transfue or grey

1.3.1 1.1.4 Create `transfusion_items_dict`

Import the labeled D_items from csv This involves importing a csv file into a new postgres table. Sometimes creating new tables in this way does not like to work with python and it's easier to just do it at the postgres command line, but your mileage may vary.

[21]:
```
# create new empty table in mimiciii schema with the following vars

cur.execute("""
DROP TABLE IF EXISTS mimiciii.transfusion_items_dict;

CREATE TABLE mimiciii.transfusion_items_dict
(Notes varchar,
 GRP char(1),
 ROW_ID int,
 ITEMID int,
 LABEL varchar,
 ABBREVIATION varchar,
 DBSOURCE varchar,
 LINKSTO varchar,
 CATEGORY varchar,
 UNITNAME varchar,
 PARAM_TYPE varchar,
 CONCEPTID varchar,
 ref varchar);""")
conn.commit()
```

[ ]:
```
#Run this command in the postgres command line to create the table if python is␣
 ↪giving issues
COPY mimiciii.transfusion_items_dict
FROM 'D:\\20180717D_ITEMS_related_to_blood_full.csv'
DELIMITER ',' CSV HEADER;
```

Verify that the table has been created correctly. It should have 132 rows total: + T=54 + G=40 + N=38

[7]:
```
cur.execute("""
SELECT grp, count(*)
    FROM mimiciii.transfusion_items_dict
GROUP BY grp;""")

colnames = [desc[0] for desc in cur.description]
print(pd.DataFrame(cur.fetchall(), columns=colnames).to_string(index=False))
```

```
grp  count
  N     38
  T     54
  G     40
```

1.3.2 1.1.5 join these labels with d_items

This will give us a table of labeled D-items based on the SME review of inputs that were transfused, non-transfused or grey. The new labeled items dict we just imported only includes the relevant transfusion-related inputs, so everything that isn't in the new dict, gets a grp label of 'N'for non-

transfused (control).

[8]:
```
cur.execute("""
DROP TABLE IF EXISTS mimiciii.D_items_labeled;

SELECT i.*, d.notes
    ,CASE WHEN grp IS NULL THEN 'N' ELSE grp END

    INTO mimiciii.D_items_labeled
FROM mimiciii.transfusion_items_dict d

RIGHT JOIN mimiciii.D_items i
    ON i.itemid=d.itemid
;""")

conn.commit()
```

1.3.3 1.1.6 Label all inputs

Join new D_items with inputs_all to give each input a grp label

[9]:
```
cur.execute("""DROP TABLE IF EXISTS mimiciii.inputs_all_labeled;""")

cur.execute("""
SELECT d.label,d.grp, i.*
    INTO mimiciii.inputs_all_labeled
FROM mimiciii.D_items_labeled d
    RIGHT JOIN mimiciii.inputs_all i
    ON i.itemid=d.itemid
;""")
conn.commit()
```

Print the number of inputs (non-lab charted items) for each of the groups. Expected values are below:

- N = 289,352,348
- T = 153,154
- G = 3872

[10]:
```
cur.execute("""
SELECT grp, count(*)
    FROM mimiciii.inputs_all_labeled
GROUP BY grp;""")

colnames = [desc[0] for desc in cur.description]
print(pd.DataFrame(cur.fetchall(), columns=colnames).to_string(index=False))
```

```
grp      count
```

```
G       3872
N  289352348
T     153154
```

1.3.4 1.1.7 Create the full list of admissions in transfused group `transfused_hadm_id`

- Create a table (`transfused_hadm_id`) of transfuse group admission ids (**hadm_id**) from the icd9 (`transfusion_icd9`) and the non-lab chart events (`inputs_all_labeled` **grp=T**) criteria
- Transfusion admissions_count = 21541

[11]:
```
cur.execute(""" DROP TABLE IF EXISTS MIMICIII.transfused_hadm_id;

SELECT DISTINCT hadm_id
    INTO mimiciii.transfused_hadm_id

FROM mimiciii.inputs_all_labeled c

    WHERE grp='T'
    AND c.hadm_id IS NOT NULL
    UNION

SELECT DISTINCT hadm_id
FROM mimiciii.transfusion_icd9
    WHERE hadm_id IS NOT NULL
;""")
```

Print total Transfused admissions

[12]:
```
cur.execute("""
SELECT count(DISTINCT hadm_id)
FROM mimiciii.transfused_hadm_id
;""")
print(pd.DataFrame(cur.fetchall(), columns=[ 'transfusion admissions_count']).
 ↪to_string(index=False))
```

```
 transfusion admissions_count
                        21541
```

1.3.5 1.1.8 Create the full list of admissions in the grey group `grey_hadm_id`

- Create a table (`grey_hadm_id`) of grey group admission ids (**hadm_id**) from the icd9 (`grey_icd9`) and the non-lab chart events (`inputs_all_labeled` **grp = G**) criteria
- grey admissions_count = 2373

[13]:
```
cur.execute(""" DROP TABLE IF EXISTS MIMICIII.grey_hadm_id;

SELECT DISTINCT hadm_id
    INTO mimiciii.grey_hadm_id
```

```
FROM mimiciii.inputs_all_labeled c

    WHERE grp='G'
    AND hadm_id IS NOT NULL
    UNION

SELECT DISTINCT hadm_id
FROM mimiciii.grey_icd9
    WHERE hadm_id IS NOT NULL
;""")
```

Print the grey admissions count

[14]:
```
cur.execute("""
SELECT count(DISTINCT hadm_id)
FROM mimiciii.grey_hadm_id
;""")
print(pd.DataFrame(cur.fetchall(), columns=[ 'grey admissions_count']).
 ↪to_string(index=False))
```

```
grey admissions_count
                 2373
```

1.3.6 1.1.9 Create list of ctrl admissions `ctrl_ids`

- make a list of hadm_ids admissions not in the transfused or grey groups.
- join all admissions from `ctrl_icd9` and `inputs_all_labeled` = N
- this basically pulls every admission **50,328**

[15]:
```
cur.execute(""" DROP TABLE IF EXISTS mimiciii.ctrl_idsa;

SELECT DISTINCT i.hadm_id
    INTO mimiciii.ctrl_idsa
FROM mimiciii.inputs_all_labeled i

    WHERE i.grp='N'
    AND i.hadm_id IS NOT NULL
    UNION

SELECT DISTINCT y.hadm_id
FROM mimiciii.ctrl_icd9 y
    WHERE y.hadm_id IS NOT NULL
;""")
```

Print the count of preliminary control admissions

[16]:
```
cur.execute(""" SELECT COUNT(DISTINCT hadm_id)
FROM  mimiciii.ctrl_idsa;""")
```

```
ncount=cur.fetchall()
print( pd.DataFrame(ncount, columns=[ 'admissions']).to_string(index=False))
```

```
admissions
     50328
```

1.3.7 1.1.10 remove admissions that belong to the transfused_hadm_id table or the grey_hadm_id

- admissions = 28,128

[17]:
```
cur.execute(""" DROP TABLE IF EXISTS mimiciii.ctrl_ids;

SELECT DISTINCT i.hadm_id
    INTO mimiciii.ctrl_ids
FROM mimiciii.ctrl_idsa i

    WHERE  i.hadm_id NOT IN (
           SELECT  x.hadm_id
           FROM mimiciii.transfused_hadm_id x)

    AND i.hadm_id NOT IN (
            SELECT g.hadm_id
            FROM mimiciii.grey_hadm_id g)
;""")
```

Print the number of control admissions

[18]:
```
cur.execute(""" SELECT COUNT(DISTINCT hadm_id)
FROM  mimiciii.ctrl_ids;""")

ncount=cur.fetchall()
print( pd.DataFrame(ncount, columns=[ 'admissions']).to_string(index=False))
```

```
admissions
     28128
```

1.4 1.1.11 Clean Up, Commit, and Close

[12]:
```
conn.commit()
cur.close()
conn.close()
```