Hierarchical EHR
A 5-Step SAS, SQL & R approach to data management

Uchechukwu Ikeaba

10/22/2019

Background & Motivation

- Massive number of patient encounter, results in high amount of stored data.

- The ability to transform clinical data into useful insights to improve patient's care is of increased necessity.

- This presentation highlights 5 steps (analytical techniques) on a fictitious hierarchical dataset.
  - Focus is on SAS, SQL & R techniques.
  - Similar techniques can be replicated using Python.

Data description and project goal

- Data consist of doctor's delivering babies (multiple patients seen by same doctor).

- Demographic information on mothers includes.
  - Sex of baby.
  - Race, house location (zip code) and age.

- Mother's undergo procedures (code).
  - **Delivery**: 720, 721, 724, 726, 728, 729, 731, 733, 736, 738, 740, 741, 742, 744.
  - **C-section**: 740, 741, 742, 744.

- Goal: For each doctor, count the number of deliveries, C-sections & black patients.

R Software approach

Hierarchical EHR data management: A 5 step approach

Uchechukwu Ikeaba

October 29, 2019

Background & Motivation

1. Massive number of patient encounter, results in high amount of stored data.
2. The ability to transform clinical data into useful insights to improve patient's care is of increased necessity.
3. This template highlights 5 basic steps (analytical techniques) on a fictitious hierarchical (multiple patients seen by same doctor) dataset.

Data Description

1. Data consist of doctor's delivering babies.
2. Demographic information obtained includes: Sex of baby, Mother's (Race, house location (zip code) and age)
3. Mother's undergo procedures (code)

- Delivery: 720, 721, 724, 726, 728, 729, 731, 733, 736, 738, 740, 741, 742, 744
- C-section: 740, 741, 742, 744

4. Goal: For each doctor, count the number of deliveries, C-sections & black patients.

Load working libraries

```
suppressPackageStartupMessages(library(readxl)) # Used to load dataset
suppressPackageStartupMessages(library(dplyr)) # Used in Step 1 (Checks for duplicate rows) & Step 3 (Create de
rived variables from merged dataset)
suppressPackageStartupMessages(library(sqldf)) # Used in Step 2 (Merge both dataset) & 5 (Prepare hierarchical
data for aggregation)
suppressPackageStartupMessages(library(stringr)) # Used in Step 3 (Create derived variables from merged dataset
```

All 5 steps are replicated using the R software

Clinical Data Structure: Data snapshot (1st ten rows)

Data1
Mother_ID
Sex of baby
Race
Age
Zip code

Data1: Mothers record

Obs	Mother_ID	SEX	RACE	AGE	ZIP
1	P003	M	B	26	19122
2	P011	F	B	22	93084
3	P023	M	B	21	16155
4	P090	F	B	20	11223
5	P092	F	W	21	54235
6	P098	M	B	26	56674
7	P231	F	B	22	33445
8	P234	M	W	24	34556
9	P289	F	W	32	15622
10	P728	F	W	29	12345

Data2
Doctor_ID
Mother_ID
PR1
PR2

Data2: Doctors record

Obs	Doctor_ID	Mother_ID	PR1	PR2
1	1001	P001	720	729
2	1001	P002	726	744
3	1001	P003	731	720
4	1001	P003	731	720
5	1002	P004	738	741
6	1009	P011	744	741
7	1002	P023	744	740
8	1006	P090	742	742
9	1008	P092	738	721
10	1008	P098	721	744

In real life applications, there are multiple tables which should be carefully explored

Step 1: Data Pre-processing (checks for duplicate, missing data,…)

- Base SAS

```
/*Mothers dataset*/

/*
 NOTE: There were 20 observations read from the data set WORK.MOTHER.
 NOTE: 3 duplicate observations were deleted.
 NOTE: The data set WORK.MOTHER2 has 17 observations and 5 variables.
 NOTE: The data set WORK.MOTHER_DUPLICATES has 3 observations and 5 variables.
 */
proc sort data = mother nodup out= mother2 dupout = mother_duplicates;
    by Mother_ID Sex Race Age Zip;
run;
```

- SQL

```
/*Mothers dataset: SAS SQL Approach*/
/*The SQL procedure to select distinct rows*/
/*
 NOTE: Table WORK.SQL_MOTHER created, with 17 rows and 5 columns.
*/
proc sql;
    create table SQL_mother as
        select distinct *
            from mother;
quit;

proc sql;
    select *
        from SQL_mother;
quit;
```

- Count duplicates (Base SAS)

```
Generate counts of duplicate datasets*/
/*Mothers dataset*/
proc freq data = mother noprint;
    tables Mother_ID*Sex*Race*Age*Zip
        / out=newdata;
run;

/*Duplicated records*/
proc sort data = newdata out=sorted;
    by count ;
        where count > 1;
run;

proc print data = sorted;run;
```

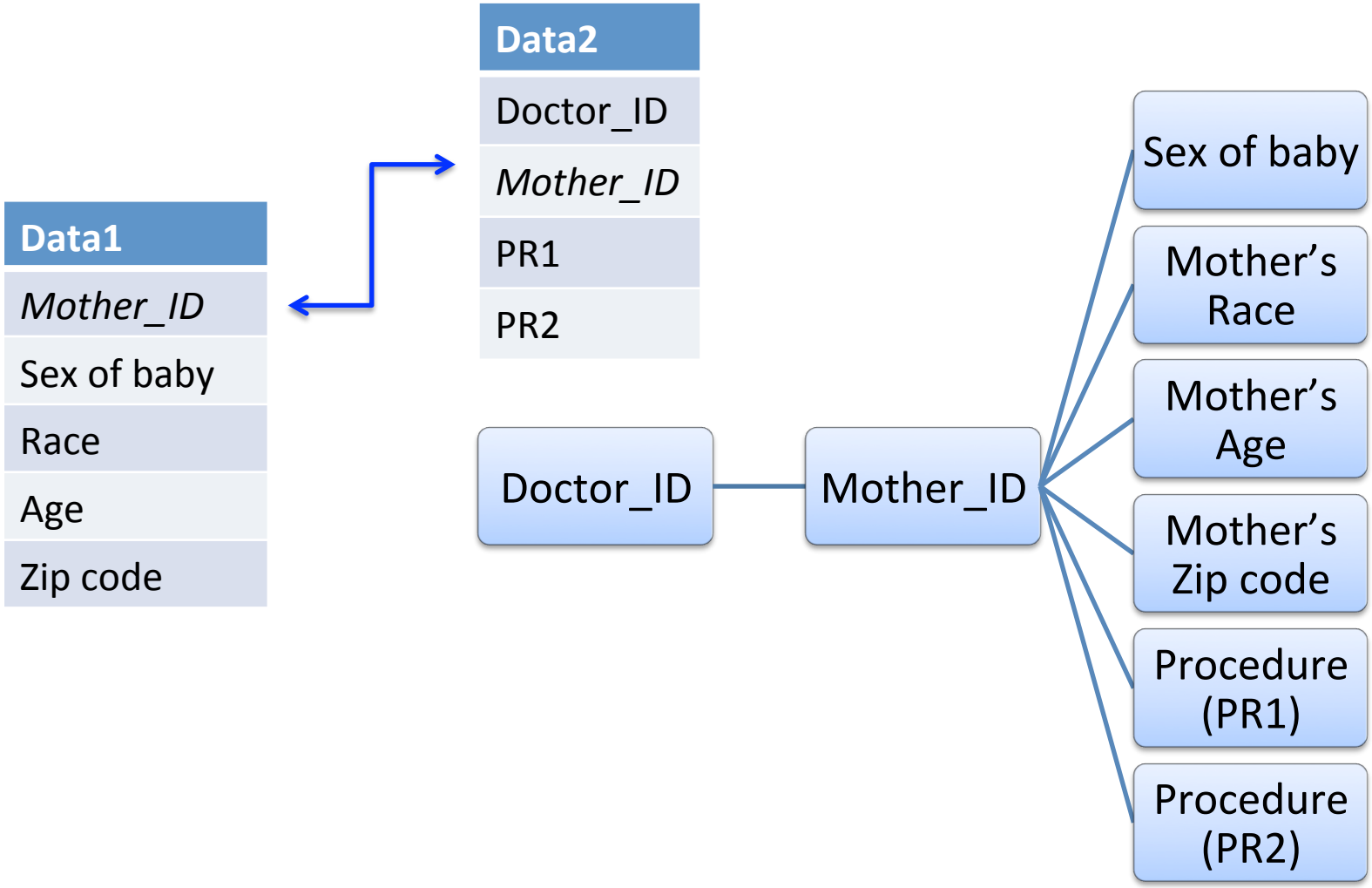
- Count duplicates (SQL)

```
/*Mothers dataset: SQL approach to count all the duplicate
  records in all columns of the table */
proc sql;
    create table SQL_Duplicates as
        select *, count(*) as Count
            from mother
                group by Mother_ID, Sex, Race, Age, Zip
                    having count(*) > 1;
quit;

/*QA*/
proc sql;
    select *
        from SQL_Duplicates;
quit;
```

Data preprocessing is an important step to reducing bias

Concept of Data Model



How do we join both tables to preserve the hierarchical structure?

Step 2: Merge both dataset

```
/*Sort both data before merge*/
proc sort data = doctor2; by Mother_ID;run;
proc sort data = mother2; by Mother_ID;run;
/*
 NOTE: There were 20 observations read from the data set WORK.DOCTOR2.
 NOTE: There were 17 observations read from the data set WORK.MOTHER2.
 NOTE: The data set WORK.LEFTJOIN has 20 observations and 8 variables.
 NOTE: The data set WORK.RIGHTJOIN has 17 observations and 8 variables.
 NOTE: The data set WORK.INNERJOIN has 17 observations and 8 variables.
 NOTE: The data set WORK.NOMATCH_DOCTOR2 has 0 observations and 8 variables.
 NOTE: The data set WORK.NOMATCH_MOTHER2 has 3 observations and 8 variables.
 NOTE: The data set WORK.FULLJOIN has 20 observations and 8 variables.
 NOTE: The data set WORK.NOMATCH_IN_BOTH has 3 observations and 8 variables.
*/
data leftjoin rightjoin innerjoin NOmatch_doctor2 NOmatch_mother2 fulljoin NOmatch_in_both;
    merge doctor2 (IN=In1) mother2 (IN=In2);
        by Mother_ID;
    IF In1=1 then output leftjoin; /*all rows in doctor2 are preserved*/
        IF In2=1 then output rightjoin; /*all rows in mother2 are preserved*/
    IF (In1=1 and In2=1) then output innerjoin; /*doctor2 are excluded if
                                                  they don't match any rows in mother2,*/
        IF (In1=0 and In2=1) then output NOmatch_doctor2;
    IF (In1=1 and In2=0) then output NOmatch_mother2;
        IF (In1=1 OR In2=1) then output fulljoin;
    IF (In1+In2)=1 then output NOmatch_in_both;
 run;
```

Base SAS

SAS SQL

```
proc sql;
    create table SQL_leftjoin as
    select *, coalesce(a.mother_id, b.mother_id) as Mother_ID
        from SQL_doctor as a
            left join SQL_mother as b
                on a.Mother_ID = b.Mother_ID;
quit;
```

Depending on the goal of the study, data can be combined in various ways

Step 3: Create derived variables from merged dataset

```
data dtmgmt_leftjoin;
    set leftjoin;
                    /*delivery*/
        if pr1 in (720, 721, 724, 726, 728, 729, 731, 733, 736, 738, 740, 741, 742, 744)
            then pr1_new = 'delivery'; else pr1_new = 'none';
        if pr2 in (720, 721, 724, 726, 728, 729, 731, 733, 736, 738, 740, 741, 742, 744)
            then pr2_new = 'delivery'; else pr2_new = 'none';

    if pr1_new = 'delivery' or pr2_new = 'delivery'
            then delivery = 'yes';
                else delivery = 'no ';

            delivery_new = (delivery = "yes"); /*Creates delivery dummy variable*/

                  /*delivery_csection*/
        if pr1 in (740, 741, 742, 744) then pr1_new_c = 'deliv_csection'; else pr1_new_c = 'none';
        if pr2 in (740, 741, 742, 744) then pr2_new_c = 'deliv_csection'; else pr2_new_c = 'none';

    if pr1_new_c = 'deliv_csection' or pr2_new_c = 'deliv_csection'
                then delivery_csection = 'yes';
                    else delivery_csection = 'no ';

       c_section = (delivery_csection = "yes"); /*Creates delivery_csection dummy variable*/
                  black = (race = "B"); /*Creates dummy variable*/
run;
```

Base SAS

Deliveries and Csections are created from the procedure codes. Race (Black) is also created

Step 3: Create derived variables from merged dataset (SQL)

```
proc sql;
    create table dtmgmt_SQL_leftjoin as
        select * ,
            case when pr1 in (720, 721, 724, 726, 728, 729, 731, 733,
                   736, 738, 740, 741, 742, 744) then "delivery"
                       else "none" end as pr1_new,
            case when pr2 in (720, 721, 724, 726, 728, 729, 731, 733,
                   736, 738, 740, 741, 742, 744) then "delivery"
                       else "none" end as pr2_new,
            case when pr1 in (740, 741, 742, 744) then "deliv_csection"
                       else "none" end as pr1_new_c,
            case when pr2 in (740, 741, 742, 744) then "deliv_csection"
                       else "none" end as pr2_new_c
from SQL_leftjoin;
quit;
proc sql;
    create table dtmgmt_SQL_leftjoin2 as
    select *,
        case when pr1_new = "delivery" or pr2_new = "delivery"
                then 'yes' else 'no' end as delivery,
        case when pr2_new_c = "deliv_csection" or pr2_new_c = "deliv_csection"
                then 'yes' else 'no' end as deliv_csection
      from dtmgmt_SQL_leftjoin;
quit;
proc sql;
    create table dtmgmt_SQL_leftjoin2a as /*Create dummy variables using SQL procedure*/
    select *,
        case when delivery = "yes" then 1 else 0 end as delivery_new,
        case when deliv_csection = "yes" then 1 else 0 end as c_section,
        case when race = "B" then 1 else 0 end as black
      from dtmgmt_SQL_leftjoin2;
quit;
```

SAS SQL

Deliveries and Csections are created from the procedure codes. Race (Black) is also created

Step 4: QC derived variables from merged dataset

```
/*QA: delivery and c_section*/
proc freq data = dtmgmt_leftjoin;
    tables delivery*delivery_new*c_section*delivery_csection
            /list missing nopercent nofreq;
run;
```

Base SAS

delivery	delivery_new	c_section	delivery_csection	Frequency	Cumulative Frequency
yes	1	0	no	7	7
yes	1	1	yes	13	20

```
/*SQL QA*/
proc sql;
    create table SQL_dtmgmt_QA as
        select  delivery, delivery_new, deliv_csection, c_section, count(*) as count
                from dtmgmt_SQL_leftjoin2a
                    group by  delivery, delivery_new, deliv_csection, c_section
                        order by  delivery, delivery_new, deliv_csection, c_section;
quit;
proc print data = SQL_dtmgmt_QA; run;
```

SAS SQL

Obs	delivery	delivery_new	deliv_csection	c_section	count
1	yes	1	no	0	7
2	yes	1	yes	1	13

Simple Quality Checks can help identify missing categories thereby increasing data accuracy

Step 5: Aggregation of hierarchical data

```
proc sort data = dtmgmt_leftjoin; by Doctor_ID; run;

proc means data = dtmgmt_leftjoin noprint;
    by Doctor_ID; /*Physician level*/
        var delivery_new c_section black;
            output out = dtmgmt_leftjoin2 (drop = _type_ _freq_)
                sum =   sum_delivery_new /*Total number of deliveries*/
                        sum_c_section  /*Total number of csection*/
                        sum_black; /*Total no of black patients*/
                    where delivery = 'yes'; /*only physicians who deliver babies*/
run;
```

Base SAS

```
proc sql;
    create table dtmgmt_SQL_leftjoin2b as
    select
        Doctor_ID,  /*Physician level*/
        sum(delivery_new) as Deliveries, /*Total number of deliveries*/
        sum(c_section) as Csections, /*Total number of csection*/
        sum(black) as Blacks /*Total no of black patients*/

        from dtmgmt_SQL_leftjoin2a
            where delivery = 'yes'
                group by Doctor_ID;
quit;
```

SAS SQL

Accurate data preparation gives reliable and accurate results

Analytical dataset at Doctors level

Raw data

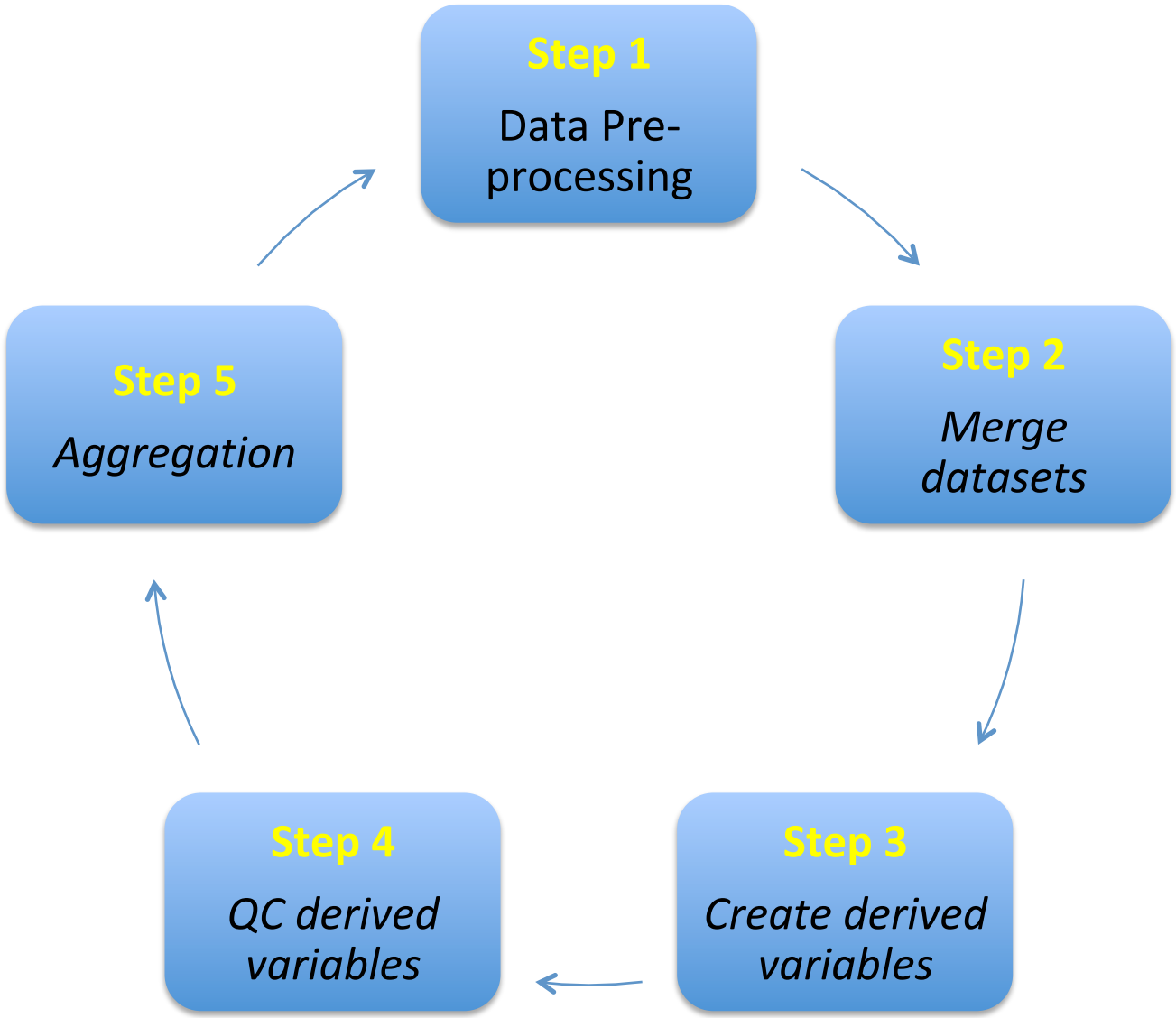
Obs	Doctor_ID	Mother_ID	PR1	PR2	SEX	RACE	AGE	ZIP
1	1001	P001	720	729			.	.
2	1001	P002	726	744			.	.
3	1001	P003	731	720	M	B	26	19122
4	1002	P004	738	741			.	.
5	1002	P023	744	740	M	B	21	16155
6	1003	P289	728	731	F	W	32	15622
7	1004	P910	738	744	F	B	29	16732
8	1004	P912	738	744	F	B	29	16732
9	1005	P784	724	742	M	B	23	23456
10	1006	P090	742	742	F	B	20	11223
11	1006	P728	280	736	F	W	29	12345
12	1007	P893	740	742	M	W	26	55332
13	1008	P092	738	721	F	W	21	54235
14	1008	P098	721	744	M	B	26	56674
15	1008	P231	740	741	F	B	22	33445
16	1008	P234	729	726	M	W	24	34556
17	1008	P789	733	744	F	W	20	58392
18	1009	P011	744	741	F	B	22	93084
19	1009	P852	740	742	F	W	21	38474
20	1010	P783	720	724	M	B	25	10099

Aggregated data

	Deliveries	Csections	Blacks
Doctor_ID			
1001	3	1	1
1002	2	2	1
1003	1	0	0
1004	2	2	2
1005	1	1	1
1006	2	1	1
1007	1	1	0
1008	5	3	2
1009	2	2	1
1010	1	0	1
Total	20	13	10

Number of Deliveries, Csections (740, 741, 742, 744) & Black patients for each doctor

Summary of analytical techniques to explore hierarchical data



SAS, SQL & R are all efficient techniques. R however has more graphic capabilities as shown in Part 2