An introduction to big data mining of electronic health records with Jupyterhub, R, SQL and SAS

Kristen Osinski, Rodney Sparapani and Bradley Taylor

H1210/H1230/H1250: July 15, 2022

Abstract

Since the Affordable Care Act mandated meaningful usage of Electronic Health Records (EHR) data systems in 2011, there has been a paradigm shift in clinical/translational research towards studying EHR administrative databanks. The Clinical and Translational Science Institute (CTSI) of Southeast Wisconsin has been at the forefront of this emerging trend. CTSI Biomedical Informatics has painstakingly created the Clinical Research Data Warehouse (CRDW) with its Honest Broker tables. The CRDW provides access to the EPIC EHR and other ancillary digital databanks. However, searching and manipulating the CRDW was challenging even for those familiar with it. Recently, in 2020, a Jupyterhub server was created so that users could explore the CRDW directly with relative ease via programming languages in common use among biostatisticians, bioinformaticians and data scientists: SQL, R and SAS. This workshop will provide you with a hands-on introduction to the CRDW via Jupyterhub. CITI training and MCWCORP username/password authentication is recommended. Room capacity limit: 90.

Learning Objectives

- Pertinent definitions/jargon for Electronic Health Records (EHR)
- A series of online resources are developed for EHR
- Background of the Clinical Research Data Warehouse (CRDW)
- An overview of the CRDW Jupyterhub environment
- Brief history of SQL, R and SAS
- Hands-on experience with the Jupyterhub by meaningful examples
- Short introduction of RASMACRO

Schedule

8:00	Speaker/attendee introductions
8:15	Bradley Taylor, Chief Research Informatics Officer
8:30	Kristen Osinski, Business Analyst IV, CTSI
9:15	Overview: slides 0:20
10:00	Snack break: getting to know each other
10:30	Hands-on with Jupyterhub: slides 21:30
12:00	Lunch provided
12:30	Hands-on with Jupyterhub: slides 31:37
2:00	Biological break
2:15	Hands-on with Jupyterhub: slides 38:50
3:45	Q and A
4:15	Closing ceremonies

Overview

- Clinical and Translational Science Institute of Southeast Wisconsin (CTSI)
- CTSI Clinical Research Data Warehouse (CRDW)
- CRDW contains adult electronic health records (EHR) at Froedtert and MCW: mainly EPIC EHR but including billing claims and ancillary databanks
- Jupyterhub access to the CRDW for adults
- Collaborative Institutional Training Initiative (CITI) required
- SQL, R and SAS: popular and mature programming languages
- SQL and R come free with Jupyterhub
- SAS can be purchased through the MCW IS Ticket system for Linux server and Windows clients, it's either free or cheap
- [https://mcwcherwellapp.mcwcorp.net/CherwellPortal]
- In this presentation, links are colored MCW green [green] and surrounded by square brackets for color equity

CRDW Data Horizon and Important Eras

- 1989: North American Association of Central Cancer Registries (NAACCR) for Froedtert and Community Memorial
- 1999 to 2018, November: GE/IDX billing
- 2001: NAACCR for St. Joseph's West Bend
- 2004: EPIC EHR debuts at Froedtert
- 2005: GE/MUSE for EKGs
- 2005 to 2007: National Provider Identifiers (NPI) transition
- 2009: American Recovery and Reinvestment Act mandates *meaningful use* of EHR
- 2012, May: EPIC EHR Community Memorial Menominee Falls
- 2013, July: EPIC EHR for Community Physicians Clinics
- 2013, September: EPIC EHR St. Joseph's West Bend
- 2015: Elekta/MOSAIQ radiotherapy dosage
- 2015, October: ICD-10 era begins
- 2020, March: COVID-19 pandemic declared

Resources

- This presentation, programs, etc. are available online [at my `github.com` repository]
- [Biostatistics, Epidemiology and Research Design (BERD)]
- [BERD Mini-grants]
- [Biostatistics Consulting Service (BCS)]
- [i2b2: informatics for integrating biology and the bedside]
- [CTSI Biomedical Informatics links]
- [CTSI Honest Broker Data Dictionary]
- [Project Jupyter]
- CRDW Jupyterhub [https://jupyter.ctsi.mcw.edu]
- [US Centers for Medicare and Medicaid Services ICD-9 to, and from, ICD-10 crosswalks]
- [US Centers for Disease Control and Prevention ICD-10-CM Browser]
- [Area Deprivation Index (ADI)]

What is an Electronic Health Record?

- Electronic Medical Record (EMR) and
  Electronic Health Record (EHR)
  are often used interchangeably
- However, there is a subtle distinction
- Technically, the EMR is merely a digital version (whether of digital provenance or scanned images) of what used to be a paper *medical record* (which is a misnomer, i.e., NOT limited to merely medicine as the name seemingly implies)
- Typically, the EMR is a collection of health-related data
- Inpatient and outpatient doctor/nursing/etc. treatment notes
- Imaging, laboratory results, vital signs, etc.
- Whereas the EHR is an information management system like EPIC providing convenient access to the EMR along with other ancillary digital capabilities such as billing/scheduling, prescription pharmaceutical orders and modern data sources including X-ray/CAT/MRI scans, chemo-/radio-therapy dosage, electrocardiograms, echocardiograms, etc.

What is an Honest Broker?

- "A neutral intermediary ... between the individual whose ... data are being studied, and the researcher. The honest broker collects and collates pertinent information ... replaces identifiers with a code, and releases only coded information to the researcher." [US Health and Human Services FAQ]
- CTSI Biomedical Informatics is the Honest Broker!
- The term originated in diplomacy meaning an entity accepted as impartial by all sides in a negotiation
- German Chancellor Otto von Bismarck was the first to use the term, by applying it to himself, as an intermediary in negotiations between Russia and the Ottoman Empire (Auray-Blais and Patenaude, BMC Medical Ethics 2006)

Honest Broker De-identification

- Jupyterhub data is brought "up-to-date" on Wednesday nights
- HIPAA de-identification provided by the Honest Broker
- For example, patient names, etc. are removed
- The Medical Record Number (MRN), `patient_mrn`, is replaced by `patient_hash` which is an encrypted key
- `patient_hash` is unchanging so that the MRNs could be retrieved if you have IRB approval for identified data
- All dates for each patient are de-identified by a single random integer from -10 to 10 (with zero excluded)
- This allows any two date differentials to be calculated exactly such as the age of a diagnosis with respect to birth date
- Geographically, we have only state of residence and ZIP code shortened to the first 3 digits
- Yet, addresses were geocoded to Census Block Groups (CBG) and the corresponding Area Deprivation Index is provided (the CBG is not of course)

A brief introduction to SQL

- Structured Query Language (SQL)
- The syntax/semantics for interacting with relational database management systems
- Originally developed by IBM: now an industry standard
- [SQL:2016 AKA ISO/IEC 9075:2016]
- [The TIOBE Index of programming language popularity] (circa 04/22)
- SQL is ranked 9
- First appeared in 1974

A brief introduction to R

- The R language is an open-source, free software GNU project purpose-built for data science that is *object-oriented*
- [https://R-project.org]
- [https://CRAN.R-project.org/manuals.html]
- Naturally vectorized language with convenient objects such as vectors, matrices, lists (objects containers) and `date.frame`'s
- On the TIOBE Index of programming language popularity (circa 04/22)
- R is ranked 11
- A derivative of S which first appeared in 1976
- Many free add-on packages
- 18991 available on the Comprehensive R Archive Network (CRAN) (circa 04/22) [https://CRAN.R-project.org]
- N.B. Jupyterhub can be used with either R or Python 3 (Python is ranked 1 on the TIOBE Index circa 04/22)

Accessing the database with R

- Via DBI package and the RPostgres backend from CRAN
- Use your MCWCORP credentials to log in online at [https://jupyter.ctsi.mcw.edu]
- BUT YOU HAVE A SEPARATE JHUSERNAME/JHPASSWORD

```
## snippet1.R
library(DBI)
library(RPostgres)
user="JHUSERNAME"
password="JHPASSWORD"
db=dbConnect(
    Postgres(),
    dbname  ="fh_jupyter_hub_hbdb",
    user    =user,
    password=password,
    host    ="localhost",
    port    =5432,
    bigint  ="integer"
)
```

A brief introduction to SAS

- The SAS language is a proprietary for-fee fourth-generation domain-specific environment for data science
- [https://SAS.com]
- [https://support.sas.com/en/documentation.html]
- Convenient naturally vectorized `DATASTEP` language
- You don't buy SAS, you rent it annually
- The MCW site license goes from June to May
- It is free for students and members of Biostatistics/CAPS
- For others its cheap, use the IS ticket system to order [https://mcwcherwellapp.mcwcorp.net/CherwellPortal] $60 for SAS on the RCC cluster and for Windows installs
- On the TIOBE Index of programming language popularity (circa 04/22)
- SAS is ranked 21
- First appeared in 1972
- The RASMACRO collection of my GPL SAS macros [https://github.com/rsparapa/rasmacro]

A brief introduction to SAS

- Why not just use R instead of SAS?
- R's strengths are not within data processing
- For example, R does not have warnings about non-unique keys whereas SAS does (always check your `.log` !)
- This begs the question: "What is a unique key?"
- Suppose your SSN is unique, does that make it a unique key?
- NO!
- Consider a table consisting of annual earnings where each row/record is a year for a given SSN?
- What is the unique key on this table?
- It is SSN AND year: NOT SSN alone!
- The R function `byvalue` adds SAS-like FIRST./LAST. automatic variables but it is not the same as the real thing with ERRORS/WARNINGS (SAS provides them, R is silent)
- For virtually all tables in the CRDW, the unique key is not obvious and often surprising: SAS is much better suited to this
- You have been forewarned

A brief introduction to Comma Separated Values

- A data exchange format that goes back to the early 1970's
- Popularized by spreadsheets in the 80's and 90's
- What we have today was solidifed by about 2005
- Standards include RFCs 4180 and 7111 among others
- See ["Comma-separated values" on Wikipedia]
- Typically, a three-letter file type of "csv", e.g., "Book1.csv"
- A text file where each field is separated by commas
- A missing value is nothing: two consecutive commas
- Fields with commas are encased in double-quotes
- Double-quotes are escaped by doubling them (like SAS does)
- Double-quotes around numbers (or anything) is read as text
- Although, a standard there are edge cases that can't be *automatically* read by common software such as R and/or SAS
- Furthermore, spreadsheets are very lax: columns can be a mixture of numbers and text that will create havoc when read

A brief introduction to Comma Separated Values

- If CSV files have so many challenges, then why bother?
- No other data exchange formats have caught on
- Alternatives like *recfiles* are not nearly as popular
- With respect to CSV, SAS fills an important niche
- SAS makes the exchange of CSV files with dates, times and date-times effectively trivial
- This is a very important concern since the EHR is rife with chronological info: so much so that you don't even want to consider making their transfer manual in any shape or form
- And we have decades of experience with CSVs
- Largely automated functions roughly in order of usefulness
- SAS: `PROC IMPORT/PROC EXPORT`
- R functions: `read.csv()/write.csv()`
- RASMACRO: `_cimport/_cexport`
  plus `_crepair`, `_constant` and `_verify` for handy features

CSV processing with R

- You can download your CSV files as we will see later
- Either with your web browser or with Secure Copy: `scp`
- `scp` is a standard command on Windows, macOS and Linux
- However, due to IRB restrictions, you may not download data for more than 9999 PATIENTS in one project
- This is the honor system since there is no mechanism to prevent malfeasance
- And you can process them with R
- R will convert character to numeric values where possible
- Except that it handles dates as character by default!
- There is no automatic detection: that is a manual process

```
CMD% scp USER@jupyter.ctsi.mcw.edu:naaccr.csv DIR
R> naaccr = read.csv("DIR/naaccr.csv")
R> str(naaccr)
R> ?str
```

A brief introduction to RASMACRO

- SAS has a powerful macro language
- RASMACRO is GPL library at [https://github.com/rsparapa/rasmacro]
- It is available on the Biostatistics Linux Cluster and the Research Computing Center cluster
- But anyone can install it
- Very useful to working with CRDW data
- `_verify` to automatically convert *character* fields to numeric if possible, i.e., some fields are unnecessarily encoded as text
- `_constant` automatically drops variables that are constant including missing
- `_crepair` for CSVs with very long variable strings
  R will create an esoteric CSV file
  that needs re-formatting so SAS can read it automatically

A brief introduction to RASMACRO

- Installing RASMACRO is fairly trivial on Windows and Linux
- Get it from github in a directory called `rasmacro`

```
git clone
https://github.com/rsparapa/rasmacro.git
```

To get updates: `cd rasmacro; git pull`
Add these lines to your `sasv9_local.cfg`
where `RASMACRO` is the path to your directory

```
/* ' these are single quotes */
-sasautos ('!SASROOT/sasautos' 'RASMACRO')
-set SASAUTOS ('!SASROOT/sasautos' 'RASMACRO')
```

CSV processing with SAS

- SAS does NOT automatically convert character to numeric
- However, the RASMACRO `_verify` can handle that
- SAS does automatically detect dates, times and datetimes!
- Also, the RASMACRO `_constant` automatically detects variable/columns that are a constant value (including missing) and removes them
- In this case, the following were superfluous and removed
- `mult_tum_rpt_as_one_prim, multiplicity_counter, cs_tumor_size, cs_lymph_nodes, cs_mets_at_dx, cs_mets_at_dx_bone, cs_mets_at_dx_brain, cs_mets_at_dx_liver, cs_mets_at_dx_lung, sequence_number_hospital`

```
* snippet1.sas ;
proc import datafile="DIR/naaccr.csv" out=naaccr;
guessingrows=max;
run;
%_verify(data=naaccr, out=naaccr);
```

Hands-on with Jupyterhub

- With your web browser, login [https://jupyter.ctsi.mcw.edu] using MCWCORP credentials, i.e., same as Outlook/etc.
- And use your Jupyterhub-only `JHUSERNAME/JHPASSWORD`
- Pressing Shift+Enter on your keyboard submits the code

```
## snippet2.R
library(DBI)
library(RPostgres)
user="JHUSERNAME"
password="JHPASSWORD"
## db object to facilitate database 2-way communication
db=dbConnect(
    Postgres(),            ## connect to PostgreSQL
    dbname  ="fh_jupyter_hub_hbdb",
    user    =user,
    password=password,
    host    ="localhost", ## loopback web connection
    port    =5432,
    bigint  ="integer"    ## see dbConnect documentation
)
```

Hands-on with Jupyterhub

- Which version of PostgreSQL are we running?
- At the time of this writing, I see version **11**.15
- The documentation can be found at [https://www.postgresql.org/docs/**11**]
- N.B. **11** stands for the major version number, e.g., if the software is upgraded to a newer version, then update accordingly

```
print(dbGetQuery(db, "select version()"))
```

```
PostgreSQL 11.15 ...
```

Hands-on with Jupyterhub

- For those on Biostatistics Cluster servers gouda or colby you can access Jupyterhub over the LAN
- Many advantages using the tools that you use on a daily basis
- Here, I'm using emacs

```
## snippet3.R
library(DBI)
library(RPostgres)
user="JHUSERNAME"
password="JHPASSWORD"
## db object to facilitate database 2-way communication
db=dbConnect(
    Postgres(),
    dbname  ="fh_jupyter_hub_hbdb",
    user    =user,
    password=password,
    host    ="garth.ctsi.mcw.edu", ## LAN connection
    port    =5432,
    bigint  ="integer"
)
```

Hands-on with Jupyterhub

- Let's see the public database tables

```
## snippet4.R
## ' these are single quotes
tables=dbGetQuery(db,
                  "select *
                   from information_schema.tables
                   where table_schema = 'public'")
print(tables$table_name)
```

Hands-on with Jupyterhub

- Let's see the public database table columns

```
## snippet5.R
## ' these are single quotes
columns=dbGetQuery(db,
                   "select *
                    from information_schema.columns
                    where table_schema = 'public'")
str(columns)
table(columns$table_name)
```

Hands-on with Jupyterhub

- We have created the `columns data.frame`
- We used the `str` function to see its *structure*: `str(columns)`
- The first four variables look like so

```
'data.frame': 671 obs. of  44 variables:
$ table_catalog : chr "fh_jupyter_hub_hbdb" ...
$ table_schema  : chr "public" "public" "public" ..
$ table_name    : chr "fh_hb_diagnosis_jupyter" ...
$ column_name   : chr "patient_hash" "encounter_has
...
```

Hands-on with Jupyterhub

- We can quickly summarize the tables in the database
- `table(columns$table_name)`
- For a selection, counts correspond to the number of columns as we have seen from the structure of the `data.frame`

```
                    fh_hb_demographics_jupyter
                                            31
                       fh_hb_diagnosis_jupyter
                                            15
fh_hb_diagnostic_results_jupyter
                                            27
                       fh_hb_mar_table_jupyter
                                            37
  fh_hb_med_orders_table_jupyter
                                            32
                 fh_hb_procs_jupyter
                                            18
               fh_hb_vitals_jupyter
                                            11
```

Hands-on with Jupyterhub

Table Name	[Title in Documentation]
`fh_hb_demographics_jupyter`	"Patient Demographics"
One record per patient: birth date, gender, race/ethnicity, death, etc.	
`fh_hb_diagnosis_jupyter`	"Diagnosis (Dx)"
Combo of EPIC/billing with ICD-9/ICD-10 diagnosis codes	
`fh_hb_diagnostic_results_jupyter`	"Diagnostic Results"
Combo of mainly EPIC with MOSAIQ (for radiotherapy dosage)	
`fh_hb_mar_table_jupyter`	"Medications Administere
EPIC in-patient records with Medi-Span pharm class/sub-class	
`fh_hb_med_orders_table_jupyter`	"Medication Ordered"
EPIC prescription orders (not fills!) along with Medi-Span	
`fh_hb_procs_jupyter`	"Procedures (Px)"
Combo of EPIC/billing with ICD-9/ICD-10 and HCPCS/CPT codes	
`fh_hb_vitals_jupyter`	"Vitals"
EPIC vital signs such as height/weight, blood pressure, temp, etc.	

Hands-on with Jupyterhub

- When does the NAACCR data start?
- Above, I said 1989: how did I get that?
- N.B. Do not confuse the SQL `date_part` function with the SAS `datepart` function: they are similar, yet quite different

```
## snippet6.R
## ' these are single quotes
naaccr=dbGetQuery(db,
           "select
            date_part('year', date_of_diagnosis_shifted)
            as dxyear from fh_hb_naaccr_jupyter")
t(table(naaccr$dxyear))
```

Hands-on with Jupyterhub

- When does the EKG data start?
- Covered in the Appendix A of the
- [CTSI Honest Broker Data Dictionary]

```
## snippet7.R
## ' these are single quotes
muse=dbGetQuery(db,
            "select
             date_part('year',acquisition_date_shifted)
             as ekgyear from ekg_test_demographics")
t(table(muse$ekgyear))
```

Hands-on with Jupyterhub

- The state of Wisconsin (like regions of the US/Canada) has/have a cancer registry that health care systems must submit their incidence data to by statuary requirements
- [NAACCR data dictionaries]
- [Latest version 22]
- NAACCR item 523: ICD-O-3 behavior codes `behavior_code_icd_o_3`
- "valid codes are 0-3" is what NAACCR specifies in their docs which leaves a lot to be desired
- So use the Surveillance, Epidemiology and End Results (SEER) program docs too
- NCI SEER is a subset of NAACCR regions in the US (including WI) with more carefully curated data and much better documentation too!
- [SEER Program Coding and Staging Manual]
- 0: benign, 1: uncertain, 2: in situ, 3: malignant

Hands-on with Jupyterhub

- NAACCR item 400: Primary Site `site_primary`
- ICD-O-3 topography code, e.g., breast cancer C50
- NAACCR item 560: `sequence_number_hospital`
- Indicates the sequence of all malignant and nonmalignant neoplasms over the lifetime of the patient
- How many patients had their first breast cancer between 2016 and 2017?
- The PostgreSQL manual seems to be very terse
- But SAS has a nice description in their manual: [SAS 9.4 SQL Procedure User's Guide, Fourth Edition]
- See the `BETWEEN` and `LIKE` operators

Hands-on with Jupyterhub

```
## snippet8.R
## ' these are single quotes
naaccr=dbGetQuery(db,
            "select * from fh_hb_naaccr_jupyter
             where (sequence_number_hospital='0'
             or sequence_number_hospital='1') and
             (date_part('year', date_of_diagnosis_shifted)
             between 2016 and 2017) and
             site_primary like 'C50%'")
addmargins(table(naaccr$sequence_number_hospital))
addmargins(table(naaccr$site_primary))
```

Hands-on with Jupyterhub

- But, how many of these are malignant?
- Notice that except for the dates/times, everything is character
- In both R and SAS, character is more painful than numeric values (SAS imports dates from CSV files as numeric)
- We can export the data as a CSV with R's `write.csv`
- The option `quote=FALSE` means don't use quotes
- However, if some fields have embedded commas, then you can't turn this option on: and we don't know if there are any
- The `na` option determines how `NA` is handled
- In this case, we set it to the CSV standard: `na=""` for nothing

```
## snippet9.R
addmargins(table(naaccr$site_primary,
           naaccr$behavior_code_icd_o_3))
write.csv(naaccr, "naaccr.csv", row.names=FALSE, na="")
str(naaccr)
?str
```

Honest Broker Residential Address Geocoding

- Geocoding is the process of taking the address of a location and returning geographic coordinates
  such as a latitude/longitude pair or a US Census Tract
- [DeGAUSS: Decentralized Geomarker Assessment for Multi-Site Studies]
- "Patient Demographics" `geocode_result`
- Self-explanatory codes that are not accurately geocoded: `po_box` and `non_address_text`
- `cincy_inst_foster_addr`: the address was not geocoded a known institutional address, not a residential address
- `imprecise_geocode`: the address was geocoded, but results were suppressed because due to the lack of precision
- `geocoded`: the address was geocoded with precision

Introduction to the Area Deprivation Index (ADI)

- [ADI documentation]
- "Patient Demographics" `adi_narank`
- A Census Block Group (CBG) with an ADI ranking of 1 indicates the lowest level of disadvantage within the nation, i.e., least deprived 1st percentile
- A CBG with an ADI ranking of 100 indicates the highest level of disadvantage, i.e., most deprived 100th percentile
- So the range is integers 1 to 100
- Some CBG are missing ADI rankings with the following codes
- `NA` the R missing value code which is undocumented
- `'PH'` for suppression due to low population and/or housing
- `'GQ'` for suppression due to a high group quarters population
- `'PH-GQ'` (or is it `'GQ-PH'`?) for suppression due to both types
- `'KVM'` (or is it `'QDI'`?) designates block groups without an ADI value due to missing data within the data's source the American Community Survey Five Year Estimates

Hands-on with Jupyterhub

```
## snippet10.R
a=dbGetQuery(db, "select geocode_result
                  from fh_hb_demographics_jupyter")
table(a$geocode_result, useNA='ifany')
b=dbGetQuery(db, "select adi_narank
                  from fh_hb_demographics_jupyter
                  where geocode_result='geocoded'")
table(b$adi_narank, useNA='ifany')
unk = (is.na(b$adi_narank) |
       b$adi_narank %in% c('GQ', 'PH', 'GQ-PH', 'QDI'))
table(unk)
b$adi=0
b$adi[unk]=NA
b$adi[!unk]=as.integer(b$adi_narank[!unk])
summary(b$adi)
plot(density(b$adi, from=1, to=100, na.rm=TRUE),
     xlab='ADI', ylab='Distribution',
     main='', sub='1:least, 100:most')
abline(h=0.01, col=8)
```

Death and Cause of Death

- Death is available; however, it may not be trustworthy
- It is common to see EHR data far beyond the death date
- Also, Cause of Death is not available
- It can be acquired from the state of Wisconsin or the US Centers for Disase Control and Prevention (CDC) National Death Index (NDI)
- NDI is death only; NDI Plus includes cause
- Of course, this costs money and cause costs more
- Typically, NDI/NDI Plus is one to two years behind
- Complete data for 2020 was released in 01/2022
- Early release for 2021 in 01/2022 was about 98% complete
- And the caveats about cause of death are substantial
- [US CDC NDI]

Anomylous Death Dates

- There appears to be an issue with the status of death and/or the death dates in the "Patient Demographics" table
- It is all relative, but the size of the "Vitals" table is much smaller than other tables such as "Encounters"
- Presumably, there would not be vital signs recorded on a date after someone has died
- So, we can create a relatively undemanding check of death dates by comparisons with their vital signs

Hands-on with Jupyterhub

```
## snippet11.R
zombies=dbGetQuery(db,
                   "select distinct a.patient_hash,
                    a.death_date_shifted,
                    a.vital_status_source
                    from fh_hb_demographics_jupyter a
                    inner join fh_hb_vitals_jupyter b
                    on a.patient_hash=b.patient_hash
                    AND a.death_date_shifted<
                    b.measure_date_shifted
                    order by patient_hash")
table(zombies$vital_status_source)
print(zombies)
```

Hands-on with Jupyterhub

Let's compare the last date of vital signs with death date

```
## snippet12.R
## ' these are single quotes
in.clause=paste0("('", paste(zombies$patient_hash,
                  collapse="','"), "')")
zdates=dbGetQuery(db,
        paste("select patient_hash, measure_date_shifted
                from fh_hb_vitals_jupyter
                where patient_hash in ", in.clause,
               "order by patient_hash,
                measure_date_shifted"))
max.zdates=c(tapply(zdates$measure_date_shifted,
                    zdates$patient_hash, max))
class(max.zdates)='POSIXct'
print(cbind(format(zombies$death_date_shifted),
            format(max.zdates)))
```

Medi-Span, GPI and RxNorm Medical Nomenclature

- [Medi-Span Generic Product Identifier (GPI)]
- The Wolters Kluwer Medi-Span brand database, called the Medispan Electronic Drug File, links the GPI code to other prescription drug classification codes
- [RxNorm] is part of Unified Medical Language System (UMLS) terminology maintained by the US National Library of Medicine (NLM)
- GPI and RxNorm codes are available on two CRDW tables
- "Medication Ordered" for medicinal prescriptions (not fills!): `fh_hb_med_orders_table_jupyter`
- "Medications Administered" for medicine given: `fh_hb_mar_table_jupyter`
- Example variables of interest
- `pharm_class`: pharmacologic class
- `pharm_subclass`: pharmacologic subclass
- `ingredient_rxcui_name`: RxNorm Concept Unique Identifier (CUI) name

Hands-on with Jupyterhub

How to make your own lookup table of drug nomenclature
N.B. RESOURCE INTENSIVE: JUST DON'T DO IT TODAY

```
## snippet13.R
medispan=dbGetQuery(db,
             "select distinct pharm_class, pharm_subclass,
              substring(gpi from 1 for 2) as gpi_group,
              substring(gpi from 3 for 2) as gpi_class,
              substring(gpi from 5 for 2) as gpi_subclass,
              ingredient_rxcui_name
              from fh_hb_mar_table_jupyter
              order by pharm_class, pharm_subclass,
              ingredient_rxcui_name")
write.csv(medispan,"medispan.csv",row.names=FALSE,na="")
```

Hands-on with Jupyterhub

What types of procedure codes do we have?
N.B. RESOURCE INTENSIVE: JUST DON'T DO IT TODAY

```
## snippet14.R
procs=dbGetQuery(db,
            "select px_type from fh_hb_procs_jupyter")
table(procs$px_type,useNA='ifany')/length(procs$px_type)

##        09        10        CH        OT      <NA>
## 0.001025 0.002704 0.879903 0.109712 0.006656
```

Hands-on with Jupyterhub

- What types of procedure codes do we have in the EHR?
- The Centers for Medicare and Medicaid Services (CMS) was an early entrant into the electronic billing space
- The Healthcare Common Procedural Coding System (HCPCS) was created by CMS for billing purposes
- HCPCS Level I codes are the [American Medical Association's Common Procedural Terminology fourth edition (CPT-4)] that typically have 5 digits (or occasionally 4 digits followed by a letter: see `snippet15.R`)
- [HCPCS Level II] codes start with a letter typically followed by 4 digits (see `snippet15.R`)

px_type	Coding system	Percentage
09	ICD-9	0.1%
10	ICD-10	0.3%
CH	HCPCS/CPT	88.0%
OT	Custom	11.0%
NA	Not available	0.7%

Cancer Research with RASMACRO

- We have already mentioned NCI SEER
- Another resource is NCI's [SEER-Medicare Linked Database]
- For example, [a resource for identifying CPT/HCPCS codes]
- [See this badly formatted list of breast cancer patient codes]
- Here, automated processing would be very challenging with R
- This is a job for ... RASMACRO!
- See the file `breastcodes.sas`

Cancer Research with RASMACRO

- For example, we have an EHR study of breast cancer survivors
- Which codes are for breast cancer treatment and which standard care?
- We need non-breast cancer men and women to compare with
- Let's contrast HCPCS codes with colorectal cancer patients
- What is left over are very likely breast cancer treatment
- Let's examine the codes billed to Medicare part A/B in 2016

Cancer Research with RASMACRO

CPT	Bills	Description
19380	1690	revision of reconstructed breast
19357	1448	breast reconstruction
19340	1281	immediate insertion of breast prosthesis
11970	1062	replacement of tissue expander with prosthesis
19371	1031	periprosthetic caspulectomy of breast
19342	1006	delayed insertion of breast prosthesis
19316	788	mastopexy
19283	717	placement of breast localization device
19328	660	removal of intact mammary implant
19350	638	nipple/areola reconstruction
19370	599	open periprosthetic caspulectomy of breast
93702	599	bioimpedance spectroscopy analysis for lymphedema
19499	520	unlisted breast procedure
77058	507	magnetic resonance imaging of the breast
19286	424	placement of breast localization device
81212	419	procedure for 185delAG, 5385insC, 6174delT variants

Cancer Research with RASMACRO

CPT	Bills	Description
19296	388	placement of radiotherapy catheter into the breast
11921	369	tattooing; 6.1 to 20 sq cm
38740	325	axillary lymphedectomy; superficial
19297	316	radiotherapy catether placement with partial masectomy
38790	295	lymphangiography injection
19086	279	breast biopsy of additional lesion
77469	273	intraoperative radiation treatment management
19126	252	excision of breast lesion identified by radiological marker
77424	252	intraoperative radiation delivery, single x-ray treatment
19361	239	breast reconstruction with latissimus dorsi flap
11971	223	removal of tissue expanders without prosthesis insertion
19330	223	removal of mammary implant material
19364	213	breast reconstruction with free flap
19325	179	mammaplasty augmentation with prosthetic implant
64462	174	paravertebral block injection including imaging guidance
11922	163	tattooing; each additional 20 sq cm

Cancer Research with RASMACRO

HCPCS	Bills	Description
L8000	816	Mastectomy bra
J9179	705	Eribulin mesylate injection
J9354	693	Inj, ado-trastuzumab emt 1mg
Q4116	427	Alloderm
L8600	335	Implant breast silicone/eq
L8030	330	Breast prosthes w/o adhesive
C1728	247	Cath, brachytx seed adm
C8906	245	Mri w/cont, breast, bi
C8905	239	Mri w/o fol w/cont, breast, un
C8907	171	Mri w/o cont, breast, bi
G8875	171	Breast cancer dx min invsive
J9207	134	Ixabepilone injection
C9726	116	Rxt breast appl place/remov
C9728	103	Place device/marker, non pro
J1950	88	Leuprolide acetate /3.75 mg
G8872	85	Intraop image confirm excise
C2639	72	Brachytx, non-stranded,i-125