Welcome to **axial**Healthcare

Statistics and Data Science Workflows in R

R Ladies

December 10, 2018

Outline of Talks

December 10, 2018

Database connections and querying with dplyr and data.table

– Amy Graves, Statistician

Creating attractive and informative map visualizations in R

– Mathilde Granke, Data Scientist

SME and RStudio Enterprise

– Meridith (Blevins) Peratikos, Statistician

Database connections and querying with dplyr and data.table

Amy Graves, Senior Healthcare Statistician

My Role at axialHealthcare

- Estimate per member per month savings attributed to axial products and services
- Typical tasks: data connection and data management, analysis, results presentation/visualization, documentation
- Constantly looking for ways to minimize data management
- This talk: connect to data stored in relational databases and manage/query data using dplyr and data.table

SQL Databases

- Relational databases (collection of data tables) - almost all SQL
- Cheap way to store a lot of data
- Examples: Postgres, Google BigQuery, Amazon Redshift…

R Packages

```r
library(rmarkdown) # for rmarkdown ppt slides
library(configr) # config file
library(raxial) # internal axial package of commonly-used functions
library(dplyr) # data manipulation
library(DBI) # database interface
library(RPostgreSQL) # Postgres connection
library(data.table) # faster data manipulation
#library(hflights) # domestic flights that departed houston in 2011
library(tictoc) # timing runs
```

Example Data

- Example code uses hflights data
- All commercial flights originating in Houston in 2011
- ~227,000 rows
- Stored on Postgres in zz_ag schema

Config File

- Purpose: store important things in one place & reference when needed
- Important things = database login info including user names and passwords, tokens, etc.
- Why? Secure way to store sensitive information so it is not stored in scripts and shared on places like github. Also, all team members have the same config file located in the same place so it's easy to collaborate (pull others' code from github, run, review)

```
config <- configr::read.config("~/.config/config.ini")
#file.edit("~/.config/config.ini")
```

Database Connection Using dplyr::src_postgres()

- You can only read data into R, not put it back on the database or alter it in any way

```
database <- "reporting_amerigroup"
con_dplyr <- dplyr::src_postgres(dbname  = database,
        host = config$postgres$host,
        port = config$postgres$port,
        user = config$postgres$username,
        password = config$postgres$password)
```

Database Connection Using DBI::dbConnect()

- You can read data into R, write back into SQL, alter in SQL, etc.
- Preferred for flexibility

```
con_dbi <- DBI::dbConnect(RPostgreSQL::PostgreSQL(),
         dbname = database,
         host = config$postgres$host,
         port = config$postgres$port,
         user = config$postgres$username,
         password = config$postgres$password)
```

Database Connection Using raxial::postgres_con()

- raxial is our internal axial r package containing common functions, including database connections
- A way for all team members to source and use common functions
- Fewer lines of code :)

```
con_dbi <- raxial::postgres_con(database)
```

Navigating Around Postgres

```r
# view all schemas within a database
DBI::dbGetQuery(con_dbi, "SELECT nspname FROM pg_catalog.pg_namespace") %>%
dplyr::arrange(nspname)

# view all tables within a schema
schema = "zz_ag"
DBI::dbGetQuery(con_dbi,
          paste0("SELECT table_name FROM information_schema.tables
                 WHERE table_schema='", schema, "'")) %>%
dplyr::arrange(table_name)
```

Connect to hflights Data in Database and read into R memory

- Use dplyr and use database connection to connect to hflights data in Postgres
- Collect() all data in memory in R
- Pulling in all quarter million records and columns into R. Not efficient for memory!

```
tic()
hflights_r <- dplyr::tbl(con_dbi, sql("SELECT * FROM zz_ag.hflights")) %>%
  collect()
toc() # 2.35 sec
```

Manipulate (query) hflights Data in R Memory using dplyr

- Average departure and arrival delays for all flights in June
- dplyr: each verb is a command; chain together commands with pipes
- Common verbs: select (select columns), filter (subset), mutate (create/recode new variables), group_by (group by before calculating something on the data), summarise (calculation on the data), arrange (order by values in a column)

Query hflights Data in R Memory Using dplyr

Continued...

```r
tic()
hflights_query1 <-
 hflights_r %>%
filter(Month == 6) %>%
group_by(Origin) %>%
summarise(mean_dep_delay = mean(DepDelay, na.rm = TRUE),
          mean_arr_delay = mean(ArrDelay, na.rm = TRUE))
toc()
# 0.06 sec of query + 2.35 sec to pull in memory = 2.41 sec.
```

Query hflights Data in R Memory Using dplyr

Continued...

```
print(hflights_query1)
## # A tibble: 2 x 3
##   Origin mean_dep_delay mean_arr_delay
##   <chr>           <dbl>          <dbl>
## 1 HOU             14.6           9.92
## 2 IAH             11.5           11.1
```

Query hflights Data in Database Using dplyr

- Same query as before: average departure and arrival delays for all flights in the month of June
- Data (227k rows) is still in database until collect() statement pulls it into R memory
- All data querying done in the database before pulling into R (!!!)

Query hflights Data in Database Using dplyr

Continued...

```
tic()
hflights_query2 <- tbl(con_dbi, sql("SELECT * FROM zz_ag.hflights")) %>%
  filter(Month == 6) %>%
  group_by(Origin) %>%
  summarise(mean_dep_delay = mean(DepDelay, na.rm = TRUE),
            mean_arr_delay = mean(ArrDelay, na.rm = TRUE)) %>%
  collect()
toc() # 0.11 sec
```

Query hflights Data in Database Using dplyr

Continued...

```
print(hflights_query2)
## # A tibble: 2 x 3
##   Origin mean_dep_delay mean_arr_delay
## * <chr>           <dbl>          <dbl>
## 1 HOU             14.6           9.92
## 2 IAH             11.5           11.1
```

Show SQL Code that dplyr is Using

```
show_query(hflights_query2)

<SQL>
SELECT "Origin", AVG("DepDelay") AS "mean_dep_delay", AVG("ArrDelay") AS
"mean_arr_delay"
FROM (SELECT * FROM zz_ag.hflights) "ottdqfhojz"
WHERE ("Month" = 6.0)
GROUP BY "Origin"
```

Join Data Using dplyr

- Many joins available in dplyr
- If you have data in different schemas, you can still join them
- If you have data in different databases, you can still join them! Though not efficient because it copies data on to the same database before joining

Join Data Using dplyr

Continued...

```
tic()
con_gbq <- raxial::gbq_con(project="axial-research",
                           dataset="research_agraves")
origin_ref <- dplyr::tbl(con_gbq, sql("SELECT * FROM research_agraves.origin_ref"))
hflights_query3 <-
  tbl(con_dbi, sql("SELECT * FROM zz_ag.hflights"))  %>%
  filter(Month == 6) %>%
  group_by(Origin) %>%
  summarise(mean_dep_delay = mean(DepDelay, na.rm = TRUE),
            mean_arr_delay = mean(ArrDelay, na.rm = TRUE)) %>%
  inner_join(origin_ref, by="Origin", copy = TRUE)
toc() # 3.1 sec
```

Join Data Using dplyr

Continued...

```
print(hflights_query3)
## # Source:   lazy query [?? x 3]
## # Database: postgres 9.6.7 [agraves@10.4.3.238:6432/reporting_amerigroup]
##   Origin mean_dep_delay mean_arr_delay
##   <chr>           <dbl>          <dbl>
## 1 IAH              11.5           11.1
```

Query Data Using data.table

Efficient & Fast on big data!

- Data must be in memory in R
- Convert data.frame into data.table
- Syntax is DT[i, j, by]
- Take data table DT, subset rows using ‘i’, then calculate ‘j’, grouped by ‘by’
- Note: fread() very quickly imports .csv files into a data.table

Query Data Using data.table

Continued...

```
tic()
hflights_dt <- data.table(hflights_r)
hflights_query4 <- hflights_dt[Month == 6L,
                               .(mean_dep_delay = mean(DepDelay, na.rm=TRUE),
                                 mean_arr_delay = mean(ArrDelay, na.rm=TRUE)),
                               Origin]
toc() #0.03 sec
```

Query Data Using data.table

Continued...

```
print(hflights_query4)
##  Origin mean_dep_delay mean_arr_delay
## 1:  HOU     14.55768       9.922661
## 2:  IAH     11.47191      11.108677
```

More Complex Query in dplyr

- Which carrier has the lowest departure and arrival delays in each origin airport, among carriers that have at least 1000 flights to an origin airport?
- Again, doing all the querying in postgres database before collecting the result into R

More Complex Query in dplyr

Continued...

```
tic()
hflights_query5 <-
  tbl(con_dbi, sql("SELECT * FROM zz_ag.hflights"))  %>%
  group_by(UniqueCarrier, Origin) %>%
  summarise(mean_dep_delay = mean(DepDelay, na.rm = TRUE),
            mean_arr_delay = mean(ArrDelay, na.rm = TRUE),
            n_flights = n()) %>%
  filter(n_flights >= 1000) %>%
  group_by(Origin) %>%
  filter(rank(mean_dep_delay) == 1 | mean_arr_delay == 1) %>%
  rename(min_dep_delay = mean_dep_delay,
         min_arr_delay = mean_arr_delay) %>%
  mutate(unique_carrier_name = if(UniqueCarrier=="FL"){"Air Florida"}
                              else if(UniqueCarrier=="US"){"US Airways"},
         origin_name = if(Origin=="HOU"){"Houston Hobby Airport"}
                      else if(Origin=="IAH"){"Houston Intercontinental Airport"}) %>%
  ungroup() %>%
  select(-c(UniqueCarrier, Origin)) %>%
  collect()
toc() # 0.29 sec
```

More Complex Query in dplyr

Continued...

```
print(hflights_query5[c("origin_name","unique_carrier_name","min_dep_delay","min_
arr_delay","n_flights")])
## # A tibble: 2 x 5
##   origin_name        unique_carrier_… min_dep_delay min_arr_delay n_flights
##   <chr>             <chr>                     <dbl>         <dbl>     <dbl>
## 1 Houston Hobby Ai… Air Florida                4.72         1.85       2139
## 2 Houston Intercon… US Airways                 1.62        -0.631      4082
```

Same Query in data.table()

- Which carrier has the lowest departure and arrival delays in each origin airport, among carriers that have at least 1000 flights to an origin airport?
- Can chain together commands in data.table by using multiple sets of brackets together (like dplyr chains commands using pipes)

Same Query in data.table()

Continued...

```
tic()
hflights_query6 <-
  hflights_dt[,
    .(mean_dep_delay=mean(DepDelay, na.rm = TRUE),
      mean_arr_delay=mean(ArrDelay, na.rm = TRUE),
      n_flights = .N),
    by=.(UniqueCarrier, Origin)][n_flights >= 1000,
    rank := frank(mean_dep_delay),
    by=.(Origin)][rank==1,
      ,
      ][,
    UniqueCarrier := c(FL="Air Florida", US="US Airways")[UniqueCarrier], ][,
    Origin := c(HOU="Houston Hobby Airport", IAH="Houston Intercontinental Airport")[Origin], ]
setnames(hflights_query6, old = c("UniqueCarrier","Origin"), new = c("unique_carrier_name","origin_name"))
toc()  #0.05 sec
```

Same Query in data.table()

Continued...

```
print(hflights_query6)
## unique_carrier_name     origin_name mean_dep_delay   mean_arr_delay n_flights rank
## 1:       US Airways  Houston Intercon...    1.622926   -0.6307692          4082    1
## 2:      Air Florida   Houston Hobby Ai...    4.716376   1.8536239          2139    1
```

Takeaways

Workflow takeaways

- config file makes storing sensitive information and sharing code safer and easier
- A package of commonly-used functions (like raxial) makes group knowledge base shareable

Database connection takeaways

- DBI connection to database allows you to not only read data from a database into r, but also manipulate and write data back on the database. Use this for most flexibility

Takeaways

dplyr takeaways

- dplyr has readable, pipeable syntax
- dplyr can connect to database and run everything on database before pulling in to R memory via collect(); efficient in memory
- dplyr can connect to many different kinds of databases and run the SQL dialect particular to the database
- dplyr can show sql queries that were run on database using show_query()
- dplyr can join tables from different databases (e.g. table on postgres, table on GBQ). use copy = TRUE when joining

Takeaways

data.table takeaways

- Most things you can do in dplyr you can also do in data.table
- data.table is faster/more efficient for extremely large data, ~10 million+ rows
- data.table can be chained together like dplyr pipes using multiple brackets
- data.table quick to read in .csv files via fread()
- To my knowledge, data.table does not have a convenient connection to databases and works only in memory

Resources

- Database connections and dplyr: https://db.rstudio.com/dplyr/
- dplyr: https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html
- data.table basics: https://s3.amazonaws.com/assets.datacamp.com/blog_assets/datatable_Cheat_Sheet_R.pdf
- advanced data.table: http://brooksandrew.github.io/simpleblog/articles/advanced-data-table/

Creating attractive and informative map visualizations in R

Mathilde Granke, Senior Research Data Scientist

What do I do at Axial?

- Identify the best providers to refer better care

Assess provider patterns and identify advanced pain management and medication-assisted treatment providers to direct patients to the highest quality of care at the most appropriate cost

Typical workflow

import data → data wrangling → data analysis & modeling → data visualization

1. efficient communication
2. reproducibility
3. production

connect to Postgres, Google BigQuery, Redshift

dplyr, data.table

lme4, stats, caret

leaflet

Objective of this Talk

- Produce a map of TN displaying providers and county-level information about the population

Steps

1. Import data (public)
2. Data analysis
3. Map

Step 1

Import Data

- Ranking in health factors (source: http://www.countyhealthrankings.org/)

```
# http://www.countyhealthrankings.org/
#The overall rankings in health factors represent what influences the health of a county.
# They are an estimate of the future health of counties as compared to other counties within a state.
# The ranks are based on four types of measures: health behaviors, clinical care, social and economic,
# and physical environment factors.
library(readxl)
tn_county_ranking <- read_excel("2018 County Health Rankings Tennessee Data.xls",
                                sheet = "Outcomes & Factors Rankings")

tn_county_ranking <- tn_county_ranking[-c(1:2),c(1,7)]
names(tn_county_ranking) <- c("fips","health_factors_rank")
tn_county_ranking$fips <- as.integer(tn_county_ranking$fips)
tn_county_ranking$health_factors_rank <- as.numeric(tn_county_ranking$health_factors_rank)
```

Step 1

Import Data

- County borders (latitude, longitude)

```
# A database matching FIPS codes to maps package county and state names
library(maps)
data(county.fips)

# Join county name and health rank on FIPS
tn_data <- left_join(county.fips, tn_county_ranking, by = 'fips') %>%
  filter(str_detect(polyname, "tennessee") == TRUE) %>%
  select(fips,
         health_factors_rank,
         polyname)

tn_map <- maps::map("county", region = "tennessee", fill = TRUE, plot = FALSE)
tn_map$fips <- tn_data$fips[match(map("county", region = "tennessee", plot=FALSE)$names, tn_data$polyname)]
tn_map$health_factors_rank <- tn_data$health_factors_rank[match(map("county", region = "tennessee", plot=FALSE)$names, tn_data$polyname)]
```

Step 1

Import Data

- Medication-assisted-treatment providers

  #https://www.samhsa.gov/medication-assisted-treatment/physician-program-data/treatment-physician-locator

```
# MAT providers in TN
#https://www.samhsa.gov/medication-assisted-treatment/physician-program-data/treatment-physician-locator

# import data
provider_data <- read.csv("Physician_Locator_2018-12-04.csv", header=TRUE, stringsAsFactors=FALSE,  encoding="UTF-8", sep=",")
# columns names
names(provider_data) <- c('title', 'first_name','last_name','degree','address','city','county','state','zip','tel','fax')
# retrieve full address for geocoding
provider_data <- provider_data %>%
  mutate(full_address = paste(address, city, state, zip))
```

Step 2

Data Analysis

- Geolocation

```r
# Geocode address
library(googleway)
#define a function that will process googles server responses
getGeoDetails <- function(address){
  #use the gecode function to query google servers
  geo_reply = google_geocode(address, key = api_key)
  answer <- data.frame(formatted_address=NA, lat = NA, long=NA)
  #return Na's if we didn't get a match:
  if (geo_reply$status != "OK"){
    answer$full_address <- provider_data$full_address[ii]
    return(answer)
  }
  answer$formatted_address <- geo_reply$results$formatted_address[1]
  answer$lat <- geo_reply$results$geometry$location[1,]$lat
  answer$long <- geo_reply$results$geometry$location[1,]$lng
  return(answer)
}

geocoded <- data.frame(full_address = character(),
                       formatted_address = character(),
                       lat = numeric(),
                       long = numeric())
for (ii in 1:length(provider_data$full_address)){
  result <- getGeoDetails(provider_data$full_address[ii])
  result$full_address <- provider_data$full_address[ii]

  print(paste("geocode address", ii, "/", length(provider_data$full_address), result$formatted_address))
  geocoded <- rbind(geocoded, result)
}
```

Step 2

Data Analysis

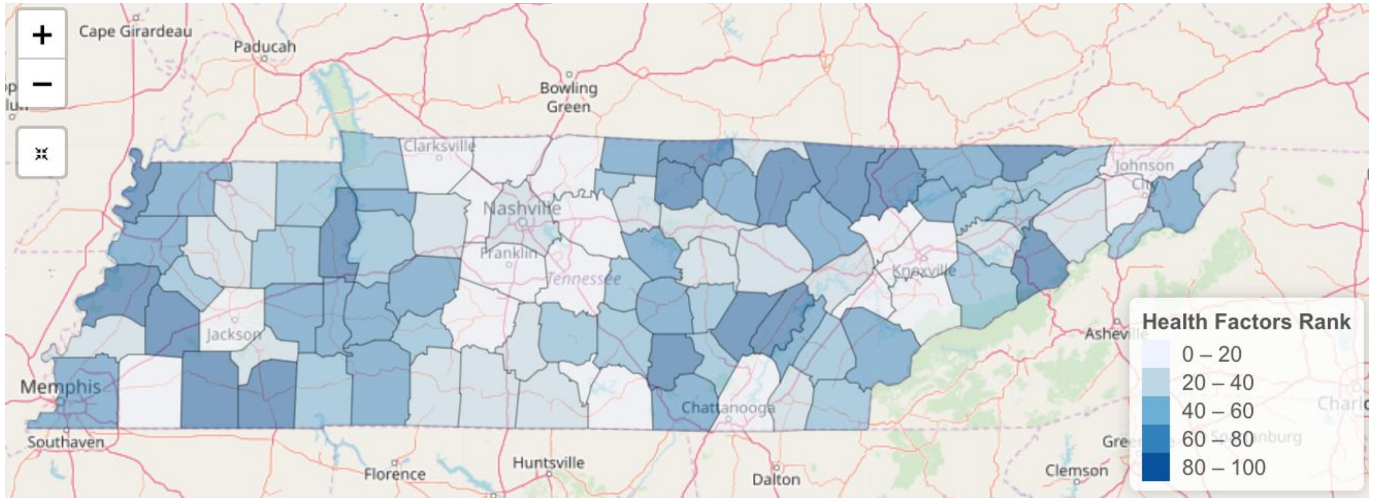
- join providers address to coordinates + assign quality of care *(random for the purpose of this exercise)*

```r
# Add latitude, longitude, formatted address
provider_data <- left_join(provider_data, geocoded, by = 'full_address') %>%
  filter(!is.na(formatted_address)) %>%
  mutate_all(funs(toupper)) %>%
  mutate(Name = paste(first_name, last_name),
         Phone = paste0('(',substr(tel,1,3),') ',substr(tel,5,12))) %>%
  select(Name,
         Address = formatted_address,
         Phone,
         lat,
         long) %>%
  distinct() %>%
  mutate(popup_info = paste(sep = "<br/>", Name, Address, Phone))

# Assign random quality assessment
quality_categories <- c("High quality providers", "Average quality providers", "Low quality providers")
provider_data$assessment <- sample(quality_categories, nrow(provider_data), replace = TRUE)
provider_data <- provider_data %>%
  mutate(clr_marker = case_when(
    assessment == "High quality providers" ~ '#08B491',
    assessment == "Average quality providers" ~ '#FCB33A',
    assessment == "Low quality providers" ~ '#D9484C'
  ))
```

Step 3

Map

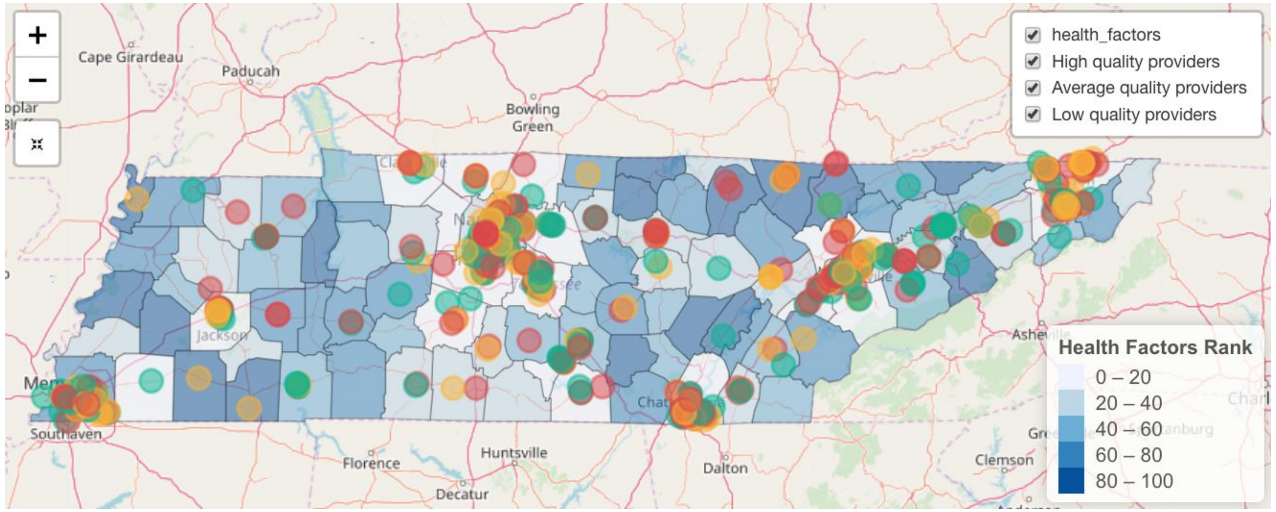
```r
library(leaflet)
# assign color to counties based on rank
mypal <- colorBin('Blues',
                  domain = tn_data$health_factors_rank,
                  bins = 5)

leaflet() %>%
  addTiles() %>%              #Add default OpenStreetMap map tiles
  addResetMapButton() %>%     #Add zoom reset button
  # Population per county
  addPolygons(data = tn_map,
            lat = ~y, lng = ~x, # coord of county border
            fillColor = ~mypal(health_factors_rank),
            fillOpacity = 0.5,
            smoothFactor = 0.5,
            stroke = TRUE,
            color = "#000",
            weight = 0.5,
            group = 'health_factors') %>%
  # Add legend
  addLegend(position = "bottomright",
          pal =  mypal,
          values = ~health_factors_rank,
          title = "Health Factors Rank",
          opacity = 1,
          data = tn_map,
          group = 'health_factors') %>%
```



Step 3

Map

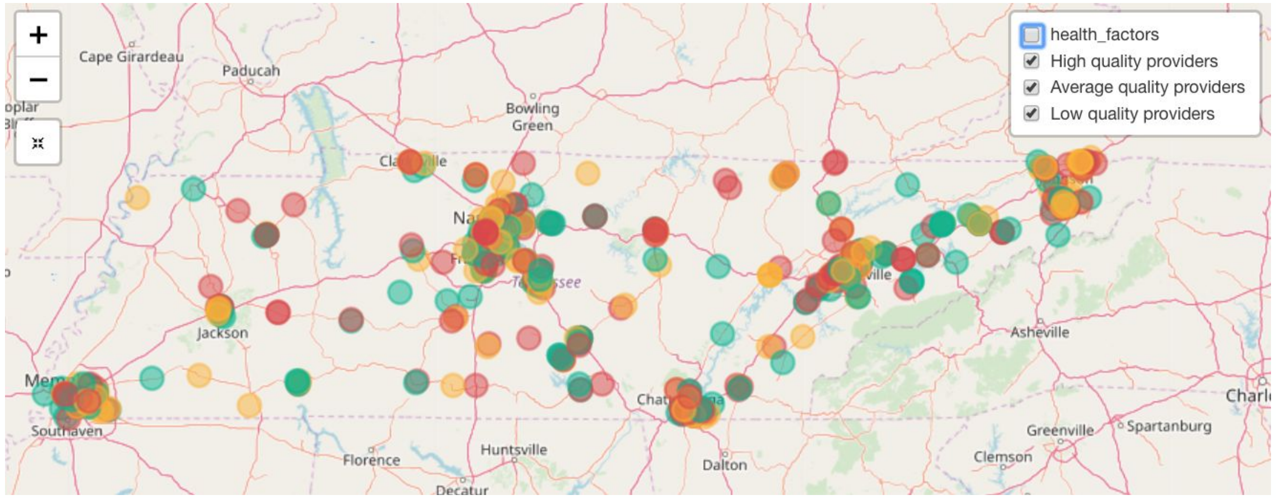
```r
# Add provider
addCircleMarkers(data = provider_data,
                 lng = ~as.numeric(long),
                 lat = ~as.numeric(lat),
                 radius = 8,
                 color = ~clr_marker,
                 fillOpacity = 0.5,
                 stroke = TRUE,
                 weight = 2,
                 popup = ~popup_info,
                 options = popupOptions(maxWidth = 800),
                 #clusterOptions = markerClusterOptions(),
                 group = ~assessment) %>%
# Layers control
addLayersControl(
  overlayGroups = c("health_factors",
                    "High quality providers",
                    "Average quality providers",
                    "Low quality providers"),
  options = layersControlOptions(collapsed = FALSE),
  position = "topright"
)
```



Step 3

Map

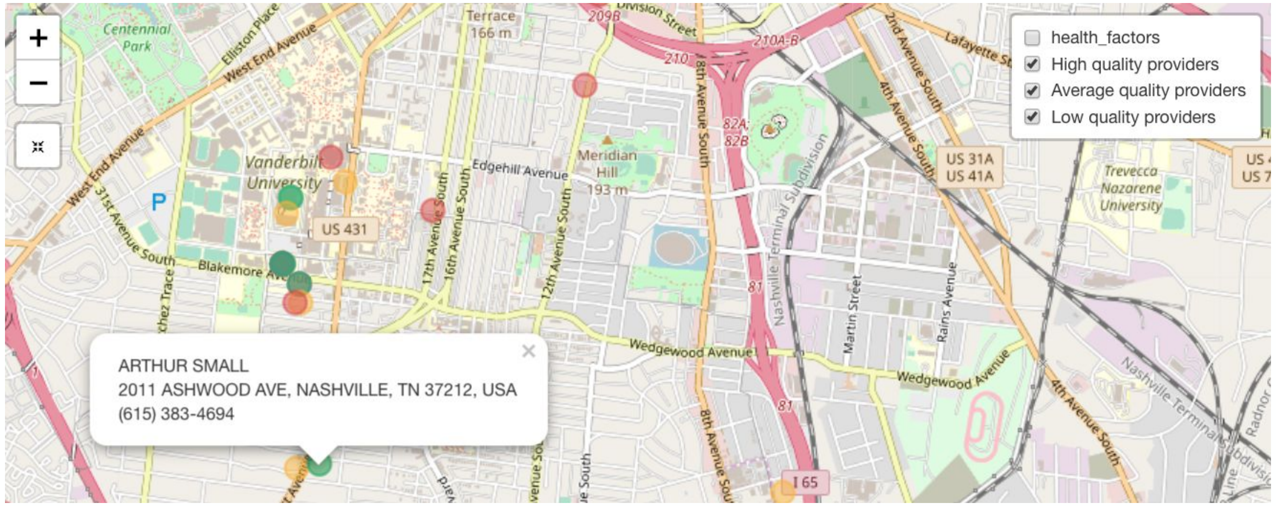
```r
# Add provider
addCircleMarkers(data = provider_data,
                 lng = ~as.numeric(long),
                 lat = ~as.numeric(lat),
                 radius = 8,
                 color = ~clr_marker,
                 fillOpacity = 0.5,
                 stroke = TRUE,
                 weight = 2,
                 popup = ~popup_info,
                 options = popupOptions(maxWidth = 800),
                 #clusterOptions = markerClusterOptions(),
                 group = ~assessment) %>%
# Layers control
addLayersControl(
  overlayGroups = c("health_factors",
                    "High quality providers",
                    "Average quality providers",
                    "Low quality providers"),
  options = layersControlOptions(collapsed = FALSE),
  position = "topright"
)
```



Step 3

Map

```r
# Add provider
addCircleMarkers(data = provider_data,
                 lng = ~as.numeric(long),
                 lat = ~as.numeric(lat),
                 radius = 8,
                 color = ~clr_marker,
                 fillOpacity = 0.5,
                 stroke = TRUE,
                 weight = 2,
                 popup = ~popup_info,
                 options = popupOptions(maxWidth = 800),
                 #clusterOptions = markerClusterOptions(),
                 group = ~assessment) %>%
# Layers control
addLayersControl(
  overlayGroups = c("health_factors",
                    "High quality providers",
                    "Average quality providers",
                    "Low quality providers"),
  options = layersControlOptions(collapsed = FALSE),
  position = "topright"
)
```

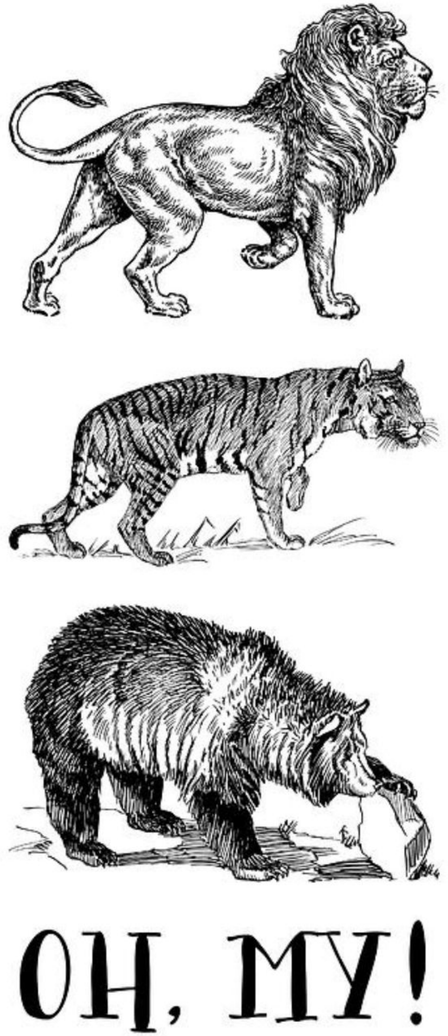


SME and RStudio Enterprise

Meridith Peratikos
Director of Statistics and Scientific Collaboration

SME and Rstudio Enterprise - Oh My!

- Subject matter expertise (SME) is critical to any statistics and data science workflow
- RStudio Enterprise tools are useful for a reproducible and scalable statistics and data science workflow
    - RStudio Server Pro
    - RStudio Connect



Developing Subject Matter (or Domain) Expertise

- PubMed or Google Scholar alerts
- Professional Society Membership
- Attend Conferences
- Talk to People

How does this benefit others across the organization?

[EXTERNAL] What's new for 'opioid' in PubMed

My NCBI <efback@ncbi.nlm.nih.gov> Unsubscribe
to me

Wed, Dec 5, 6:27 AM (4 days ago)

This message contains My NCBI what's new results from the National Center for Biotechnology Information (NCBI) at the U.S. National Library of Medicine (NLM).
Do not reply directly to this message.

Sender's message: Here is the digest "opioid" articles:

Sent on Wednesday, 2018 December 05
Search: **opioid**

View complete results in PubMed (results may change over time).

Edit saved search settings, or unsubscribe from these e-mail updates.

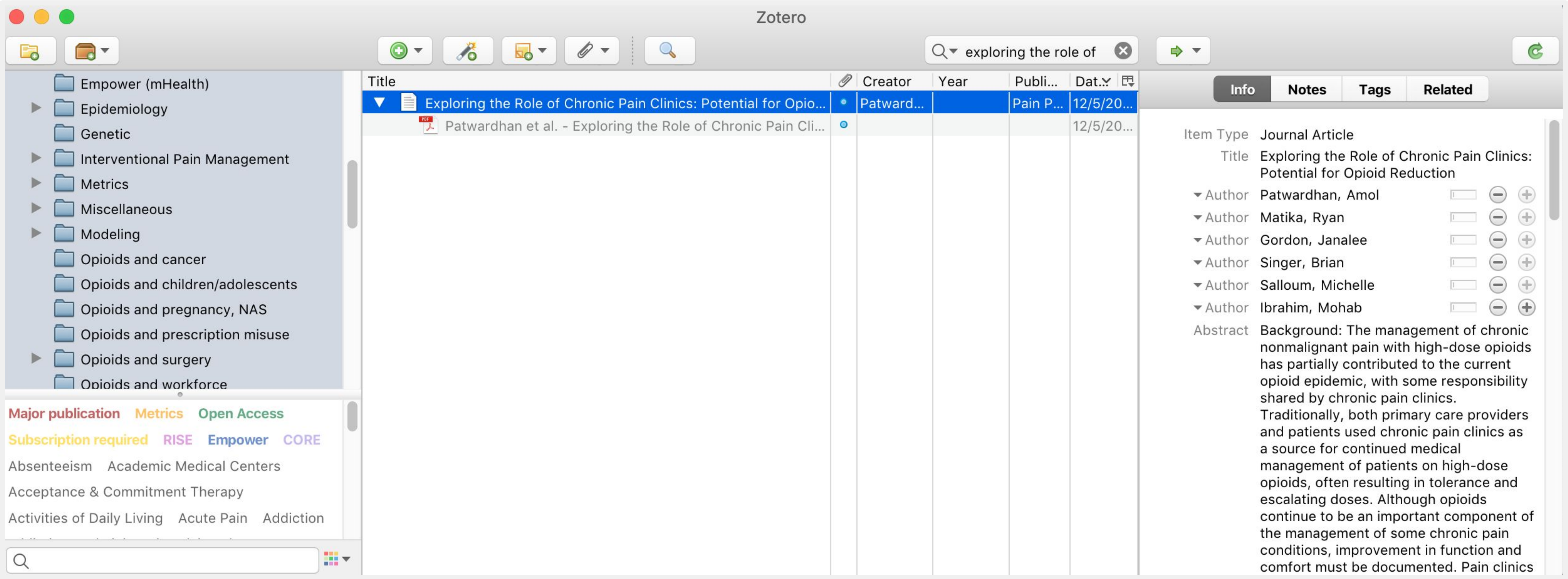
PubMed Results

Items 1 - 20 of 240

1. Prevalence and patterns of opioid misuse and opioid use disorder among primary care patients who use tobacco.
   John WS, Zhu H, Mannelli P, Subramaniam GA, Schwartz RP, McNeely J, Wu LT.
   Drug Alcohol Depend. 2018 Nov 26;194:468-475. doi: 10.1016/j.drugalcdep.2018.11.011. [Epub ahead of print]
   PMID: 30513477 [PubMed - as supplied by publisher]
2. Biochemical and pharmacological investigation of novel nociceptin/OFQ analogues and N/OFQ-RYYRIK hybrid peptides.
   Erdei AI, Borbély A, Magyar A, Szűcs E, Ötvös F, Gombos D, Al-Khrasani M, Stefanucci A, Dimmito MP, Luisi G, Mollica A, Benyhe S.
   Peptides. 2018 Dec 1. pii: S0196-9781(18)30242-0. doi: 10.1016/j.peptides.2018.11.010. [Epub ahead of print]
   PMID: 30513351 [PubMed - as supplied by publisher]
3. Educational lectures enhance knowledge about the human immunodeficiency virus (HIV) and reduce risky behavior and fear among methadone maintenance treatment patients.
   Barak I, Adelson M, Sason A, Livnat Y, Schreiber S, Peles E.
   Subst Abus. 2018 Dec 4:1-5. doi: 10.1080/08897077.2018.1528492. [Epub ahead of print]
   PMID: 30513276 [PubMed - as supplied by publisher]

19. Exploring the Role of Chronic Pain Clinics: Potential for Opioid Reduction.
    Patwardhan A, Matika R, Gordon J, Singer B, Salloum M, Ibrahim M.
    Pain Physician. 2018 Nov;21(6):E603-E610.
    PMID: 30508991 [PubMed - in process]
20. Evaluation of Primary Care Physician Chronic Pain Management Practice Patterns.
    Provenzano DA, Kamal KM, Giannetti V.
    Pain Physician. 2018 Nov;21(6):E593-E602.
    PMID: 30508990 [PubMed - in process]

Building a reference library



Disseminating weekly updates of references

Zotero Library Weekly Update as of Friday, December 7th

Meridith Peratikos <mperatikos@axialhealthcare.com>
to dlaxiallibrary

Fri, Dec 7, 9:38 AM (2 days ago)

Publish to Promote

In 2019, axialHealthcare will launch our publish to promote initiative. For a preview of this initiative and a snapshot of our current publish to promote activity, check out [Publish to Promote on Confluence].

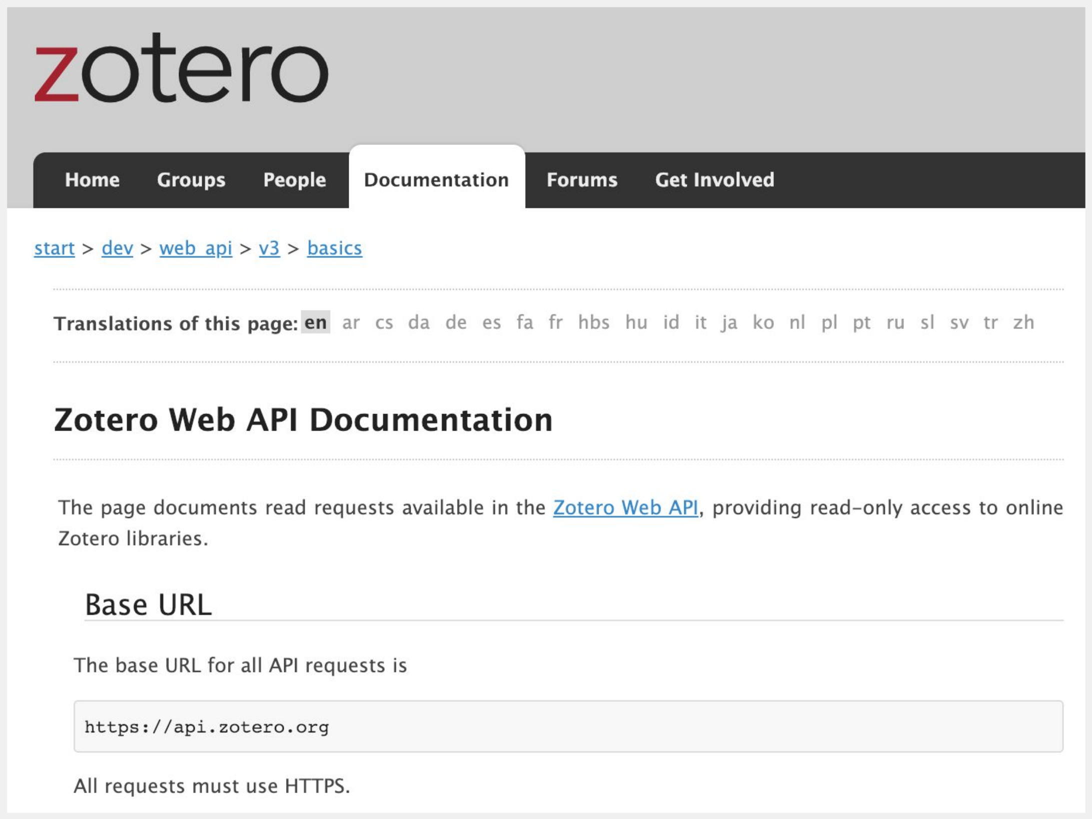
Zotero Library Weekly Update

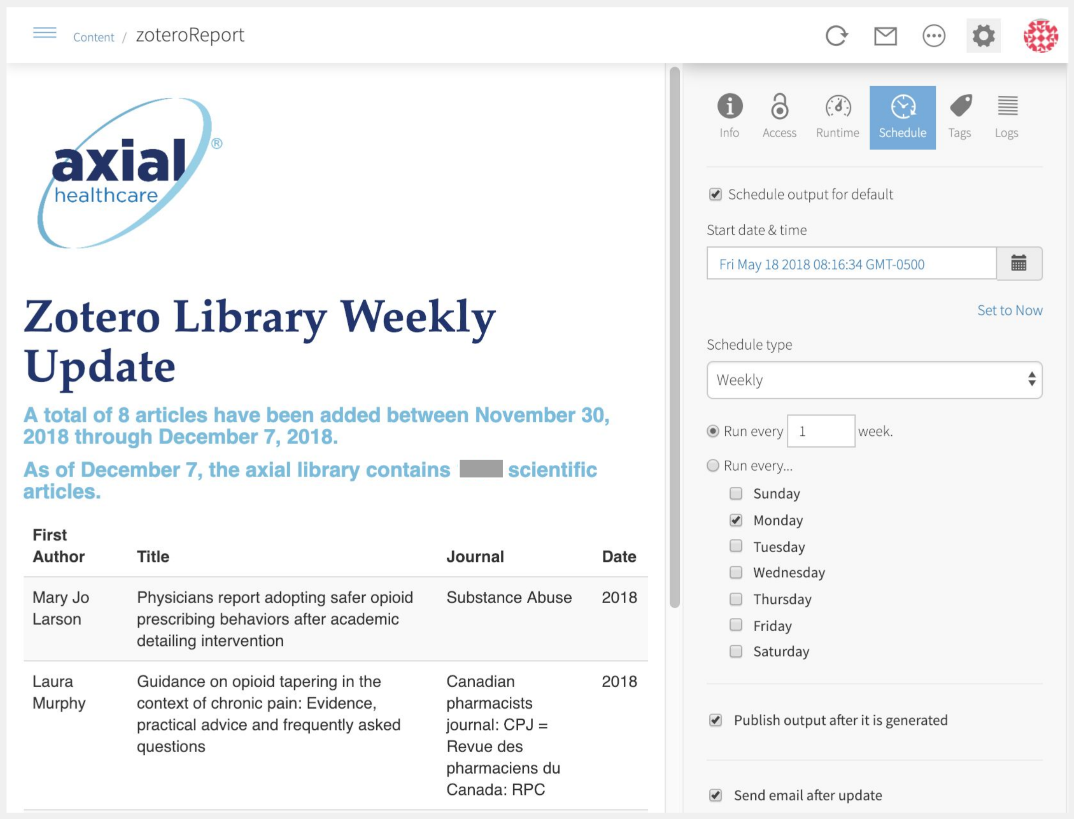
A total of 8 articles have been added between November 30, 2018 through December 7, 2018.

As of December 7, the axial library contains [redacted] scientific articles.

First Author	Title	Journal	Date
Mary Jo Larson	Physicians report adopting safer opioid prescribing behaviors after academic detailing intervention	Substance Abuse	2018
Laura Murphy	Guidance on opioid tapering in the context of chronic pain: Evidence, practical advice and frequently asked questions	Canadian pharmacists journal	2018
Beth D. Darnall	Patient-Centered Prescription Opioid Tapering in Community Outpatients With Chronic Pain	JAMA internal medicine	2018
Deborah Dowell	CDC Guideline for Prescribing Opioids for Chronic Pain–United States, 2016	JAMA	2016
Amol Patwardhan	Exploring the Role of Chronic Pain Clinics: Potential for Opioid Reduction	Pain Physician	2018
Gualberto Ruaño	Fundamental Considerations for Genetically- Guided Pain Management with Opioids Based on CYP2D6 and OPRM1 Polymorphisms	Pain Physician	2018
Nitika Sanger	Association Between Socio-Demographic and Health Functioning Variables Among Patients with Opioid Use Disorder Introduced by Prescription: A Prospective Cohort Study	Pain Physician	2018

Behind the scenes of weekly update





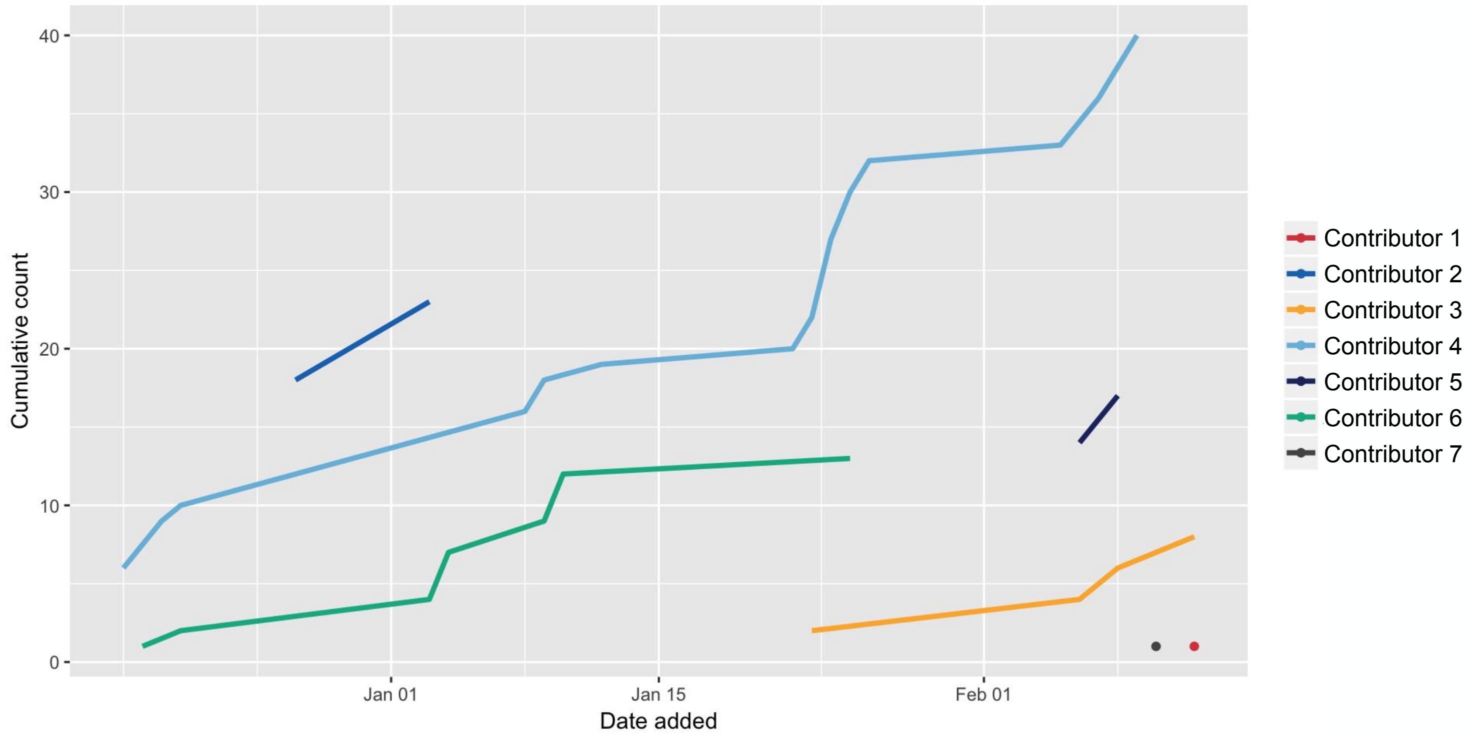
Jacquelyn Neal (R&D intern) wrote an Rmarkdown report.
It sourced axial's reference library from Zotero's API using RCurl, httr, jsonlite, purrr, readr packages.
The report is published to Rstudio Connect which generates weekly email updates.

Related reference library outputs

Since we had the data, Jacquelyn and I generated some useful reports and visualizations.

- Individual library contribution report
- Word Cloud for blog post (wordcloud2 package)
- Governance report (not pictured)

Figure: Cumulative article contributions over past 2 months

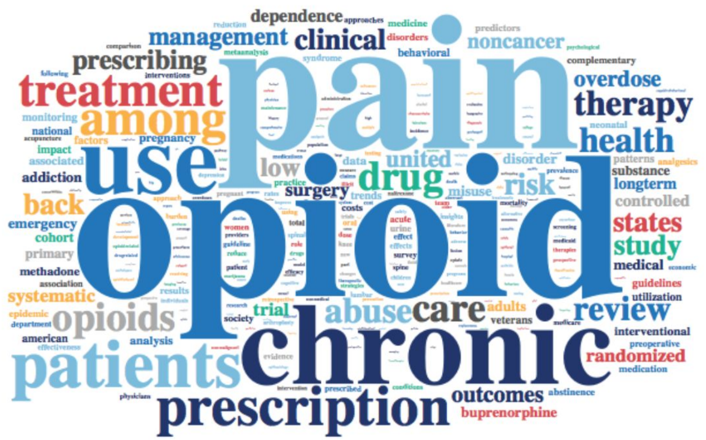


Opioid, Pain & Chronic: How 3 Words Can Impact Our Solutions

Expertise, Latest Posts | December 21, 2017

In order to successfully offer clinicians and patients the highest quality pain care resources and tools, our solutions need to be informed by the latest evidence-based research. To do this, our data science and analytics team curates an extensive library of relevant literature, which is then used to define our methodology and inform capabilities like proprietary care pathways for pain care and opioid therapy.

Using titles from over 850 articles in the library, our data science and analytics team developed a word cloud to highlight the most frequently used words. The top three most used words? ***Opioid***, ***Pain*** and ***Chronic***. Not far behind in frequency are ***Use***, ***Treatment***, ***Patients*** and ***Prescription***. These terms offer a unique and accurate glimpse of the past year at axialHealthcare.

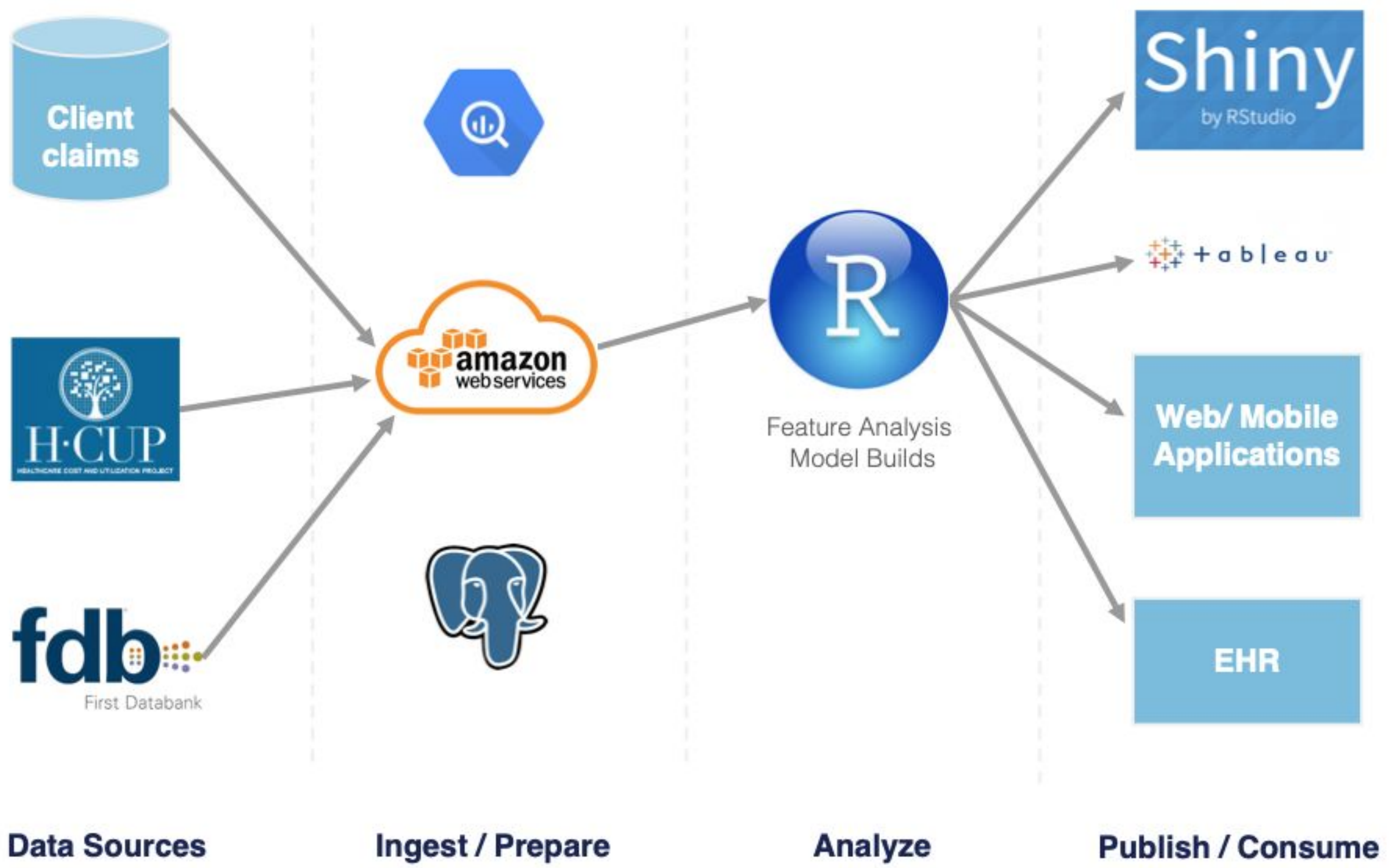


Sharing Subject Matter (Domain) Expertise

- PubMed or Google Scholar alerts [build library]
- Professional Society Membership [build library and contacts]
- Attend Conferences [lunch & learn]
- Talk to People [build contacts]

It is in your best interest to develop SME across the organization.

Enhancing our Workflow with Rstudio Server Pro & Connect



RStudio Server Pro

- Run multiple sessions and projects simultaneously
- Execute long running scripts that do not affect local machine
- Better performance on the server (compared with your macbook)
- Live collaboration tools in shared session (think google docs)
- Access your code in any browser without cloning repos
- Stable dev environment - not susceptible to OS X updates
- Identical to axial’s R production environment
- Backend drivers for easier database connections
- Work safely with databases while traveling abroad

RStudio Connect

- Rmarkdown reports or shiny apps
- Restrict access
- Vanity URL (Fave)
- Schedule reports with/without email delivery
- Cannot track usage statistics (yet?)

RStudio Connect

https://rstudioconnect.axialistas.com/pain-opioid-usa/

Content / bhi-maps

mperatikos

Pain and opioids in the United States commercial coverage population

The data source is a random sample of commercially covered Blues members enrolled during a one year period given by July 1, 2016 through June 30, 2017.

Select Region and Indicator

Region:

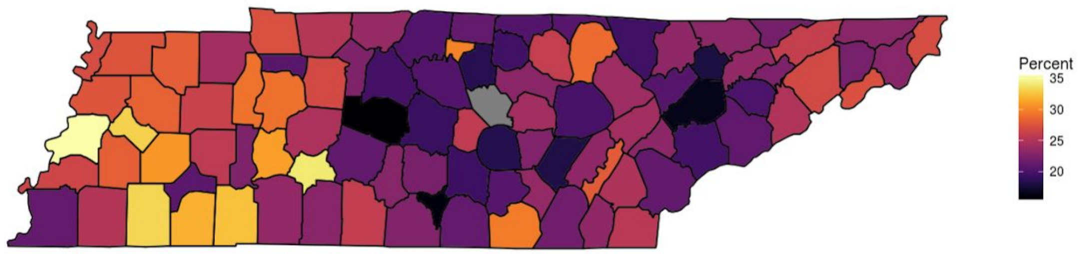
Tennessee

Indicator:

% with opioid Rx

	Tennessee	USA
Member count	154,046	5,692,251
% with pain condition	47.8%	47.3%
% with chronic pain	17.1%	16.8%
% with opioid Rx	21.7%	17.5%
% with chronic opioid treatment	2.8%	1.9%
% with severe risk of OUD	0.95%	0.66%
% with diagnosed OUD	0.90%	0.68%
% with opioid overdose	0.02%	0.03%

% with opioid Rx



Info Access Runtime Schedule Tags Logs

Who can view this application

Everyone

Who can change this application

Meridith Peratikos mperatikos

Add collaborator

Who runs this application on the server

The default user rstudio-connect

Administrators can create a custom "vanity" url to access this application. Your custom URL will be appended to your domain to form the complete path to your application.

/pain-opioid-usa/

Your custom url:

https://rstudioconnect.axialistas.com/pain-opioid-usa/

An Rmarkdown & Shiny App with Open Source Data

- R Markdown report:
  http://rpubs.com/mayjh/drug_prescription

- Shiny app:
  https://mayjh.shinyapps.io/medicare_part_d_drug_prescription__overdose/

- Code:
  https://github.com/mayjh/r-ladies-demo

Biographies

Amy J. Graves, MS, MPH
Senior Healthcare Statistician, axialHealthcare
Amy's role in Medical Economics ia dedicated to estimating healthcare savings. She specializes in causal inference methods using observational data.

Mathilde Granke, PhD
Senior Research Data Scientist, axialHealthcare
Finding and sharing the stories hiding in data is Mathilde's passion. Her expert scientific understanding is applied to the largest pain database in healthcare to gain insight into pain management and opioid prescription practices.

Meridith (Blevins) Peratikos, MS
Director of Statistics and Scientific Collaboration, axialHealthcare
Adjunct Faculty, Department of Biostatistics, Vanderbilt University Medical Center
A career researcher committed to contributing to an evidence-based understanding of high impact healthcare problems, Meridith leads a team of scientists that conducts research studies, methodological oversight, and literature review.

Jianhong (May) Shen, PhD (contributed Shiny app from open data)
Research Data Scientist, axialHealthcare
Committed to enhancing the well-being of patients in pain, Jianhong mines the literature and data to develop evidence-based solutions to issues on pain care and opioid misuse.