RAPID CALCULATION OF MEDICATION ADHERENCE USING PARALLEL COMPUTING WITH R AND PYTHON

Nick Davis, PhD

Assistant Professor of Research

Department of Medical Informatics

University of Oklahoma, Tulsa

School of Community Medicine

nicholas-davis@ouhsc.edu

@argoneus

Oklahoma Supercomputing Symposium

September 24, 2014

Medication Adherence Definitions

- Adherence – fill prescriptions in a timely fashion
- Compliance – take medication as directed
- Persistence – deals with overall duration of drug therapy
- Subtle but distinct differences, but studies will sometimes use compliance and adherence interchangeably

Methodology

- Adherence is generally defined as the rate at which patients fill their prescriptions as indicated
- Drug claims data is considered to be an efficient and generally accurate means of assessing medication adherence
- One caveat is claims data is not a perfect representation of whether a patient is actually taking the medication
  - Patient may fill prescription and still fail to take medication

Methodology (cont.)

- Adherence generally expressed as a percentage
- Can be viewed roughly as how often/ frequently patients take medication as directed
- Most studies have suggested a threshold of 80%
- Specific conditions may require a higher cutoff (e.g. 95%)

Methodology (cont.)

- Nonadherence is common for cardiovascular disease patients
- Psychiatric illness patients struggle with adherence, but have the greatest potential benefit – 58% among patients with psychoses
- *“Since adherence is enhanced when patients are involved in medical decisions about their care and in monitoring their care, the traditional model of the authoritarian provider should be replaced by the more useful dynamic of shared decision making by the health care provider and the patient.”*

Motivation

- Medication adherence called the "next frontier" in healthcare quality improvement
- Non-adherence is related to greater morbidity and mortality in chronic disease
- Non-adherence estimated to increase healthcare costs by over $170 billion annually in the US alone
- Patient treatment and economic considerations suggest that non-adherence is a serious health concern that encourages research with a goal of impacting health outcomes and treatment costs
- *"Drugs don't work in patients who don't take them"* – former US Surgeon General C. Everett Koop

Measurement

- Centers for Medicare and Medicaid (CMS) have a number of recommended measures
- As suggested in a recent Pharmacy Quality Alliance (PQA) report, the Proportion of Days Covered (PDC) is the recommended medication adherence measure for electronic pharmacy claims
- PDC is defined as

$$\frac{\text{Number of days in period “covered” by medication}}{\text{Number of days in period}}$$

- PDC is a more conservative estimate of adherence than related measures in dealing with patients switching drugs

Data characteristics

- For this study, electronic pharmacy claims data from the Oklahoma Health Care Authority (OHCA) is used
- OU Tulsa receives a monthly pharmacy Medicaid claims feed from OHCA
- > 3.9 million claims records currently in data set
- Duration of claims is 1994 – present
- >97% of the data is from 2009 onward
- Over 134,000 unique Oklahoma Medicaid patients represented
- Over 20,000 unique medications represented

Data elements

Patient ID	Drug name
Date of Service (filled by patient)	Drug code
Date Prescribed	Drug class
DOB/Age	Drug quantity
Sex	Days supplied
Race	ProviderName (first, middle, last)
County	CoPay

Additional data sources

- NPI DB from CMS
- RxNorm <-> National Drug Code (NDC) crosswalk tables from UMLS
- Both have been loaded in DBs in the BI data warehouse
- Capture Provider details and RxNorm, VA Drug Class for most records

OHCA Data preparation

- Data aggregated by patient and medication combination
- The study period/duration is one year
- The total number of days covered by medication is determined by the number of days supplied for each prescription
- Key advantage:

  Data is “embarrassingly parallel”

Example calculation

- Mitch has a prescription of a renin-angiotensin inhibitor for a cardiovascular condition. Each Rx is a 30 day supply (one pill per day), with refills provided on a monthly basis.

Adherence =

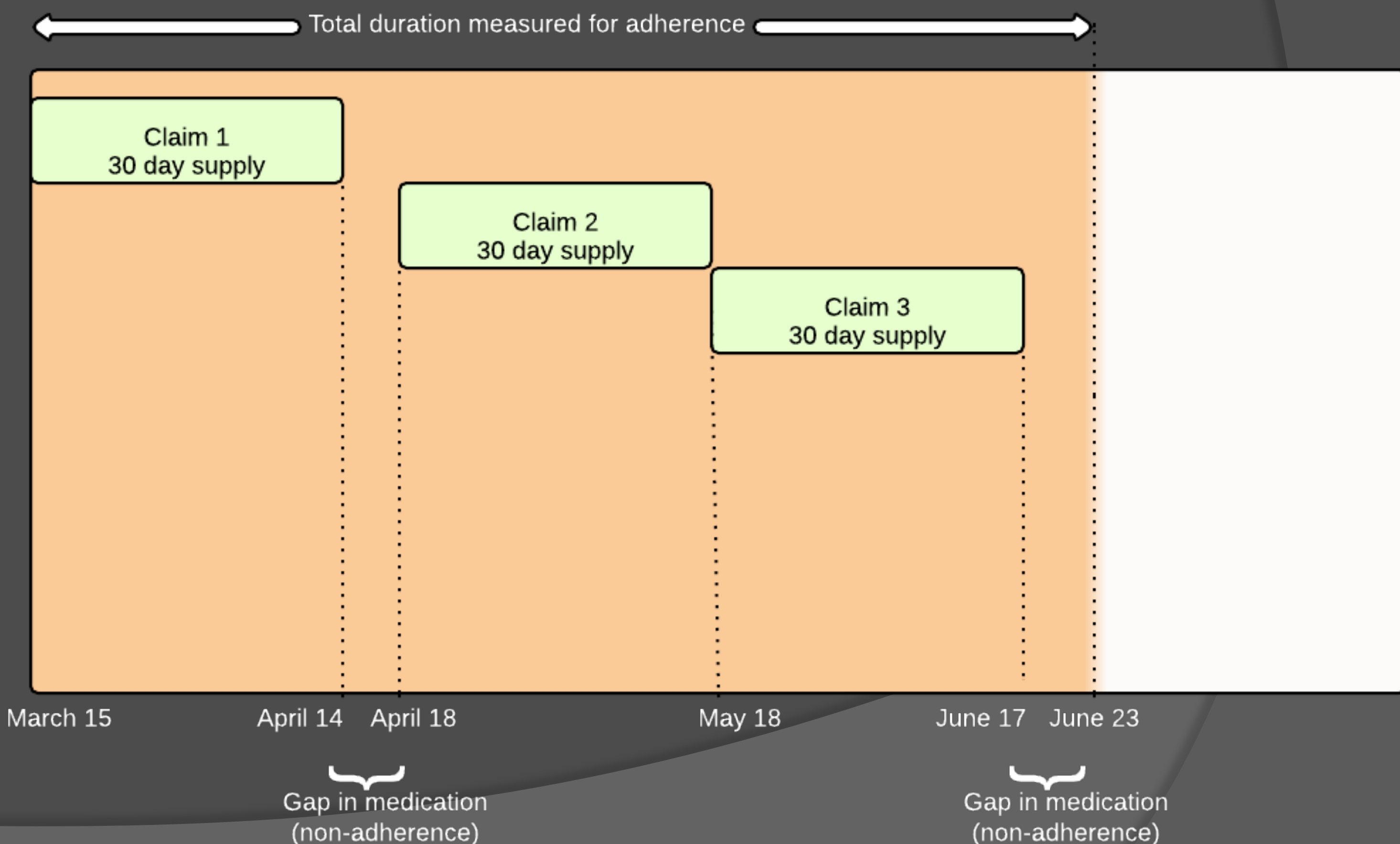
$$\frac{30 + 30 + 30}{\text{Days between June 23 and March 15}}$$

=

$$\frac{90}{100}$$

=

90%

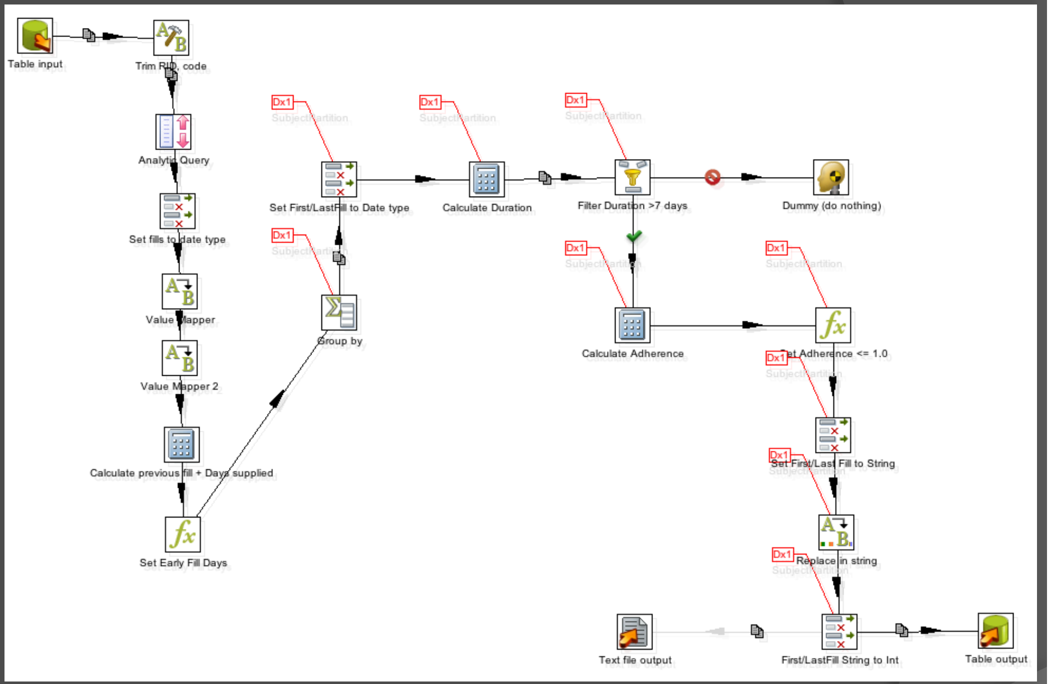


Testbed

- Dell Precision T7610
- Intel Xeon E5-2697 2.7GHz, 12 cores
- 128 GB DDR3 RAM
- OS - Ubuntu Linux 14.04, kernel 3.13.0-24
- 100Mb/s Ethernet

Language choice

- OUSCM analytics environment leverages Pentaho
  - \+ Simple GUI-based programming
  - \+ Java based so relatively fast execution
  - – Difficult to extend or modify, not a “real” programming language
  - – Not widely used
- R and Python
  - \+ R is a premier language and platform for data analysis/data science applications
  - \+ Python has the Pandas library to provide R-like data structures (also Numpy/Scipy)
  - \+ Both are open source, large communities, lots of examples, libraries, documentation
  - – Interpreted languages, slower than compiled programs



R Code

```r
calc_adherence <- function(fills, year){
  filldates <- as.Date(as.character(fills$FirstDateofService), format("%Y%m%d"))
  days_supplied <- fills$DaysSupplied
  first_fill <- min(filldates)
  last_fill <- max(filldates)

  duration <- as.Date(paste(year,"1231", sep=""), format("%Y%m%d")) - first_fill + 1
  med_days <- vector(mode = 'integer', length=as.numeric(duration))
  for (i in seq(duration)){
    for (j in seq(length(filldates))){
      if (filldates[j] <= first_fill + i - 1 && first_fill + i - 1 <= filldates[j] + days_supplied[j] - 1)
        med_days[i] <- 1
    }
  }

  early_fill_days <- calc_early_fill(filldates, days_supplied)
  days_covered <- sum(med_days) + early_fill_days
  adh <- days_covered / as.numeric(duration)
  if (adh > 1) adh <- 1.0
  ad <- data.frame(PatientID=fills$PatientID[1], DrugCode=fills$DrugCode[1], DrugName=fills$DrugName[1],
                   DrugStrength=fills$DrugStrength[1], OHCALabel=fills$OHCALabel[1],
                   OHCADrugClassName=fills$OHCADrugClassName[1], VADrugClassName=fills$VADrugClassName[1],
                   FirstFill=first_fill,
                   LastFill=last_fill, Duration=duration, DaysCovered=days_covered, MedicationDays=sum(med_days),
                   Sex=fills$RecipientSexCode[1], Age=fills$Age[1], Race=fills$RecipientRaceCode[1],
                   County=fills$RecipientCountyCode[1], CoPay=fills$CoPay[1], Zip=fills$Zip[1],
                   DrugClass=fills$DrugTherapyClass[1], PrescribingProviderNPI=fills$PrescriberPhysicianProviderNPI[1],
                   ProviderFirstName=fills$ProviderFirstName[1], ProviderLastName=fills$ProviderLastName[1],
                   DrugQuantity=fills$DrugQuantity[1], LastDaysSupplied=tail(days_supplied, n=1),
                   TotalDaysSupplied=sum(fills$DaysSupplied), EarlyDays=early_fill_days, Year=as.integer(year), Method="PDC",
                   Adherence=adh)
}
```

R Code (cont.)

```
# split population by PatientIDs to partition the data for parallel processing
system.time(ad_pop_idx <- split(seq_len(nrow(ad_pop)), ad_pop$groupID))

system.time(par_adh <- mclapply(ad_pop_idx, function(i, ap, fun) fun(ap[i,], "2013"), ad_pop, calc_adherence, mc.cores = 1))

# merge list of data frames into a single data frame
system.time(adh_all <- rbindlist(par_adh))

completeFun <- function(data, desiredCols) {
  completeVec <- complete.cases(data[,desiredCols])
  data[completeVec, ]
}

# remove NAs for subjects with strange fill data (less than 7 days covered, leading to infinite adherence measures and
# other similar strangeness)
adh_filt <- completeFun(data.frame(adh_all), "Adherence")
# calculate mean adherence for all patients
mean(adh_filt$Adherence)
```

R Code (cont.)

```
# split population by PatientIDs to partition the data for parallel processing
system.time(ad_pop_idx <- split(seq_len(nrow(ad_pop)), ad_pop$groupID))

system.time(par_adh <- mclapply(ad_pop_idx, function(i, ap, fun) fun(ap[i,], "2013"), ad_pop, calc_adherence, mc.cores = 1))

# merge list of data frames into a single data frame
system.time(adh_all <- rbindlist(par_adh))

completeFun <- function(data, desiredCols) {
  completeVec <- complete.cases(data[,desiredCols])
  data[completeVec, ]
}

# remove NAs for subjects with strange fill data (less than 7 days covered, leading to infinite adherence measures and
# other similar strangeness)
adh_filt <- completeFun(data.frame(adh_all), "Adherence")
# calculate mean adherence for all patients
mean(adh_filt$Adherence)
```

Python Code

```
def calc_adherence(fills, year = None):
    filldates = [ datetime.datetime.strptime(str(fills['FirstDateofService'][i]), '%Y%m%d').date()
            for i in fills['FirstDateofService'].index.values.tolist() ]
    days_supplied = fills['DaysSupplied']
    first_fill = min(filldates)
    last_fill = max(filldates)
    duration = ((datetime.date(int(year), 12, 31) if year else last_fill) -\
            first_fill).days + 1
    med_days = [0] * duration
    for i in range(duration):
        for j in range(len(filldates) if year else len(filldates) - 1):
            if filldates[j] <= first_fill + datetime.timedelta(days=i) \
             <= filldates[j] +\
            datetime.timedelta(days=int(days_supplied.irow(j))) - datetime.timedelta(days=1):
                med_days[i] = 1
    early_fill_days = calc_early_fill(filldates, days_supplied)
    days_covered = sum(med_days) + early_fill_days
    adh = days_covered / float(duration)
    if adh > 1:
        adh = 1.0
    ad = pd.DataFrame({'PatientID' : fills['PatientID'][0],
                    'DrugCode' : fills['DrugCode'][0],
                    'RxNorm'   : fills['RxNorm'][0],
                    'RxNormLabel' : fills['RxNormLabel'][0],
                    'DrugName' : fills['DrugName'][0],
                    'DrugStrength' : fills['DrugStrength'][0],
                    'OHCALabel' : fills['OHCALabel'][0],
                    'OHCADrugClassName' : fills['OHCADrugClassName'][0],
                    'VADrugClassName' : fills['VADrugClassName'][0],
                    'FirstFill' : first_fill,
                    'LastFill' : last_fill,
                    'Duration' : duration,
                    'DaysCovered' : days_covered,
                    'MedicationDays' : sum(med_days),
                    'Sex' : fills['RecipientSexCode'][0],
                    'Age' : fills['Age'][0],
                    'Race' : fills['RecipientRaceCode'][0],
                    'County' : fills['RecipientCountyCode'][0],
                    'CoPay' : fills['CoPay'][0],
                    'Zip' : fills['Zip'][0],
                    'DrugClass' : fills['DrugTherapyClass'][0],
                    'PrescribingProviderNPI' : fills['PrescriberPhysicianProviderNPI'][0],
                    'ProviderFirstName' : fills['ProviderFirstName'][0],
                    'ProviderLastName' : fills['ProviderLastName'][0],
                    'DrugQuantity' : fills['DrugQuantity'][0],
                    'LastDaysSupplied' : days_supplied.irow(-1),
                    'TotalDaysSupplied' : sum(fills['DaysSupplied']),
                    'EarlyDays' : early_fill_days,
                    'Year': int(year) if year != None else None,
                    'Method': "PDC" if year != None else "smPDC",
                    'Adherence' : adh  }, index = [fills['groupID'][0]])
    return ad
```

Python Code (cont.)

```python
def run_analysis(year = '2013', pdc = True):
    """Execute a series of steps required for calculating medication adherence"""
    ad_pop = getRecords(True, year)
    # set data type of groupID to int64
    ad_pop['groupID'] = ad_pop['groupID'].astype(int)
    # split population by groupIDs to partition the data for parallel processing
    ad_pop_idx = ad_pop.groupby('groupID')
    # Run the adherence calculation in parallel for each partition of data
    start = time.clock()
    par_adh = Parallel(n_jobs=-1)(delayed(calc_adherence)(
        ad_pop.iloc[ad_pop_idx.groups[i]].reset_index(), (year if pdc else None)) for i in ad_pop_idx.groups)
    end = time.clock()
    print("calc: %f"%(end - start))
    # merge list of data frames into a single data frame
    start = time.clock()
    adh_all = pd.concat(par_adh)
    end = time.clock()
    print("concat: %f"%(end - start))
    # remove rows with NAs in Adherence column
    adh = adh_all.dropna(axis=0, subset=['Adherence'])
    # calculate mean adherence across all patients
    print("mean adherence: %f"%adh['Adherence'].mean())
    return adh
```

Python Code (cont.)

```python
def run_analysis(year = '2013', pdc = True):
    """Execute a series of steps required for calculating medication adherence"""
    ad_pop = getRecords(True, year)
    # set data type of groupID to int64
    ad_pop['groupID'] = ad_pop['groupID'].astype(int)
    # split population by groupIDs to partition the data for parallel processing
    ad_pop_idx = ad_pop.groupby('groupID')
    # Run the adherence calculation in parallel for each partition of data
    start = time.clock()
    par_adh = Parallel(n_jobs=-1)(delayed(calc_adherence)(
        ad_pop.iloc[ad_pop_idx.groups[i]].reset_index(), (year if pdc else None)) for i in ad_pop_idx.groups)
    end = time.clock()
    print("calc: %f"%(end - start))
    # merge list of data frames into a single data frame
    start = time.clock()
    adh_all = pd.concat(par_adh)
    end = time.clock()
    print("concat: %f"%(end - start))
    # remove rows with NAs in Adherence column
    adh = adh_all.dropna(axis=0, subset=['Adherence'])
    # calculate mean adherence across all patients
    print("mean adherence: %f"%adh['Adherence'].mean())
    return adh
```

Runtimes

- 219K rows of data
- Each row represents a unique Patient/ Medication/Fill date combination
- Reported times represent the average of 3 runs

Language	Number of cores	Overall Runtime	Calculation Runtime
R	1 (sequential)	1674.5 s	1652.7 s
Python	1 (sequential)	420.1 s	405.6 s
R	12	821.7 s	799.8 s
Python	12	132.9 s	115.6 s

Conclusions/Questions?

- R and Python (with Pandas) are excellent languages for data analysis
- Parallelizing code is often trivial, with some caveats
- Faster runtimes lead to richer exploration of the data