JHSPH Interview Response

Larry D. Lee Jr.

October 22, 2024

Abstract: This technical note answers the questions presented in JHSPH's Coding Challenge. In this note, I review the mathematical theory behind combining datasets that use different stratifications and present code that combines these datasets.

Introduction

This technical note answers the questions presented in JHSPH's Coding Challenge. In this note, I review the mathematical theory behind combining datasets that use different stratifications and present code that combines these datasets.

Mathematical Background

In this section, I briefly review the mathematical background needed to understand how to combine stratified datasets.

General Notation

Let $s : set\ T$ represent a set of entities of type T. In general, T can represent anything, however, in this technical note, T will represent people. We will use the dot notation to reference attributes. For example, $s_i.age : \mathbb{R}^+$ will represent a person's age. Whenever we have a set s, $|s|$ will denote the number of elements within it.

Partitions

A finite **partition** of a set s is a finite set of sets $p : set\ (set\ T)$ such that $s = \bigcup_{i=1}^{n} p_i$ and $\forall i, j \in \mathbb{N}, i \neq j \implies p_i \cap p_j = \emptyset$ where $n = |p|$. A **partition function** $f : T \rightarrow \mathbb{N}$ is a function that accepts a value s_i from s and returns the index j of the subset p_j that contains s_i in a partition p.

As a special case we will often consider contiguous clopen interval partitions

(**CCIPs**) of the positive real number line $\mathbb{R}^+$.[1] We will represent these partitions as a list of intervals in ascending order and will represent the type of such partitions as $[Interval]$. Note that there are no gaps between these intervals thus:

$$\forall p : [Interval], \forall x \in \mathbb{R}, x \geq p_0.start \ \wedge x < p_n.end \implies \exists p_i \in p, x \in p.$$

Stratification

A **stratification** is a partition in which elements from s are allocated to different subsets p based some property value such as age.

In a stratification we take a set of entities of type T, idenfity a property, such as age, that spans some domain, such as $\mathbb{R}^+$, partition that domain p, and then project entities into those partition sets.

Rates

Let $f : T \to \mathbb{B}$ represent a predicate such as the function that accepts a person and returns true iff the person has a given disease. We define the **frequency** of f over a set s as:

$$freq\ (f, s) := |\{x \in s \mid f\ (x)\}|.$$

The **rate** of a f over a set s is defined as:

$$rate\ (f, s) := \frac{freq\ (f, s)}{|s|}.$$

Let p denote a finite disjoint partition of s into n subsets, then:

$$freq\ (f, s) := \sum_{i=1}^{n} freq\ (f, p_i)$$

and accordingly:

$$rate\ (f, s) = \frac{\sum_{i=1}^{n} freq\ (f, p_i)}{|s|} = \frac{\sum_{i=1}^{n} rate\ (f, p_i)\ |p_i|}{\sum_{i=1}^{n} |p_i|}. \tag{1}$$

[1]We call intervals that are closed at their lower bound and open at their upper bound **clopen**.

We say that a dataset gives the rates of a predicate f over a partition p when the dataset lists the rate of f for every element (subset of s) in p.

Rate Ratios

Consider the case in which we have two different predicates $f, g : T \to \mathbb{B}$. These two functions may represent different diseases such as HIV and Gonorrhea for instance. Often, we want to compare their rates. Their **rate ratio** over a set s is given by:

$$rateRatio(f, g, s) := \frac{rate\ (f, s)}{rate\ (g, s)} = \frac{freq\ (f, s)}{freq\ (g, s)}.$$

Notice that the rate ratio can only be defined when the rates given apply to the same underlying set. We say that two sets are **compatible** if they are equal. We can calculate risk ratios using rates over two compatible sets. Also note that the risk ratio of f and g can be calculated by taking the ratio of their frequencies over s.

Refinement and Coarsening

Let $p, q : set(set\ T)$ represent two partitions of a set s. We say that q is a **refinement** of p iff for every element p_i in p there exist one or more elements in q that collectively represent a partition of p. In other words, there are one or more elements in q whose union equals p_i.

$$\forall p_i \in p, \exists r \subseteq q, p_i = \bigcup r.$$

Conversely, we say that p is a **coarsening** of q if q is a refinement of p.

We slightly overload our terminology by saying that r is p_i's **refinement** in q.'

We can apply this notion to contiguous clopen interval partitions of $\mathbb{R}^+$. Let $p, q : [Interval]$ represent two CCIPs of $\mathbb{R}^+$. We say that q **refines** p iff:

$$refines?\ (q, p) := \forall p_i \in p, \exists (j, n \in \mathbb{N}), p_i = \bigcup_{k=j}^{j+n} q_k.$$

Conversely, we say that p **coarsens** q. We say that q is **more granular** than p when q refines p.

Rates over Coarsening

Let $p, \pi : [Interval]$ be two CCIPs where π generalizes p. Let's assume furthermore that we have rates for some predicate f over p. We can use (1) to calculate the corresponding rates of f over π.

If we know the proportion $prop(p_i) := |p_i|/|s|$ of entities in the underlying set s that are grouped within each interval p_i, we can rewrite (1) as:

$$rate\ (f, \pi) = \sum_{i=1}^{n} rate\ (f, p_i)\ prop\ (p_i).$$

Rate Ratios over Coarsening

We can calculate the rate ratios over two partitions p and q when q is a refinement of p. The rate ratios will be defined for elements (sets of s) defined by the less granular partition p. For every element p_i in p, let r_i represent p_i's refinement in q. Then the risk ratio of f and g over p is given by:

$$rateRatio(f, g, p_i, r_i) := \frac{rate\ (f, p_i)}{rate\ (g, r_i)} = \frac{freq\ (f, p_i)}{freq\ (g, r_i)}.$$

Generalization

Let's say that $p, q, \pi : [Interval]$ represent three CCIPs of $\mathbb{R}^+$. We say that π **generalizes** p and q iff:

$$generalizes?\ (\pi, p, q) := refines?(\pi, p) \wedge refines?\ (\pi, q).$$

Notice that the intervals in π start and end at boundary points shared by both p and q.

We say that π is the **most granular generalization** of p and q iff π refines every other generalization of p and q.

Rounding Errors

When working with real world datasets we have to account for the accumulation of rounding errors when combining rates for partitions. Imagine that we have an interval and a rate of 0.53. If we extended the accuracy of the rate to four significant digits, we could find that the true rate was anything from 0.5250 to 0.5349.

Regression Estimates

In this technical note, we've considered datasets that use different data stratifications and discussed ways to define new stratifications that we can use to combine frequencies and rates. This approach produces the most accurate results across composite datasets. However, this discussion would be incomplete without discussing the use of regression analysis to estimate rates across populations.

Consider the case in which we have a set s of entities of type T. T in this case will represent people. We can stratify the people in s using a partition p. For instance, we can stratify people by age where each age group corresponds to a five year interval. Significantly however we can gradually refine our partition by using a series of increasingly shorter intervals. For instance, we can refine p into 1 year intervals, then 1 month intervals, then 1 day intervals, etc. Theoretically, we can continue this refinement process until we reach a limit of a continuous time distribution.

Let's say that we have a set s of people (entities of type T). We stratify this population by age using equal intervals (for example 1 year intervals) p_i and then calculate the rates of some condition f for each subgroup.

Answers

The Interview challenge opens with the following questions:

> PART 1: Using these four datasets, calculate the following quantities, or if they cannot be calculated exactly, explain why
>
> 1. The ratio of gonorrhea rate to HIV diagnosis rate among people ages 25-44, and among people ages 45 or greater, in 2020
> 2. The ratio of heroin use rate to HIV diagnosis rate among people ages 12 and older in 2020

I've included a code package that includes code written in both R and OCaml to answer these questions. We can calculate the frequency of Gonorrhea and HIV for each of the age ranges included in their datasets. We can then calculate the sum of people in both datasets who have these conditions in the 25-44 age cohorts. Dividing these sums we find that:

1. (a) The ratio of gonorrhea rate to HIV diagnosis rate among people aged 25-44 is **19.8**.

Note that this calculation is approximate but accurate to 3 significant digits. We relied on underlying rate estimates for Gonorrhea and HIV. Both of these datasets rounded rate estimates to the nearest tenth of 1/100,000th. Thus, given an estimate such as 25.7 per 100,000, the true rate is anywhere between 25.65 and 25.74. These errors can compound enough to shift our estimate but not by enough to alter the figure given. If we extend our estimate to four significant digits the potential error ranges from 19.82 to 19.84.

1. (b) The ratio of gonorrhea rate to HIV diagnosis rate among people aged 45 and older is **6.7**

This calculation is accurate to the number of digits shown. However, given the rounding error implicit in the underlying rate estimates for Gonorrhea and HIV, the true ratio could range from 6.72 to 6.75.

1. We cannot accurately estimate the ratio of heroin use rate to HIV diagnosis rate for people ages 12 and older.

The reason we cannot estimate this datum is because the age partitions used by the HIV rate dataset and the population age distribution are not compatible (as defined above). The best that we can do is to use some form of curve fitting or interpolation to estimate population density over age and then use this to estimate the number of people falling within the age ranges used by the heroin dataset.

The interview challenge continues by giving a specification for a function that "... takes three datasets as inputs (two with rates per population and one with population) and returns the ratio of these rates by the most granular age brackets possible". The `getRateRatio` function defined in the code sample given in Appendix 1 implements this function specification.

Appendix 1: R Code Sample

```r
# This package contains definitions that can be used to represent and combine
# stratified datasets.
#
# It defines a datatype named interval that represents half open intervals of
# the real number line. These intervals can be used to partition the real number
# line into a finite number of intervals. A population can be divided across
# these intervals and information about the subpopulation associated with each
# group can be added.
#
# At the end of this file, we use these definitions to describe the prevalence
# of HIV and gonorrhea across Baltimore City.

UNBOUND=quote(unbound)

# Constructs an annotated interval
# These objects represent the closed-open intervals of the postive real number
# line.
# Annotated partitions are represented by list of interval objects
interval <- function (info, start, end = UNBOUND) {
  list (info=info, start=start, end=end)
}

# Accepts two real numbers x and y who may be "unbound" and compares them.
boundCompare <- function (x, y) {
  if (x == y) {
    return (0)
  } else if (x < y || y == UNBOUND) {
    return (-1)
  } else if (x > y || x == UNBOUND) {
    return (1)
  }
}

# Accepts two real numbers x and y who may be "unbound" and returns true iff x
# is less than or equal to y.
boundLte <- function (x, y) boundCompare (x, y) <= 0

# Accepts two real numbers who may be "unbound" and returns the smaller of them.
boundMin <- function (x, y) {
  if (x == UNBOUND) {
    y
  } else if (y == UNBOUND) {
    x
  } else {
    min (x, y)
  }
}

# Accepts three arguments:
#
# @param f a function that accepts two contiguous series of annotated intervals
#   that span the same range of the positive real number line and merges them
# @param xs, an annotated partition
# @param ys, an annotated partition
#
# And returns a new annotated partition based on the most granular intervals
# spanned by xs and ys whose annotations are derived from f.
rebase <- function (f, xs, ys) {
  zs <- list ()
  i <- 1
  j <- 1
  while (i <= length (xs) && j <= length (ys)) {
    x <- xs[[i]]
    y <- ys[[j]]
    if (x$start == y$start) {
      xInfos <- c ()
      yInfos <- c ()
      start <- x$start
      while (i <= length (xs) && j <= length (ys)) {
        x <- xs[[i]]
        y <- ys[[j]]
        if (
          boundCompare (x$end, y$end) == 0 ||
          (y$end == UNBOUND && i == length (xs)) ||
          (x$end == UNBOUND && j == length (ys))
        ) {
          zs <- append (zs,
            list (interval (
              info=f (
                c (xInfos, x$info),
                c (yInfos, y$info)
              ),
              start=start,
              end=boundMin (x$end, y$end)
          )))
          i <- i + 1
          j <- j + 1
          break
        } else if (boundCompare (x$end, y$end) == -1) {
          xInfos <- c (xInfos, x$info)
          i <- i + 1
        } else { # x$end > y$end
          yInfos <- c (yInfos, y$info)
          j <- j + 1
        }
      }
    } else if (x$start < y$start) {
      i <- i + 1
    } else { # x$start > y$start
      j <- j + 1
    }
  }
  zs
}

# Accepts two partitions:
# @param ps a partition that represents the number of people in a population who
#   fall within a contiguous set of age ranges
# @param rs a partition that represents the proportion of people within a
#   contiguous set of age ranges who have some condition
# and returns a partition that gives the absolute number of people who have the
# condition within the most granular set of age ranges spanned by both
# ps and rs.
# @warn
getFrequency <- function (ps, rs) {
  rebase (
    function (sizes, rates) {
      if (length (rates) != 1) {
        stop ("Error: the population partition is not a \"refinement\" of the rate partitio
      }
      rates[1]*sum (sizes)
    }, ps, rs
  )
}

# Accepts three arguments:
# @start the age range lower bound
# @end the age range upper bound
# @freqs the number of people who have a given condition for a set of age ranges
# and returns the total number of people who have the given condition between
# the start and end age range (inclusive).
getFreqSum <- function (start, end, freqs) {
  sum <- 0
  for (freq in freqs) {
    if (start <= freq$start && boundLte (freq$end, end)) {
      sum <- sum + freq$info
    }
  }
  sum
}

# Accepts two partitions:
# @param xs a partition that gives the number of people within a contiguous set
#   of age ranges who have condition x
# @param ys a partition that gives the number of people within a contiguous set
#   of age ranges who have condition y
# and returns the rate ratios of the two conditions over the most granular
# partition spanned by both xs and ys.
# Note that xs and ys do not have use the same partition intervals.
getFreqRateRatio <- function (xs, ys) {
  rebase (
    function (freqs0, freqs1) {
      sum (freqs0)/sum (freqs1)
    }, xs, ys
  )
}
```

```
# Accepts three partitions:
# @param ps a partition that gives the number of people who fall into a
#   contiguous set of real intervals
# @param xs a partition that gives the proportion of people within a range of
#   intervals who have a condition x
# @param ys a partition that gives the proportion of people within a range of
#   intervals who have a condition y
# and returns the rate ratio of the number of people who have condition x and y
# over the most granular common partition that spans xs and ys.
getRateRatio <- function (ps, xs, ys) {
  xFreq <- getFrequency (ps, xs)
  yFreq <- getFrequency (ps, ys)
  getFreqRateRatio (xFreq, yFreq)
}

# A partition listing the number of Baltimore residents who's ages fall within
# a contiguous set of age ranges.
population = list (
  interval (36355, 0, 4),
  interval (33773, 5, 9),
  interval (33590, 10, 14),
  interval (33872, 15, 19),
  interval (37183, 20, 24),
  interval (53357, 25, 29),
  interval (54804, 30, 34),
  interval (43408, 35, 39),
  interval (34271, 40, 44),
  interval (30273, 45, 49),
  interval (33423, 50, 54),
  interval (37639, 55, 59),
  interval (36895, 60, 64),
  interval (29868, 65, 69),
  interval (22486, 70, 74),
  interval (13910, 75, 79),
  interval (8977, 80, 84),
  interval (9073, 85)
)

# A partition listing the rates of HIV amongst Baltimore residents falling
# within certain age ranges.
hivRates = list (
  interval (45.6e-5, 13, 24),
  interval (53.6e-5, 25, 34),
  interval (46.6e-5, 35, 44),
  interval (26.7e-5, 45, 54),
  interval (5.4e-5, 55, 64),
  interval (30.4e-5, 65)
)

#  A partition list the rates of Gonorrhea amongst Baltimore residents falling
# within certain age ranges.
gonorrheaRates = list (
  interval (25.7e-5, 0, 14),
  interval (2021.6e-5, 15, 19),
  interval (2647.6e-5, 20, 24),
  interval (1477.8e-5, 25, 29),
  interval (1047.3e-5, 30, 34),
  interval (773.0e-5, 35, 39),
  interval (490.3e-5, 40, 44),
  interval (298.6e-5, 45, 54),
  interval (139.5e-5, 55, 64),
  interval (23.6e-5, 65)
)

# A partition listing the rates of heroin use amongst Baltimore residents
# falling within certain age ranges
# Note: The handout appears to have a typo in which heroin usage rates are
# reported as "rates per 100,000"
heroinRates = list (
  interval (0.00112, 12, 17),
  interval (0.005426598, 18, 25),
  interval (0.009849983, 26)
)

# The number of people within certain age ranges who have HIV
hivFreq = getFrequency (population, hivRates)

# The number of people within certain age ranges who have Gonorrhea
gonorrheaFreq = getFrequency (population, gonorrheaRates)

# The number of people within certain age ranges who use heroin
heroinFreq = getFrequency (population, heroinRates)

# The rate ratio of gonorrhea and HIV for people aged 25 to 44
# Note: the answer to question 1.(a)
gonorrheaHivFreq2544 =
  getFreqSum (25, 44, gonorrheaFreq)/
  getFreqSum (25, 44, hivFreq)

# The rate ratio of gonorrhea and HIV for people aged 45 and older
# Note: the answer to question 1.(b)
gonorrheaHivfreq =
  getFreqSum (45, UNBOUND, gonorrheaFreq)/
  getFreqSum (45, UNBOUND, hivFreq)
```