Johns Hopkins School of Public Health: Sr. Programmer Analyst Coding Challenge

Larry D. Lee Jr.

October 18, 2024

Abstract

In this brief technical note I present the mathematical theory behind ratio calculations across stratified datasets. I then apply this theory to work through an example dataset and close by presenting code samples in R and OCaml.

Introduction

In real world analysis, we often have to combine and analyze datasets using different data formats. Frequently, when working with time series and population datasets, the information is grouped together using different stratifications. As a result it's important for us to develop techniques for combining discrepant datasets. In this technical note, I present the mathematics needed to combine these datasets and illustrate these mathematical techniques. In the last chapter, I present code samples in R and OCaml that perform these operations.

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** $f : T \to \mathbb{N}$ 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.

The elements of this scheme are presented in the following diagram:

TODO: create diagram

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:

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

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

and accordingly:

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

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:

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

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.

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:

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

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.

Rates over Generalizations

TODO revise the algorithm: intervals include proportions and rates. Generate both the generalized patitions and the pair of imputed rates.

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. If we combine these rates

Regression Estimates

Confidence Intervals

Code Sample

The following algorithm accepts two partitions p and q and returns their most granular generalization.

```
-- Represents clopen intervals on R+
type Interval = Interval {
  start :: Int,
  end   :: Int
}

-- Accepts two interval partitions and returns their union - i.e. the most granualar partit
scan :: [Interval] -> [Interval] -> [Interval]
scan [] _ = []
scan _ [] = []
scan xs@(x:xs') ys@(y:ys')
  | x.start = y.start = stretch x.start xs ys
  | x.start < y.start = scan xs' ys
  | x.start > y.start = scan xs ys'

-- stretches a current interval and then resumes scanning for common intervals.
stretch :: Int -> [Interval] -> [Interval] -> [Interval]
stretch _ [] _ = []
stretch _ _ [] = []
stretch start xs@(x:xs') ys@(y:ys')
  | x.end = y.end = (Interval start x.end):(scan xs' ys')
  | x.end < y.end = stretch start xs' ys
  | x.end > y.end = stretch start xs ys'
```

Notice that the algorithm runs in linear time.

Illustrated Example

Let's focus specifically on the following scenario. We have a population of people represented by the set s and two datasets. D_0 and D_1. The first dataset stratifies the people in s by age using partition p. The second stratifies people by age using different age ranges. Its partition is denoted by q. Thus, every set in p and q corresponds to an interval on the positive real number line $\mathbb{R}^+$. We do not require q to be a refinement of p or vice versa. Instead, given an interval p_i in p, there may or may not exist one or more intervals in q that partitions p_i - i.e. there may or may not exist a refinement of p_i in q. We want to identify the most granular partition π of $\mathbb{R}^+$ that is compatible with both p and q and then calculate the rate ratios for every element in π.

Conclusion

Appendix 1: Rates Over Generalized Partitions

In the following code snippet, we present a set of functions that accept two partitions for which we have rates and returns the corresponding rates for their most granular generalization.

```
-- Represents information about entities in an interval
type Info a = Info {
  rate       :: a,
  proportion :: Float
}

-- Represents clopen intervals on R+
type Interval a = Interval {
  info  :: Info a,
  start :: Int,
  end   :: Int
}

-- Accepts two interval partitions and returns their union - i.e. the most granualar partit
scan :: [Interval Float] -> [Interval Float] -> [Interval (Float, Float)]
scan [] _ = []
scan _ [] = []
scan xs@(x:xs') ys@(y:ys')
  | x.start = y.start = stretch x.start (Info 0 0) xs ys
  | x.start < y.start = scan xs' ys
  | x.start > y.start = scan xs ys'

-- stretches a current interval and then resumes scanning for common intervals.
stretch :: Int -> Info (Float, Float) -> [Interval Float] -> [Interval Float] -> [Interval
```

```haskell
stretch _ _ [] _ = []
stretch _ _ _ [] = []
stretch start info xs@(x:xs') ys@(y:ys')
  | x.end = y.end =
    (Interval
      {
        rate = (
          fst info.rate + x.info.rate * x.info.proportion,
          snd info.rate + y.info.rate * y.info.proportion
        ),
        proportion = info.proportion + x.info.proportion
      }
      start x.end):
    (scan xs' ys')
  | x.end < y.end = stretch start
    {
      rate = (
        fst info.rate + x.info.rate * x.info.proportion,
        snd info.rate
      ),
      proportion = info.proportion + x.info.proportion
    }
    xs' ys
  | x.end > y.end = stretch start
    {
      rate = (
        fst info.rate,
        snd info.rate + y.info.rate + y.info.proportion
      ),
      proportion = info.proportion
    }
    info xs ys'

-- Accepts two interval partitions and returns the rate ratios for their generalized partio
getRateRatios :: [Interval Float] -> [Interval Float] -> [Interval Float]
getRateRatios p q = map (\x -> x {info = (fst x.info)/(snd x.info)}) $ scan p q
```