Chapter 2: The t-a-p Model

David Eubanks

Table of contents

This chapter examines the assumptions and implications of the t-a-p model in detail to build a theoretical foundation for estimation and inference. This work also paves the way to connect the three parameter model to widely-used rater agreement statistics (kappas) and machine learning algorithms.

1 Introduction

Suppose that we have N subjects to be classified by at least two raters each, and each subject belongs (in truth) to one of two categories Class 1 or Class 0, abbreviated to C_1, C_0. For example, if faculty members review portfolios of student work, the two classes could be "pass" and "fail." We often use ordered classifications like "A, B, C, D, F" or "Excellent, Good, Fair, Poor" but these can be converted to binary classifications by grouping the classes. For example "A, B, C" could be compared to "D, F" or "Excellent, Good" could be compared to "Fair, Poor." There are also extensions of the t-a-p model that work directly on ordinal or

non-binary categorical scales. The exposition is easiest to understand for the binary case, so we will start there.

We assume that each subject is independently assigned to one of the two classes by each of R observers (raters). For now, think of R as fixed, so that there are RN total ratings, but that condition is relaxed later on, so that the number of ratings per subject can vary, which is common in real data.
Rater-assigned categories are distinguished from true classes in notation by the use of hats to suggest an estimated value. Ratings of Class 1 are denoted $\widehat{C_1}$, but some of them may be C_0 in truth. This distinction between rated (estimated) values and true values leads to the idea of true/false positives/negatives and the confusion matrix described in the introductory chapter. However, the confusion matrix does not capture all the details we are interested in. For example, the true positive rate of classification in the matrix doesn't distinguish between ratings that are true for a good reason and those that are accidentally true (maybe the rater is just flipping coins). To solve that problem we need the idea of rater accuracy.

We will assume that an assigned rating corresponds to the *belief* of a rater. This rules out raters who are intentionally being deceptive, for example. Then we will say that a rater makes an *accurate* assignment of $\widehat{C_1}$ or $\widehat{C_0}$ for a subject if

1. the rater-assigned class is the true class, and
2. the rater has justification for the choice.

These requirements operationalize the Justified True Belief (JTB) definition of knowledge used by philosophers who study epistemology. Inaccurate ratings are those where one of the two listed conditions fails. If the rater's belief is false and chooses the wrong category or if they choose the correct category but because of an invalid justification. The latter case corresponds loosely to Gettier-type problems, where the chain of reasoning reaches a correct conclusion, but because of flaws in perception the logic isn't sound. A clear case of failing the justification requirement is if raters flip coins to choose categories. Coin flipping is entirely random, but even good raters have some randomness inherent to the classifications. That randomness is the usual starting point for chance-corrected agreement statistics.

The rating process just described lends itself to a tree diagram that illustrates three variables as conditional probabilities: (1) the true Class 1 rate t, (2) rater accuracy a, and (3) the probability p of choosing $\widehat{C_1}$ when rating inaccurately. On the diagram, it's convenient to use a bar over a variable to denote its complementary probability, e.g. $\bar{a} = 1 - a$.

Each rating is conditional on a subject's true classification (Class 1 or Class 0), which will often be unknown, so that we can only observe the rater-assigned categories $\widehat{C_1}$ and $\widehat{C_0}$. The rating data will be denoted by c_{ij}, where $i = 1 \dots N$ are the subjects and $j = 1 \dots R$ are the raters, who independently classify subjects as 1 or 0 according to the $\widehat{C_1}$ or $\widehat{C_0}$assignment, respectively. This is convenient, since $k_i := \sum_j c_{ij}$ is the number of $\widehat{C_1}$ ratings for subject i, and the average of the classifications made by raters is $\mathrm{E}[\widehat{C_1}] = \sum c_{ij}/(NR)$.

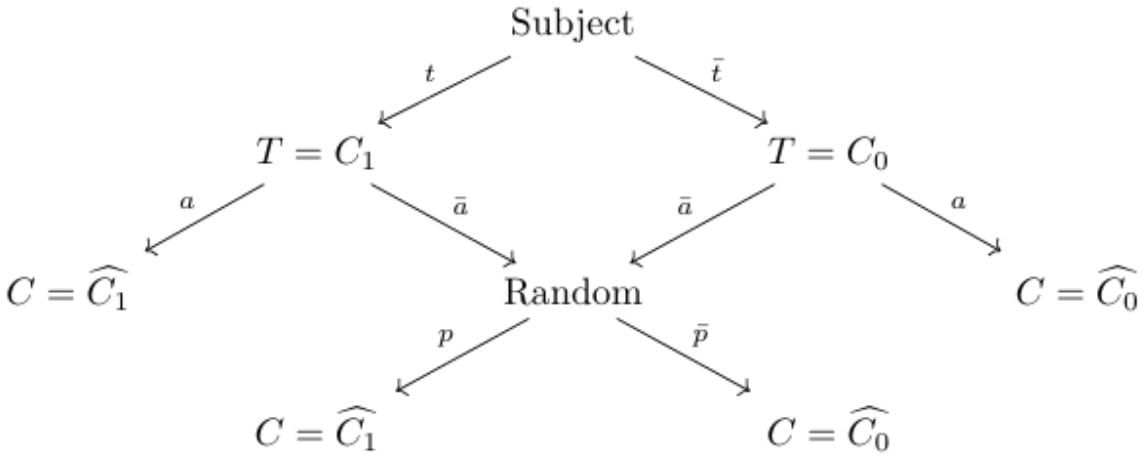

Figure 1: Assumed rating process, where T is a hidden variable recording the true classification of each subject and C recorded the assigned class by a rater.

Example: A wine judge independently and honestly assesses a vintage for excellence. The two categories are C_1 = "excellent" and C_0 = "not excellent." After judging four wines, the situation might be that in the table below.

Table 1: Sample wine ratings showing normally unknown truth values.

Wine	Truth	Accurate?	Classification
1	Excellent	Yes	Excellent
2	Excellent	No	Not Excellent
3	Not Excellent	No	Not Excellent
4	Not Excellent	Yes	Not Excellent
5	Not Excellent	Yes	Not Excellent

If the judge makes an accurate assessment, the classification recorded matches the true value. But for the third wine, the judge got the correct answer even though the process was flawed and somewhat random (an inaccurate rating). For the second wine, the inaccuracy resulted in the wrong classification being assigned. Those two inaccurate cases are illustrated in the process diagram in the middle, marked Random.

2 Conditional Probabilities

The tree diagram in figure @fig-tree models the assignments of ratings c_{ij} over subject i and rater j, and can be read by multiplying the conditional probabilities on the edges from the top

down to find the probability of a given classification being $c_{ij} = \widehat{C_1}$.

If a subject is not classified accurately, the classification for that rater is assumed to be made at random, with probability p of choosing $\widehat{C_1}$ regardless of the true class. So the conditional probability of a $\widehat{C_1}$ classification when the subject really is C_1 is $\Pr(\widehat{C_1}|C_1) = a+a'p$. Similarly, $\Pr(\widehat{C_1}|C_0) = a'p$. This model assumes that guess rates for the two classes are the same independent of the true classification. More complex models are introduced later.

The binary (Bernoulli) probability that for a given subject with a given true classification, a rater will assign a $\widehat{C_1}$ rating is shown in the table below. This comes from reading the tree diagram from the top down, multiplying the branches.

Table 2: Conditional probabilities of a single rater assigning a $\widehat{C_1}$ rating to a subject.

	True C1	True C0
Classified C1	$a + \bar{a}p$	$\bar{a}p$

We can use these probabilities to find the expected number of ratings of Class 1.

3 Binomial Mixtures

The rating process posed in the t-a-p model is illustrated with the tree diagram and table of example ratings above. But both of those entail the use of the hidden true classification value for each subject. There are cases where that can be known, but in general it is not. What we usually have to work with is a table of ratings, from which we must infer the hidden variables like rater accuracy. The collection the ratings under the t-a-p assumptions fall into a well-studied probability distribution called a binomial mixture.

```
# set the parameters
N_r = 5    # number of raters for each subject
N_s = 100  # number of subjects (not used here)

t = .3 # fraction of subjects that are in fact class 1
a = .7 # probability of a rater rating a subject accurately
p = .2 # probability of a rater rating a subject as class 1 when rating inaccurately

# find the conditional probabilities for each class

  # probabilities of a class 1 rating for a class 0 subject
  c0_probs = dbinom(0:N_r, N_r, prob = (1-a)*p)
```

```
  # probabilities of a class 1 rating for a class 1 subject
  c1_probs = dbinom(0:N_r, N_r, prob = a + (1-a)*p)

# create the mixture with t as the mixing parameter
mixture = c0_probs * (1-t) + c1_probs * t

# Plot the conditional probabilities
plot(0:N_r, c0_probs, type="b", col="blue", pch=16, ylim=c(0, max(c(c0_probs, c1_probs, mixtu
     ylab="Probability", xlab="Number of class 1 ratings for a given subject", main="Binomial

# Add the second component (c1_probs)
lines(0:N_r, c1_probs, type="b", col="red", pch=16)

# Add the mixture as a black dashed line
lines(0:N_r, mixture, type="b", col="black", lty=2, pch=16)

# Add a legend
legend("topright", legend=c("Class 0 Probabilities", "Class 1 Probabilities", "Mixture"),
       col=c("blue", "red", "black"), lty=c(1, 1, 2), pch=16)
```

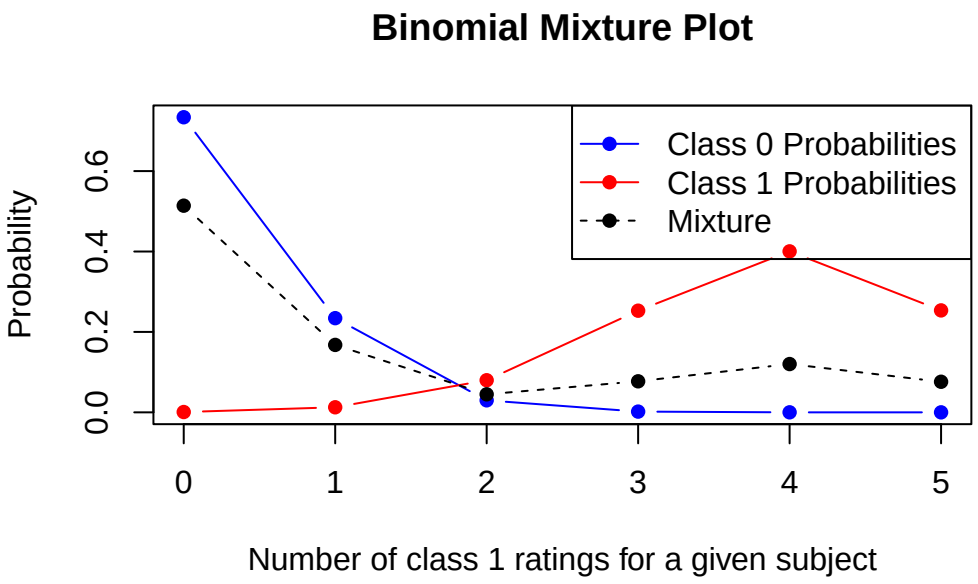

Figure 2: Binomial mixture showing the probability of the number of class 1 ratings for a given subject. In blue is the distribution for true-class 0 subjects, in red is the distribution for true-class 1 subjects, and the black line is the mixture based on relative proportions.

Figure 2 shows an example of how probabilities combine to create the mixture. Given the three t-a-p parameters plus the number of raters per subject, we apply the binomial probability density function using the probabilities found at the end of the previous section. The mixture is created by weighting these two distributions by their frequency in the sample space. In this case, the parameters are $t = .3$, $a = .7$, and $p = .2$.

For true Class 1 cases, the probability of a rater assigning a $\widehat{C_1}$ rating is $ta + t\bar{a}p$, and for true Class 0 cases it is $\bar{t}\bar{a}p$ (see the table at the end of the last section). Those are probabilities for a single rating. If we have N_r raters, then anywhere between zero and N_r of them could assign a $\widehat{C_1}$ rating to a given subject. The binomial distribution gives the probability for each of those possible count outcomes. For true Class 1 subjects the probability of k raters assigning a $\widehat{C_1}$ rating is

$$
\begin{aligned}
Pr(k|C_1) &= \binom{N_r}{k}(a+\bar{a}p)^k(1-a-\bar{a}p)^{N_r-k} \\
&= \binom{N_r}{k}(a+\bar{a}p)^k(\bar{a}-\bar{a}p)^{N_r-k} \\
&= \binom{N_r}{k}(a+\bar{a}p)^k(\bar{a}\bar{p})^{N_r-k}
\end{aligned}
$$

That distribution is represented by the red line in Figure 2. Notice that the most outcome is that four of the five raters assign a $\widehat{C_1}$ rating. The reason that's not five is that the parameter $p = .2$ means that for inaccurate ratings, the "guess" is much more likely to assign $\widehat{C_0}$ than $\widehat{C_1}$. The effect is to deflate the number of $\widehat{C_1}$ ratings for true Class 1 subjects.

For true Class 0 cases (the blue line in the plot), it is

$$
Pr(k|C_0) = \binom{N_r}{k}(\bar{a}p)^k(1-\bar{a}p)^{N_r-k}
$$

These are the two probability distributions are shown in Figure 2, with the given parameters applied. The code is included so that you can try variations on your own.

But there are not necessarily the same number of true Class 1 and Class 0 cases. The fraction of Class 1 cases is assumed to be t. The mixture of the two

$$
Pr(k) = t\binom{N_r}{k}(a+\bar{a}p)^k(\bar{a}\bar{p})^{N_r-k} + \bar{t}\binom{N_r}{k}(\bar{a}p)^k(1-\bar{a}p)^{N_r-k}
$$

This is the mixture distribution that is assumed to represent the count data. To proceed with an investigation of the t-a-p model, we first count up the number of $\widehat{C_1}$ ratings for each subject. If there are the same number of raters for each subject, these counts will form a histogram that corresponds to the black dashed line in the figure, for some set of t-a-p parameters. The job then is to find out what those parameters are.

So for a real data set, the red and blue plots in Figure 2 are assumed to exist, but are not directly accessible to us. Instead we can see something like the mixture (black dashed line). But even that isn't exact, because it is subject to sampling error: the histogram of counts won't exactly correspond to the ideal probabilities. The larger the number of subjects rated, the closer the empirical proportions will, in theory, converge to the ideal mixture distribution.

For general information on binomial mixtures of discrete data see @agresti2003categorical chapter 14.

4 Simulation

The interactive app provided with the tapModel R package allows you to specify t-a-p parameters and generate a data set from them. Once the package is installed, you can run the app with `tapModel::launchApp()`. Navigate to the Simulate Data tab.

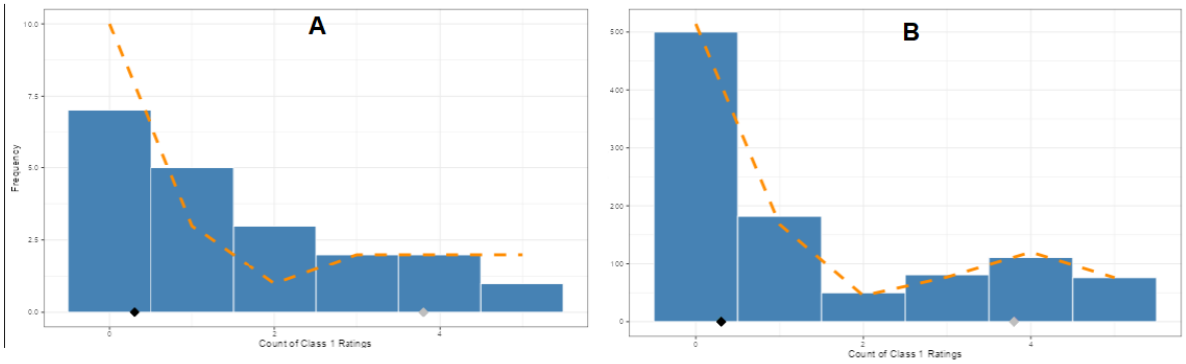

Figure 3: Rating distributions with t = .3, a = .7, p = .2. Plot A has 20 subjects and Plot B has 1000. The blue bars are the histograms of the simulated data and the orange dashed line is the true mixture probability distribution.

Figure 3 shows two count histograms with identical parameters except for the number of subjects (20 versus 1000). The histograms are the result of applying the t-a-p binomial mixture distribution with five raters on each subject and parameter set $(t = .3, a = .7, p = .2)$. Notice that the smaller sample size on the left (plot A) doesn't match the distribution line as well as the one on the right. This is the effect of random sampling. The smaller the sample, the more likely it is that the empirical counts don't look like the distribution.

This leaves us with two problems: how do we estimate the parameters, and how much should we trust the results? These are classical problems from statistics.

The accuracy rate a will affect the subject distributions. If $a = 0$ the ratings will be distributed as Bernoulli(p), independently of the subjects being rated. If $a = 1$, then all raters agree on the true value for each subject. Therefore the way we can reconstruct a from data is through the distribution of the within-subject ratings. The method used here can be seen as a latent class model with binomial mixture distributions. For a nice discussion of these ideas in practice see [**?**], which helpfully notes that binomial mixtures are statistically identifiable if the number of cases exceeds a low threshold [**?**].

We would like to know the true proportion t of the subjects belonging to C_1 regardless of how they were rated, rater accuracy a, and the proportion p of inaccurate assignments that are $\widehat{C_1}$. That goal describes the general model illustrated in the following section. A hierarchical version is given subsequently.

note: see https://www.jarad.me/courses/stat544/slides/Ch05/Ch05a.pdf

and https://cran.r-project.org/web/packages/mixtools/vignettes/mixtools.pdf

https://joss.theoj.org/papers/10.21105/joss.01505

I should address the label-switching problem. I think that's taken care of by way the classifications are pre-specified, AND accuracy is generic.

5 Fitting the Model

The first question about the model illustrated in figure **??** is whether it is computationally useful. Using known parameter values for p, a, t to generate simulated ratings, can we then recover the parameters from the data? The answer is yes, with some provisos. Given a data set c_{ij}, we can fit a general model to estimate the three parameters t, a, and p using maximum likelihood to fit a binomial mixture model. The log likelihood function for the binomial mixture described by figure **??** with N subjects and $R > 1$ raters is

$$\ell(t, a, p; k_1, \ldots, k_N) = \sum_{i=1}^{N} \log \left(t \binom{R}{k_i} (a + a'p)^{k_i} (a'p')^{R-k_i} + t' \binom{R}{k_i} (a'p)^{k_i} (1 - a'p)^{R-k_i} \right)$$

where $k_i = \sum_j C_{ij}$ the number of Class 1 ratings for subject i. The sum over the logs is justified by the assumption that ratings are independent. It is straightforward to implement the function in the Bayesian programming language Stan [**?**], using uniform $(0, 1)$ priors for the three parameters (see the Discussion section to access the source code).

To test the computational feasibility of this method, ratings were simulated using a range of values of t, a, and p. The 729 trials each simulated 300 subjects with five raters each, using all combinations of values ranging from .1 to .9 in increments of .1 for each of t, a, and p. The Stan engine uses a Markov chain Monte Carlo (MCMC) algorithm to gather representative samples from the joint probability density of the three parameters. Each run used 1,000 iterations (800 after warm-up) with four chains each.

[need to cite @gelman2020bayesian and use those methods]

```
regenerate = FALSE

if(regenerate == TRUE){
  param_grid <- expand.grid(t = seq(.1,.9,.1),a = seq(.1,.9,.1),p = seq(.1,.9,.1))

  n_sims <- nrow(param_grid)
  N <- 300 # number of subjects
```

```
  R <- 5   # number of raters

  out <- data.frame()
  fixed_model_spec <- stan_model("code/fixed_effects.stan")

  for(i in 1:n_sims){
    ratings <- generate_ratings(N, R, param_grid$p[i], param_grid$a[i], param_grid$t[i])

    # get the number of Class 1 ratings per subject in a vector
    counts <- ratings %>%
      group_by(SubjectID) %>%
      summarize(k = sum(RatedClass)) %>%
      select(k) %>%
      pull()

    kappa <- sqrt(fleiss(counts,R))

    fixed_model <- sampling(object = fixed_model_spec,
                            data = list(N = N, R = R, S = 1, count = counts),
                            iter = 1000,
                            warmup = 200,
                            thin = 1)

      stats <- rbind( broom::tidy(fixed_model),
                      data.frame(term = "root kappa",estimate = kappa, std.error = NA,
                                 stringsAsFactors = FALSE)) %>%
                      mutate(p = param_grid$p[i], a = param_grid$a[i], t = param_grid$t[i])

      out <- rbind(out,stats)
    }
    write_csv(out,"data/fit_test_sim.csv")
} else {
  out <- read_csv("data/fit_test_sim.csv")
}

# format for plotting
pdf <- out %>%
#  filter(a > .2) %>%
  select(parameter = term, value = estimate, actual_p = p, actual_a = a, actual_t = t) %>%
  spread(parameter, value) %>%
  rename(estimated_a = accuracy, estimated_p = p, kappa_a = `root kappa`, estimated_t = t) %
  mutate(Sim = row_number()) %>%
```

```
  gather(parameter, value, -Sim) %>%
  separate(parameter, into = c("type","parameter"), sep = "_") %>%
  spread(type, value)

  ggplot(pdf, aes(x = actual, y = estimated, group = actual)) +
    geom_boxplot() +
    geom_abline(slope = 1, intercept = 0) +
    facet_grid(. ~ parameter) +
    theme_bw()
```

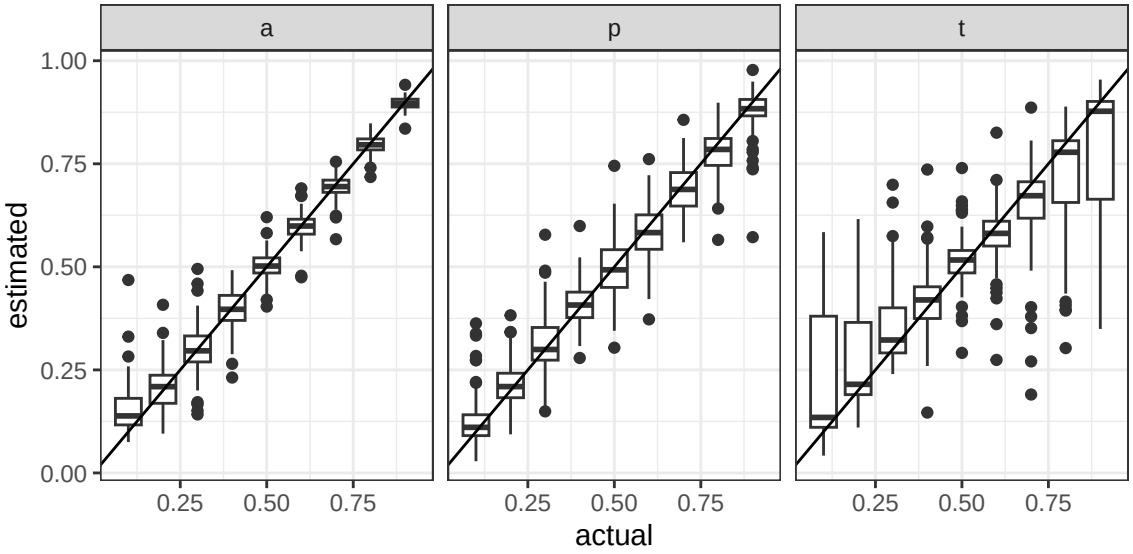

Figure 4: Box and whisker plots show parameter estimates from simulations of rater data *t*-*a*-*p* values ranging from .1 to .9. The diagonal line marks perfect estimates.

The accuracy measure a and the Class 1 "guess rate" p are stable across scenarios in figure 4, but the estimated true fraction of Class 1 cases t is sensitive to values of a near zero. To see this, substitute $a = 0$ into the likelihood function to get

$$\begin{aligned}\ell(t, p; a = 0, k_1, \ldots, k_N) &= \sum_{i=1}^{N} \log\left(t\binom{R}{k_i}p^{k_i}(p')^{R-k_i} + t'\binom{R}{k_i}p^{k_i}(p')^{R-k_i}\right) \\ &= \sum_{i=1}^{N} \log\left(\binom{R}{k_i}p^{k_i}(1-p)^{R-k_i}\right)\end{aligned}$$

showing that t is under-determined when $a = 0$, and we should expect poor behavior as a nears zero. This is intuitive: if the raters are only guessing, they should give us no information

about the true C_1 rate. If the data in figure 4 are filtered to $a > .2$ the estimates of t greatly improve. Aside from extreme values of a affecting the estimation of t, a visual inspection of the scatter plots of the parameter estimates shows no correlations.

6 Indentifiability

The tap model is not identifiable in the sense that, given a set of ratings, the likelihood function is not uniquely maximized for a single choice of (t, a, p). As an example, consider a set of ratings of subjects with two raters each, where for each subject one rater assigns Class 1 and the other rater assigns Class 0. The likelihood function is the product of the likelihood for each subject, and these will be identical, so maximizing likelihood for one subject is the same as for the whole data set. By symmetry, $a \leq 1/2$ and likelihood will be maximized when $a + \bar{a}p = 1/2$ so that any value of $p = \frac{1/2-a}{1-a}$ is a solution. Intuitively, this choice ranges from accuracy of .5 and $p = 0$ to accuracy of 0 and $p = 1/2$.

Other non-identifiable cases occur when one or more of the tap parameters is zero or one. If a = 1, it doesn't matter what p is, for example. These degenerate cases aside, the tap paramterization of the binomial mixture has an advantage over the usual way this is done, to estimate the mean of each of the two binomial distributions individually. If $Pr[mixture] = t\text{binom}(\mu = c_1) + (1-t)\text{binom}(mu = c_0)$ we must then add the constraint that $\mu_1 \geq \mu_2$ to avoid label switching. The tap parameterization allows all three of the parameters to be in [0,1] without such a problem, because a is the difference between the two means: $c_1 - c_0 = a + \bar{a}p - \bar{a}p$.

Non-uniqueness is a well-known problem with mixture models, and it is less of a problem for Bayesian MCMC estimation, which provides a full posterior distribution for the parameters, which may be multi-modal if there are two competing solutions. This gives us a way to detect degenerate solutions and look for a different model if desired.

In the case where we allow individual parameters for accuracy and random class assignment, the t-a0a1-p0p1 case, we may get data sets that are not identifiable for label-switching reasons. That is, we may be able to find the maximum likelihood mixture of binomial distributions, but not know which is Class 0 and which is Class 1. This is not a model problem, but a data problem.

With the $a0, a1, p0, p1$ parameters, label switching means that we swap $\bar{t}$ for t and the two corresponding means swap as well. That requires some second set of the parameters $a0', a1', p0', p1'$ such that

$$a_1 + \bar{a_1}p_1 = \bar{a_0}'p_0'$$
$$a_1' + \bar{a_1}'p_1' = \bar{a_0}p_0$$

Then we can swap the primes for the non-primes, and exchange t for 1-t and the distribution is the same. There are four free parameters and two constraints shown above, plus the constraints that all parameters are in [0,1]. A example of a label-switching set is when $\mu_0 = \bar{a_0}p_0 > a_1 + \bar{a_1}p_1 = \mu_1$, so that there are more class 1 ratings for class 0 cases that there are for true class 1 cases. Suppose a0 = .5, a1 = .2, p0 = .7 and p1 = .1, we have $\mu_0 = (.5)(.7) = .35$ and $\mu_1 = .2 + (.8).1 = .28$, so the mean of the class 1 ratings is to the left of the mean of the class 0 ratings. This can't happen with the three-parameter model. It's not a problem with the model, however, but an expression of the ratings and raters. For example, suppose these are wine tasting ratings, and it's easier to for judges to agree on poor wine than excellent wine (this seems to be the case in practice). Suppose further that some of the raters are motivated to over-rate poor wines, pushing up the p0 rate to .7 from some lower value. This rating inflation makes the poor wines seem better than they are, and by comparison the good wines seem worse. From the data, we can't tell which class is which under these model assumptions and with this data. However, the MCMC results will show us a multi-model distribution.

[I'm not sure we can ignore t, because the rows of data create the likelihood. Need to test this with real data using the simulator]

7 Overdispersion

The tap model assumes fixed averages over raters and subjects for the three parameters. The most sensible of these assumptions is that there is a single value for t that represents the fraction of Class 1 cases, although this idea has been challenged by @???. That leaves two parameters that are certainly oversimplified in the tap model, so that counts of Class 1 ratings per subject are likely to have more variance than a binomial model would. This is due to anticipated variance in rater ability and the difficulty in rating subjects, resulting in overdispersion. A general approach to this problem is to allow each rater to have a different accuracy rate a_j and each subject to have a different guessing rate p_i. This is the hierarchical model, which is described later.

Original suggestion to use beta-binomial @williams1975394, modern look at it in @ascari2021new.

8 Other Properties

As is shown in Appendix A, rater accuracy a is proportional to the correlation between the true and assigned classifications. If C is the true classification and T is the assigned classification, then

$$\text{Cor}(T, C) = a\frac{\sigma_T}{\sigma_C}$$

The t-a-p model focuses on raters more than subjects. For example, some rater models include a difficulty parameter for each subject, assuming that some are harder to classify than others. The t-a-p model does not include this, per se, but a random-effects version of the model is introduced later that allows for subject-specific coefficients p_i, which adjusts the guessing rate per subject when a classification is made inaccurately. See @paun2018comparing for a discussion of difficulty parameters in machine-learning models similar to tap.