A Bayesian & EHR-derived Latent Phenotyping model for COPD

Kristen McGreevy
Ernest Shen

August 15, 2019

Contents

Introduction

Chronic obstructive pulmonary disease (COPD) is a chronic-life limiting illness that affects around 16 million adults in the US. The disease terminology is an umbrella term that describes progressive lung diseases including emphysema, chronic bronchitis, refractory (non-reversible) asthma, and some forms of bronchiectasis. Currently, COPD is the fourth most frequent cause of death worldwide and corresponds to $15 billion in direct healthcare costs. [1] At the same time, with 1 in 5 smokers developing COPD and with the continued high prevalence of smoking, tobacco control efforts are unlikely to eliminate the problem of COPD in the foreseeable future. [2] Furthermore, it is estimated that 50% of people with COPD are not be diagnosed and do not receive treatment. [1]

Spirometry is the "Gold Standard" for testing lung function and for diagnosing patients with COPD. The test involves forcefully exhaling into the device and records two measurements. One is forced vital capacity (FVC), which is the amount of air that you can forcefully exhale. This is similar to the total volume in your lungs. The second measure is forced expiratory volume one (FEV1), which is the amount of air you can force from your lungs in one second. To diagnose someone as COPD, the ratio of these two measurements is taken as FEV1/FVC, which is a percentage of the lung air volume you can exhale in 1 second. For healthy people, this should be close to 1, indicating almost all air can be exhaled in 1 second. People are diagnosed as having COPD when their FEV1/FVC is below 0.7 (70%).

	FEV1/FVC	FEV1 % pred
Normal	> 0.95	≥ 80
Mild COPD	≤ 0.7	≥ 80
Moderate COPD	≤ 0.7	50 - 80
Severe COPD	≤ 0.7	30 - 50
Very Severe COPD	≤ 0.7	< 30

Motivation

While this test is good at indicating a person's lung function, not all patients receive this test and the observation of this test in EHR is patient-driven. In actuality, only about 20% of COPD patients have spirometry measures in their EHR. The missingness of patients' spirometry is likely Missing Not at Random (MNAR). Intuitively, healthy people without COPD may never have a spirometry test performed. Other factors, such as ethnicity, may also prevent someone from receiving care, lowering their probability of having a test performed. Because of this knowledge, the missingness of the spirometry test is *dependent* on factors like the COPD disease state. Understanding the relationship of spirometry test missingness will improve models that seek to identify a person's COPD phenotype.

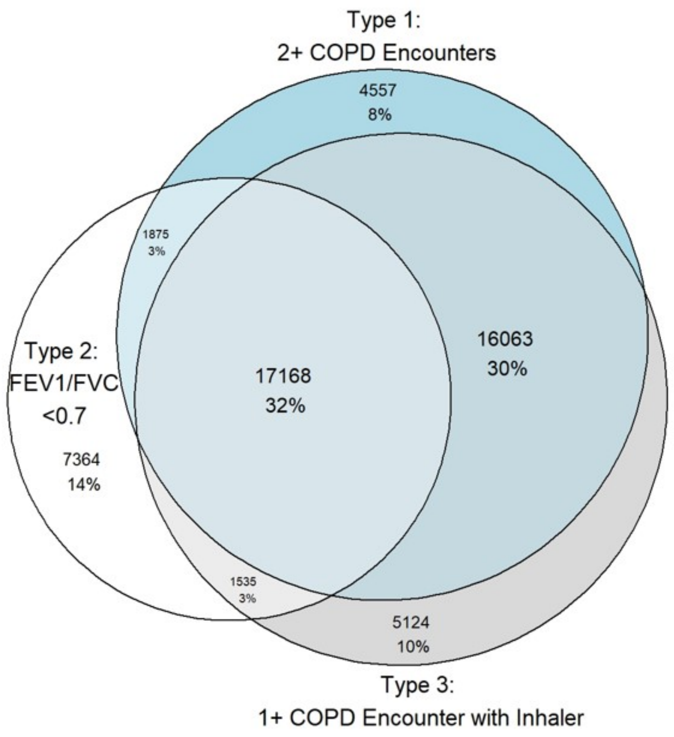

Figure 1. Venn Diagram of Rule-based methods for diagnosing COPD from Electronic Health Records.

If only the Gold Standard (Type 2) in Figure 1 was used to diagnose patients from EHR, only 52% of the COPD population would be correctly identified since 48% of COPD patients are missing a spirometry test. Therefore, there is a large need to improve diagnosis of COPD in healthcare using EHR.

Research Objective

To develop a model that describes the probability somebody has COPD from their observed and missing Electronic Health Records (EHR).

> A **Bayesian Latent Variable Analysis** will be used to understand the relationships among EHR variables. This methodology will relate the dependent relationships among medications, comorbiditites, and diagnostic codes to the underlying COPD disease status. Researchers have used this methodology to diagnose children with Type 2 Diabetes using Electronic Health Records.[3]
>
> A model that successfully relates EHR variables to a COPD phenotype will provide an individualized probability of COPD. This probability will also retain the uncertainty in the measure, as a probability close to 50% indicates uncertainty in a person's disease status.

The figure below displays the conceptual model framework in a Directed Acyclic Graph (DAG). Observed characteristics are modeled as their own function, and then the final likelihood for each person is calculated from all the models being combined.

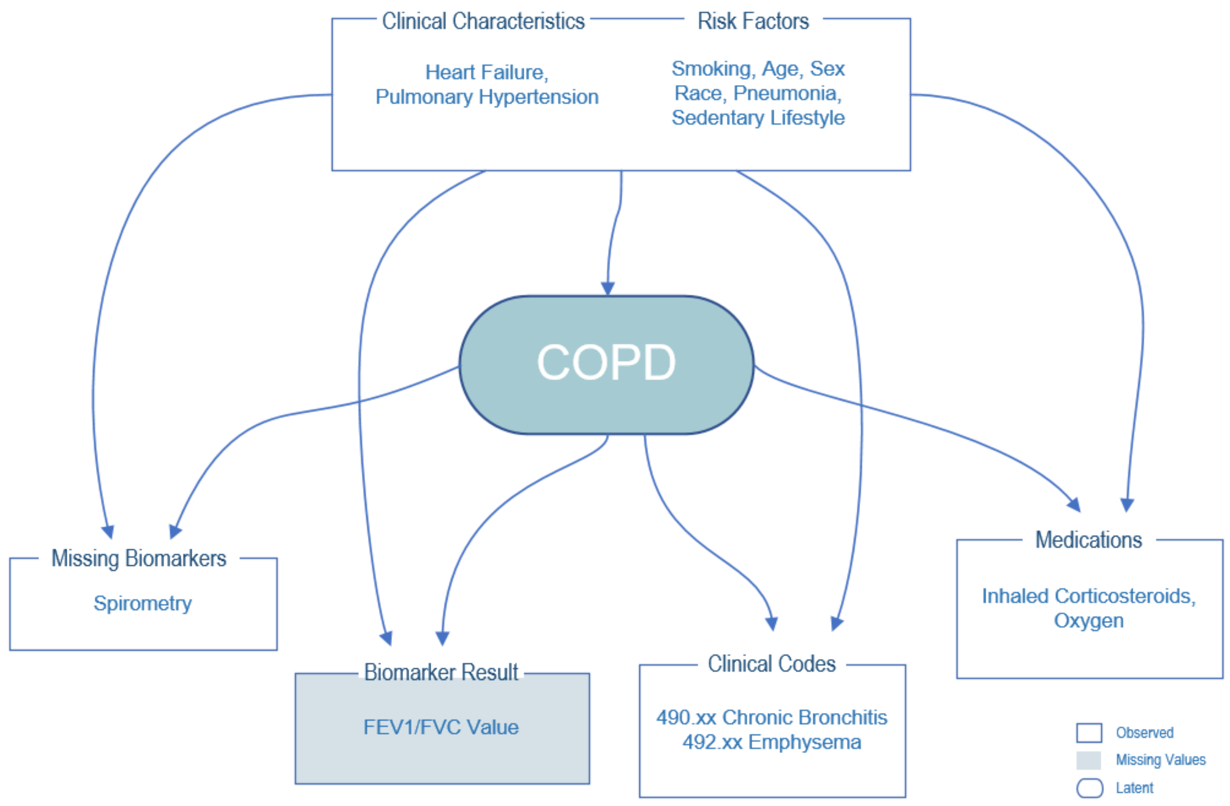

Figure 2. DAG relating Observable EHR characteristics to the latent COPD phenotype

Data

The data used come from a PORTAL-funded study on COPD patients. The data was collected from 2011 - 2015 which used ICD-9 diagnostic codes.

The operational definitions used in this analysis are as follows: Any adult age 40 or older with at least one COPD encounter during the study period was defined as **at-risk**. Any adult age 40 or older with 2 encounters for COPD during the study period OR pulmonary function testing demonstrating fixed obstruction (defined by an FEV1 / FVC ratio < 0.7) OR EHR problem list active designation of COPD OR any one visit AND a dispensing of inhaled therapy consistent with COPD was defined as **having COPD**. When building the model described below, the definition of having COPD was those patients with highly-specific COPD, which is having 2 or more encounters, an inhaler, and a FEV1/FVC ratio < 0.7.

Patients were included in the dataset if they were classified as at-risk, which for this purpose was over 40 years of age and had at least one COPD encounter. There were 56k observations in this dataset. 53k satisfied at least one rule-based decision rule for COPD. 7k had highly specific COPD. Variables that were extracted include age, insurance type, smoking, vaccination, hospital, PCP, or specialty doctor visits, BMI, self-reported exercise, comorbidities, asthma, heart failure, healthcare utilization, inhalers, prescriptions, and FEV1/FVC (when available).

Additional variables were made in SAS Enterprise by Kristen McGreevy. Indicator variables were made that reduced the variable information into binary categories. Indicator variables for missing covariates were made. Variables were made that adjusted for time to allow for relative comparisons among patients. For example, number of PCP visits was converted to number of PCP visits per member-year. Then, the average visits/year was used to classify people into binary categories of above or below average. This was is important because observations did not have the same length of time observed over the 2011-2015 period.

Variables for education and income were geocoded variables, meaning they were *representative* values based on census blocks. Therefore, we did not have individual level data regarding these constructs. We excluded these variables because of lack of interpretability, lack of reproducibility by other researchers without census data, and anticipated lack of information that would be gained.

A total of 228 variables were available for inclusion in models, however redundant variables with high collinearity were removed, resulting in approximately 140 uniquely informative variables.

Model Specification

The relationship we seek to describe is that of both observed and missing components in EHR data that can identify someone with COPD. There is a dependent relationship among the observable EHR traits and the phenotype of interest, COPD, as shown in Figure 2. The presence/absence of a spirometry test, the spirometry result, diagnostic ICD-9 codes, and COPD-specific prescriptions are dependent on an individual's COPD disease status. Risk factors and clinical characteristics can also contribute to the COPD disease state and influence the other observable traits. Because many of the observable traits in EHR data can be influenced by other factors, the dependency among the factors and outcome need to be modeled as such. Thus, a Bayesian Latent Variable Analysis is the most appropriate statistical method. For information regarding this method, please reference >>><<<.

Let COPD be the Latent Phenotype of interest. The equation we are extracting and applying is the posterior likelihood of the phenotype based on a person's demographic characteristics, biomarker availability (spirometry), the biomarker value, diagnostic and clinical codes, and prescriptions from EHR.

Let

i	index patients
$\boldsymbol{X}$	be the set of covariates for an individual
D	index the COPD disease states, where 0 means no COPD and 1 indicates having COPD
R	indicate the presence or absence of the spirometry biomarker test
Y	be the FEV1/FVC value from the spirometry test
	β^D, β^R, and β^Y denote the association between patient characteristics and the phenotype of interest, the underlying phenotype and availability of biomarkers, and biomarker values, respectively.
W	indicate ICD-9 clinical codes which are indexed by k
P	indicate medications or prescribed therapies which are indexed by l
	$\boldsymbol{\beta}_k^W = (\beta_{k0}^W, ..., \beta_{k,M+1}^W)$ and $\boldsymbol{\beta}_l^P = (\beta_{l0}^P, ..., \beta_{l,M+1}^P)$ denote the association between patient characteristics to clinical codes and medications, respectively.

Then the Likelihood of the ith patient's COPD Phenotype given his/her observable traits is modeled as

$$L(\beta^D, \beta^R, \beta^Y, \boldsymbol{\beta}^W, \boldsymbol{\beta}^P, \tau^2 | \boldsymbol{X_i}) = \sum_{d=0,1} \Bigg[P(D_i = d | \beta^D, \boldsymbol{X}_i) \tag{1}$$

$$f(R_i | D_i = d, \boldsymbol{X}_i, \beta^R) f(Y_i | D_i = d, \boldsymbol{X}_i, \beta^Y, \tau^2)^{R_i} \tag{2}$$

$$\prod_{k=1}^{K} f(W_{ik} | D_i = d, \boldsymbol{X_i}, \boldsymbol{\beta}_k^W) \prod_{l=1}^{L} f(P_{il} | D_i = d, \boldsymbol{X}_i, \boldsymbol{\beta}_l^P) \Bigg] \tag{3}$$

(1) There are two disease states, having COPD and not having COPD, which are modeled as Bernoulli random variables. The likelihoods conditioned on each disease state are calculated and summed across to calculate a person's final likelihood of COPD.

(2) Biomarker availability is a Bernoulli model (having spirometry or not). The biomarker value is Normally distributed. If a person does not have a spirometry test, the second likelihood equation is raised to 0, therefore only information from the *lack* of a biomarker is contributed to their likelihood.

(3) Clinical codes and prescriptions are Bernoulli models. Each code and prescription have their own relation and dependence on the latent phenotype, therefore there are K and L likelihood equations multiplied based on the number of clinical codes and prescriptions, respectively.

The model specifies that the mean biomarker levels are shifted by a quantity β^Y for patients with the phenotype of interest compared to those without COPD. Similarly, sensitivity and specificity of binary indicators for clinical codes, medications, and presence of spirometry (biomarker) are given by combinations of regression parameters. For instance, in a model with no patient covariates, specificity of the kth code is given by 1 - expit(β^W_{k0}) while sensitivity is given by expit($\beta^W_{k0} + \beta^W_{k1}$). expit refers to the equation that converts logistic regression parameters to probabilities, where $\text{expit}(\cdot) = \frac{exp(\cdot)}{1+exp(\cdot)}$

This model allows for a unique combination of available data elements for each patient, which incorporates the missing data relationships into the likelihood instead of imputing the value or excluding an observation. The likelihood for each individual consists of the product of the likelihood contributions for all available measurements for that individual for each variable type. If the jth biomarker is missing, $R_{ij} = 0$, which results in exclusion of the likelihood contribution for Y_{ij}.
Note: Our model has only one biomarker (j=1) but is used as an example because clinical codes and prescriptions are available for all observations.

Prior knowledge about the classification accuracy of biomarkers and clinical codes can be incorporated through suitable choice of priors for these parameters. In general, for a normally distributed biomarker with mean and variance given by μ_1 and σ_1^2, in individuals with COPD and μ_0 and σ_0^2 in individuals without COPD, the area under the curve (AUC) is given by $\Phi(\frac{\mu_1-\mu_0}{\sqrt{\sigma_1^2+\sigma_0^2}})$ where $\Phi(\cdot)$ represents the standard normal cumulative distribution function. We assume a common variance between COPD phenotypes and a difference in biomarker means of β^Y, which corresponds to an AUC of $\Phi(\frac{\beta^Y}{\sqrt{2\tau^2}})$.

Model Fitting

There are two unique R scripts that have been developed using the code provided in Hubbard et al. (2019) as a reference. It includes functions and JAGS model specifications to estimate a Bayesian Latent Variable model for phenotyping COPD. The model joins information from demographic characteristics, biomarker availability (spirometry), the biomarker value, clinical codes, and prescriptions.

Parameter estimation is carried out using Markov Chain Monte Carlo (MCMC) methods. Specifically, Just Another Gibbs Sampler (JAGS) was used through the R package runjags. The posterior, patient-specific probability of COPD membership, expit($\boldsymbol{X}_i\boldsymbol{\beta}^D$), was sampled to describe a patient's latent COPD phenotype from a probabilistic manner. The distribution of likelihoods and the location of a person's likelihood within the distribution determines their binary phenotype using a cutpoint that was optimized for both sensitivity and specificity in the code below.

R Packages

Computation was performed on a Windows machine. The packages `rjags` and `runjags` are built for Windows systems. If on a Linux machine, `R2jags` can be used. Software also exists for running the opposite machine virtually on your own, which may be of interest. `rjags`, `runjags`, `MCMCvis`, `ggplot2`, and `dplyr` are packages used in this document.

source_functions R Script

This first script defines the functions to be used in the JAGS model, which is called in the `run_file_COPD` script.

`expit` is the logistic link function used to sample from the posterior distribution. This equation converts logistic regression parameters to probabilities, where $\text{expit}(\cdot) = \frac{exp(\cdot)}{1+exp(\cdot)}$.

```
library(rjags)
library(runjags)
```

```r
expit <- function(x) exp(x) / (1 + exp(x))
```

`parse_miss` is a function for identifying patterns of missingness and converts the data into lists for use in JAGS estimation.

```r
parse_miss <- function(dat){

  # Get names of columns that contain missing values
  miss_var_names <- colnames(dat)[ apply(dat, 2, function(x) sum(is.na(x))) > 0]
  # extract the original columns from the dataset with missing values
  miss_dat <- dat[, miss_var_names]
  # concatenates output to a single string for each row if missing
  dat$comb <- apply(miss_dat, 1, function(x) paste0(is.na(x), collapse = '-'))
  # all the unique combinations of missingness
  map <- unique(dat$comb)
  map2 <- unique(dat$comb)
  map <- 1:length(map) # changes it to a vector of integers
  # names gives each elementbased on logic of missing
  names(map) <- map2
  dat$comb <- map[dat$comb]

  #split divides the data in the dat into the groups defined by dat$comb
  split_dat <- split(dat, dat$comb)
  # returns the result of the as.list function to split_dat
  dl <- lapply(split_dat, as.list)
  # combines the groupings that were defined into ll
  ll <- do.call('c', dl)
  # substring function is taking the names of the groups which are separated by periods.
  names(ll) <- paste0(substring(text = names(ll),
                      first = as.numeric(gregexpr("\\.", names(ll) )) + 1,
                      last = nchar(names(ll))),
                    substring(text = names(ll),
                      first = 1,
                      last = as.numeric(gregexpr("\\.", names(ll))) - 1))

  ind <- unlist(lapply(ll, function(l) length(l) != sum(is.na(l)) ))
  ll <- ll[ind]
  return(ll)
}

# counts of missing biomarkers in data.frame
count_miss <- function(d){
  miss_FEV1 <- mean(is.na(d$FEV1_FVC))
  return(mFEV1 = miss_FEV1)
}
```

JAGS Model

We specify the JAGS model using JAGS syntax. The logit model is used because we are interested in a binary outcome of COPD (has / does not have). `Npat` is the number of observations who are missing the FEV1/FVC value. The model is organized in 5 steps.

1. Define Variables' Dependency
2. Define Variables' Distributions

3. Define Latent Phenotype and Missing Biomarker Models
4. Specify Priors
5. Calculate Latent Phenotype Probability

Syntax

Standard syntax in JAGS differs slightly from R. Information/ precision is specified instead of variance. Define equations before you define variables or their distribution. JAGS runs bottom-up, and defining equations and variables in the correct order will reduce computation time.

STEP 1

Define logit models for variables you expect to differ based on an individual's COPD status. We expect medications that help COPD and comorbidities associated with the disease to be more prevalent in those patients who have COPD, therefore these variables are dependent on their latent phenotype.

$$\texttt{logit(}\underbrace{\texttt{pSABA[i]}}_{\substack{\text{probability of}\\\text{SABA prescription}}}\texttt{) <- }\underbrace{\texttt{codeSABA\_b\_int}}_{\substack{\text{intercept: prescription}\\\text{rate for people without}\\\text{COPD}}} + \underbrace{\texttt{codeSABA\_b\_dm}}_{\substack{\text{slope: difference in}\\\text{prescription rate for}\\\text{people with COPD}}} * \underbrace{\texttt{latentCOPD[i]}}_{\substack{\text{latent COPD status}\\\text{(0 or 1)}}}$$

The logit models in this section describe the probability of a prescription or clinical code given their COPD disease state.

```
jagsmod_COPD <- "
model{
# Specify likelihood
for(i in 1:sum(Npat)){

logit(pCOPD[i]) <- codeCOPD_b_int + codeCOPD_b*latentCOPD[i] # COPD by definition
logit(pSABA[i]) <- codeSABA_b_int + codeSABA_b_dm*latentCOPD[i] # SABA prescription
logit(pSAAC[i]) <- codeSAAC_b_int + codeSAAC_b_dm*latentCOPD[i] # SAAC prescription
logit(pLABA[i]) <- codeLABA_b_int + codeLABA_b_dm*latentCOPD[i] # LABA prescription
logit(pLAMA[i]) <- codeLAMA_b_int + codeLAMA_b_dm*latentCOPD[i] # LAMA prescription
logit(pICS[i]) <- codeICS_b_int + codeICS_b_dm*latentCOPD[i]    # ICS prescription
logit(pSCS[i]) <- codeSCS_b_int + codeSCS_b_dm*latentCOPD[i]    # SCS prescription
logit(pLSCS[i]) <- codeLSCS_b_int + codeLSCS_b_dm*latentCOPD[i] # LSCS prescription
logit(pTHEO[i]) <- codeTHEO_b_int + codeTHEO_b_dm*latentCOPD[i] # Theopylline prescription
logit(pOXYG[i]) <- codeOXYG_b_int + codeOXYG_b_dm*latentCOPD[i] # Oxygen Use

# comorbidities by charlson quartiles
logit(pcharlson_q3_ind[i]) <- codecharlson_q3_ind_int + codecharlson_q3_ind_b*latentCOPD[i]
logit(pcharlson_q2_ind[i]) <- codecharlson_q2_ind_int + codecharlson_q2_ind_b*latentCOPD[i]
logit(pcharlson_q4_ind[i]) <- codecharlson_q4_ind_int + codecharlson_q4_ind_b*latentCOPD[i]
```

The probabilities (such as `pCOPD`) etc are not elements in the dataset, these are estimated. Parameters beginning with `code` are also estimated in the JAGS model; these are intercept and slope coefficients. `latentCOPD` is 0 or 1 and is modeled iteratively to estimate the parameters.

STEP 2

We define the distribution that variables in our dataset originate from. In our model, we converted variables into indicators, therefore they should be modeled as Bernoulli random variables. Variables on the left-hand side (LHS) are variables in the dataset, and the probability associated with each random variable is the probability used in the logit model from STEP 1.

$$\underbrace{\texttt{meds1[i]}}_{\text{Short-acting beta-agonist variable}} \sim \underbrace{\texttt{dbern(}}_{\text{bernoulli distributed}} \underbrace{\texttt{pSABA[i]}}_{\text{probability of SABA prescription}})$$

`meds1` is defined as a Bernoulli random variable, which means it can take the values 0 or 1 with a probability `pSABA`. This probability is dependent on COPD as defined in the equations above.

```
COPD[i] ~ dbern(pCOPD[i])
meds1[i] ~ dbern(pSABA[i])
meds2[i] ~ dbern(pSAAC[i])
meds3[i] ~ dbern(pLABA[i])
meds4[i] ~ dbern(pLAMA[i])
meds5[i] ~ dbern(pICS[i])
meds6[i] ~ dbern(pSCS[i])
meds7[i] ~ dbern(pLSCS[i])
meds8[i] ~ dbern(pTHEO[i])
oxygen[i] ~ dbern(pOXYG[i])

charlson_q3_ind[i] ~ dbern(pcharlson_q3_ind[i])
charlson_q2_ind[i] ~ dbern(pcharlson_q2_ind[i])
charlson_q4_ind[i] ~ dbern(pcharlson_q4_ind[i])
```

STEP 3

Define your models of interest. We have two models, one is the model that relates missing the biomarker to factors such as person's latent COPD status and their smoking status. For COPD, this is a person's presence or absence of FEV1/FVC, the "Gold Standard". This is not the value of the biomarker, just whether it exists or not. Recall the presence and value make the biomarker doubley informative.

$$\texttt{logit(} \underbrace{\texttt{pMiss\_FEV[i]}}_{\text{probability of missing spirometry test}} \texttt{) <-} \underbrace{\texttt{a0}}_{\text{intercept: missing rate for people without COPD}} + \underbrace{\texttt{a\_COPD\_miss}}_{\text{slope: difference in missing rate for people with COPD}} * \underbrace{\texttt{latentCOPD[i]}}_{\text{latent COPD status (0 or 1)}} + \underbrace{\texttt{...}}_{\text{other variables in dataset that influence missingness}}$$

The second model relates the latent phenotype to risk factors available in EHR. Variables included in these models must *not* be missing and must *not* have a prior defined above (because then the model is a loop). If covariates have missing values, use one or both approaches:

a: make indicator variables for covariate missingness.
  once you fit initial models, you can check the assumption of missingness at random

b: impute missing values before fitting the JAGS model.

We recommend doing both and compare the results.

Note: Be intentional with your research. You must specify the distribution variables originate from, and if they can be 0, 1, or missing, that is not a Bernoulli Random Variable because three distinct states are possible. You can make the model unnecessarily complex by modeling the variable as multinomial, or you can make indicator variables. If there is reason to doubt a variable is missing at random, the assumptions needed for imputation are not met. Methods have been developed that impute MNAR variables, however these methods are not used in this research project.

"Do the best experiments you can, and always tell the truth. - Sydney Brenner"

$$\texttt{logit(}\underbrace{\texttt{rho\_COPD[i]}}_{\text{probability of latent COPD phenotype for observation i}}\texttt{) <-}\underbrace{\texttt{r0.5[i]}}_{\text{intercept: individual random effects}} + \underbrace{\ldots}_{\text{other variables in dataset that influence COPD disease state}}$$

```
# Missing in biomarker FEV1/FVC #
logit(pMiss_FEV[i]) <- a0 + a1*smoke_miss[i] + a2*smoke_ind[i]
+ a_COPD_miss*latentCOPD[i] + a4*primary_63[i] + a5*primary_miss[i]
+ a6*specialty_12[i] + a7*specialty_miss[i] + a8*ins_dual_ind[i]
+ a9*ins_medicaid_ind[i] + a10*ins_medicare_ind[i] + a11*hbpc[i]
+ a12*pneumonia[i] + a13*influenza[i] + a14*pulmonary_rehab[i]
+ a17*age_69[i]
+ a18*age_79[i] + a19*age_80plus[i] + a20*race_amind[i]
+ a21*race_asian[i] + a22*race_black[i] + a23*race_hawai[i] + a24*race_multi[i]
+ a25*race_unkno[i]
+ a26*depress_miss[i] + a27*hf_miss[i] + a28*pulhyper_miss[i]
+ a29*diab_miss[i] + a30*asthma_miss[i] + a31*bronch_miss[i]
+ a32*depress_val[i] + a33*hf_val[i] + a34*pulhyper_val[i]
+ a35*diab_val[i] + a36*asthma_val[i] + a37*bronch_val[i]

# similar to STEP 2 above, specify the variable in your dataset with the distribution
# and probability above
FEV1_miss_ind[i] ~ dbern(pMiss_FEV[i])

# Latent PHENOTYPE 5 / COPD BY BIOMARKER DEFINITION model#
logit(rho_COPD[i]) <-  r0.5[i] + r2.5*smoke_ind[i] + r3.5*smoke_miss[i]
+ r4.5*age_69[i] + r5.5*age_79[i] + r6.5*age_80plus[i] + r7.5*hispanic_ind[i]
+ r8.5*ins_dual_ind[i] + r9.5*ins_medicaid_ind[i] + r10.5*ins_medicare_ind[i]
+ r15.5*influenza[i] + r16.5*pneumonia[i]
+ r17.5*pulmonary_rehab[i] + r18.5*marriage[i] + r19.5*marriage_miss[i]
+ r20.5*primary_63[i]
+ r145.5*hf_miss[i]
+ r146.5*pulhyper_miss[i] + r147.5*diab_miss[i] + r148.5*depress_miss[i]
+ r150.5*chronic_pain_miss[i] + r151.5*asthma_miss[i]
+ r152.5*bronch_miss[i]
+ r155.5*hf_val[i] + r156.5*pulhyper_val[i] + r157.5*diab_val[i]
+ r158.5*depress_val[i] + r159.5*anxiety_val[i] + r160.5*chronic_pain_val[i]
+ r161.5*asthma_val[i] + r162.5*bronch_val[i]
+ r164.5*pulm_ind_val[i]

}
# This is the end of specifying the likelihood for the bayesian model.
```

For people who have spirometry data, their FEV1/FVC value is believed to follow a normal distribution. The model indicates the mean FEV1/FVC value will be shifted based on the latent COPD phenotype, but the variability is assumed to be the same.

$$\underbrace{\texttt{FEV1\_FVC[i]}}_{\text{FEV1/FVC value for observation i}} \sim \underbrace{\texttt{dnorm(}}_{\text{normally distributed}} \underbrace{\texttt{FEV1\_b\_int}}_{\text{average FEV1/FVC value for someone without COPD}} + \underbrace{\texttt{FEV1\_b\_dm}}_{\text{difference in average FEV1/FVC value for people with COPD}} * \underbrace{\texttt{latentCOPD[i]}}_{\text{latent COPD status}}, \underbrace{\texttt{FEV1\_tau}}_{\text{Fisher's information (inverse-variance)}})$$

```
# Loop over subjects with non-missing FEV1/FVC #
for (i in 1:(Npat[1])){
  FEV1_FVC[i] ~ dnorm(FEV1_b_int + FEV1_b_dm*latentCOPD[i], FEV1_tau)
}
```

STEP 4A

Specify priors for the coefficients in STEP 1. Because we used logit models, the prior values are calculated as $ln(\frac{\text{prevalence}}{1-\text{prevalence}})$. Intercepts are expected to be around the population average (because that corresponds to people without a latent COPD phenotype). The slope priors are relatively uninformative. We don't expect coefficients to have drastic impact, therefore priors are specified to be centered on 0 with variance of 10. In JAGS, `dnorm(0,0.1)` corresponds to a relatively uninformative normally distributed prior centered on 0, with a variance of 10. We specify precision $(\frac{1}{\text{variance}})$, not variance.

Intercepts are uniformly distributed and slopes are normally distributed.

```
codeCOPD_b_int ~ dunif(-4,-2)        # prevalence between 2 and 5 %
codeCOPD_b_dm ~ dnorm(0,0.1)
codeSAAC_b_int ~ dunif(-7,-2.2)      # 0 - 10 %
codeSAAC_b_dm ~ dnorm(0,0.1)
codeSABA_b_int ~ dunif(-0.62, 0.62)  # first line drug prescription, 35 - 65 %
codeSABA_b_dm ~ dnorm(0,0.1)
codeLABA_b_int ~ dunif(-1.75, -0.85) # 15 - 30 %
codeLABA_b_dm ~ dnorm(0,0.1)
codeLAMA_b_int ~ dunif(-7, -0.85)    # 0 - 30 %
codeLAMA_b_dm ~ dnorm(0,0.1)
codeICS_b_int ~ dunif(-1.75, -0.62)  # 15 - 35 %
codeICS_b_dm ~ dnorm(0,0.1)
codeSCS_b_int ~ dunif(-7, -1.1)      # 0 - 25 %
codeSCS_b_dm ~ dnorm(0,0.1)
codeLSCS_b_int ~ dunif(-7, -2.2)     # 0 - 10 %
codeLSCS_b_dm ~ dnorm(0,0.1)
codeTHEO_b_int ~ dunif(-7, -2.95)    # 0 - 5 %
codeTHEO_b_dm ~ dnorm(0,0.1)
codeOXYG_b_int ~ dunif(-2.2, -1.1)    # 10 - 25
codeOXYG_b_dm ~ dnorm(0,0.1)

codecharlson_q3_ind_int ~ dunif(-1.39, -0.85)      # 20 - 30
codecharlson_q3_ind_b ~ dnorm(0,0.1)
codecharlson_q2_ind_int ~ dunif(-1.39, -0.85)      # 20 - 30
codecharlson_q2_ind_b ~ dnorm(0,0.1)
codecharlson_q4_ind_int ~ dunif(-1.39, -0.85)      # 20 - 30
codecharlson_q4_ind_b ~ dnorm(0,0.1)

FEV1_b_int ~ dnorm(70,10) # informative mean and informative information
FEV1_b_dm ~ dnorm(-0.2,1) # informative prior based on AUC = 0.95
FEV1_tau ~ dgamma(0.001,1000)
FEV1_sigma <- 1/FEV1_tau    # because we have to specify information matrix
```

STEP 4B

Specify priors for the coefficients in the missing biomarker and latent phenotype models from STEP 3. The missing model has less informative priors. The latent phenotype model uses priors that have high confidence in variables being unimportant. These are centered at 0 with a small variance (0.1). This allows variable selection similar to LASSO methods, where coefficients are forced to zero if they have small impact, resulting in a simpler, more parsimonious model.

$$\underbrace{\texttt{a0}}_{\substack{\text{parameter in missing}\\ \text{biomarker model}}} \sim \underbrace{\texttt{dnorm(}}_{\substack{\text{normally}\\ \text{distributed}}} \underbrace{\texttt{0}}_{\text{zero effect}} , \underbrace{\texttt{0.1}}_{\text{variance of 10}})$$

$$\underbrace{\texttt{r1.5}}_{\text{parameter in latent phenotype model}} \sim \underbrace{\texttt{dnorm(}}_{\text{normally distributed}} \underbrace{0}_{\text{zero effect}} , \underbrace{10}_{\text{variance of 0.1}})$$

```
# missing biomarker model
a0 ~ dnorm(0,0.1)
a1 ~ dnorm(0,0.1)
a2 ~ dnorm(0,0.1)
# ... repeat for all missing spirometry coefficients #
a_COPD_miss ~ dnorm(0,0.1)

# latent phenotype model
r1.5 ~ dnorm(0,10)
r2.5 ~ dnorm(0,10)
r3.5 ~ dnorm(0,10)
# ... repeat for all latent phenotype coefficients #
r164.5 ~ dnorm(0,10)
```

STEP 5

The last part in the model uses all information provided to estimate the probability of each person's latent phenotype. Their latent phenotype is sampled from the bernoulli distribution with the corresponding individual probability `rho_COPD`.

`r0.5[i]` are individual level random effects.

```
# calculate probability of latent COPD for each person
for (i in 1:sum(Npat)){
 latentCOPD[i] ~ dbern(rho_COPD[i])
  r0.5[i] ~ dunif(-7,-1)
}
}
"
```

This is the end of the JAGS model specification. Now that we have made our JAGS model, we can estimate our function using `run.jags()`.

`run_bayes_COPD()` is a wrapper function for `run.jags()`. It takes the dataset and arguments to pass into `run.jags` as input. It samples from the posterior distribution and returns the probability of COPD and other model parameters.

```
run_bayes_COPD<-function(original.data, jagsmod, Burnin, sample, method, adapt, monitor){

  # Store original data set
  temp <- original.data

  # Select only NUMERIC columns of data set needed by JAGS
  pedsdata <- temp[, c("meds1","meds2","meds3","meds4", "meds6", "meds7",
                       "meds8", "oxygen", "ipc",
                       "specialty_12", "specialty_miss", "hbpc", "oppc",
                       "marriage", "marriage_miss",
                       "charlson_q2_ind", "charlson_q3_ind", "charlson_q4_ind",
                       "pneumonia", "influenza", "pulmonary_rehab", "hispanic_ind",
                       "ins_dual_ind", "ins_medicaid_ind", "ins_medicare_ind",
                       "smoke_ind", "smoke_miss", "age_69", "age_79", "age_80plus",
                       "exercise_inactive", "exercise_insufficient", "exercise_miss",
                       "primary_63", "primary_miss", "hispanic_miss",
                       "bmi_under", "bmi_over", "bmi_obese", "bmi_miss",
                       "race_amind", "race_asian", "race_black",
                       "race_hawai", "race_multi", "race_unkno",
                       "hf_miss",
                       "pulhyper_miss", "diab_miss", "depress_miss",
                       "anxiety_miss", "chronic_pain_miss", "asthma_miss",
                       "bronch_miss", "snf_ind_miss", "pulm_ind_miss",
                       "hf_val", "pulhyper_val", "diab_val",
                       "depress_val", "anxiety_val", "chronic_pain_val",
                       "asthma_val", "bronch_val", "snf_ind_val",
                       "pulm_ind_val"
                       )]
  # Create numeric binary indicator variables with any character variables
  # JAGS cannot evaluate almost anything with NA's, so as mentioned previously,
  # ensure the variables have missing indicators or are imputed.

  pedsdata$COPD <- ifelse((!is.na(temp$FEV1_FVC)) & (0 < temp$FEV1_FVC) & (temp$FEV1_FVC<= 70), 1, 0)
  pedsdata$phen1 <- as.numeric(temp$phenotype_1)
  pedsdata$sex <- ifelse((!is.na(temp$SEX)) & (temp$SEX == "female"), 1, 0)
  pedsdata$FEV1_miss_ind <- ifelse(is.na(temp$FEV1_FVC) == TRUE, 1, 0)

  # Create missingness indicators
  pedsdata$FEV1_miss_ind <- ifelse(is.na(temp$FEV1_FVC) == TRUE, 1, 0)

  # Add vector with number of occurences of each missingness pattern to data
  misspat <- paste(pedsdata$FEV1_miss_ind)
  pedsdata <- pedsdata[order(misspat), ]
  outdata <- data.frame(pedsdata,T1D = temp[order(misspat), "FEV1_FVC"])

  dat_list <- list()
  dat_list$`Npat` <- as.vector(table(misspat))
  dat_list <- c(dat_list, as.list(pedsdata))

  # Run JAGS
  results<-run.jags(jagsmod_COPD,
                    monitor = monitor,
                    data = dat_list,
                    burnin = Burnin, sample = sample, n.chains = 3,
                    method = 'rjags', adapt = adapt, silent.jags = TRUE,
                    inits = NA)

  # Format MCMC samples
  results_mcmc <- as.mcmc.list(results)
  mymodel.mcmc <- as.mcmc(results)
  mymodel.mat <- as.matrix(mymodel.mcmc)
  mymodel.dat <- as.data.frame(mymodel.mat)

  output <- list(mcmc_list = results_mcmc, outdata = outdata)

  # specify to return more output from above based on how you need results formatted
  return(output)
  return(mymodel.dat)
}
```

This is the end of the `source_functions_COPD` R script.

run_file R Script

This script will call the `source_functions_COPD` script above, as long as they are saved in the same directory, or you can specify it directly (punny). The Bayesian model will be run, parameters will be monitored for convergence, posterior probabilities will be extracted, and sensitivity and specificity will be calculated.

The parameters to monitor for convergence should be the coefficients in the models specified above.

You may have to increase the memory limit used. I needed to run `memory.limit(5000)` first to ensure enough memory was reserved for vector allocation. Take a random subset to see if the model is correctly specified (highly recommended). Set the seed for reproducibility, but use random.org to generate true random numbers for your seed.

```r
set.seed(0835)
index <- sample(1:nrow(original.data), 10000, replace=FALSE)
random_subset <- original.data[index, ]

source("source_functions_COPD.R")

# Set parameters to monitor for convergence
monitor <- c("rho_COPD","FEV1_b_int","FEV1_b_dm", "FEV1_tau", "codeCOPD_b_int", "codeCOPD_b",
             "codeSAAC_b_int", "codeSAAC_b_dm", "codeSABA_b_int",
             "codeSABA_b_dm", "codeLABA_b_int", "codeLABA_b_dm", "codeLAMA_b_int",
             "codeLAMA_b_dm", "codeSCS_b_int",
             "codeSCS_b_dm", "codeLSCS_b_int", "codeLSCS_b_dm", "codeTHEO_b_int",
             "codeTHEO_b_dm", "codeOXYG_b_int", "codeOXYG_b_dm",
             "codecharlson_q3_ind_int", "codecharlson_q3_ind_b",
             "codecharlson_q2_ind_int", "codecharlson_q2_ind_b",
             "codecharlson_q4_ind_int", "codecharlson_q4_ind_b", "a_COPD_miss",
             "a0", "a1", "a2", "a4", "a5", "a6", "a7",
             "a8", "a9", "a10", "a11", "a12", "a13", "a14",
             "a17", "a18", "a19", "a20", "a21", "a22", "a23", "a24", "a25",
             "a26", "a27", "a28", "a29", "a30", "a31", "a32", "a33", "a34", "a35",
             "a36", "a37",
             "r2.5",
             "r3.5", "r4.5", "r5.5", "r6.5", "r7.5", "r8.5", "r9.5",
             "r10.5", "r15.5", "r16.5", "r17.5", "r18.5", "r19.5",
             "r20.5",
             "r145.5", "r146.5", "r147.5", "r148.5",
             "r150.5", "r151.5", "r152.5",
             "r155.5", "r156.5", "r157.5", "r158.5", "r159.5", "r160.5",
             "r161.5", "r162.5", "r164.5")
```

We are ready to run the latent phenotyping model! There are seven inputs you will need to specify in order for the model to be run.

`original.data`	is the dataframe to build the model from
`jagsmod`	is the JAGS model we specified above
`monitor`	is the list of parameter estimates to monitor for convergence
`Burnin`	is the number of iterations from the MCMC sample before parameter estimates are recorded. This hopefully allows the sampling to get to a higher probability region in the distribution.
`sample`	is the number of samples from the posterior distribution to take
`method`	is the type of sampling method
`adapt`	specifies the number of iterations for the model to begin adaptation. This allows the chain to learn about the proposed distribution and update the sampling scheme accordingly

```
start.time <- Sys.time()
COPD_bayes <- run_bayes_COPD(original.data, jagsmod = jagsmod_COPD,
                             monitor = monitor, Burnin = 1000, sample = 5000,
                             method = 'simple', adapt = 1000)
end.time <- Sys.time()
run.time <- end.time-start.time

# Save MCMC object and data set sorted by missingness pattern
outdata <- COPD_bayes[[2]]
COPD_bayes2 <- COPD_bayes[[1]]

# Extract posterior probabilities of COPD
rho <- COPD_bayes2[[1]][, which(regexpr("rho_COPD", colnames(COPD_bayes2[[1]])) > 0)]
# Extract other model parameters
parm <- COPD_bayes2[[1]][, which(regexpr("rho_COPD", colnames(COPD_bayes2[[1]])) <= 0)]
```

Summarize Posterior Means and CIs

```
# Transform parameters for missing FEV1/FVC to Odds Ratios
parm[, c("a_COPD_miss")] <- exp(parm[, c("a_COPD_miss")])
postmeans <- apply(parm, 2, mean)
postci    <-  apply(parm, 2, quantile, probs = c(0.025, 0.975))


# this includes ONLY the variables that had priors specified
codesens <- apply(parm[,c("codeCOPD_b_int", "codeSAAC_b_int", "codeSABA_b_int",
                          "codeLABA_b_int", "codeLAMA_b_int",
                          "codeICS_b_int", "codeSCS_b_int",
                          "codeLSCS_b_int", "codeTHEO_b_int",
                          "codeOXYG_b_int", "codecharlson_q3_ind_int",
                          "codecharlson_q2_ind_int", "codecharlson_q4_ind_int")]
                  + parm[,c("codeCOPD_b", "codeSAAC_b_dm",
                            "codeSABA_b_dm", "codeLABA_b_dm",
                            "codeLAMA_b_dm", "codeICS_b_dm",
                            "codeSCS_b_dm", "codeLSCS_b_dm",
                            "codeTHEO_b_dm", "codeOXYG_b_dm",
                            "codecharlson_q3_ind_b", "codecharlson_q2_ind_b",
                            "codecharlson_q4_ind_b")],2, function(x){ mean(expit(x)) })
codesens.ci <- apply(parm[,c("codeCOPD_b_int", "codeSAAC_b_int", "codeSABA_b_int",
                             "codeLABA_b_int", "codeLAMA_b_int",
                             "codeICS_b_int", "codeSCS_b_int",
                             "codeLSCS_b_int", "codeTHEO_b_int",
                             "codeOXYG_b_int", "codecharlson_q3_ind_int",
                             "codecharlson_q2_ind_int", "codecharlson_q4_ind_int")]
                    + parm[,c("codeCOPD_b", "codeSAAC_b_dm",
                               "codeSABA_b_dm", "codeLABA_b_dm",
                               "codeLAMA_b_dm", "codeICS_b_dm",
                               "codeSCS_b_dm", "codeLSCS_b_dm",
                               "codeTHEO_b_dm", "codeOXYG_b_dm",
                               "codecharlson_q3_ind_b", "codecharlson_q2_ind_b",
                               "codecharlson_q4_ind_b")],
                    2,function(x){quantile(expit(x), probs = c(0.025, 0.975))})
# only intercepts
codespec <- apply(parm[,c("codeCOPD_b_int", "codeSAAC_b_int", "codeSABA_b_int",
                          "codeLABA_b_int", "codeLAMA_b_int",
                          "codeICS_b_int", "codeSCS_b_int",
                          "codeLSCS_b_int", "codeTHEO_b_int",
                          "codeOXYG_b_int", "codecharlson_q3_ind_int",
                          "codecharlson_q2_ind_int", "codecharlson_q4_ind_int")],
                  2,function(x){mean(1-expit(x))})
codespec.ci <- apply(parm[,c("codeCOPD_b_int", "codeSAAC_b_int", "codeSABA_b_int",
                             "codeLABA_b_int", "codeLAMA_b_int",
                             "codeICS_b_int", "codeSCS_b_int",
                             "codeLSCS_b_int", "codeTHEO_b_int",
                             "codeOXYG_b_int", "codecharlson_q3_ind_int",
                             "codecharlson_q2_ind_int", "codecharlson_q4_ind_int")],
                     2,function(x){quantile(1-expit(x), probs = c(0.025, 0.975))})
```

This is the end of `run_file_COPD` script. At this point, you will have a few useful objects

`COPD_bayes`	is the large list that has 2 elements.
`COPD_bayes2`	is the MCMC list.
`outdata`	is the corresponding data.
`run.time`	is the total computation time
`rho`	contains the observation level probabilities of COPD. Recall values closer to 0.5 indicate higher uncertainty.
`parm`	is an object with the model parameters that were specified in the monitor step above
`codesens`	is the medication and comorbidty sensitivity
`codespec`	is the medication and comorbidity specificity

If you want to have an audible notification when the task finishes running, use package `beepr` (#4 is a personal favorite and is reminiscent of the Cookie Monster).

Variable Selection

Initial Inclusion	All variables.
Exclusion from Latent Model	5 independent runs using 1,000 unique, random observations had at least 4 runs where the parameter estimate converged (Rhat < 1.1) and the estimate was < 0.07.
Inclusion in Missing Model	any missing indicator variable that was influential in the Latent Variable model was included. This meant of the 5 independent runs, at least 3 converged (Rhat < 1.1) and the estimate was ≥ 0.07.

Results

Results are excluded because they have not been published and KP is in the process of applying the results to improve clinical outcomes.

Visualizations

Some visualizations and diagnostics of the posterior.

It is recommended to exclude `rho` variables because each observation has a probability, so a lot of plots would be made. `ISB` allows variables with similar names to be excluded or not.

Diagnostics

Convergence appears well satisfied if Rhat is near 1. Values above 1.1 should be investigated - we suggest looking at traceplots and credible intervals.

```r
library(MCMCvis)
library(ggplot2)

# caterpillar plots of posterior median with 95% Credible Intervals
MCMCplot(COPD_bayes, ISB = FALSE, excl = 'rho', ref_ovl = TRUE,
         rank = TRUE, main="1000 obs, 2000 burnin, 4 chains")

# traceplots of each coefficient for convergence.
MCMCtrace(COPD_bayes,  ISB = FALSE, excl = 'rho', pdf = FALSE)

# Basically the same caterpillar figure as above, but with more control over the
# graphical display. This code will need modified, but an example plot
# for how it would look is displayed below.
ggplot(included_latent, aes(x=parameters, y=mean, color=factor(overlaps_zero))) +
  geom_pointrange(aes(ymin = low, ymax = high)) +
  ggtitle("Latent Model Variables ") +
  labs(x=" ", y="95% Credible Intervals") + theme_bw() +
  theme(axis.text.x = element_text(size=10), plot.title = element_text(lineheight=.8, hjust = 0.5))+
  coord_flip() +
  scale_x_discrete(labels = included_latent$Label) +
  scale_colour_manual(values=c("#1c9099", "#99CCCC"))
```

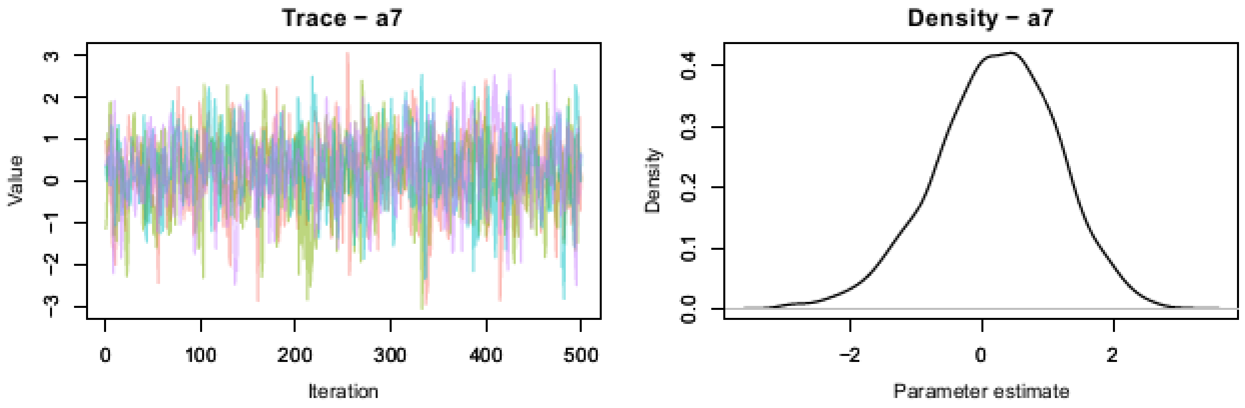

Figure 3. Example traceplot from MCMCtrace that demonstrates good convergence and mixing

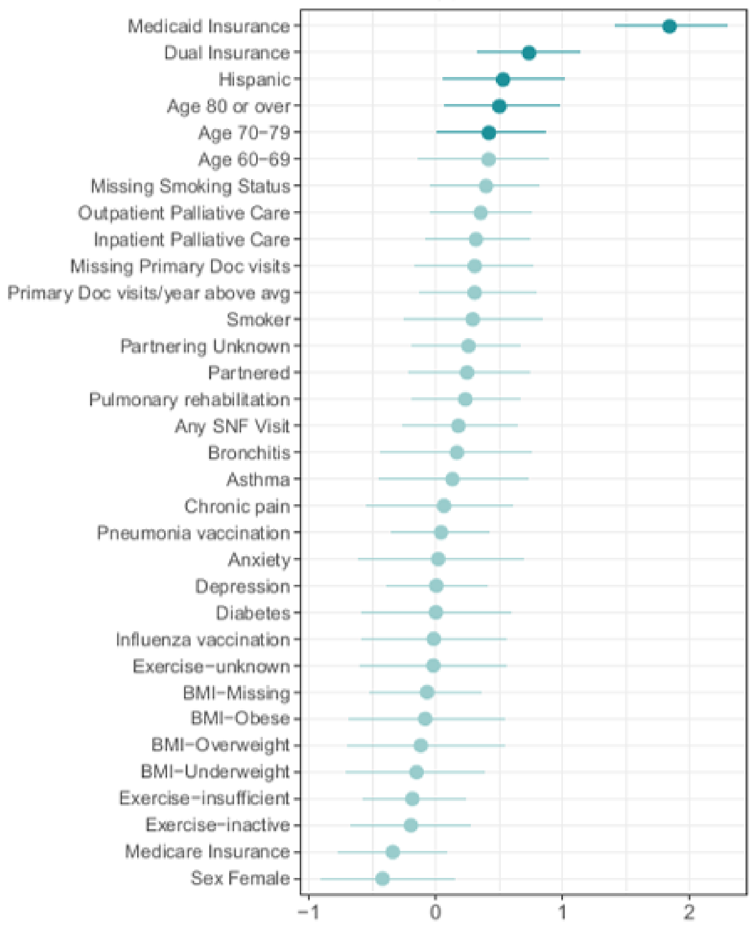

Figure 4. Caterpillar plot displaying parameters on Log Odds scale with 95% Credible Intervals using ggplot

From the caterpillar plots, parameters with mean values below 0 tend to decrease the odds of COPD. Parameters above 0 tend to increase the odds of COPD. For example, in Figure 4, the Hispanic parameter estimate is centered near 0.5, which corresponds to $exp(0.5) = 1.65$. Therefore, someone who is Hispanic has 1.65 times the Odds of having COPD compared to someone who is not Hispanic when other factors are held constant.

Apply JAGS Model R Function

Once a satisfiable model is built, this function can be used to apply your newly estimated JAGS model to a new dataset for internal or external validation and/or phenotype prediction.

The developed function, **`apply_jags_model`**, applies the estimated Bayesian Latent Variable Model to new data for phenotype prediction. While this can be done directly in rjags, values for prediction must be pre-specified before rjags is run and constructed; therefore the model parameter estimates are subject to vary and will require substantial computation for each new prediction. Using my developed function allows full reproducibility of phenotype identification of a cohort or individual person. This document is intended to serve as a template for other researchers to apply to their favorite latent variable model.

The function deconstructs the JAGS model output into four components as specified in the full Likelihood Equation. The parameter estimates are separated by model (Missing Spirometry, FEV1/FVC Distribution, Latent Phenotype, and Dependent Factors). The values are applied to the factors in the new dataset, and then individual likelihoods are calculated.

The `apply_jags_model` function requires four inputs: `new_data`, `variable_labels`, `training_data`, and `jags_model_output`.

`new_data`	is the dataframe with unique observations on each row and variables as columns. These variables should have the same name as used in the initial jags model.
`variable_labels`	is a dataframe with the variable names, their corresponding label, and the coefficient name that multiplies each coefficient. For example, if `beta_1*sex` is part of your model, the dataframe will be `variable_labels <- data.frame(parameters=("beta_1"), var=("sex"), Label=("Sex:  Female"))` This dataframe should have three variables named parameters, var, and Label.
`training_data`	is the data that was used to construct the rjags model. This is used to calculate sensitivity and specificity.
`jags_model_output`	is the two object list that contains the posterior results and output data. Specifically, the first object must be an MCMC list. `results_mcmc <- as.mcmc.list(results)` was used in the source_function script

If researchers are reproducing each step in this project, the `source_function` and `run_file` scripts need to be run before this function since it requires output from those scripts. Otherwise, a model which was already made using JAGS needs to be run first.

```r
# making the variable_labels dataframe
labels_table <- data.frame(
    parameters = c("a1", "a2",   "a3",   "a_COPD_miss", "a4", "a5",   "a6", "a7", "a8",
                  "a9", "a10", "a11",   "a12", "a13","a14", "a15", "a16", "a17", "a18",
                  "a19", "a20", "a21", "a22", "a23","a24", "a25",
                  "a26", "a27", "a28", "a29", "a30", "a31", "a32",
                  "r1.5", "r2.5",
                  "r3.5","r4.5", "r5.5", "r6.5", "r7.5","r8.5", "r9.5", "r10.5",
                  "r11.5", "r12.5", "r13.5","r15.5", "r16.5", "r17.5", "r18.5", "r19.5",
                  "r20.5", "r21.5",  "r22.5", "r23.5", "r58.5","r121.5", "r122.5",
                  "r123.5","r124.5", "r125.5", "r126.5", "r127.5", "r128.5","r129.5",
                  "r130.5","r131.5", "r132.5", "r133.5",   "r134.5", "r135.5", "r136.5",
                  "r137.5", "r138.5",  "r139.5",   "r140.5","r141.5", "r142.5","r143.5",
                  "r144.5",  "r145.5", "r146.5","r147.5", "r148.5",  "r149.5", "r150.5",
                  "r151.5", "r152.5", "r153.5", "r154.5",  "r155.5", "r156.5", "r157.5",
                  "r158.5", "r159.5", "r160.5", "r161.5", "r162.5", "r163.5","r164.5",
                  "a0", "codecharlson_q2_ind_b", "codecharlson_q2_ind_int",
                  "codecharlson_q3_ind_b", "codecharlson_q3_ind_int",
                  "codecharlson_q4_ind_b", "codecharlson_q4_ind_int",
                  "codeCOPD_b", "codeCOPD_b_int", "codeICS_b_dm", "codeICS_b_int",
                  "codeLABA_b_dm", "codeLABA_b_int", "codeLAMA_b_dm", "codeLAMA_b_int",
                  "codeLSCS_b_dm", "codeLSCS_b_int", "codeOXYG_b_dm", "codeOXYG_b_int",
                  "codeSAAC_b_dm", "codeSAAC_b_int", "codeSABA_b_dm", "codeSABA_b_int",
                  "codeSCS_b_dm", "codeSCS_b_int", "codeTHEO_b_dm", "codeTHEO_b_int",
                  "FEV1_b_dm", "FEV1_b_int"),
         var =  c("smoke_miss", "smoke_ind", "sex", "latentCOPD", "primary_63",
                  "primary_miss", "specialty_12", "specialty_miss", "ins_dual_ind",
                  "ins_medicaid_ind", "ins_medicare_ind", "hbpc", "pneumonia", "influenza",
                  "pulmonary_rehab", "a15", "a16", "age_69", "age_79", "age_80plus",
                  "race_amind", "race_asian", "race_black", "race_hawai", "race_multi",
                  "race_unkno", "a26", "a27", "a28", "a29", "a30", "a31", "a32",
                  "r1.5",  "smoke_ind", "smoke_miss", "age_69", "age_79", "age_80plus",
                  "hispanic_ind", "ins_dual_ind", "ins_medicaid_ind", "ins_medicare_ind",
                  "r11.5", "r12.5", "r13.5", "influenza", "pneumonia", "pulmonary_rehab",
                  "marriage", "marriage_miss", "primary_63", "r21.5",  "r22.5", "r23.5",
                  "r58.5", "r121.5", "r122.5", "r123.5","r124.5", "r125.5", "r126.5",
                  "r127.5", "r128.5","r129.5", "r130.5","r131.5", "r132.5", "r133.5",
                  "r134.5", "r135.5", "r136.5", "r137.5", "r138.5",  "r139.5",
                  "r140.5","r141.5", "r142.5","r143.5", "r144.5", "hf_miss",
                  "pulhyper_miss", "diab_miss", "depress_miss", "r149.5",
                  "chronic_pain_miss", "asthma_miss", "bronch_miss", "r153.5", "r154.5",
                  "hf_val", "pulhyper_val", "diab_val", "depress_val", "anxiety_val",
                  "chronic_pain_val", "asthma_val", "bronch_val", "r163.5", "pulm_ind_val",
                  "a0", "charlson_q2_ind", "codecharlson_q2_ind_int", "charlson_q3_ind",
                  "codecharlson_q3_ind_int", "charlson_q4_ind", "codecharlson_q4_ind_int",
                  "codeCOPD_b", "codeCOPD_b_int", "meds5", "codeICS_b_int",
                  "meds3", "codeLABA_b_int", "meds4", "codeLAMA_b_int",
                  "meds7", "codeLSCS_b_int", "oxygen", "codeOXYG_b_int",
                  "meds2", "codeSAAC_b_int", "meds1", "codeSABA_b_int",
                  "meds6", "codeSCS_b_int", "meds8", "codeTHEO_b_int",
                  "FEV1_b_dm", "FEV1_b_int"),
       Label = c("Missing Smoking Status", "Smoker", "Sex Female", "missing COPD Dx",
                 "Primary Doc visits/year above avg",  "Missing Primary Doc visits",
                 "Specialty doctor visits/year above avg", "Missing Specialty Doc visits",
                 "Dual Insurance", "Medicaid Insurance", "Medicare Insurance",
                 "Home Based Palliative Care", "Pneumonia vaccination",
                 "Influenza vaccination", "Pulmonary rehabilitation", "Hispanic",
                 "Missing Ethnicity",  "Age 60-69", "Age 70-79", "Age 80 or over",
                 "Race-American Indian",  "Race-Asian", "Race-Black",
                 "Race-Hawaiian or Pacific Islander", "Race-Mixed", "Race-Unknown",
                 "BMI-Underweight", "BMI-Overweight", "BMI-Obese", "BMI-Missing",
                 "Exercise-inactive", "Exercise-insufficient", "Exercise-unknown",
                 "Sex Female", "Smoker",  "Missing Smoking Status", "Age 60-69",
                 "Age 70-79", "Age 80 or over", "Hispanic", "Dual Insurance",
                 "Medicaid Insurance", "Medicare Insurance", "Exercise-inactive",
                 "Exercise-insufficient", "Exercise-unknown",
                 "Influenza vaccination", "Pneumonia vaccination",
                 "Pulmonary rehabilitation", "Partnered", "Partnering Unknown",
                 "Primary Doc visits/year above avg", "Missing Primary Doc visits",
                 "Inpatient Palliative Care", "Outpatient Palliative Care",
                 "Missing Ethnicity", "BMI-Underweight", "BMI-Overweight", "BMI-Obese",
                 "BMI-Missing","Race-American Indian", "Race-Asian", "Race-Black",
                 "Race-Hawaiian or Pacific Islander", "Race-Mixed", "Race-Unknown",
                 "COPD Hospitalizations - Missing", "COPD Hospitalizations",
                 "COPD Emergency Department Visit - Missing",
                 "COPD Emergency Department Visit",
                 "COPD Observational visits - Missing", "COPD Observational visits",
                 "Any Hospitalization - Missing", "Any Hospitalization",
                 "Any Emergency Department Visit - Missing",
                 "Any Emergency Department Visit", "Any Observational visit - Missing",
                 "Any Observational visit", "Any Urgent Care Visit - Missing",
                 "Any Urgent Care Visit", "Heart Failure - Missing",
                 "Pulmonary Hypertension - Missing", "Diabetes - Missing",
                 "Depression - Missing",   "Anxiety - Missing", "Chronic pain - Missing",
                 "Asthma - Missing",  "Bronchitis - Missing", "SNF visit - Missing",
                 "Pulmonary visits - Missing",   "Heart Failure", "Pulmonary Hypertension",
                 "Diabetes", "Depression", "Anxiety", "Chronic pain",  "Asthma",
                 "Bronchitis", "Any SNF Visit", "Any Pulmonary Visit",
                 "Missing Model Intercept", "Charlson Q2 Slope", "Charlson Q2 Intercept",
                 "Charlson Q3 Slope", "Charlson Q3 Intercept", "Charlson Q4 Slope",
                 "Charlson Q2 Intercept", "Latent COPD Slope", "Latent COPD Intercept",
                 "ICS Slope", "ICS Intercept", "LABA slope", "LABA intercept",
                 "LAMA slope", "LAMA intercept", "LSCS slope", "LSCS intercept",
                 "Oxygen slope", "Oxygen intercept", "SAAC slope", "SAAC intercept",
                 "SABA slope", "SABA intercept", "SCS slope", "SCS intercept",
                 "Theophylline slope", "Theophylline intercept",
                 "FEV1/FVC Missing slope", "FEV1/FVC Missing intercept" ))
```

The training dataset is used to extract information on diagnostic accuracy. This will be used to estimate the number of people expected to have COPD and will be used in part to be the threshold for the new dataset's diagnostic cutoff. Parameter values and names are extracted from the jags model.

```
apply_jags_model <- function(new_data, variable_labels,
                             training_data, jags_model_output){
  x <- new_data
  labels_table <- variable_labels
  jags_model_output <- jags_model_output[[1]]

  rho <- jags_model_output[[1]][, which(regexpr("rho_COPD",
                                  colnames(jags_model_output[[1]])) > 0)]
  parms <- jags_model_output[[1]][, which(regexpr("rho_COPD",
                                  colnames(jags_model_output[[1]])) <= 0)]
  parms[, c("a_COPD_miss")] <- exp(parms[, c("a_COPD_miss")])
  postCOPD_training <- apply(rho, 2, mean)

  library(MCMCvis)
  library(dplyr)

  values <- MCMCsummary(jags_model_output, excl = "rho_COPD")
  small <- as.data.frame(values[, c("Rhat", "mean")])
```

```r
colnames(small) <- c("Rhat", "mean")

# extracts the parameter names in the same order as the output
param_names <- colnames(parms) %>% as.data.frame
colnames(param_names) <- "parameters"
summary_table <- cbind(param_names, small)
```

The model has four components. `latent` contains variable coefficients which multiplied $\boldsymbol{X}_i$ in the logit latent COPD model (`logit(rho_COPD[i]) <- ...`).

```r
latent <- c("r1.5", "r2.5",
            "r3.5","r4.5", "r5.5", "r6.5", "r7.5","r8.5", "r9.5", "r10.5",
            "r11.5", "r12.5", "r13.5","r15.5", "r16.5", "r17.5", "r18.5", "r19.5",
            "r20.5", "r21.5",  "r22.5", "r23.5", "r58.5","r121.5", "r122.5",
            "r123.5","r124.5", "r125.5", "r126.5", "r127.5", "r128.5","r129.5",
            "r130.5","r131.5", "r132.5", "r133.5",   "r134.5", "r135.5", "r136.5",
            "r137.5", "r138.5",  "r139.5",   "r140.5","r141.5", "r142.5","r143.5",
            "r144.5",  "r145.5", "r146.5","r147.5", "r148.5",  "r149.5", "r150.5",
            "r151.5", "r152.5", "r153.5", "r154.5",  "r155.5", "r156.5", "r157.5",
            "r158.5", "r159.5", "r160.5", "r161.5", "r162.5", "r163.5","r164.5")
dependence <- c("codeCOPD_b_int", "codeCOPD_b",
                "codeSAAC_b_int", "codeSAAC_b_dm", "codeSABA_b_int",
                "codeSABA_b_dm", "codeLABA_b_int", "codeLABA_b_dm", "codeLAMA_b_int",
                "codeLAMA_b_dm", "codeSCS_b_int",
                "codeSCS_b_dm", "codeLSCS_b_int", "codeLSCS_b_dm", "codeTHEO_b_int",
                "codeTHEO_b_dm", "codeOXYG_b_int", "codeOXYG_b_dm",
                "codecharlson_q3_ind_int", "codecharlson_q3_ind_b",
                "codecharlson_q2_ind_int", "codecharlson_q2_ind_b",
                "codecharlson_q4_ind_int", "codecharlson_q4_ind_b")
FEV_vars <- c("FEV1_b_int","FEV1_b_dm", "FEV1_tau")

latent_results <- summary_table[summary_table$parameters %in% latent, ]
dependence_results <- summary_table[summary_table$parameters %in% dependence, ]
missing_results <- summary_table[!(summary_table$parameters %in% latent)
                                 & !(summary_table$parameters %in% dependence)
                                 & !(summary_table$parameters %in% FEV_vars), ]

latent_var <- merge(latent_results, labels_table,
                    by = intersect(names(labels_table), names(latent_results)),
                    all.x = TRUE, sort = FALSE, no.dups = TRUE)
depend_var <- merge(dependence_results, labels_table,
                    by = intersect(names(labels_table), names(dependence_results)),
                    all.x = TRUE, sort = FALSE, no.dups = TRUE)
missing_var <- merge(missing_results, labels_table,
                     by = intersect(names(labels_table), names(missing_results)),
                     all.x = TRUE, sort = FALSE, no.dups = TRUE)
```

Each piece of the model has its parameter estimates from rjags in different vectors. Now, we need to multiply the vector of parameter estimates with each individual's observed characteristics. To do so, we will
(1) extract the corresponding variables from the new dataset,
(2) arrange them in the order that the parameter estimates above are ordered,
(3) multiply the matrix and vector, and
(4) calculate the likelihood component that will be incorporated into the full likelihood equation.

Latent Model Component

$P(D_i = d|\beta^D, \boldsymbol{X}_i)$

This corresponds to (1) in the Likelihood Equation.

```r
# getting the parameters for the latent variable model
r <- latent_var[, "mean"]
r_var <- latent_var[, "var"]
# getting the variables from the dataset to multiply
x_r <- x[, names(x) %in% r_var]

# names of the variables in the order they were extracted
t_xr <- as.data.frame(names(x_r), ncol=1)
colnames(t_xr) <- "var"
# merge to get the parameters in the correct order
t_2 <- merge(t_xr, latent_var, by = intersect(names(t_xr), names(latent_var)),
             all.x = TRUE, sort = FALSE, no.dups = TRUE)

# get the parameter estimates in the correct order to multiply;
beta_r <- as.matrix(t_2[, "mean"], nrow=1)
x_R <- as.matrix(x_r)
# must convert NAs to zeros in order to multiply out correctly.
x_R[is.na(x_R)] <- 0
# matrix multiplication X*b
likelihood_latent_parameters <- x_R %*% beta_r

# undetermined if needed, but edit if we need to calculate individual random effects
# it seems to be doing a good job without these effects

########### MODIFY HERE #############
# random_effects <- some model
# likelihood_latent_parameters2 <- likelihood_latent_parameters + random_effects
# solve for pi / probabiliity
# probability_latent_COPD <- expit(likelihood_latent_parameters2)
# probability_latent_noCOPD <- 1-probability_latent
########### MODIFY HERE #############

# solve for pi / probabiliity
probability_latent_COPD <- expit(likelihood_latent_parameters)
probability_latent_noCOPD <- 1 - probability_latent_COPD
```

The results are the two probabilities needed in (1) of the Likelihood Equation.
`probability_latent_COPD` is $P(D_i = 1|\beta^D, \boldsymbol{X}_i)$
`probability_latent_noCOPD` is $P(D_i = 0|\beta^D, \boldsymbol{X}_i)$

Missing Biomarker Component

$f(R_i|D_i = d, X_i, \beta^R)$

This corresponds to the first component of (2) in the Likelihood Equation.

```r
a <- missing_var[,"mean"]
a_var <- missing_var[,"var"]
x_a <- x[, names(x) %in% a_var]
# getting the variables from the dataset to multiply
# names of the variables in the order they were extracted
t_xa <- as.data.frame(names(x_a), ncol = 1)
colnames(t_xa) <- "var"
# merge to get the parameters in the correct order
t_2a <- merge(t_xa, missing_var, by = intersect(names(t_xa), names(missing_var)),
              all.x = TRUE, sort = FALSE, no.dups = TRUE)

# get the parameter estimates in the correct order to multiply;
beta_a <- as.matrix(t_2a[,"mean"], nrow = 1)
x_A <- as.matrix(x_a)
# get the intercept term
int_a <- missing_results[missing_results$parameters == "a0", "mean"]
# get the dependence on latent COPD variable
miss_var_a <- missing_results[missing_results$parameters == "a_COPD_miss", "mean"]

# must convert NAs to zeros in order to multiply out correctly.
x_A[is.na(x_A)] <- 0
# matrix multiplication X*b
likelihood_missing_parameters <- x_A%*%beta_a

# complete the linear equation by putting in the intercept and dependent factor;
likelihood_missing_parameters_COPD <- likelihood_missing_parameters + int_a + miss_var_a
likelihood_missing_parameters_noCOPD <- likelihood_missing_parameters + int_a

# this is probability of missing Spirometry.
probability_missing_COPD <- expit(likelihood_missing_parameters_COPD)
probability_missing_noCOPD <- expit(likelihood_missing_parameters_noCOPD)


# now to get the component of the function that gets multiplied,
# need to know if FEV is missing or not.
# stored in FEV1_miss_ind variable
x$FEV1_miss_ind <- ifelse(is.na(x$FEV1_FVC) == TRUE, 1, 0)
# if missing, use probability from above.
likelihood_missing_COPD <- ifelse(x$FEV1_miss_ind == 1,
                                  probability_missing_COPD,
                                  1 - probability_missing_COPD)
likelihood_missing_noCOPD <- ifelse(x$FEV1_miss_ind == 1,
                                    probability_missing_noCOPD,
                                    1 - probability_missing_noCOPD)
```

The likelihood equation for biomarker availability is a singular probability.
Recall if $X \sim Bern(\pi)$ then $L(\pi|X) = \pi^x(1-\pi)^{1-x}$ for $x = 0, 1$.

`probability_missing_COPD` is the likelihood of missing spirometry if the individual has COPD. `probability_missing_noCOPD` is the likelihood of missing spirometry if the individual does not have COPD. `likelihood_missing_COPD` and `likelihood_missing_noCOPD` adjust the probabilities based on whether the person has the biomarker.

The table below may help conceptualize the probabilities used for each individual based on what is observed in their EHR for the biomarker presence.

EHR Observation	Conditioned On	$f(R_i \vert D_i = d, X_i, \beta^R)$
has biomarker ($R_i = 1$)	has COPD ($D_i = 1$)	`1-probability_missing_COPD`
	no COPD ($D_i = 0$)	`1-probability_missing_noCOPD`
does not have biomarker ($R_i = 0$)	has COPD ($D_i = 1$)	`probability_missing_COPD`
	no COPD ($D_i = 0$)	`probability_missing_noCOPD`

Biomarker Value Component

$f(Y_i|D_i = d, \boldsymbol{X}_i, \beta^Y, \tau^2)^{R_i}$
Second component of (2) in the Likelihood Equation.
The biomarker (FEV1/FVC value) is believed to be normally distributed.

```r
# need to get normal distribution likelihood for a person's FEV1 / FVC value

norm_mean_COPD <- summary_table[summary_table$parameters == "FEV1_b_int", "mean"] +
                  summary_table[summary_table$parameters == "FEV1_b_dm", "mean"]
norm_mean_noCOPD <- summary_table[summary_table$parameters == "FEV1_b_int", "mean"]
FEV1_tau <- summary_table[summary_table$parameters == "FEV1_tau", "mean"]

# normal distribution likelihood and then raised to the power of (have FEV or not)

x_mu_COPD <- x$FEV1_FVC - norm_mean_COPD
x_mu_noCOPD <- x$FEV1_FVC - norm_mean_noCOPD
norm_constant <- 1 / (sqrt(2 * pi * (1 / FEV1_tau)))
denom <- 2 * (1 / FEV1_tau)

# if not missing the spirometry value, then incorporate the code into the likelihood
norm_FEVCOPD_likelihood <- ifelse(x$FEV1_miss_ind == 0,
                                  norm_constant * exp(-(x_mu_COPD)^2 / denom), 1)
norm_FEVnoCOPD_likelihood <- ifelse(x$FEV1_miss_ind == 0,
                                    norm_constant * exp(-(x_mu_noCOPD)^2 / denom), 1)
```

Clinical Codes and Prescriptions Component

$\prod_{k=1}^{K} f(W_{ik}|D_i = d, \boldsymbol{X_i}, \boldsymbol{\beta}_k^W) \prod_{l=1}^{L} f(P_{il}|D_i = d, \boldsymbol{X}_i, \boldsymbol{\beta}_l^P)$

The likelihood contributions for the prescriptions and clinical codes each have their own likelihood function. This corresponds to (3) in the Likelihood Equation.

```r
# calculate the probability for each of four conditions
  hasdep_COPD <- apply(parms[, c("codeSAAC_b_int", "codeSABA_b_int", "codeLABA_b_int",
                             "codeSCS_b_int", "codeLSCS_b_int", "codeTHEO_b_int",
                             "codeOXYG_b_int", "codecharlson_q3_ind_int",
                             "codecharlson_q2_ind_int", "codecharlson_q4_ind_int")]
                       + parms[, c("codeSAAC_b_dm", "codeSABA_b_dm", "codeLABA_b_dm",
                               "codeSCS_b_dm", "codeLSCS_b_dm", "codeTHEO_b_dm",
                               "codeOXYG_b_dm", "codecharlson_q3_ind_b",
                               "codecharlson_q2_ind_b", "codecharlson_q4_ind_b")],
                       2, function(x){ mean(expit(x)) })
  hasdep_noCOPD <- apply(parms[, c("codeSAAC_b_int", "codeSABA_b_int",
                               "codeLABA_b_int", "codeSCS_b_int",
                               "codeLSCS_b_int", "codeTHEO_b_int",
                               "codeOXYG_b_int", "codecharlson_q3_ind_int",
                               "codecharlson_q2_ind_int", "codecharlson_q4_ind_int")],
                      2, function(x){ mean(expit(x)) })

hasnodep_COPD <- apply(parms[, c("codeSAAC_b_int", "codeSABA_b_int", "codeLABA_b_int",
                               "codeSCS_b_int", "codeLSCS_b_int", "codeTHEO_b_int",
                               "codeOXYG_b_int", "codecharlson_q3_ind_int",
                               "codecharlson_q2_ind_int", "codecharlson_q4_ind_int")]
                      + parms[, c("codeSAAC_b_dm", "codeSABA_b_dm", "codeLABA_b_dm",
                                 "codeSCS_b_dm", "codeLSCS_b_dm", "codeTHEO_b_dm",
                                 "codeOXYG_b_dm", "codecharlson_q3_ind_b",
                                 "codecharlson_q2_ind_b", "codecharlson_q4_ind_b")],
                      2, function(x){ mean(1- expit(x)) })
hasnodep_noCOPD <- apply(parms[, c("codeSAAC_b_int", "codeSABA_b_int", "codeLABA_b_int",
                                 "codeSCS_b_int", "codeLSCS_b_int", "codeTHEO_b_int",
                                 "codeOXYG_b_int", "codecharlson_q3_ind_int",
                                 "codecharlson_q2_ind_int", "codecharlson_q4_ind_int")],
                        2, function(x){ mean(1- expit(x)) })


d <- depend_var[, "mean"]
d_var <- depend_var[, "var"] # variable names
nam_code <- as.data.frame(names(hasdep_COPD), ncol = 1)
nam_code2 <- as.data.frame(names(hasdep_noCOPD), ncol = 1)
nam_code3 <- as.data.frame(names(hasnodep_COPD), ncol = 1)
nam_code4 <- as.data.frame(names(hasnodep_noCOPD), ncol = 1)

nam_val <- as.data.frame(hasdep_COPD, ncol = 1)
nam_val2 <- as.data.frame(hasdep_noCOPD, ncol = 1)
nam_val3 <- as.data.frame(hasnodep_COPD, ncol = 1)
nam_val4 <- as.data.frame(hasnodep_noCOPD, ncol = 1)

colnames(nam_code) <- c("parameters") # rename so that you can overwrite
colnames(nam_val) <- c("hasdep_COPD")
nam_code <- cbind(nam_code, nam_val)

colnames(nam_code2) <- c("parameters")
colnames(nam_val2) <- c("hasdep_noCOPD")
nam_code2 <- cbind(nam_code2, nam_val2)

colnames(nam_code3) <- c("parameters")
colnames(nam_val3) <- c("hasnodep_COPD")
nam_code3 <- cbind(nam_code3, nam_val3)

colnames(nam_code4) <- c("parameters")
colnames(nam_val4) <- c("hasnodep_noCOPD")
nam_code4 <- cbind(nam_code4, nam_val4)

# merge the dataframes by ppl w or w/o COPD;
expit_COPD <- merge(nam_code, nam_code3,
                    by = intersect(names(nam_code), names(nam_code3)),
                    all = TRUE, sort = FALSE, no.dups = TRUE)
expit_noCOPD <- merge(nam_code2, nam_code4,
                      by = intersect(names(nam_code2), names(nam_code4)),
                      all = TRUE, sort = FALSE, no.dups = TRUE)
```

```r
x_d <- x[, names(x) %in% d_var]

# names of the variables in the order they were extracted
t_xd <- as.data.frame(names(x_d), ncol=1)
colnames(t_xd) <- "var"
# merge to get the parameters in the correct order
t_2d <- merge(t_xd, depend_var, by = intersect(names(t_xd), names(depend_var)),
              all.x = TRUE, sort = FALSE, no.dups = TRUE)


# relabel so that they have the same label and then merge the mean value over.
P3 <- data.frame(
  parameters = c("codecharlson_q2_ind_int", "codecharlson_q3_ind_int",
               "codecharlson_q4_ind_int", "codeCOPD_b_int", "codeLABA_b_int",
               "codeLAMA_b_int", "codeLSCS_b_int", "codeOXYG_b_int", "codeSAAC_b_int",
               "codeSABA_b_int", "codeSCS_b_int", "codeTHEO_b_int"),
  Label = c("Charlson Q2 Slope",  "Charlson Q3 Slope", "Charlson Q4 Slope",
            "Latent COPD Slope", "LABA slope", "LAMA slope", "LSCS slope",
            "Oxygen slope", "SAAC slope", "SABA slope",  "SCS slope",
            "Theophylline slope") )

# this is their sensitivity
relabeled_COPD <- merge(expit_COPD, P3,
                        by = intersect(names(expit_COPD), names(P3)),
                        all.x = TRUE, sort = FALSE, no.dups = TRUE)
relabeled_noCOPD <- merge(expit_noCOPD, P3,
                          by = intersect(names(expit_noCOPD), names(P3)),
                          all.x = TRUE, sort = FALSE, no.dups = TRUE)

# now each variable has its own row with sensitivity with it.
with_var_COPD <- merge(relabeled_COPD, t_2d, by = "Label",
                       all.x = TRUE, sort = FALSE, no.dups = TRUE)
with_var_noCOPD <- merge(relabeled_noCOPD, t_2d, by = "Label",
                         all.x = TRUE, sort = FALSE, no.dups = TRUE)
# now same order as the variables that were extracted from the dataset;
sorted_2_COPD <- merge(t_xd, with_var_COPD, by = "var",
                       all.x = TRUE, sort = FALSE, no.dups = TRUE)
sorted_2_noCOPD <- merge(t_xd, with_var_noCOPD, by = "var",
                         all.x = TRUE, sort = FALSE, no.dups = TRUE)

# 4 vectors of values to potentially multiply through.
beta_hasdep_COPD <- as.matrix(sorted_2_COPD[, "hasdep_COPD"], nrow = 1)
beta_hasnodep_COPD <- as.matrix(sorted_2_COPD[, "hasnodep_COPD"], nrow = 1)
beta_hasdep_noCOPD <- as.matrix(sorted_2_noCOPD[, "hasdep_noCOPD"], nrow = 1)
beta_hasnodep_noCOPD <- as.matrix(sorted_2_noCOPD[, "hasnodep_noCOPD"], nrow = 1)

x_D <- as.matrix(x_d)
# must convert NAs to zeros in order to multiply out correctly.
x_D[is.na(x_D)] <- 0

# by each row, extract the values of the beta's based on the person's logical array
inverted_x_D <- ifelse(x_D == 0,1,0)
```

```r
# for COPD = 1
likelihoods_2_multiply_dep_COPD <- sweep(x_D, MARGIN = 2, beta_hasdep_COPD, `*`)
likelihoods_2_multiply_nodep_COPD <- sweep(inverted_x_D, MARGIN = 2,
                                           beta_hasnodep_COPD, `*`)

# for COPD = 0
likelihoods_2_multiply_dep_noCOPD <- sweep(x_D, MARGIN = 2, beta_hasdep_noCOPD, `*`)
likelihoods_2_multiply_nodep_noCOPD <- sweep(inverted_x_D, MARGIN = 2,
                                             beta_hasnodep_noCOPD, `*`)
likelihood_noCOPD <- likelihoods_2_multiply_dep_noCOPD +
                     likelihoods_2_multiply_nodep_noCOPD

# want each person to have a few functions multiplied together.
# remove any NAs for vector multiplication
likelihood_COPD[is.na(likelihood_COPD)] <- 1
likelihood_noCOPD[is.na(likelihood_noCOPD)] <- 1

# each person's individual likelihood of medications and codes
likelihood_dependence_COPD <- apply(likelihood_COPD, MARGIN = 1, prod)
likelihood_dependence_noCOPD <- apply(likelihood_noCOPD, MARGIN = 1, prod)
```

`likelihood_dependence_COPD` is the product of K clinical codes and L prescription likelihood functions for each individual conditioned on having COPD. `likelihood_dependence_noCOPD` is the product of the codes and prescription likelihood functions conditioned on not having COPD.

EHR Observation	Conditioned On	$f(W_{ik} \mid D_i = d, X_i, \beta_k^W)$
has clinical code k ($W_{ik} = 1$)	has COPD ($D_i = 1$)	expit(intercept $+\beta_k^W$)
	no COPD ($D_i = 0$)	expit(intercept)
does not have clinical code k ($W_{ik} = 0$)	has COPD ($D_i = 1$)	1 —expit(intercept $+\beta_k^W$)
	no COPD ($D_i = 0$)	1 —expit(intercept)

Combining Likelihoods

All three components of the likelihood equation have been computed for both disease states: having and not having COPD. The product of individual components and the summation of the likelihoods for each latent phenotype will be calculated for individuals.

The estimated prevalence of COPD in the training dataset is used as the cutoff for diagnosing people's latent phenotype; if 93% of people were diagnosed with COPD in the original jags model, then individuals in the new dataset with likelihood values at or below the 93rd percentile are phenotyped with COPD. Individuals above this percentile are phenotyped as not having COPD. This method was suggested by Hubbard et al. (2019). The section also calculates the sensitivity and specificity at different cutpoints, and the optimal is plotted with rule-based diagnostic methods.

```r
# elementwise multiplication of likelihood components
like_COPD <- probability_latent_COPD * likelihood_missing_COPD *
             norm_FEVCOPD_likelihood * likelihood_dependence_COPD
like_noCOPD <- probability_latent_noCOPD * likelihood_missing_noCOPD *
               norm_FEVnoCOPD_likelihood * likelihood_dependence_noCOPD

# final individual likelihood
individual_likelihood_COPD <- like_COPD + like_noCOPD

latent_model_count <- round(sum(postCOPD_training), 0)
```

```
# this is the estimated prevalence of COPD in the training data
latent_model_prediction <- latent_model_count / nrow(training_data)
quantile_pred <- quantile(individual_likelihood_COPD, probs=latent_model_prediction)

# using different cutpoints to find the optimal sensitivity and specificity
has_COPD <- ifelse(individual_likelihood_COPD <= quantile_pred, 1, 0)
has_COPD_min2 <- ifelse(individual_likelihood_COPD <= quantile_pred - .002, 1, 0)
has_COPD_min4 <- ifelse(individual_likelihood_COPD <= quantile_pred - .004, 1, 0)
has_COPD_min6 <- ifelse(individual_likelihood_COPD <= quantile_pred - .006, 1, 0)
has_COPD_min10 <- ifelse(individual_likelihood_COPD <= quantile_pred - .010, 1, 0)
has_COPD_min11 <- ifelse(individual_likelihood_COPD <= quantile_pred - .011, 1, 0)
has_COPD_min12 <- ifelse(individual_likelihood_COPD <= quantile_pred - .012, 1, 0)
has_COPD_min13 <- ifelse(individual_likelihood_COPD <= quantile_pred - .013, 1, 0)
has_COPD_min14 <- ifelse(individual_likelihood_COPD <= quantile_pred - .014, 1, 0)
has_COPD_min15 <- ifelse(individual_likelihood_COPD <= quantile_pred - .015, 1, 0)
has_COPD_min20 <- ifelse(individual_likelihood_COPD <= quantile_pred - .020, 1, 0)
has_COPD_plus2 <- ifelse(individual_likelihood_COPD <= quantile_pred + .002, 1, 0)
has_COPD_plus4 <- ifelse(individual_likelihood_COPD <= quantile_pred + .004, 1, 0)
has_COPD_plus6 <- ifelse(individual_likelihood_COPD <= quantile_pred + .006, 1, 0)
has_COPD_plus10 <- ifelse(individual_likelihood_COPD <= quantile_pred + .010, 1, 0)
has_COPD_plus12 <- ifelse(individual_likelihood_COPD <= quantile_pred + .012, 1, 0)
has_COPD_plus15 <- ifelse(individual_likelihood_COPD <= quantile_pred + .015, 1, 0)
has_COPD_plus20 <- ifelse(individual_likelihood_COPD <= quantile_pred + .020, 1, 0)

# merge the predictions with true COPD phenotype
lol <- cbind(has_COPD, x[, "COPD"])
lol <- as.data.frame(lol)
colnames(lol) <- c("has_COPD", "COPD")

lol_all <-cbind(has_COPD_min20, has_COPD_min15, has_COPD_min14, has_COPD_min13,
                has_COPD_min12, has_COPD_min11, has_COPD_min10, has_COPD_min6,
                has_COPD_min4, has_COPD_min2, has_COPD, has_COPD_plus2,
                has_COPD_plus4,  has_COPD_plus6, has_COPD_plus10, has_COPD_plus12,
                has_COPD_plus15, has_COPD_plus20, x[, "COPD"])
lol_all <- as.data.frame(lol_all)
colnames(lol_all) <- c("has_COPD_min20", "has_COPD_min15", "has_COPD_min14",
                       "has_COPD_min13", "has_COPD_min12", "has_COPD_min11",
                       "has_COPD_min10", "has_COPD_min6", "has_COPD_min4",
                       "has_COPD_min2", "has_COPD", "has_COPD_plus2", "has_COPD_plus4",
                       "has_COPD_plus6", "has_COPD_plus10", "has_COPD_plus12",
                       "has_COPD_plus15", "has_COPD_plus20", "COPD")
# the number correctly identified
sum(lol[, 1] * lol[, 2])
# the number of people with the phenotype
sum(lol[, 2])

# calculate sensitivity
new_data_sensitivity <- sum(lol[, 1] * lol[, 2]) / sum(lol[, 2])

sens_min2 = sum(has_COPD_min2 * x$COPD) / sum(x$COPD)
sens_min4 = sum(has_COPD_min4 * x$COPD) / sum(x$COPD)
sens_min6 = sum(has_COPD_min6 * x$COPD) / sum(x$COPD)
sens_min10 = sum(has_COPD_min10 * x$COPD) / sum(x$COPD)
```

```r
sens_min11 = sum(has_COPD_min11 * x$COPD) / sum(x$COPD)
sens_min12 = sum(has_COPD_min12 * x$COPD) / sum(x$COPD)
sens_min13 = sum(has_COPD_min13 * x$COPD) / sum(x$COPD)
sens_min14 = sum(has_COPD_min14 * x$COPD) / sum(x$COPD)
sens_min15 = sum(has_COPD_min15 * x$COPD) / sum(x$COPD)
sens_min20 = sum(has_COPD_min20 * x$COPD) / sum(x$COPD)
sens_plus2 = sum(has_COPD_plus2 * x$COPD) / sum(x$COPD)
sens_plus4 = sum(has_COPD_plus4 * x$COPD) / sum(x$COPD)
sens_plus6 = sum(has_COPD_plus6 * x$COPD) / sum(x$COPD)
sens_plus10 = sum(has_COPD_plus10 * x$COPD) / sum(x$COPD)
sens_plus12 = sum(has_COPD_plus12 * x$COPD) / sum(x$COPD)
sens_plus15 = sum(has_COPD_plus15 * x$COPD) / sum(x$COPD)
sens_plus20 = sum(has_COPD_plus20 * x$COPD) / sum(x$COPD)

# compare all diagnostic methods #

built_latent_model = latent_model_prediction
new_data_pred = sum(has_COPD * x$COPD) / sum(x$COPD)
phenotype_1 = sum(x$phenotype_1, na.rm = TRUE) / sum(x$COPD)
phenotype_2 = sum(x$phenotype_2, na.rm = TRUE) / sum(x$COPD)
phenotype_3 = sum(x$phenotype_3, na.rm = TRUE) / sum(x$COPD)
phenotype_4 = sum(x$phenotype_4, na.rm = TRUE) / sum(x$COPD)
phenotype_5 = sum(x$phenotype_5, na.rm = TRUE) / sum(x$COPD)
phenotype_6 = sum(x$phenotype_6, na.rm = TRUE) / sum(x$COPD)

sens_threshold <-cbind(sens_min20, sens_min15, sens_min14, sens_min13,
                       sens_min12, sens_min11, sens_min10, sens_min6,
                       sens_min4, sens_min2, new_data_pred, sens_plus2,
                       sens_plus4, sens_plus6, sens_plus10, sens_plus12,
                       sens_plus15, sens_plus20)
```

```r
# SPECIFICITY #
# can't do for built model if using original data because everyone has COPD
built_latent_model_spec = NA
new_data_spec = sum(lol$has_COPD == 0 & lol$COPD == 0) / sum(lol$COPD == 0)
# can't do for phenotypes because phenotype is not in new dataset.
# (who are ppl w/o COPD)
phenotype_1_spec = NA
phenotype_2_spec = NA
phenotype_3_spec = NA
phenotype_4_spec = NA
phenotype_5_spec = NA
phenotype_6_spec = NA

spec_min2 = sum(lol_all$has_COPD_min2 == 0 & lol_all$COPD == 0) /
            sum(lol_all$COPD == 0)
spec_min4 = sum(lol_all$has_COPD_min4 == 0 & lol_all$COPD == 0) /
            sum(lol_all$COPD == 0)
spec_min6 = sum(lol_all$has_COPD_min6 == 0 & lol_all$COPD == 0) /
            sum(lol_all$COPD == 0)
spec_min10 = sum(lol_all$has_COPD_min10 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_min11 = sum(lol_all$has_COPD_min11 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
```

```r
spec_min12 = sum(lol_all$has_COPD_min12 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_min13 = sum(lol_all$has_COPD_min13 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_min14 = sum(lol_all$has_COPD_min14 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_min15 = sum(lol_all$has_COPD_min15 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_min20 = sum(lol_all$has_COPD_min20 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_plus2 = sum(lol_all$has_COPD_plus2 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_plus4 = sum(lol_all$has_COPD_plus4 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_plus6 = sum(lol_all$has_COPD_plus6 == 0 & lol_all$COPD == 0) /
             sum(lol_all$COPD == 0)
spec_plus10 = sum(lol_all$has_COPD_plus10 == 0 & lol_all$COPD == 0) /
              sum(lol_all$COPD == 0)
spec_plus12 = sum(lol_all$has_COPD_plus12 == 0 & lol_all$COPD == 0) /
              sum(lol_all$COPD == 0)
spec_plus15 = sum(lol_all$has_COPD_plus15 == 0 & lol_all$COPD == 0) /
              sum(lol_all$COPD == 0)
spec_plus20 = sum(lol_all$has_COPD_plus20 == 0 & lol_all$COPD == 0) /
              sum(lol_all$COPD == 0)

spec_threshold <-cbind(spec_min20, spec_min15, spec_min14, spec_min13,
                       spec_min12, spec_min11, spec_min10, spec_min6,
                       spec_min4, spec_min2, new_data_spec, spec_plus2,
                       spec_plus4, spec_plus6, spec_plus10, spec_plus12,
                       spec_plus15, spec_plus20)
sens_and_spec <- rbind(sens_threshold, spec_threshold)

# the optimal cut point is chosen as the one that has the
# best sensitivity and specificity when added together.
maxes <- colSums(sens_and_spec)
cutoff <- names(which.max(maxes))
best_specification <- as.data.frame(t(sens_and_spec[, cutoff]), ncol = 2)

compare <- as.data.frame(rbind(built_latent_model, new_data_pred,
                               best_specification[, 1], phenotype_1, phenotype_2,
                               phenotype_3, phenotype_4, phenotype_5, phenotype_6))
spec_compare <- as.data.frame(rbind(built_latent_model_spec, new_data_spec,
                                    best_specification[, 2], phenotype_1_spec,
                                    phenotype_2_spec, phenotype_3_spec,
                                    phenotype_4_spec, phenotype_5_spec,
                                    phenotype_6_spec))
model_names <- c("Built Latent Model", "New Data Prediction",
                 "New Model Best Combo", "Phenotype 1", "Phenotype 2", "Phenotype 3",
                 "Phenotype 4", "Phenotype 5", "Phenotype 6")
diagn_compare <- cbind(model_names, compare, spec_compare)
colnames(diagn_compare) <- c("Models", "Sensitivity", "Specificity")

# this extracts the best sensitivity and specificity.
```

```
  lol_all_vals <- lol_all
  colnames(lol_all_vals) <- c("sens_min20", "sens_min15", "sens_min14", "sens_min13",
                              "sens_min12", "sens_min11", "sens_min10", "sens_min6",
                              "sens_min4", "sens_min2", "new_data_pred",
                              "sens_plus2", "sens_plus4", "sens_plus6",
                              "sens_plus10", "sens_plus12", "sens_plus15",
                              "sens_plus20", "COPD")
  to_research <- lol_all_vals[,c(cutoff, "COPD")]

  # plot sensitivity of different diagnostic measures ##
  sens_compare <- ggplot(diagn_compare, aes(x = Models, y = Sensitivity)) +
    geom_point() +
    labs(x = "Diagnosis Method", y = "Sensitivity") + theme_bw() +
    theme(axis.text.x = element_text(angle = 60, hjust = 1, size = 12))

  return (list(individual_likelihood_COPD, sens_compare, summary_table,
               diagn_compare, sens_and_spec, to_research))
}
```

Function Results

The function outputs a 6 item list. The example call to the functions saves the list to `trial`:

```
trial <- apply_jags_model(new_data = smol3, variable_labels = labels_table,
                          training_data = random_subset_7_31,
                          jags_model_output = COPD_bayes_7_31)
```

- `[[1]]` is the vector of each person's likelihood. Once a final model is selected, these likelihood values can be condensed into ranges with probability of COPD status and the assigned phenotype.
- `[[2]]` is a simple plot of the sensitivities of each diagnostic method. The built latent model and new data predicition points are using the suggested threshold from Hubbard et al. (2019). "New model best combo" plots the optimal cutpoints' sensitivity.
- `[[3]]` is the parameter estimates used in the model.
- `[[4]]` is a table of the sensitivity and specificity of each measure that is plotted in `[[2]]`.
- `[[5]]` is an object with the sensitivity and specificity for the new data at the different cutpoints. This can be used if a ROC is desired.
- `[[6]]` is a two column vector with the COPD phenotyping for the new dataset and their true COPD disease status. 1 indicates COPD, 0 indicates no COPD.

To plot the sensitivity, call `trial[[2]]`

Limitations

As a next step in the future, we could modify the diagnostic measures to use DiLRs to improve prediction. To do this, each person's likelihood is multiplied by the sensitivity of the range they are in and then this adjusted likelihood is used for diagnosis.

References

1. The Top 10 Causes of Death. 2019. World Health Organization Factsheet. http://www.who.int/mediacentre/factsheets/fs310/en/.
2. What Causes COPD? COPD Foundation, www.copdfoundation.org/What-is-COPD/Understanding-COPD/What-Causes-COPD.aspx.
3. Hubbard, Rebecca, Jing Huang, Joanna Harton, Arman Oganisian, Grace Choi, Levon Utidjian, Ihuoma Eneli, Charles Bailey, and Yong Chen. 2019. "A Bayesian Latent Class Approach for EHR-Based Phenotyping." Statistics in Medicine 38: 74-87. doi:10.1002/sim.7953.
4. Centers for Disease Control and Prevention, National Center for Health Statistics. Compressed Mortality File 1999-2016 on CDC WONDER Online Database, released June 2017. Series 20 No. 2U, 2016, Vital Statistics Cooperative Program. http://wonder.cdc.gov/cmf-icd10.html.

Contributions

Kristen McGreevy wrote the document, modified and annotated the R code, built the model, and developed the predictive function.
Ernest Shen developed the project idea, built the conceptual framework, and worked with Kristen throughout the project. Dr. Shen's subject knowledge and statistical expertise made this project feasible. Dr. Shen ensured accuracy of methods and interpreted the results.
Janet S. Lee pulled and managed the data which includes making the original variables.
Michael Gould supported this project via implementation and funding. Dr. Gould provided research direction and project feedback.

Correspondence

This document was made by Kristen McGreevy as part of the Care Improvement Research Team at Kaiser Permanente. The R scripts adapted from Hubbard can be found at https://github.com/rhubb/Latent-phenotype.

Suggested Citation:

McGreevy, K.M., Shen, E.. 2019. "A Bayesian & EHR-derived Latent Phenotyping model for COPD". Research and Evaluation Department, Kaiser Permanente Southern California