IHME

Measuring what matters

ETHIOPIAN PUBLIC HEALTH INSTITUTE

NATIONAL DATA MANAGEMENT CENTER FOR HEALTH

DATA ANALYTICS AND VISUALIZATION UNIT

DRIVERS OF VACCINATION: MANUAL FOR DATA CLEANING, MANIPULATION AND ANALYSIS

PREPARED BY

DATA ANALYSIS AND VISUALIZATION UNIT IN COLLABORATION WITH INSTITUTE FOR HEALTH METRICS AND EVALUATION (IHME) AT UNIVERSITY OF WASHINGTON

August 24, 2023

ADDIS ABABA, ETHIOPIA

Summary

Having insights from Institute for Health Metrics and Evaluation (IHME) analysis which shows that progress in reducing zero-dose children which stalled in a number of regions in Ethiopia post-2010, the study is intended to quantify the relative contribution of the three drivers (intent to vaccine, community access and facility readiness) of vaccine coverage in Ethiopia. In collaboration with IHME, data analytic and visualization unit at National Data Management Center for Health (NDMC) at Ethiopian public health institute (EPHI) in collaboration with IHME systematically compiled and analyzed data from different sources to produce harmonized geospatial estimates of the three drivers, linked the estimates of the three drivers to available household surveys and estimated the predicted relationship between the probability that a child is vaccinated and levels of the three drivers using Machine Learning (ML) methods. This manual presents the data pre-processing approach as well as steps for fitting ML model for quantifying the relative contributions of the three domains on the likelihood of vaccination. The analysis was carried out using R statistical software. Thus, this document presents the R code used together with the results of the analysis

Acknowledgements

On behalf of the NDMC, we would like express our sincere gratitude to IHME for the technical support provided to us in data preparation and analysis, machine learning modeling and interpretation of results for our drivers of vaccination project. The IHME team's expertise and guidance were invaluable in helping us to understand the drivers of vaccination coverage in Ethiopia. We especially appreciate the time that you took to answer our questions and to help us to troubleshoot problems. We believe that the results of our project will have the potential to make a significant contribution to our understanding of the drivers of vaccination coverage in Ethiopia. We are confident that these results would not have been possible without your engagement. We are grateful for your commitment to advancing global health research. Your work is making a real difference in improving the health of the world population. Thank you!

Contents

Acronyms

CCEI Cold chain equipment inventory. 2, 27, 29–31, 33, 34, 48

EPHI Ethiopian public health institute. i

EPI Expanded programme on immunization. 23, 40

IHME Institute for Health Metrics and Evaluation. i, ii

MICE Multivariate imputation by chained equations. iii, 44

ML Machine Learning. i, 1, 2, 47, 61, 64

NDMC National Data Management Center for Health. i, ii

RI routine immunization. 8, 9, 11, 30, 31, 38, 40, 41, 54, 55, 58–60

SARA Service Availability and Readiness Assessment. iii, 1–10, 14–17, 19, 20, 22, 23, 25, 27, 37, 41, 43, 44

WHO World Health Organizations. 1, 3, 27

List of Figures

Chapter 1

Introduction

In this analysis, the drivers of improvements in childhood vaccination were quantified. The analysis was done following three-step analytical approach. The first step is to assemble individual-level dataset with linked variables on vaccination drivers across a range of data sources (surveys, administrative data and routine systems). We then model the relationship between drivers and likelihood of vaccination coverage to understand which drivers have the biggest impact on vaccination. Finally, we decompose the relationship of drivers on vaccination coverage over time to understand which drivers were associated with the greatest change.

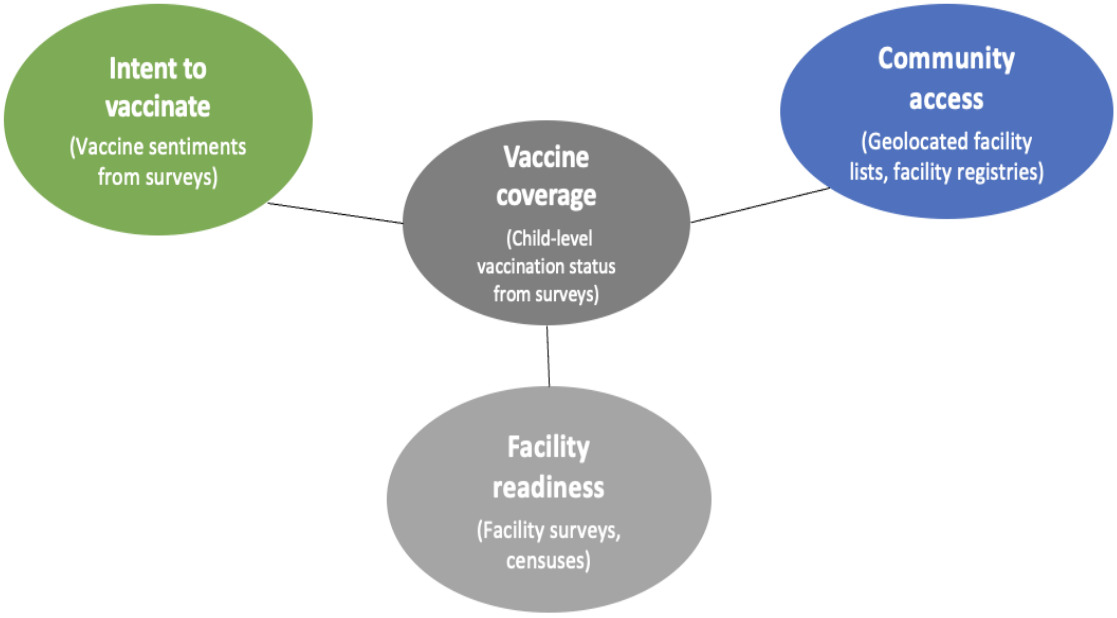

Figure 1.1: Conceptual Frame Work

As a general approach, the existing survey and administrative data over time were linked first, followed by construction of child-level cohort with estimates of intent (caregiver perceptions), access (travel time to closest facility with RI), and readiness (World Health Organizations (WHO)-SARA based index of vaccine service delivery). Tree-based ML methods were utilized to model interacting-nonlinear re-

lationship between domains, SHapley additive ExPlanations (SHAP) to decompose modeled contributions to likelihood of coverage [1].

In this manual, the overall procedures for data cleaning, manipulation and modeling were presented. The second chapter presents data pre-processing and data manipulation steps for all sources of data including SARA, Cold chain equipment inventory (CCEI). The third chapter presents steps of computing the three domains (readiness, intent, access) and the fourth chapter presents implementation of ML models for the cleaned data. All analysis were carried out using R software, thus the codes presented can directly be implemented using R software [2].

Chapter 2

Data Pre-Processing

2.1 Pre-processing 2018 Service Availability and Readiness Assessment (SARA) Data

SARA is a data set collected by. In this analysis, SARA data set is used to extract information on the readiness domain. We used WHO indicators for readiness of facilities for providing vaccines to children. The following sections describes the steps carried out to pre-process SARA 2018 dataset with objective of making the data ready for computing facility readiness indicators from it.

We started with loading the necessary packages or libraries used for data manipulation in R. The following code is used to do so.

```
  libs <- c('tidyverse', 'RColorBrewer', 'ggplot2', 'reshape', 'knitr',
  'data.table', 'geepack', 'mlmRev', 'lme4', 'xtable', 'readstata13',
  'MASS',  'lubridate', 'varhandle', 'readxl')
for(l in libs){
  if(!require(l,character.only = TRUE, quietly = TRUE))
  {message( sprintf('Did not have the required package
  << %s >> installed. Downloading now ... ',l))
    install.packages(l)
  }
  library(l, character.only = TRUE, quietly = TRUE)
}
```

The above code will check if the listed libraries are available in our computer, load (if available) and install and load (if missing!).

In other words, the code will install tidyverse, RColorBrewer, ggplot2, reshape, knitr, data.table, geepack, mlmRev, lme4, xtable, readstata13, MASS, lubridate, varhandle and readxl. The code first creates a vector of package names called libs. Then, it loops through the vector and checks if each package is installed. If the package is not installed, the code will print a message and then install the package. Finally, the code loads each package into the R environment.

After installing the necessary package and loading them in to the R environment,

we then set the working directory and clear our working environment using the following code. Any convenient folder can be set as working directory.

```
#set working directory:
setwd('Desktop/facility_readiness/SARA2018')
#clear working space:
rm(list = ls())
```

This will help us to ease our work by assigning any variables of our interest and save it to our working environment. After cleaning our working environment, we proceed with uploading the SARA 2018 survey in csv format. Note that we first need to set working directory and locate our data set in that folder for easy work.

```
# Load the dataset
sara_2018<-read.csv('All SARA 2018.csv')
#Convert all variables to lowercase for easier coding
names(sara_2018) <- tolower(names(sara_2018))
```

The following code first creates a new variable called iso3 and assigns it the value "ETH" for Ethiopia, the country where the SARA 2018 survey was conducted. The code then renames the following variables (adopted from SARA 2018 questionnaire):

- q001: fac_id
- q003: fac_name
- weight: fac_weight
- q005: admin1
- $q005_name$: $admin1\_name$
- q006: admin2
- $q006\_name$: $admin2\_name$
- q007: fac_type
- qmonth: svy_month
- qday: svy_day
- qyear: svy_year

```
sara_2018 <- sara_2018 %>%
 #Renaming subset of variables
 mutate(iso3="ETH",
        svy="SARA 2018",
        fac_id=q001,
        fac_name=str_to_title(q003),
        fac_weight=weight, #all blank
        admin1=q005,
        admin1_name=str_to_title(q005_name),
        admin2=q006,
        admin2_name=str_to_title(q006_name),
        fac_type=str_to_title(q007),
        svy_month=qmonth,
        svy_day=qday,
        svy_year=qyear)
```

In the above code, the *str_to_title*() function is used to capitalize the first letter of each word in a string. This is done for the fac_name, $admin1\_name$, and $admin2\_name$ variables to make the names more readable.

The fallowing code first renames the q009 variable in SARA 2018 data set (see appendix) to urban and changes the values to 0 for rural and 1 for urban.

The code then standardizes the following variables:

- result=as.numeric(factor(result)) converts the result variable from a factor to a numeric vector. A factor is a categorical variable that can have a limited number of values. The levels of the factor are converted to integers, and the order of the levels is preserved
- *svy_complete* : This variable reflects whether the survey was completed or not. The values 2, 3, and 4 are all codes for non-completion, so they are converted to 0. The value 1 is the code for completion, so it is left unchanged.
- q5001b: This variable contains the reason for non-completion. The text is trimmed to remove leading and trailing spaces.
- $comments\_about_{r}espondent$: This variable contains comments about the respondent. The text is trimmed to remove leading and trailing spaces.
- *any_other_comments*: This variable contains any other comments. The text is trimmed to remove leading and trailing spaces.
- operational: This variable indicates whether the facility is operational. The values 2 and 3 are codes for non-operational, so they are converted to 0. The value 1 is the code for operational, so it is left unchanged.
- $fac\_comments1$: This variable contains comments about specific questions. The text is trimmed to remove leading and trailing spaces and converted to lowercase.
- $fac\_comments2$: This variable contains any other comments. The text is trimmed to remove leading and trailing spaces and converted to lowercase. The above code also creates a new variable for the survey year, which is calculated from the $svy_{d}ate$ variable.

```
sara_2018 %>%
  mutate(urban=ifelse(q009==2,0,q009),
         #Facility weights are all 0 for svy_complete==0
         result=as.numeric(factor(result)),
         svy_complete=ifelse(q5001a==2 | q5001a==3 | q5001a==4,0,NA),
         svy_complete=ifelse(q5001a==1, 1, svy_complete),
         q5001b==str_trim(q5001b), comments_about_respondent=
         str_trim(comments_about_respondent),
         any_other_comments=str_trim(any_other_comments),
         operational=ifelse(svy_complete==1,1,NA),
         operational=ifelse(q5001a==4,0,operational),
    fac_comments1=str_trim(tolower(comments_on_specific_questions)),
         fac_comments2=str_trim(tolower(any_other_comments))
```

Facility ownership

The facility ownership variable in the sara 2018 dataset is coded as follows. in SARA 2018. the fac_own variable is a categorical variable that indicates the ownership of the facility with possible values of government/public, NGO/Not-for-profit, private-for-profit and mission/faith-based.

- The code first checks if the value of q008 is 1. If it is, the fac_own variable is assigned the value "Government/public".
- The code then checks if the value of q008 is 2. If it is, the fac_own variable is assigned the value "NGO/not-for-profit".
- The code then checks if the value of q008 is 96 and the value of $q008\_a$ is "suger factory". If it is, the fac_own variable is assigned the value "Government/public". This is because the Matahara General Hospital, which is owned by the government, is coded as 96 in the q008 variable.
- The code then checks if the value of q008 is 96 and the value of $q008\_a$ is "company employees cl". If it is, the fac_own variable is assigned the value "Private-for-profit". This is because the Sheble Transport Lower Clinic, which is owned by a private company, is coded as 96 in the q008 variable.
- The code then checks if the value of q008 is 96 and the value of $q008\_a$ is "INSTITUTIONAL CLINI". If it is, the fac_own variable is assigned the value "Private-for-profit". This is because the Awash Emenebered Medium Clinic, which is owned by a private company, is coded as 96 in the q008 variable.
- The code then checks if the value of q008 is 3. If it is, the fac_own variable is assigned the value "Private-for-profit".
- Lastly, the code then checks if the value of q007 is 4. If it is, the fac_own variable is assigned the value "Mission/faith-based".

```
sara_2018 %>% mutate(
         fac_own=ifelse(q008==1, "Government/public", ""),
         fac_own=ifelse(q008==2, "NGO/Not-for-profit", fac_own),
        fac_own=ifelse((q008==96&q008_a=='suger
        factory'),"Government/public", fac_own),
        fac_own=ifelse((q008==96&q008_a=='company employees
        cl'),"Private-for-profit", fac_own),
        fac_own=ifelse((q008==96&q008_a=='INSTITUITIONAL
        CLINI'),"Private-for-profit", fac_own),
         fac_own=ifelse(q008==3, "Private-for-profit", fac_own),
         fac_own=ifelse(q007==4, "Mission/faith-based", fac_own))
```

Facility type

The following code is used to re-code the facility type variable in the SARA 2018 dataset. The fac_type variable is a categorical variable that indicates the type of facility with possible values of referral hospital, general hospital, primary hospital,

health centre, health post, higher clinic, medium clinic, lower clinic and MCH specialized center.

- The code first checks if the value of fac_type is 1. If it is, the fac_type variable is assigned the value "Referral hospital".
- The code then checks if the value of fac_type is 2. If it is, the fac_type variable is assigned the value "General hospital".
- The code then checks if the value of fac_type is 3. If it is, the fac_type variable is assigned the value "Primary hospital".
- The code then checks if the value of fac_type is 4. If it is, the fac_type variable is assigned the value "Health centre".
- The code then checks if the value of fac_type is 5. If it is, the fac_type variable is assigned the value "Health post".
- The code then checks if the value of fac_type is 6. If it is, the fac_type variable is assigned the value "Higher clinic".
- The code then checks if the value of fac_type is 7. If it is, the fac_type variable is assigned the value "Medium clinic".
- The code then checks if the value of fac_type is 8. If it is, the fac_type variable is assigned the value "Lower clinic".
- The code then checks if the value of fac_type is 9. If it is, the fac_type variable is assigned the value "MCH specialized center".

```
sara_2018 %>% mutate(
     fac_type=ifelse(fac_type=="1","Referral hospital", fac_type),
     fac_type=ifelse(fac_type=="2","General hospital", fac_type),
     fac_type=ifelse(fac_type=="3","Primary hospital", fac_type),
     fac_type=ifelse(fac_type=="4","Health centre", fac_type),
     fac_type=ifelse(fac_type=="5","Health post", fac_type),
     fac_type=ifelse(fac_type=="6","Higher clinic", fac_type),
     fac_type=ifelse(fac_type=="7","Medium clinic", fac_type),
     fac_type=ifelse(fac_type=="8","Lower clinic", fac_type),
     fac_type=ifelse(fac_type=="9","MCH specialized center", fac_type)
```

In SARA 2018 data set, the variables $q013\_b$, $q013\_c$, and $q013_d$variables contain the degrees, minutes, and seconds of the latitude, respectively. The $q014\_b$, $q014\_c$, and $q014\_d$ variables contain the degrees, minutes, and seconds of the longitude, respectively. The code you provided is to convert the GPS coordinates from degrees, minutes, and seconds to decimal degrees.

The code first creates two new variables, latnum and longnum. The latnum variable is assigned the value of $q013\_b$ plus the value of $q013\_c$ divided by 60 plus the value of $q013\_d$ divided by 3600. The longnum variable is assigned the value of $q014\_b$ plus the value of $q014\_c$ divided by 60 plus the value of $q014\_d$ divided by 3600.

```
sara_2018 %>% mutate(
  latnum=q013_b + ((q013_c/60)+(q013_d/3600)),
  longnum=q014_b +((q014_c/60)+(q014_d/3600)))
```

Service availability and stocking

The following code is used to re-code routine immunization (RI) service availability and stocking in the SARA 2018 dataset. The *ri_services* variable indicates whether the facility provides RI services. The *ri_services_today* variable indicates whether the facility provides RI services at the time of data collection

The code first creates two new variables, *ri_services* and *ri_services_today* variables. The *ri_services* variable is assigned the value of 0 if the value of q1100 is 2, which indicates that the facility does not provide RI services. The *ri_services* variable is assigned the value of 1 if the value of q1100 is 1, which indicates that the facility does provide RI services.

The *ri_services_today* variable is assigned the value of 0 if the value of q1101 is 2, which indicates that the facility does not provide routine immunization services at the time of data collection. The *ri_services_today* variable is assigned the value of 1 if the value of q1101 is 1, which indicates that the facility does provide routine immunization services at the time of data collection.

```
sara_2018 %>% mutate(
    ri_services=ifelse(q1100==2,0,NA),
    ri_services=ifelse(q1100==1,1,ri_services),
    ri_services_today=ifelse(q1101==2,0,NA),
    ri_services_today=ifelse(q1101==1,1,ri_services_today))
```

Guideline availability is re-coded using *nat_child_ri_guidelines* and *any_child_ri_guidelines* variables. The *nat_child_ri_guidelines* variable indicates whether the facility has national child RI guidelines. The *any_child_ri_guidelines* variable indicates whether the facility has any child RI guidelines.

- The code first creates two new variables, *nat_child_ri_guidelines* and *any_child_ri_guidelines*. The *nat_child_ri_guidelines* variable is assigned the value of 0 if the value of q1105 is 2 or 3, which indicates that the facility does not have national child RI guidelines or that the guidelines are not seen while it is assigned the value of 1 if the value of q1105 is 1, which indicates that the facility has national child RI guidelines.
- The *any_child_ri_guidelines* variable is assigned the value of the *nat_child_ri_guidelines* variable. This is because the *any_child_ri_guidelines* variable is a more general measure of whether the facility has any child RI guidelines, including national guidelines.

Note that SARA 2018 does not ask about vaccine-specific frequency of services. The code then trims the whitespace from the *q*1103_*a* and *q*1104_*a* variables using the *str_trim*() function which removes leading and trailing whitespace from character strings.

```
 sara_2018 %>% mutate(
#Guideline availability:
#reported not seen + no guidelines recoded as No
nat_child_ri_guidelines=ifelse(q1105==2 | q1105==3,0,NA),
```

```
nat_child_ri_guidelines=ifelse(q1105==1,1,nat_child_ri_guidelines),
any_child_ri_guidelines=nat_child_ri_guidelines,
q1103_a=str_trim(as.character(q1103_a)),
q1104_a=str_trim(as.character(q1104_a)))
```

Facility-based services

The following code is used to capture the frequency of facility-based immunization services in the SARA 2018 dataset. The f_freq_daily, fac_freq_weekly, $fac\_freq\_monthly$, and $fac\_freq\_quarterly$ variables indicate whether the facility provides immunization services daily, weekly, monthly, or quarterly, respectively. The fac_freq_any variable indicates whether the facility provides any facility-based immunization services. The code first creates a new variable called f_freq_daily and assigns the value of 1 if the value of q1103 is 1, which indicates that the facility provides immunization services daily. The code then creates a new variable called fac_freq_weekly and assigns the value of 1 if the value of q1103 is 2, which indicates that the facility provides immunization services weekly.

The code then checks if the value of $q1103\_a$ is one of the following strings: "two days per weak", "2 days per weak", "3 days per weak", "twice per week", "per week two times", "2 times aweek", " twice aweek", "twice a week", "every other day", "2TIMES PER WEEK" OR "2times per week". If the value of $q1103\_a$ is one of these strings, the code assigns the value of 1 to the fac_freq_weekly variable. This is because these strings all indicate that the facility provides immunization services weekly. The code then creates a new variable called $fac\_freq\_monthly$ and assigns the value of 1 if the value of q1103 is 3, which indicates that the facility provides immunization services monthly.

The code then checks if the value of $q1103\_a$ is one of the following strings: "every 15 days", "every two weeks" OR "Three Times monthly". If the value of $q1103\_a$ is one of these strings, the code assigns the value of 1 to the $fac\_freq_monthly$ variable. This is because these strings all indicate that the facility provides immunization services at least monthly.

The code then creates a new variable called $fac\_freq\_quarterly$ and assigns the value of 1 if the value of q1103 is 4, which indicates that the facility provides immunization services quarterly. The code then creates a new variable called fac_freq_any and assigns the value of 1 if any of f_freq_daily, fac_freq_weekly, $fac\_freq\_monthly$, and $fac\_freq\_quarterly$ variables are equal to 1. This is because this variable indicates whether the facility provides any facility-based immunization services.

The code then checks if the value of $ri\_services$ is 0, which indicates that the facility does not provide RI services. If the value of $ri\_services$ is 0, the code assigns the value of 0 to the fac_freq_any variable. This is because a facility that does not provideRI services cannot provide facility-based immunization services.

```
 sara_2018 %>% mutate(
#Facility-based services
fac_freq_daily=ifelse(q1103==1,1,NA),
```

```
fac_freq_weekly=ifelse(q1103==2,1,NA),
#Recode 2x per week as weekly
fac_freq_weekly=ifelse(q1103_a=="two days per weak" |
q1103_a=="2 days per weak" | q1103_a=="3 days per weak" |
                         q1103_a=="twice per week" |
                         q1103_a=="per week two times" |
                         q1103_a=="2 times aweek" |
                         q1103_a=" twice aweek" |
                         q1103_a=="twice a week"|
                         q1103_a=="every other day"|
                         q1103_a=="2TIMES PER
                         WEEK"|q1103_a=="2times per
                         week",1,fac_freq_weekly),

fac_freq_monthly=ifelse(q1103==3,1,NA),
#Recode every 15 days and every two weeks as at least monthly
# recode 3x monthly as quarterly
fac_freq_monthly=ifelse(q1103_a=="every 15 days" |
q1103_a=="every two weeks"|q1103_a=="Three Times
monthly",1,fac_freq_monthly),
fac_freq_quarterly=ifelse(q1103==4,1,NA),
      # frequency of any immunization services (yes for all
      daily, weekly, monthly or quarterly yeses)
#no if the facility is not providing RI services

fac_freq_any=ifelse(fac_freq_daily==1 | fac_freq_weekly==1 |
fac_freq_monthly==1 | fac_freq_quarterly==1,1,NA),
fac_freq_any=ifelse(ri_services==0,0,fac_freq_any),
#recode open ended questions
fac_freq_any=ifelse(q1103_a=="no facility services" |
q1103_a=="NO GIVE THE SITE" | q1103_a=="provides outreach only"
| q1103_a=="no infacilty vaccination" |
                       q1103_a=="doesn't give in the facil" |
                       q1103_a=="only outreach immunizaion" |
                       q1103_a=="NO GIVE ONSITE IMMUNIZATI" |
                       q1103_a=="provide at otreach only" |
                       q1103_a=="doesnt provide at
                       facilit",0,fac_freq_any))
```

Finally, the code checks if the value of $q1103\_a$ is one of the following strings: "no facility services", "NO GIVE THE SITE", "provides outreach only", "no infacilty vaccination", "doesn't give in the facil", "only outreach immunizaion", "NO GIVE ONSITE IMMUNIZATI", "provide at otreach only", "doesnt provide at facilit". If the value of $q1103\_a$ is one of these strings, the code assigns the value of 0 to the fac_freq_any variable. This is because these strings all indicate that the facility does not provide any facility-based immunization services.

Outreach-based services

The following code is used to re-code outreach-based services variable in the SARA 2018 dataset:

- out_freq_daily: A binary variable indicating whether the respondent re-

ported that RH services are provided on a daily basis (1) or not (0).

- *out_freq_weekly*: A binary variable indicating whether the respondent reported that RH services are provided on a weekly basis (1) or not (0).
- *out_freq_monthly*: A binary variable indicating whether the respondent reported that RH services are provided on a monthly basis (1) or not (0).
- *out_freq_quarterly*: A binary variable indicating whether the respondent reported that RH services are provided on a quarterly basis (1) or not (0).
- *out_freq_any*: A binary variable indicating whether the respondent reported that RH services are provided on a daily, weekly, monthly, or quarterly basis (1) or not (0).

The code also takes into account the following factors when creating these variables:

- The value of the q1104 variable, which asks how often RI services are provided.
- The value of the *q*1104_*a* variable, which is an open-ended question asking for more detail about the frequency of RI service provision.
- The value of the *ri_services* variable, which indicates whether the respondent's facility provides RI services.

```
sara_2018 %>% mutate(
    #Outreach-based services

    #Daily service provision -- q1104==1
    out_freq_daily=ifelse(q1104==1,1,NA),

    #Weekly service provision -- q1104==2
    out_freq_weekly=ifelse(q1104==2,1,NA),
    #Recode 2 or 3x per week as indicated in open-ended question as
    weekly service provision
    out_freq_weekly=ifelse(q1104_a=="2 days per weak" |
    q1104_a=="2times per week", 1,out_freq_weekly),

    #Monthly service provision -- q1104==3
    out_freq_monthly=ifelse(q1104==3,1,NA),

    #Recode every 2 weeks, every 15 days, 2-5 days per month as
    indicated in open-ended question as monthly service provision
    out_freq_monthly=ifelse(q1104_a=="every 15 days" |
                            q1104_a=="two day's per month" |
                            q1104_a=="2 times amonth" |
                            q1104_a=="3 times per
                            month",1,out_freq_monthly),
    #Quarterly service provision -- q1104==4
    out_freq_quarterly=ifelse(q1104==4,1,NA),

    #Any outreach-based service provision
        ### To follow-up: what about "campaigns only" and/or "when
        #there's a campaign" -- should that count for "any [routine]
        #outreach"?
    out_freq_any=ifelse(out_freq_daily==1 | out_freq_weekly==1 |
```

```
out_freq_monthly==1 | out_freq_quarterly==1,1,NA),
#Recode every 6 months as any outreach service provision - less
than quarterly
out_freq_any=ifelse(ri_services==0,0,out_freq_any),
#Recode open-ended question responses around no outreach-based servic
out_freq_any=ifelse(ri_services==1 & (q1104==96 &
(q1104_a=="no out each" | q1104_a=="No out reach" |
q1104_a=="NO OUT REACH" | q1104_a=="NO OUTREACH" |
q1104_a=="no outreach" | q1104_a=="n0 0ut reach" |
q1104_a=="n0 outreach" | q1104_a=="no outreache" |
q1104_a=="no outrech" | q1104_a=="no outreche" |
q1104_a=="NO outrich" | q1104_a=="no outrich" |
q1104_a=="not outreach t all" |
q1104_a=="their is no out reach"|
q1104_a=="NO out reach service" |
q1104_a=="no out  reach service"|
q1104_a=="No out reach service" |
q1104_a=="NO OUT REACH SERVICE" |
q1104_a=="No outreach service" |
q1104_a=="no outreach service" |
q1104_a=="no out reach serivce" |
q1104_a=="n0 outreach service" |
q1104_a=="no outreached service" |
q1104_a=="no outrich service" |
q1104_a=="no service at outreach"|
q1104_a=="no out reach services" |
q1104_a=="no outreach services" |
q1104_a=="no out reach servise" |
q1104_a=="no ut each serives" |
q1104_a=="no out each sevices" |
q1104_a=="no outreach ervise" |
q1104_a=="only in the facility" |
q1104_a=="Only in the facility" |
q1104_a=="in the facility only" |
q1104_a=="IN THE FACILITY ONLY" |
q1104_a=="In the facility only" |
q1104_a=="IN THE FACILITY ONLY" |
q1104_a=="facility only" |
q1104_a=="in facility only" |
q1104_a=="only in the faclity" |
q1104_a=="only at the facility" |
q1104_a=="only given inthe facility" |
q1104_a=="No out each program" |
q1104_a=="no out reach program" |
q1104_a=="no outreach pg" |
q1104_a=="no any outreach program"|
q1104_a=="no in outreach program" |
q1104_a=="NO OUTREACH PROGRAM" |
q1104_a=="no program to outreach" |
q1104_a=="NEVER GIVEN AS OUTREACH" |
q1104_a=="NO GIVE" |
q1104_a=="do not provide service" |
```

```
    q1104_a=="donot provides" |
    q1104_a=="dont offer outreach servi" |
    q1104_a=="dont provide outreach" |
    q1104_a=="no give outreach" |
    q1104_a=="no given out side" |
     q1104_a=="no out reach given" |
     q1104_a=="no outreach given" |
     q1104_a=="no provide" |
     q1104_a=="not give" |
    q1104_a=="not given  outreach servic" |
    q1104_a=="not provide"|
    q1104_a=="not provided" |
    q1104_a=="not prvided"|
    q1104_a=="notoffered" |
    q1104_a=='no outreach immunization'|
    q1104_a=="no outreach vaccine" |
    q1104_a=="not aut reach at all" |
    q1104_a=="not Applicable" |
    q1104_a=="not pplicable" |
    q1104_a=="Not applicable" |
    q1104_a=="nOT APPLICABLE" |
     q1104_a=="NOT APPLICABLE" |
     q1104_a=="N/A" | q1104_a=="N/a" |
    q1104_a=="No" | q1104_a=="no" |
    q1104_a=="not done" |
    q1104_a=="not done outrech" |
    q1104_a=="out reach not done" |
    q1104_a=="outreach not done" ),0,out_freq_any))
```

Availability of vaccines in the stock: Penta

The availability of vaccines in the stock variable is re-coded using the following code.

- *vxinstock_penta* : A binary variable indicating whether pentavalent vaccine is available in the stock (1) or not (0).
- *nostockout3mths_penta*: A binary variable indicating whether there was no stock-out of pentavalent vaccine in the past 3 months (1) or yes stockout (0) or not observed today (0).
- *neveravail_penta*: A binary variable indicating whether pentavalent vaccine is never available (1) or not (0).

The code takes into account the following factors when creating these variables:

- The value of the *q*1116_02 variable, which asks about the availability of pentavalent vaccine in the stock.
- The value of the *q*1117_02 variable, which asks about whether there was a stock-out of pentavalent vaccine in the past 3 months.

```
sara_2018 %>% mutate(
    #Availability of vaccines in the stock
    #Pentavalent/DTP
    #Code as 0 if available not but valid, reported but not seen, not
    available today, never available
```

```
    vxinstock_penta=ifelse(q1116_02==2 |
    q1116_02==3 | q1116_02==4 | q1116_02==5,0,NA),
    vxinstock_penta=ifelse(q1116_02==1,1,vxinstock_penta),
    #Observed at least one valid
    #stock-out (pentavalent, 1=no stock-out, 0= yes stockout
    OR not observed today)
    nostockout3mths_penta=ifelse(q1117_02==1 |
    q1116_02==4,0,NA), nostockout3mths_penta=
    ifelse(q1117_02==2,1, nostockout3mths_penta),
    nostockout3mths_penta=ifelse(is.na(vxinstock_penta),NA,
    nostockout3mths_penta),
    #Pentavalent not available if (product not offered OR never
    available)
    neveravail_penta=ifelse(q1117_02==4 | q1116_02==5,1,NA))
```

Availability of vaccines in the stock: BCG

For BCG, the following code is used to check its availability using the SARA 2018 dataset:

- *vxinstock_bcg*: This variable indicates whether BCG vaccine is in stock. A value of 0 indicates that the vaccine is not in stock, and a value of 1 indicates that it is in stock. The variable is coded as 0 if available but not valid, reported but not seen, not available today and never available.
- *nostockout3mths_bcg* : This variable indicates whether there has been no stockout of BCG vaccine in the past 3 months. A value of 0 indicates that there has been a stockout, and a value of 1 indicates that there has not been a stockout.
- *neveravail_bcg* : This variable indicates whether BCG vaccine has never been available. A value of 1 indicates that the vaccine has never been available, and a value of NA indicates that the information is not available.

```
sara_2018 %>% mutate(
    #BCG
    vxinstock_bcg=ifelse(q1116_04==2 | q1116_04==3 |
    q1116_04==4 | q1116_04==5,0,NA),
    vxinstock_bcg=ifelse(q1116_04==1,1,vxinstock_bcg),
    #Observed at least one valid
    nostockout3mths_bcg=ifelse(q1117_04==1 | q1116_04==4,0,NA),
    nostockout3mths_bcg=ifelse(q1117_04==2,1, nostockout3mths_bcg),
    nostockout3mths_bcg=ifelse(is.na(vxinstock_bcg),NA,
    nostockout3mths_bcg), neveravail_bcg=ifelse(q1117_04==4 |
    q1116_04==5,1,NA))
```

Availability of vaccines in the stock: Measles

The following code will will create new variables for measles containing vaccine availability and stockout from SARA 2018 dataset:

- v*xinstock_mcv*: This variable indicates whether measles containing vaccine is in stock. A value of 0 indicates that the vaccine is not in stock, and a value

of 1 indicates that it is in stock. The variable is coded as 0 if available but not valid, reported but not seen, not available today or never available (see *q*1116_01 in SARA 2018 questionnaire).

- *nostockout3mths_mcv* : This variable indicates whether there has been no stockout of measles containing vaccine in the past 3 months. A value of 0 indicates that there has been a stockout, and a value of 1 indicates that there has not been a stockout.

- *neveravail_mcv*: This variable indicates whether measles containing vaccine has never been available. A value of 1 indicates that the vaccine has never been available, and a value of NA indicates that the information is not available.

```
sara_2018 %>% mutate(
    #Measles containing vaccine
    vxinstock_mcv=ifelse(q1116_01==2 | q1116_01==3 | q1116_01==4 |
    q1116_01==5,0,NA),
    vxinstock_mcv=ifelse(q1116_01==1,1,vxinstock_mcv),
    #Observed at least one valid
    nostockout3mths_mcv=ifelse(q1117_01==1 | q1116_01==4,0,NA),
    nostockout3mths_mcv=ifelse(q1117_01==2,1, nostockout3mths_mcv),
    nostockout3mths_mcv=ifelse(is.na(vxinstock_mcv),
    NA,nostockout3mths_mcv),
    neveravail_mcv=ifelse(q1117_01==4 | q1116_01==5,1,NA))
```

Availability of vaccines in the stock: Polio

The code below creates new variables for the availability of polio vaccines, OPV and IPV. The *vxinstock_opv* variable indicate whether the vaccine is in stock and is coded as 0 if available but not valid, reported but not seen, not available today and never available (see *q*1116_03 in SARA 2018 questionnaire). The *nostockout3mths_opv* variables indicate whether there has been no stockout of the vaccine in the past 3 months where as the variable *neveravail_opv* indicate whether the vaccine has never been available. In addition, the *vxinstock_ipv* and *nostockout3mths_ipv* variables were set to NA because IPV vaccine is not included in the SARA 2018 dataset.

```
sara_2018 %>% mutate(
    #Polio - OPV
    vxinstock_opv=ifelse(q1116_03==2 | q1116_03==3 | q1116_03==4 |
    q1116_03==5,0,NA),
    vxinstock_opv=ifelse(q1116_03==1,1,vxinstock_opv),
    #Observed at least one valid
    nostockout3mths_opv=ifelse(q1117_03==1 | q1116_03==4,0,NA),
    nostockout3mths_opv=ifelse(q1117_03==2,1, nostockout3mths_opv),
    nostockout3mths_opv=ifelse(is.na(vxinstock_opv),NA,
    nostockout3mths_opv),
    neveravail_opv=ifelse(q1117_03==4 | q1116_03==5,1,NA),
    #Polio - IPV (not inclucded in SARA 2018)
    vxinstock_ipv=NA, nostockout3mths_ipv=NA,
    neveravail_ipv=NA)
```

Availability of vaccines in the stock: PCV

The following code will create the new variables for the availability of pneumococcal conjugate vaccine (PCV). The *vxinstock_pcv* variable indicate whether the vaccine is in stock and is coded as 0 if available but not valid, reported but not seen, not available today, never available (see *q*1116_06 in SARA 2018 questionnaire). The *nostockout3mths_pcv* variable indicate whether there has been no stockout of the vaccine in the past 3 months. The *neveravail_pcv* variable indicate whether the vaccine has never been available.

```
sara_2018 %>% mutate(
    #PCV
    vxinstock_pcv=ifelse(q1116_06==2 | q1116_06==3 |
    q1116_06==4 | q1116_06==5,0,NA),
    vxinstock_pcv=ifelse(q1116_06==1,1,vxinstock_pcv),
    #Observed at least one valid
    nostockout3mths_pcv=ifelse(q1117_06==1 | q1116_06==4,0,NA),
    nostockout3mths_pcv=ifelse(q1117_06==2,1, nostockout3mths_pcv),
    nostockout3mths_pcv=ifelse(is.na(vxinstock_pcv),NA,
    nostockout3mths_pcv),
    neveravail_pcv=ifelse(q1117_06==4 | q1116_06==5,1,NA))
```

Availability of vaccines in the stock: Rota

The following codes creates the new variables for the availability of rotavirus vaccine (RotaC). The *vxinstock_rotac* variables indicate whether the vaccine is in stock whic is coded as 0 if available not but valid, reported but not seen, not available today, never available (see *q*1116_05 in SARA 2018 questionnaire). In addition, the *nostockout3mths_rotac* variables indicate whether there has been no stockout of the vaccine in the past 3 months while the *neveravail_rotac* variables indicate whether the vaccine has never been available.

```
sara_2018 %>% mutate(
    #RotaC
    vxinstock_rotac=ifelse(q1116_05==2 | q1116_05==3 | q1116_05==4 |
    q1116_05==5,0,NA), vxinstock_rotac=ifelse(q1116_05==1,1,vxinstock_rot
    #Observed at least one valid
    nostockout3mths_rotac=ifelse(q1117_06==1 | q1116_06==4,0,NA),
    nostockout3mths_rotac=ifelse(q1117_06==2,1,
    nostockout3mths_rotac),
nostockout3mths_rotac=ifelse(is.na(vxinstock_rotac),NA,
    nostockout3mths_rotac),
    neveravail_rotac=ifelse(q1117_06==4 | q1116_06==5,1,NA))
```

Storage and Cold Chain

Regrading storage and cold chain, new variable named $fac\_storesvx$ is created that indicates whether the facility stores vaccines. The variable will be set to NA because the SARA 2018 dataset does not have any specific questions on vaccine storage.

```
sara_2018 %>% mutate(
```

```
    #Storage and cold chain
    #no specific questions on vaccine storage
    fac_storesvx=NA)
```

Storage and Cold Chain: Functional Fridge

Regarding functional fridge availability, the follwoing code will will create a new variable called *fridge_avail* that indicates whether the facility has a functional fridge. The variable will be set to 0 if the fridge is available but not functional, available but not known if functioning, or not available at all (see $q1108$ in SARA 2018 questionnaire). The variable will be set to 1 if the fridge is functional.

```
sara_2018 %>% mutate(
    #Functional fridge
    fridge_avail=ifelse(q1108==2 | q1108==3 | q1108==4,0,NA),
    fridge_avail=ifelse(q1108==1,1,fridge_avail))
```

Storage and Cold Chain: Cold-chain monitoring system

For the cold chain monitoring system, the follwing code will create:

- *coldchain_system* : Indicates whether the facility has a cold chain monitoring system. The system is considered to be present if either a thermometer or a logger is available and functioning. The variable is is recoded as 0 if neither Thermometer nor logger is available (see $q1111$ in SARA 2018 questionnaire).
- *coldchain_system*$2x$_30*days*: Indicates whether the cold chain monitoring system has been checked twice in the past 30 days.
- *fridge_temp* : Indicates the temperature of the fridge. The temperature is considered to be within the recommended range of 2 to 8 degrees Celsius if the value is 1. The temperature is considered to be out of range if the value is 3. The temperature recording is not available if the value is 2 or 4.
- *vx_carriers*: Indicates whether the facility has vaccine carriers (cold boxes). The carriers are considered to be present if they are observed. These questions are asked irrespective of storage response. The variable is recoded 1 if observed other wise 0 reported not seen OR not available (see $q1107\_03$ in SARA 2018 questionnaire).

```
sara_2018 %>% mutate(
    coldchain_system=ifelse(q1111_01a==3 & q1111_02a==3,0,NA),
    coldchain_system=ifelse(q1111_02a !=1 & (q1111_01a==2 |
    q1111_01a==3),0,coldchain_system),
    coldchain_system=ifelse(q1111_01a !=1 & (q1111_02a==2 |
    q1111_02a==3),0, coldchain_system),
    coldchain_system=ifelse(q1111_02a !=1 & q1111_01a==1 &
    (q1111_01b==2 | q1111_01b==8), 0, coldchain_system),
    coldchain_system=ifelse(q1111_01a !=1 & q1111_02a==1 &
    (q1111_02b==2 | q1111_02b==8), 0, coldchain_system),
    #cold chain system recoded yes if either of Thermometer or
    #logger is observed and functioning
    coldchain_system=ifelse((q1111_01a==1 & q1111_01b==1) |
```

```
    (q1111_02a==1 & q1111_02b==1),1,coldchain_system),
    coldchain_system2x_30days=ifelse(coldchain_system==0 |
    (coldchain_system==1 & (q1112 !=1 | q1113 !=1)),0,NA),
    coldchain_system2x_30days=ifelse(coldchain_system==1 & q1112==1 &
    q1113==1,1,coldchain_system2x_30days),
    fridge_temp=ifelse(q1114==1, "Between 2 and 8 degrees C", ""),
    fridge_temp=ifelse(q1114==3, "Out of range", fridge_temp),
    fridge_temp=ifelse(q1114==2 | q1114==4,"Recording not available",
    fridge_temp),
    vx_carriers=ifelse(q1107_03==1,1,NA),
    vx_carriers=ifelse(q1107_03==2 | q1107_03==3,0,vx_carriers))
```

Storage and Cold Chain: Ice packs for vaccine carriers For vaccine carriers ice packs, the following new variables were created using the below code:

- *vx_carriers_icepacks*: Indicates whether the vaccine carriers have a set of ice packs. The ice packs are considered to be present if they are observed.
- *ipc_runningwater*: Indicates whether the facility has running water (piped, bicket with tap, or pour pitcher)
- *ipc_soap* : Indicates whether the facility has soap (hand washing soap or liquid soap).
- *ipc_sharpsbox* : Indicates whether the facility has a sharps box (immunization specific).
- *ipc_sharpsbox_gen*: Indicates whether the facility has a sharps box (general).
- *ipc_latexgloves* : Indicates whether the facility has latex gloves (general).
- *ipc_disinfectant*: Indicates whether the facility has disinfectant (general).

```
sara_2018 %>% mutate(
    #Set of ice packs for vaccine carriers
    vx_carriers_icepacks=ifelse(q1107_04==1,1,NA),
    vx_carriers_icepacks=ifelse(q1107_04==2 |
    q1107_04==3,0,vx_carriers_icepacks),
    ipc_runningwater=ifelse(q600_01==2 | q600_01==3,0,NA),
    ipc_runningwater=ifelse(q600_01==1,1,ipc_runningwater),

    #Soap (hand washing soap OR liquid soap):
    ipc_soap=ifelse(q600_02==2 | q600_02==3,0,NA),
    ipc_soap=ifelse(q600_02==1,1,ipc_soap),

    #Sharps box -- immunization specific and then general
    #immunization specific
    ipc_sharpsbox=ifelse(q1107_02==2 | q1107_02==3,0,NA),
    ipc_sharpsbox=ifelse(q1107_02==1,1,ipc_sharpsbox),
    #general
    ipc_sharpsbox_gen=ifelse(q600_06==2 | q600_06==3,0,NA),
    ipc_sharpsbox_gen=ifelse(q600_06==1,1,ipc_sharpsbox_gen),

    #Latex gloves -general
    ipc_latexgloves=ifelse(q600_04==2 | q600_04==3,0,NA),
```

```
    ipc_latexgloves=ifelse(q600_04==1,1,ipc_latexgloves),

    #Disinfectant -general
    ipc_disinfectant=ifelse(q600_07==2 | q600_07==3,0,NA),
    ipc_disinfectant=ifelse(q600_07==1,1,ipc_disinfectant))
```

Storage and Cold Chain: Disposable syringes or needles

For disposable syringes or needles (for both immunization specific and general), the following new variables were created using the below code:

- *ipc_dispsyringes*: Indicates whether the facility has disposable syringes or needles (immunization specific). The syringes or needles are considered to be present if they are observed (see *q*1107_01 in SARA 2018 questionnaire).
- *ipc_dispsyringes_gen*: Indicates whether the facility has disposable syringes or needles (general). The syringes or needles are considered to be present if at least one of the two questions (*q*600_08 and *q*600_09) is answered "yes".
- *ipc_pedalwastebin*: Indicates whether the facility has a pedal waste bin (with plastic liner) (general). The waste bin is considered to be present if it is observed see *q*600_05 in SARA 2018 questionnaire)

```
sara_2018 %>% mutate(
    #Disposable syringes or needles
    # immunization specific
    ipc_dispsyringes=ifelse(q1107_01==1,1,NA),
    ipc_dispsyringes=ifelse(q1107_01==2 |
    q1107_01==3,0,ipc_dispsyringes),
    # general
    ipc_dispsyringes_gen=ifelse(q600_08 !=1 & q600_09 !=1 &
    !is.na(q600_08) & !is.na(q600_09),0,NA),
    ipc_dispsyringes_gen=ifelse(q600_08==1 |
    q600_09==1,1,ipc_dispsyringes_gen),

    #Pedal waste bin (with plastic liner) -general
    ipc_pedalwastebin=ifelse(q600_05==2 | q600_05==3,0,NA),
    ipc_pedalwastebin=ifelse(q600_05==1,1,ipc_pedalwastebin),
    #Other waste bin (conditional on not having a pedal waste
    #bin available)
    ipc_othwastebin=NA,
    #Alcohol hand rub
    ipc_alcoholhandrub=ifelse(q600_03==2 | q600_03==3,0,NA),
    ipc_alcoholhandrub=ifelse(q600_03==1,1,ipc_alcoholhandrub))
```

The following line of codes creates new variables from SARA 2018 questionnaire for masks, gowns, eye protection, vaccination records(such as immunization cards/child health booklet, tally sheets, registers).

```
    sara_2018 %>% mutate(
    #Masks, Gowns, Eye Protection -- not explicitly asked about
    ipc_masks=NA,
    ipc_gowns=NA,
    ipc_eyeprotect=NA,
```

```
#IPC safety guidelines (page 11)
ipc_guidelines=ifelse(q422==2 | q422==3,0,NA),
ipc_guidelines=ifelse(q422==1,1,ipc_guidelines),

#Vaccination records (immunization cards/child
#health booklet, tally sheets, registers)

supply_blankimmuncards=ifelse(q1107_05==2 | q1107_05==3,0,NA),
supply_blankimmuncards=ifelse(q1107_05==1,1,
supply_blankimmuncards),
supply_tallysheets=ifelse(q1107_06==2 | q1107_06==3,0,NA),
supply_tallysheets=ifelse(q1107_06==1,1,supply_tallysheets),
supply_immunregister=ifelse(q1107_07==2 | q1107_07==3,0,NA),
supply_immunregister=ifelse(q1107_07==1,1,supply_immunregister))
```

Infrastructure

The following code generated new variable for measurement of availability of infrastructure such as electricity, functional generator and land line (phone) from from SARA data set with the following columns:

- electricity: The source of electricity for the facility. Possible values are "Connected to grid, always available", "Connected to grid, sometimes interrupted", and "Not connected to grid". The *ifelse()* function in this case is used to create new columns based on the values of existing columns. That is, the electricity column is created by using the *ifelse()* function to check the values of the q410 and q412 columns. If q410 is equal to 1 and q412 is equal to 1, then the value of electricity is set to "Connected to grid, always available". Otherwise, the value of electricity is set to the empty string.

- $funct\_generator$: Whether the generator is functional. Possible values are 1 (functional) and 0 (not functional).

- $funct\_solar$: Whether the solar system is functional. Possible values are 1 (functional) and 0 (not functional).

- $funct\_computer$: Whether the computer is functional. Possible values are 1 (functional) and 0 (not functional).

- $funct\_phone$: Whether there is a functioning landline or cell phone. Possible values are 1 (functional) and 0 (not functional).

- $healthdata\_system$: Whether the facility has a health services data system with regular updates. Possible values are NA (data not collected) and 1 (yes) and 0 (no).

- $cleanliness\_score8pts$: The cleanliness score of the facility, on a scale of 0 to 8. Possible values are NA (data not collected) and an integer from 0 to 8.

- $clientfees\_any$: Whether the facility charges any fees to clients. Possible values are NA (data not collected) and 1 (yes) and 0 (no).

- *clientfees_healthcard*: Whether the facility charges a fee for health cards. Possible values are NA (data not collected) and 1 (yes) and 0 (no).
- *clientfees_vaccines*: Whether the facility charges a fee for vaccines. Possible values are NA (data not collected) and 1 (yes) and 0 (no).
- *clientfees_syringe* : Whether the facility charges a fee for syringes. Possible values are NA (data not collected) and 1 (yes) and 0 (no).

The NA values in the *healthdata_system*, *cleanliness_score8pts*, *clientfees_any*,*clientfees_health* *clientfees_vaccines*, and *clientfees_syringe* columns indicate that these data were not collected for the SARA dataset.

```
sara_2018 %>% mutate(
    #Infrastructure
    electricity=ifelse(q410==1 & q412==1, "Connected to grid,
    always available", ""),
    electricity=ifelse(q410==1 & (q412==2 | q412==3),
    "Connected to grid, sometimes interrupted", electricity),
    #Not connected to grid if no electric source,
    #(generator OR solar system are main source of electricity)
    electricity=ifelse(q408==2 | (q410==2 | q410==3), "Not connected
    to grid", electricity),
    funct_generator=ifelse((q410==2 | q411==2) & q413==1,1,NA),
    funct_generator=ifelse((q410==2 | q411==2) &
    q413==2,0,funct_generator),
    funct_generator=ifelse(q408==2,0,funct_generator),

    funct_solar=ifelse((q410==3 | q411==3) & q416==1,1,NA),
    funct_solar=ifelse((q410==3 | q411==3) & (q416==2 |
    q416==3),0,funct_solar),
    funct_solar=ifelse(q408==2,0,funct_solar),

    funct_computer=ifelse(q403==2,0,q403),
    #Functioning landline or cell phone
    funct_phone=ifelse((q400==1 | q401==1),1,NA),
    funct_phone=ifelse(q400==2 & q401==2,0,funct_phone),

    #Has a health services data system with regular updates -
    data not collected
    healthdata_system=NA,
    healthdata_system=NA,

    #Cleaniness indicators - not collected for SARA
    cleanliness_score8pts=NA,

    #Fees - not collected in SARA
    clientfees_any=NA,
    clientfees_any=NA,

    #Conditional on indicating separate fees

    clientfees_healthcard=NA,
    clientfees_vaccines=NA,
```

```
        clientfees_syringe=NA)
```

Staff and Training

The code below creates the following new columns from SARA 2018 data set. Note in this case also that, the *felse()* function is used to create new columns based on the values of existing columns.

- *staff_gp*: The number of general and specialist medical doctors (MDs) at the facility.
- *staff_nurses*: The number of nursing professionals at the facility.
- *staff_chw*: The number of health extension workers (CHWs) at the facility.
- *supv_lastvisit*: The time since the last supervisor visit. Possible values are "None", "Within past 1 month", "Within past 3 months", and "More than 3 months ago".
- *supv_pharmacy*: Whether the supervisor reviewed the pharmacy (e.g. drug stock out, expiry, records, etc.). Possible values are 0 (no) and 1 (yes).
- *supv_staffing*: Whether the supervisor reviewed the staffing (e.g. staff available and training). Possible values are 0 (no) and 1 (yes).
- *supv_data*: Whether the supervisor reviewed the data (e.g. completeness, quality, and timely reporting). Possible values are 0 (no) and 1 (yes).

```
sara_2018 %>% mutate(
    #Staff and training#
    #Number of Staffs with qualifications:
    staff_total=NA, #Focus on sub-group
    #Generalist and specialist MDs
    staff_gp=q200_01a + q200_02a,
    #Nursing professionals
    staff_nurses=q200_04a,
    #Health extension workers
    staff_chw=q200_12a,
    #Meetings - management, staff-community meetings: data not
    #collected in SARA
    freq_mgmtmtgs=NA,
    any_mgmtmtgs=NA,
    freq_staffcommmtgs=NA,
    any_staffcommmtgs=NA,
    #Last supervisor visit
    supv_lastvisit=ifelse(eth_19==2, "None",NA),
    supv_lastvisit=ifelse(eth_19==1 & q430==1, "Within past 1 month",
    supv_lastvisit),
    supv_lastvisit=ifelse(eth_19==1 & q430==2, "Within past 3 months",
    supv_lastvisit),
    supv_lastvisit=ifelse(eth_19==1 & q430==3, "More than 3 months
    ago", supv_lastvisit),
    # Pharmacy, Staffing, Data
    supv_pharmacy=ifelse(q430==1 & q431_01==2,0,NA),
    supv_pharmacy=ifelse(q430==1 & q431_01==1,1,supv_pharmacy),
```

```
    supv_staffing=ifelse(q430==1 & q431_02==2,0,NA),
    supv_staffing=ifelse(q430==1 & q431_02==1,1,supv_staffing),
    supv_data=ifelse(q430==1 & q431_03==2,0,NA),
    supv_data=ifelse(q430==1 & q431_03==1,1,supv_data))
```

Supervisory actions

Regarding supervisory actions, the following code will create the following columns from SARA 2018 dataset:

- *supv_pharmacy* : Whether the supervisor reviewed the pharmacy (e.g. drug stock out, expiry, records, etc.). Possible values are 0 (no) and 1 (yes).
- *supv_staffing*: Whether the supervisor reviewed the staffing (e.g. staff available and training). Possible values are 0 (no) and 1 (yes).
- *supv_data*: Whether the supervisor reviewed the data (e.g. completeness, quality, and timely reporting). Possible values are 0 (no) and 1 (yes).
- *prov_epitrain_2yrs*: Whether at least one provider has had different types of training with respect to Expanded programme on immunization (EPI) in the last 2 years. Possible values are 0 (no) and 1 (yes). The *prov_epitrain_2yrs* column is created by using the *ifelse()* function to check the values of the q1106_01 through q1106_07 columns under the SARA 2018 dataset. If all of these values are equal to 3, then the value of *prov_epitrain_2yrs* is set to 0, indicating that no provider has had any training in EPI in the last 2 years. Otherwise, the value of *prov_epitrain_2yrs* is set to 1, indicating that at least one provider has had some training in EPI in the last 2 years.

```
sara_2018 %>% mutate(
    #Supervisory actions - if no last supervisory list, recode to 0
    supv_pharmacy=ifelse(supv_lastvisit=="None",0,supv_pharmacy),
    supv_staffing=ifelse(supv_lastvisit=="None",0,supv_staffing),
    supv_data=ifelse(supv_lastvisit=="None",0,supv_data),

    #At least one EPI training in the last 2 years
    prov_epitrain_2yrs=ifelse(q1106_01==3 & q1106_02==3 & q1106_03==3 &
    q1106_04==3 & q1106_05==3 & q1106_06==3 & q1106_07==3,0,NA),
    prov_epitrain_2yrs=ifelse(q1106_01==1 | q1106_02==1 | q1106_03==1 |
    q1106_04==1 | q1106_05==1 | q1106_06==1 |
    q1106_07==1,1,prov_epitrain_2yrs))
```

Specific types of training

For specific type of training, the following columns were created using the code below:

- *prov_iiptrain_2yrs*: Whether immunization service provider has received training in immunization in practice or equivalent immunization service delivery in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_vxmangtrain_2yrs*: Whether immunization service provider has received training in vaccine management/handling and cold chain in the last 2

years. Possible values are 0 (no) and 1 (yes).

- *prov_datatrain_2yrs*: Whether immunization service provider has received training in data reporting and monitoring of service delivery in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_surveiltrain_2yrs*: Whether immunization service provider has received training in disease surveillance and reporting in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_injsafetrain_2yrs*: Whether immunization service provider has received training in injection safety and waste management in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_redtrain_2yrs*: Whether immunization service provider has received training in Reaching Every District (RED) in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_pcvrotatrain_2yrs*: Whether the provider has received training in PCV or rotavirus vaccine in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_episupv_2yrs*: Whether the provider has received supportive supervision in EPI in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_iipsupv_2yrs*: Whether the provider has received supervision in immunization in practice or equivalent immunization service delivery in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_vxmangsupv_2yrs*: Whether the provider has received supervision in vaccine management/handling and cold chain in the last 2 years. Possible values are 0 (no) and 1 (yes).

```
sara_2018 %>% mutate(
    #Specific types of training in the last two years
    #Immunization in practice
    prov_iiptrain_2yrs=ifelse(q1106_01==1,1,NA), #
    prov_iiptrain_2yrs=ifelse(q1106_01==3,0,prov_iiptrain_2yrs),
    #Vaccine management/handling and cold chain
    prov_vxmangtrain_2yrs=ifelse(q1106_02==1,1,NA), #
    prov_vxmangtrain_2yrs=ifelse(q1106_02==3,0,prov_vxmangtrain_2yrs),

    #Data reporting and monitoring of service delivery
    prov_datatrain_2yrs=ifelse(q1106_03==1,1,NA), #
    prov_datatrain_2yrs=ifelse(q1106_03==3,0,prov_datatrain_2yrs),
    #Disease surveillance and reporting
    prov_surveiltrain_2yrs=ifelse(q1106_04==1,1,NA), #
    prov_surveiltrain_2yrs=ifelse(q1106_04==3,0,
    prov_surveiltrain_2yrs),
    #Injection safety and waste management
    prov_injsafetrain_2yrs=ifelse(q1106_05==1,1,NA), #
    prov_injsafetrain_2yrs=ifelse(q1106_05==3,0,
    prov_injsafetrain_2yrs),
    #Reaching Every District (RED)
    prov_redtrain_2yrs=ifelse(q1106_06==1,1,NA), #
    prov_redtrain_2yrs=ifelse(q1106_06==3,0,prov_redtrain_2yrs),
```

```
    #PCV or rotavirus vaccine training
    prov_pcvrotatrain_2yrs=ifelse(q1106_07==1,1,NA), #
    prov_pcvrotatrain_2yrs=ifelse(q1106_07==3,0,
    prov_pcvrotatrain_2yrs),
    #Supportive training
    prov_episupv_2yrs=ifelse(q1106_01==3 & q1106_02==3 & q1106_03==3 &
    q1106_04==3 & q1106_05==3 & q1106_06==3 & q1106_07==3,0,NA),
    prov_episupv_2yrs=ifelse(q1106_01==2 | q1106_02==2 | q1106_03==2 |
    q1106_04==2 | q1106_05==2 | q1106_06==2 |
    q1106_07==2,1,prov_episupv_2yrs),
    #Immunization in practice or equivalent immunization service
    delivery training
    prov_iipsupv_2yrs=ifelse(q1106_01==2,1,NA), #
    prov_iipsupv_2yrs=ifelse(q1106_01==3,0,prov_iipsupv_2yrs),
    #Vaccine management/handling and cold chain
    prov_vxmangsupv_2yrs=ifelse(q1106_02==2,1,NA), #
    prov_vxmangsupv_2yrs=ifelse(q1106_02==3,0,prov_vxmangsupv_2yrs))
```

Data reporting and monitoring of service delivery

Data reporting and monitoring of service delivery is recoded and the following columns were created from SARA 2018 data set:

- *prov_datasupv_2yrs*: Whether immunization service provider has received supervision in data reporting and monitoring of service delivery in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_surveilsupv_2yrs*: Whether immunization service provider has received supervision in disease surveillance and reporting in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_injsafesupv_2yrs*: Whether immunization service provider has received supervision in injection safety and waste management in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_redsupv_2yrs*: Whether immunization service provider has received supervision in Reaching Every District (RED) in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *prov_pcvrotasupv_2yrs*: Whether immunization service provider has received supervision in PCV or rotavirus vaccine in the last 2 years. Possible values are 0 (no) and 1 (yes).
- *open_hours_perday*: The number of hours per day that the facility is open to clients. Possible values are "4 hours or less", "5 to 8 hours", "9 to 16 hours", "17 to 23 hours", and "24 hours".
- *op_ip_avail*: Whether the facility provides outpatient services only or outpatient and inpatient services. Possible values are "Outpatient services (no routine inpatient)" and "Outpatient services (with routine inpatient)".
- *ip_beds*: The number of inpatient beds in the facility.

```
sara_2018 %>% mutate(
    #Data reporting and monitoring of service delivery
```

```
        prov_datasupv_2yrs=ifelse(q1106_03==2,1,NA), #
        prov_datasupv_2yrs=ifelse(q1106_03==3,0,prov_datasupv_2yrs),
        #Disease surveillance and reporting
        prov_surveilsupv_2yrs=ifelse(q1106_04==2,1,NA),
        prov_surveilsupv_2yrs=ifelse(q1106_04==3,0,prov_surveilsupv_2yrs),

        #Injection safety and waste management
        prov_injsafesupv_2yrs=ifelse(q1106_05==2,1,NA), #
        prov_injsafesupv_2yrs=ifelse(q1106_05==3,0,prov_injsafesupv_2yrs),
        #Reaching Every District (RED)
        prov_redsupv_2yrs=ifelse(q1106_06==2,1,NA), #
        prov_redsupv_2yrs=ifelse(q1106_06==3,0,prov_redsupv_2yrs),
        #PCV or rotavirus vaccine training
        prov_pcvrotasupv_2yrs=ifelse(q1106_07==2,1,NA), #
        prov_pcvrotasupv_2yrs=ifelse(q1106_07==3,0,prov_pcvrotasupv_2yrs),
        #basic client amenities (page 10)
        #Outputs and broader services
        open_hours_perday=ifelse(q417==1,"4 hours or less",""),
        open_hours_perday=ifelse(q417==2,"5 to 8 hours",open_hours_perday),
        open_hours_perday=ifelse(q417==3,"9 to 16 hours",open_hours_perday),
        open_hours_perday=ifelse(q417==4,"17 to 23
        hours",open_hours_perday),
        open_hours_perday=ifelse(q417==5,"24 hours",open_hours_perday),
        #outpatient only (page 3)
        op_ip_avail=ifelse(q010==1,
        "Outpatient services (no routine inpatient)", NA),
        op_ip_avail=ifelse(q010==2, "Outpatient services (with routine
        inpatient)", op_ip_avail),
        #inpatient beds (page 7)
        ip_beds=q301)
```

Finally, the following code selects the variables which are used for facility readiness indicator from the processed data using **dplyr** library.

```
sara_2018 %>%
        dplyr::select(
        #Facility info
        iso3, svy, fac_id, fac_name, admin1, admin1_name,
        admin2, admin2_name,
        urban, fac_type, fac_own, fac_weight,
        svy_year, svy_month, svy_day, svy_complete,
        operational, latnum,
        longnum, fac_comments1, fac_comments2,
        #RI services and supplies
        ri_services, ri_services_today, any_child_ri_guidelines,
        #Service frequency
        starts_with("fac_freq"), starts_with("q1103"),
        starts_with("out_freq"),
        starts_with("q1104"),
        #Vaccine-specific indicators
        vxinstock_penta, nostockout3mths_penta, neveravail_penta,
        vxinstock_bcg, nostockout3mths_bcg,
        neveravail_bcg,
```

```
vxinstock_mcv, nostockout3mths_mcv,
neveravail_mcv,
vxinstock_opv, nostockout3mths_opv, neveravail_opv,
vxinstock_pcv, nostockout3mths_pcv, neveravail_pcv,
vxinstock_rotac, nostockout3mths_rotac, neveravail_rotac,
vxinstock_mr, nostockout3mths_mr, neveravail_mr,
vxinstock_ipv, nostockout3mths_ipv, neveravail_ipv,
fac_storesvx, fridge_avail, coldchain_system,
coldchain_system2x_30days, fridge_temp, vx_carriers,
vx_carriers_icepacks, supply_blankimmuncards, supply_tallysheets,
supply_immunregister,
#IPC
ipc_runningwater, ipc_soap, ipc_sharpsbox, ipc_sharpsbox_gen,
ipc_latexgloves, ipc_disinfectant, ipc_dispsyringes,
ipc_dispsyringes_gen, ipc_pedalwastebin,
ipc_othwastebin, ipc_alcoholhandrub, ipc_masks, ipc_gowns,
ipc_eyeprotect, ipc_guidelines,
#Infrastructure
electricity, funct_generator, funct_solar, funct_computer,
funct_phone, healthdata_system, cleanliness_score8pts,
#Fees - not collected
clientfees_any, clientfees_healthcard, clientfees_vaccines,
clientfees_syringe,
#Staff and training
staff_total, staff_gp, staff_nurses, staff_chw, any_mgmtmtgs,
freq_mgmtmtgs, any_staffcommmtgs, freq_staffcommmtgs,
supv_lastvisit, supv_pharmacy, supv_staffing, supv_data,
#Immunization specific training
starts_with("prov"),
#Outpatient/inpatient services
open_hours_perday, op_ip_avail, ip_beds,
#Indicators to compare
q1101, q1102_01, q1102_02, q1102_03, q1103, q1103_a, q1104, q1104_a)
```

The pre-processed data set is saved with new file name (*sara_geolocated_ETH2018.csv*) under working directory. The following command is used to do so.

```
write.csv(sara_2018, file= 'sara_geolocated_ETH2018.csv',
row.names=FALSE)
```

We would like to remind the readers that data pre-processing for SARA 2016 can be adopted directly using the above R code since the questionnaires for both SARA 2016 and SARA 2018 are the same.

2.2 Pre-processing 2020 Cold Chain Equipment Inventory Survey

In this analysis, the CCEI data was used to construct indices for facility readiness and community access domain. The following sections describes the steps for pre-processing CCEI survey data for computing indicators of WHO readiness indicator

paramers from the survey for Ethiopian case. We started with loading the required libraries as we did in the previous sections.

The for loop you provided will iterate through the list of packages *tidyverse, RColorBrewer, ggplot2, reshape, knitr, data.table, geepack, lme4, xtable, readstata13, MASS, lubridate, varhandle, readxl, sf* and install and load each package that is not already installed. The *message()* function will print a message to the console indicating which package is being installed. The *require()* function is used to load a package into the R environment. The *character.only = TRUE* argument tells the *require()* function to only load the package if it is a character vector. The *quietly = TRUE* argument tells the *require()* function to not print a message to the console if the package is already installed.

```
#Load libraries
libs <- c('tidyverse', 'RColorBrewer', 'ggplot2', 'reshape', 'knitr',
'data.table', 'geepack',  'lme4', 'xtable', 'readstata13', #'mlmRev',
'MASS', 'lubridate', 'varhandle', 'readxl', 'sf')
for(l in libs){
  if(!require(l,character.only = TRUE, quietly = TRUE)){\
    message( sprintf('Did not have the required package << %s >>
    installed. Downloading now ... ',l))
    install.packages(l)
  }
  library(l, character.only = TRUE, quietly = TRUE)
}
#Clear workspace
rm(list=ls())
```

Then we loaded our raw data using *read.dta* function from *heaven* library as follows:

```
#Upload facility data in .csv format
ccei_facility <- haven::read_dta(eth_coldchain, "STATA Data
Set_CCEI_Final_2020.dta") %>% as.data.table()
```

Processing the data

We started with converting all variables to lowercase for easier coding (using *tolower()*) function and creating a separate data frame with processed data, keeping "original" data as it is. The following code is used to do this:

```
#Convert all variables to lowercase for easier coding
     names(ccei_facility) <- tolower(names(ccei_facility))
        #Create a separate data frame with processed data,
         keeping "original" data as it is
         ccei <- ccei_facility %>%
```

Next, we created the following new variables that are easier to understand and interact with onward code better using the below code:

- iso3: The ISO 3166-1 alpha-3 code for Ethiopia
- svy: The name of the survey, which is the Cold Chain Assessment 2020

- *fac_id*: A unique identifier for each facility, starting at 10000
- *fac_name*: The name of the facility.
- *fac_weight*: The survey weight for the facility. This is not reported in the CCEI dataset, so the value is NA.
- admin1: The name of the administrative region where the facility is located.
- *admin*1*_name*: The name of the administrative region where the facility is located, recoded using the CCEI codebook.
- admin2: The name of the zone where the facility is located.
- admin3: The name of the woreda where the facility is located.
- admin4: The name of the kebele where the facility is located.
- *fac_type*: The type of facility, recoded using the CCEI codebook.
- *svy_month*: The month in which the facility was surveyed.
- *svy_day*: The day in which the facility was surveyed.
- *svy_year*: The year in which the facility was surveyed.

```
ccei_facility %>%
    mutate(
    #Survey and facility characteristics
    iso3="ETH",
    svy="Cold Chain Assessment 2020",
    #Create facility ID starting at 10000
    fac_id=(row_number() + 9999),
    fac_name=str_to_title(q003),
    #fac_weight=NA, # no weights reported
    admin1=region,
    # admin 1 is numeric, recode using codebook here
    admin1_name=ifelse(admin1==1, "Tigray", NA),
    admin1_name=ifelse(admin1==2, "Afar", admin1_name),
    admin1_name=ifelse(admin1==3, "Amhara", admin1_name),
    admin1_name=ifelse(admin1==4, "Oromia", admin1_name),
    admin1_name=ifelse(admin1==5, "Somali", admin1_name),
    admin1_name=ifelse(admin1==6, "Benshangul Gumz", admin1_name),
    admin1_name=ifelse(admin1==7, "S.N.N.P", admin1_name),
    admin1_name=ifelse(admin1==8, "Gambella", admin1_name),
    admin1_name=ifelse(admin1==9, "Harari", admin1_name),
    admin1_name=ifelse(admin1==10, "Addis Ababa", admin1_name),
    admin1_name=ifelse(admin1==11, "Dire Dawa", admin1_name),
    # admin 2-4 are characters -- no need to recode as did with admin 1
    admin2=str_to_title(zone), admin3=str_to_title(woreda),
    admin4=str_to_title(kebele))
```

Facility Type

```
ccei_facility %>%
    mutate(
    # fac type is numeric, recode using codebook
```

```
    fac_type=q007,
    fac_type=ifelse(fac_type==1, "ESPA Central Store", fac_type),
    fac_type=ifelse(fac_type==2, "ESPA Hub", fac_type),
    fac_type=ifelse(fac_type==3, "Woreda Vaccine Store", fac_type),
    fac_type=ifelse(fac_type==4, "Primary Hospital", fac_type),
    fac_type=ifelse(fac_type==5, "General Hospital", fac_type),
    fac_type=ifelse(fac_type==6, "Referral Hospital", fac_type),
    fac_type=ifelse(fac_type==7, "Health Centre", fac_type),
    fac_type=ifelse(fac_type==8, "Health Post", fac_type),
    svy_month=qmonth,
    svy_day=qday,
    svy_year=qyear,
    # ownership -- only public included in this survey
fac_own="Government/public")
```

Fixing latitude and Longitude that Exceeds 1000

Then number of GPS values appear to be the same (e.g., 14 LAT, 38 LONG in Tigray) and some portion are just not possible since they're in a different GPS format). The following code is used to fix this issue:

```
ccei_facility %>% mutate(
#GPS
latnum_orig=glatitude,
latnum=ifelse(latnum_orig==0,NA,latnum_orig),
longnum_orig=glongitude,
longnum=ifelse(longnum_orig==0,NA,longnum_orig),

#Fixing lat-long that exceed 1000
latnum_alt=ifelse(latnum > 15 & latnum < 340, latnum/10,NA),
latnum_alt=ifelse(latnum >= 340 & latnum < 1600,
latnum/100,latnum_alt),
latnum_alt=ifelse(latnum >= 1600 & latnum < 16000,
latnum/1000,latnum_alt),
latnum_alt=ifelse(latnum >= 16000 & latnum < 160000,
latnum/10000,latnum_alt),
longnum_alt=ifelse(longnum > 300 & longnum < 1000, longnum/10,NA),
longnum_alt=ifelse(longnum > 999 & longnum < 10000,
longnum/100,longnum_alt),
longnum_alt=ifelse(longnum > 9999 & longnum < 100000,
longnum/1000,longnum_alt),
#Replace with fixed GPS
latnum=ifelse(!is.na(latnum_alt),latnum_alt, latnum),
longnum=ifelse(!is.na(longnum_alt),longnum_alt,longnum))
```

RI Service Availability

Regarding the RI services, from the CCEI survey questionnaire, we can see that

- Q12: 1= Vaccine storage only, 2= Immunization services without on site storage, 3=vaccine storage and immunization services, 4= no vaccine storage and immunization services
- Q13: Types of vaccination services offered: Q13_a=Static, Q13_b=Outreach, Q13_c=Mobile

- Q14: Frequency of facility based services: 1=Daily, 2=Weekly, 3= Every two weeks, 4=Monthly, 5=Other
- specify Q14O

The above questions are recorded using *metate()* function as follows:

```
ccei_facility %>% mutate(
vxavail_fac_any=ifelse(q13_a==1,1,NA),
vxavail_fac_any=ifelse(q13_a==0,0,vxavail_fac_any),
vxavail_out_any=ifelse(q13_b==1,1,NA),
vxavail_out_any=ifelse(q13_b==0,0,vxavail_out_any),
vxavail_mobile_any=ifelse(q13_c==1,1,NA),
vxavail_mobile_any=ifelse(q13_c==0,0,vxavail_mobile_any),
fac_storesvx=ifelse(q12== 2 | q12==4,0,NA),
fac_storesvx=ifelse(q12==1 | q12==3,1,fac_storesvx))
```

The following code is used for counting *yes* for facility based or outreach based or mobile based

```
ccei_facility %>% mutate(
ri_services=ifelse(vxavail_fac_any==1 | vxavail_out_any==1
| vxavail_mobile_any==1,1,NA),
ri_services=ifelse(vxavail_fac_any==0 & vxavail_out_any==0
& vxavail_mobile_any==0,0,ri_services),
#Response where no immunization services OR storage
#these have 0 for q13a and b
ri_services=ifelse(q12==4,0,ri_services))
```

At this point we have columns needed for community access. We then computed columns needed for readiness indicator.

RI Service Frequency

CCEI 2020 does not ask about frequency of vaccine-specific RI services. Thus, we re-coded any facility based RI services. Possible responses for open ended question for q14 about frequency of facility based services are also cleaned up as follows:

```
ccei <- ccei %>%
  mutate(
    q14o=str_trim(as.character(q14o)) %>% tolower(),
    # daily service provision q14 = 1 or various responses in other
    fac_freq_daily=ifelse(q14 == 1, 1, NA),
    fac_freq_daily=ifelse(q14o %in% c("5 times per day",
    "5 times per week", "everyday exept bcg/m",
    "monthly and daily", "daily and monthly",
    "two times per aday", "2 times per aday"), 1, fac_freq_daily),

    # weekly service provision q14 = 2 or various responses in other
    fac_freq_weekly=ifelse(q14 == 2, 1, NA),
    fac_freq_weekly=ifelse(q14o %in% c("twice/week",
    "2* per week", "tewice a week",  "3 days in a week",
    "twice a week", "times per month4", "four times per month",
```

```
    "three times per week", "ten times per month",
    "2 times per week", "3 times per week", "two times per
    week", "5times per month", "6 times per month",
    "twice weekly", "2days per week", "2times/week",
    "2times per week", "2x/week", "two time per week",
    "two in week mondat a", "two in week", "monday and thursday",
    "mondat and thursday", "per week 2 days",
    "2 days in week", "twice per week", "2 dayes in week",
    "2 per week", "biweekly", "bi weekly", "three times weekly",
    "twice aweek",  "two times aweeks", "two times per weeks",
    "2 days per week", "3 days per week", "2daays per week",
    "3 times/week", "two times/week", "three times/week",
    "two days in a week",  "2 days in a week",
    "twice in a week", "4 days per week",  "twice in aweek",
    "3 times a week", "two days per week", "two days in a weak",
    "2 days/wk", "2 dayswith in eek", "2x per week",
    "mo thu", "tuesday wedensday",  "two  times per week",
    "two times/ wk", "twice per weak",  "three times in aweek",
    "two days in weeks", "two dayis in week",
    "one day in week", "one day in a week",
    "10day per month", "2 days per weak", "10 days per month",
    "10 days per week", "three times a week",
    "2 perweek",  "two times weekly", "two times per wk",
    "two times in week", "2 times a week",
    "2 times per a week", "times perweek", "3 times per aweek",
    "3 times per a week", "4 times per a week", "2 times aweek",
    "2 times per aweek", "2 times per aweeek",
    "3 times per awwk", "4 times per aweek", "2 days every week",
    "two dayes per weeak",  "three day per a week",
    "2x a week", "two times a week",  "twice week", "once
    aweek", "once in aweek", "twicwe in aweek",
    "twice in a weeek", "3days per week", "1 day per week",
    "2days weekly", "twicw weekly", "twies perweek",
    "two days perweek", "weeklysometimes ever",
    "2 days aweak", "two days a weak",  "2 days a weak",
    "two days aweak", "3 days a weak", "2days a weak",
    "2 dats in aweek", "two days a week",
    "two days withn a wee",  "2 day per week",
    "wikly 3day", "5 times in month",  "7 days in a month",
    "6 times in a month",  "5days per month",
    "2 days in aweek"),  1, fac_freq_weekly))
```

We then code monthly or every two weeks as monthly service provision. Monthly service provision q14 = 4, every two weeks q14 = 3 or various reponses in q14o are recoded as follows:

```
ccei <- ccei %>%
  mutate(
    fac_freq_monthly=ifelse(q14 == 3 | q14 == 4, 1, NA),
    fac_freq_monthly=ifelse(q14o %in% c("3/month",
    "twice monthly",  "three times per mont",
    "third times a month", "3 times per month",
     "2 times per month", "two times per month",
```

```
    "3 times amonth", "2 days in month", "three times/mont",
    "thee times per month", "2times per month",
    "every three week", "two days per month",
   "three days per month", "three times amonth",
   "3 times in a month", "three times a month", "monthly",
   "every 3 week", "3times per month", "3times permonth",
   "every 17 days", "3 days per month",  "3times amonth",
   "2 times amonth", "3 times a month",
   "2 times per a month", "2 times a month",
   "3 times per a month",  "2 times per amonth",
   "3 time per amonth", "3 times per amonth",
    "2 timespeer amonth", "twice a month", "tice a month",
    "three x per a month",  "3x times a month",
    "3x a month", "2x a month", "three times in a month",
    "twice in amonth", "twice in month", "3timesin amonth",
    "every 3 weeks", "2 days a month",
    "three times in month"),  1, fac_freq_monthly))
```

The quarterly service provision for various responses in q14o are recoded as follows:

```
ccei <- ccei %>%
  mutate(
  fac_freq_quarterly=ifelse(q14o  %in%  c("once in two month",
  "every 45 days", "every two month", "in two or three mont",
  "two month", "in two or three mont", "every two month"), 1, NA))
```

Any facility-based services provision is then re-coded as follows:

```
ccei <- ccei %>%
  mutate(
  # any facility-based services provided?
  fac_freq_any = ifelse(fac_freq_daily==1 | fac_freq_weekly==1
  | ac_freq_monthly==1 | fac_freq_quarterly == 1,1,NA),
  # recode every six month as facility service provision
 #(less than #quarterly)
 fac_freq_any = ifelse(ri_services == 1 & q14o %in%
 c("every 6 month (twic)", "every 6 month"), 1,
 fac_freq_any), fac_freq_any=ifelse(ri_services==0 | q14o %in%
 c("0", "no service", "not given", "out reach only",
 "no vaccination", "no serice"),0,fac_freq_any))
```

Outreach-based services

The 2020 CCEI survey does not ask about frequency of outreach but can extract information on the number of outreach sites which is recoded as follows:

```
ccei <- ccei %>%
  mutate(
    num_outreach_site = as.numeric(q17),
    num_mobile_site = as.numeric(q18),
    out_freq_any = ifelse(ri_services == 1 & (vxavail_out_any
    == 1 | vxavail_mobile_any == 1), 1, NA),
    out_freq_any = ifelse(ri_services == 1 & (vxavail_out_any
    == 0 & vxavail_mobile_any == 0), 0, out_freq_any),
```

```
    out_freq_any = ifelse(ri_services == 0, 0, out_freq_any))
```

Vaccine stocks

The 2020 CCEI survey does not ask about data on stocks but do have info on stockouts in last 3 months which is recoded as follows:

```
ccei <- ccei %>%
  mutate(
   #Penta
   #vxinstock_penta = NA,
   nostockout3mths_penta = ifelse(q21s_c==1, 0, NA),
   nostockout3mths_penta = ifelse(q21s_c==0, 1, nostockout3mths_penta),
   #neveravail_penta = NA,
   #BCG
    #vxinstock_bcg = NA,
    nostockout3mths_bcg = ifelse(q21s_a==1, 0, NA),
    nostockout3mths_bcg = ifelse(q21s_a==0, 1, nostockout3mths_bcg),
    #neveravail_bcg = NA,
    #MCV
    #vxinstock_mcv = NA,
    nostockout3mths_mcv = ifelse(q21s_f==1, 0, NA),
    nostockout3mths_mcv = ifelse(q21s_f==0, 1, nostockout3mths_mcv),
    #neveravail_mcv = NA,
    #Polio - OPV
    #vxinstock_opv = NA,
    nostockout3mths_opv = ifelse(q21s_b==1, 0, NA),
    nostockout3mths_opv = ifelse(q21s_b==0, 1, nostockout3mths_opv),
    #neveravail_opv = NA,
    #Polio - IPV
    #vxinstock_ipv = NA,
    nostockout3mths_ipv = ifelse(q21s_e==1, 0, NA),
    nostockout3mths_ipv = ifelse(q21s_e==0, 1, nostockout3mths_ipv),
    #neveravail_ipv = NA,
    #PCV
    #vxinstock_pcv = NA,
    nostockout3mths_pcv = ifelse(q21s_d==1, 0, NA),
    nostockout3mths_pcv = ifelse(q21s_d==0, 1, nostockout3mths_pcv),
    #neveravail_pcv = NA,
    #RotaC
    #vxinstock_rotac = NA,
    nostockout3mths_rotac= ifelse(q21s_g==1, 0, NA),
    nostockout3mths_rotac = ifelse(q21s_g==0, 1, nostockout3mths_rotac),
    #neveravail_rotac = NA,
    ##TT
    #vxinstock_tt = NA,
    nostockout3mths_tt= ifelse(q21s_h==1, 0, NA),
    nostockout3mths_tt = ifelse(q21s_h==0, 1, nostockout3mths_tt)
    #neveravail_tt = NA)
```

Storage and cold chain

Regarding functional fridge (number of functional fridges that the facility owns), if reported 0 functional fridges or all fridges they report status for are non functional or uninstalled or obsolete, then we code it as 0.

```
ccei <- data.table(ccei)
fridge_cols <- c(paste0("h_0", 1:9), paste0("h_", 10:20)
ccei[, (fridge_cols) := lapply(.SD, function(x) x == "1"),
.SDcols = fridge_cols]
ccei[, fridge_avail := ifelse(q31b == 0 | rowSums(.SD) ==
0,0,NA), .SDcols= fridge_cols])
```

If report other than 0, functional fridges and in individual fridge columns report at least 1 functional.

```
ccei[, fridge_avail := ifelse(q31b>0 & rowSums(.SD)
>0,1,fridge_avail), .SDcols = fridge_cols]#Available AND functional
```

Temperature monitoring system in use in at least 1 fridge.

```
#Matches SARA where don't ask about number of fridges
coldchain_cols <- c(paste0("j_0", 1:9), paste0("j_", 10:20))
ccei[, (coldchain_cols) := lapply(.SD, function(x) x != ""),
.SDcols = coldchain_cols]
ccei[, coldchain_system := ifelse(rowSums(.SD) == 0, 0, NA),
.SDcols = coldchain_cols]
ccei[, coldchain_system := ifelse(rowSums(.SD) > 0, 1,
coldchain_system), .SDcols = coldchain_cols]
```

Facility Infrastructure

```
ccei <- ccei %>% mutate(
    # Electricity
    electricity = ifelse(q24 == 1 | q24 == 2, "Connected to
    grid, more than 16 hours per day", ""),
    electricity = ifelse(q24 == 3 , "Connected to grid,
    4 to 8 hours per day", electricity),
    electricity = ifelse(q24 == 4 , "Connected to grid,
    less than 4 hours per day", electricity),
    electricity = ifelse(q24 == 5 , "Not connected to  grid", electricity),
    #Functional generator available
    funct_generator = ifelse(q29 == 1, 1, NA),
    funct_generator = ifelse(q29 == 0, 0, funct_generator),
     #Functional solar power available
    funct_solar = ifelse(q28 == 1 & q28a == 1, 1, NA),
    funct_solar = ifelse(q28 == 0, 0, funct_solar),
    funct_solar = ifelse(q28 == 1 & (q28a == 2 | q28a == 3),
    0, funct_solar))
```

Adding admin2 (zone) names based on GPS

```
    #Adding admin 2 names based on GPS
    shp <- read_rds(file.path(work_root,
    "lbd_standard_admin_2.shp"))
```

```
shp.admin2 <- shp %>% subset(ADM0_NAME == "Ethiopia") #%>%
as(Class="Spatial")
temp <- st_as_sf(ccei[!(is.na(latnum) | is.na(longnum))],
coords= c("longnum", "latnum"), crs = 4326)
temp <- st_join(temp, shp.admin2, join = st_within)
temp <- cbind(temp, st_coordinates(temp)) %>% as.data.table
setnames(temp,
old = c("X", "Y"), new = c("longnum", "latnum"))
ccei <- rbind(ccei[is.na(longnum) | is.na(latnum)], temp[,
-c("geometry"), with = F], fill = TRUE)
```

Then, the processed variables are selected and ordered using *select()* function from *dplyr* library as follows:

```
#Selecting processed variables and ordering them
ccei <- ccei %>% dplyr::select(
 #Facility info
 iso3, svy, svy_year, svy_month, svy_day, fac_id,
 fac_name, admin1, admin1_name, admin2, admin3, admin4,
 fac_type, latnum, longnum, latnum_orig, longnum_orig,
 starts_with("vxavail"), fac_storesvx,
#geography as assinged by shapefile
 ADM1_NAME, ADM2_NAME,
 #RI services
 ri_services,
 #Service frequency
 starts_with("fac_freq"), starts_with("q13"), q14, q14o,
 starts_with("out_freq"), starts_with("num_"),
 #Vaccine-specific indicators
 nostockout3mths_penta,
 nostockout3mths_bcg,
 nostockout3mths_mcv,
 nostockout3mths_opv,
 nostockout3mths_pcv,
 nostockout3mths_rotac,
 nostockout3mths_tt,
 nostockout3mths_ipv,
 fridge_avail, coldchain_system,
 #Infrastructure
 electricity, funct_generator, funct_solar, funct_phone)
```

We then save the geolocated and initially extracted, processed dataset on our working directory using the code below:

```
write.csv(ccei, file= "ccei_geolocated_readiness.csv",
row.names=FALSE)
```

Chapter 3

Missing Value Managment

As part of computing the indexes for the three domains, we conducted missigness checking for SARA survey. The result is presented as follows:

3.1 Multiple Imputation for SARA 2018

```
 #Load libraries
libs <- c('tidyverse', 'RColorBrewer', 'ggplot2', 'reshape2', 'viridis',
'data.table', 'GGally', 'expss', 'fedmatch',
'measurements',  'fuzzyjoin', 'geepack', 'mice', 'mlmRev', 'lme4')

for(l in libs){
  if(!require(l,character.only = TRUE, quietly = TRUE)){
    message(sprintf('Did not have the required package << %s >>
    installed. Downloading now ... ',l))
    install.packages(l)
  }
  library(l, character.only = TRUE, quietly = TRUE)
}

#Clear workspace
rm(list=ls())
```

After loading the required packages and cleaning the working environment, we can now call the necessary color schemes and load the required data sets. (Note that, now we are going to use the cleaned version of SARA data set which is pre-processed following steps presented under the previous chapter).

```
#Setting up tans/yellows
brewer <- brewer.pal(11, "PuOr")
dktan <- brewer[1]
medtan <- brewer[2]
lttan <- brewer[3]
golden <- brewer[4]
ltgolden <- brewer[5]
whitish <- brewer[6]
```

```
##Setting up blues
brewer <- brewer.pal(9, "Blues")
blue <- brewer[6]
medblue <- brewer[7]
dkblue <- brewer[8]
vdkblue <- brewer[9]

#Exemplar color palette
dkblue <- "#00416b"
teal <- "#0098a7"
orange <- "#ff6400"
purple <- "#472677"
lightGrey <- 'grey90'

#Other colors
magenta <- "#c92196"
eghblue <- "0a6bd1"
eghgreen <- "87d45c"
eghyellow <- "ffb634"

#Teal shades/tints
dkteal <- "#005159"
medteal <- "#4cb6c1"
ltteal <- "#99d5db"

#Orange shades/tints
dkorange <- "#cc5000"
medorange <- "#ff8332"
ltorange <- "#ffa266"
black <- "black"
```

After setting the working environment, we then load data (located in Desktop with file name of sara_geolocated_ETH2018.csv) as follows:

```
sara2018 <- read.csv("Desktop/sara_geolocated_ETH2018.csv")
```

Now, we have loaded the required packages, cleaned the working environment and loaded our data set to our working environment. We proceed to computing the missingness of key indicators pertaining to readiness.

The following code is used to compute 'true' missingness for key indicators pertaining to readiness. The first step is to account for skip patterns (also knows as what should be recoded as zeros versus true missingness). We first subset to facilities with completed surveys and that are operational, and where RI status is known.

```
sara2018_processed <- sara2018 %>%
  filter(svy_complete==1 & operational==1 &
    !is.na(ri_services)) %>%
    mutate(
    #RI services today: if no RI services at all, none today
     ri_services_today=ifelse(ri_services==0,0, ri_services_today),
```

```
        #Service frequency
        #Facility based services
        fac_freq_any=ifelse(ri_services==0,0, fac_freq_any),
        fac_freq_quarterly=ifelse(ri_services==0,0,
        fac_freq_quarterly),
        fac_freq_quarterly=ifelse(!is.na(fac_freq_any) &
        is.na(fac_freq_quarterly),0, fac_freq_quarterly),
        fac_freq_monthly=ifelse(ri_services==0,0, fac_freq_monthly),
        fac_freq_monthly=ifelse(!is.na(fac_freq_any) &
        is.na(fac_freq_monthly),0, fac_freq_monthly),
        fac_freq_weekly=ifelse(ri_services==0,0, fac_freq_weekly),
        fac_freq_weekly=ifelse(!is.na(fac_freq_any) &
        is.na(fac_freq_weekly),0, fac_freq_weekly),
        fac_freq_daily=ifelse(ri_services==0,0, fac_freq_daily),
        fac_freq_daily=ifelse(!is.na(fac_freq_any) &
        is.na(fac_freq_daily),0, fac_freq_daily),
        #Outreach services
        out_freq_any=ifelse(ri_services==0,0, out_freq_any),
        out_freq_quarterly=ifelse(ri_services==0,0,
        out_freq_quarterly),
        out_freq_quarterly=ifelse(!is.na(out_freq_any) &
        is.na(out_freq_quarterly),0,out_freq_quarterly),
        out_freq_monthly=ifelse(ri_services==0,0,
        out_freq_monthly), out_freq_monthly=ifelse(!is.na(out_freq_any) &
        is.na(out_freq_monthly),0,out_freq_monthly),
        out_freq_weekly=ifelse(ri_services==0,0,
        out_freq_weekl)
        ,out_freq_weekly=ifelse(!is.na(out_freq_any) &
        is.na(out_freq_weekly),0,out_freq_weekly),
        out_freq_daily=ifelse(ri_services==0,0,out_freq_daily),
        out_freq_daily=ifelse(!is.na(out_freq_any) &
         is.na(out_freq_daily),0,out_freq_daily))
```

The following code is used to check for cold chain related monitoring system (such as availability of fridge)

```
sara2018_processed <- sara2018 %>%
    mutate(
    #Storage + cold chain
    fridge_avail=ifelse(ri_services==0,0,fridge_avail),
    coldchain_system=ifelse(ri_services==0,0,coldchain_system),
        #Accounting for not asking if there's a cold chain
        #monitoring system if there is no fridge
        coldchain_system=ifelse(fridge_avail==0 &
        is.na(coldchain_system),0,coldchain_system),
        coldchain_system2x_30days=ifelse(ri_services==0,0,
        coldchain_system),
        coldchain_system2x_30days=ifelse(fridge_avail==0 & is.na(coldchain_
        #Converting fridge temp to binary variable
        #fridge_temp=as.character(fridge_temp),
        fridge_temp2to8=ifelse(fridge_temp==
        "Between 2 and 8 degrees C",1,NA),
        fridge_temp2to8=ifelse(fridge_temp=="Out of
```

```
        range",0,fridge_temp2to8),
        #Reflects that there's no fridge and no cold chain
         #monitoring system -- if fridge_avail==0 &
        #cold_chain_system !=0, then #it's likely that the
        #fridge is not functional but exists
        fridge_temp2to8=ifelse(fridge_avail==0 & coldchain_system==0 &
        is.na(fridge_temp),0,fridge_temp2to8))
```

Infrastructure

```
sara2018_processed <- sara2018 %>%
    mutate(electricity=as.character(electricity),
    electricity_gridalways=ifelse(electricity !="Connected
    to grid, always available",0,NA),
    electricity_gridalways=ifelse(electricity=="Connected
    to grid, always available",1,electricity_gridalways))
```

Vaccination supplies

```
sara2018_processed <- sara2018 %>%
    mutate(
        #Vaccination supplies
        vx_carriers=ifelse(ri_services==0,0,vx_carriers),
        vx_carriers_icepacks=ifelse(ri_services==0,0,
        vx_carriers_icepacks),
        supply_blankimmuncards=ifelse(ri_services==0,0,
        supply_blankimmuncards),
        supply_tallysheets=ifelse(ri_services==0,0,supply_tallysheets),
        supply_immunregister=ifelse(ri_services==0,0,
        supply_immunregister))
```

RI Guidelines

```
sara2018_processed <- sara2018 %>%
    mutate(
        #Guidelines - if no RI services, replace with 0
        any_child_ri_guidelines=ifelse(ri_services==0,0,
        any_child_ri_guidelines))
```

Infection prevention and control (IPC)

```
sara2018_processed <- sara2018 %>%
    mutate(
    #IPC
    ipc_sharpsbox=ifelse(ri_services==0,0,ipc_sharpsbox),
    ipc_dispsyringes=ifelse(ri_services==0,0,ipc_dispsyringes))
```

EPI Training

```
sara2018_processed <- sara2018 %>%
    mutate(
        #Epi training
        prov_epitrain_2yrs=ifelse(ri_services==0,0,
        prov_epitrain_2yrs),
        prov_iiptrain_2yrs=ifelse(ri_services==0,0,
        prov_iiptrain_2yrs), prov_vxmangtrain_2yrs=ifelse(ri_services==0,0,
        ifelse(ri_services==0,0, prov_datatrain_2yrs),
        prov_surveiltrain_2yrs=ifelse(ri_services==0,0,
        prov_surveiltrain_2yrs), prov_injsafetrain_2yrs=ifelse(ri_services=
        prov_injsafetrain_2yrs),
        prov_redtrain_2yrs=ifelse(ri_services==0,0,
        prov_redtrain_2yrs),
        prov_pcvrotatrain_2yrs=ifelse(ri_services==0,0,
        prov_pcvrotatrain_2yrs))
```

Vaccine stocks

The available vaccines for SARA 2018 are pentavalent, MCV, OPV, BCG, PCV, Rotavirus. Note that the skip logic for SARA 2018 does not ask about vaccine stocks if no RI services today and if the fridge is not functional. Thus it is recoded as zero.

```
sara2018_processed <- sara2018 %>%
mutate(
#Pentavalent
vxinstock_penta=ifelse(ri_services==0,0,vxinstock_penta),
vxinstock_penta=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, vxinstock_penta),
nostockout3mths_penta=ifelse(ri_services==0,0,
nostockout3mths_penta),
nostockout3mths_penta=ifelse(ri_services_today==0
& fridge_avail !=1,0, nostockout3mths_penta),
neveravail_penta= ifelse(ri_services==0,0,neveravail_penta),
neveravail_penta=ifelse(vxinstock_penta== 1
|nostockout3mths_penta==1,0,neveravail_penta),
neveravail_penta=ifelse(ri_services_today==0 &
fridge_avail !=1, 0, neveravail_penta),
#MCV
vxinstock_mcv=ifelse(ri_services==0,0,vxinstock_mcv),
vxinstock_mcv=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, vxinstock_mcv),
nostockout3mths_mcv=ifelse(ri_services==0,0,
nostockout3mths_mcv),
nostockout3mths_mcv=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, nostockout3mths_mcv),
neveravail_mcv=ifelse(ri_services==0,0,neveravail_mcv),
neveravail_mcv=ifelse(vxinstock_mcv==1 |
nostockout3mths_mcv==1,0,neveravail_mcv),
neveravail_mcv=ifelse(ri_services_today==0 &
fridge_avail!=1,0, neveravail_mcv),
#BCG
```

```
vxinstock_bcg=ifelse(ri_services==0,0,vxinstock_bcg),
vxinstock_bcg=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, vxinstock_bcg),
nostockout3mths_bcg=ifelse(ri_services==0,0,
nostockout3mths_bcg),
nostockout3mths_bcg=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, nostockout3mths_bcg),
neveravail_bcg=ifelse(ri_services==0,0,neveravail_bcg),
neveravail_bcg=ifelse(vxinstock_bcg==1 |
nostockout3mths_bcg==1,0,neveravail_bcg),
neveravail_bcg=ifelse(ri_services_today==0 &
fridge_avail !=1,0, neveravail_bcg),
#OPV
vxinstock_opv=ifelse(ri_services==0,0,vxinstock_opv),
vxinstock_opv=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, vxinstock_opv),
nostockout3mths_opv=ifelse(ri_services==0,0,
nostockout3mths_opv),
nostockout3mths_opv=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, nostockout3mths_opv),
neveravail_opv=ifelse(ri_services==0,0,neveravail_opv),
neveravail_opv=ifelse(vxinstock_opv==1 |
nostockout3mths_opv==1,0,neveravail_opv),
neveravail_opv=ifelse(ri_services_today==0 &
fridge_avail !=1, 0, neveravail_opv),
#PCV
vxinstock_pcv=ifelse(ri_services==0,0,vxinstock_pcv),
vxinstock_pcv=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, vxinstock_pcv),nostockout3mths_pcv=
ifelse(ri_services==0,0,nostockout3mths_pcv),
nostockout3mths_pcv=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, nostockout3mths_pcv),
neveravail_pcv=ifelse(ri_services==0,0,neveravail_pcv),
neveravail_pcv=ifelse(vxinstock_pcv==1 |
nostockout3mths_pcv==1,0,neveravail_pcv),
neveravail_pcv=ifelse(ri_services_today==0 &
fridge_avail !=1, 0, neveravail_pcv),
#Rotavirus
vxinstock_rotac=ifelse(ri_services==0,0,vxinstock_rotac),
vxinstock_rotac=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, vxinstock_rotac),
nostockout3mths_rotac=ifelse(ri_services==0,0,
nostockout3mths_rotac),
nostockout3mths_rotac=ifelse(ri_services_today !=1 &
fridge_avail !=1,0, nostockout3mths_rotac),
neveravail_rotac=ifelse(ri_services==0,0,neveravail_rotac),
neveravail_rotac=ifelse(vxinstock_rotac==1 |
ostockout3mths_rotac==1,0,neveravail_rotac),
neveravail_rotac=ifelse(ri_services_today==0 &
fridge_avail !=1, 0, neveravail_rotac))
```

Data Preparation for multiple imputation

```
sara2018_formi <- sara2018_processed %>%
    mutate(fac_id=as.character(fac_id)) %>%
    dplyr::select(
    fac_id,
    #Vaccination indicators for potential prediction
    #RI services and supplies
    ri_services, ri_services_today,
    any_child_ri_guidelines,
    #Service frequency
    starts_with("fac_freq"), starts_with("out_freq"),
    #Vaccine-specific indicators
    vxinstock_penta, nostockout3mths_penta,
    neveravail_penta,
    vxinstock_bcg, nostockout3mths_bcg, neveravail_bcg,
    vxinstock_mcv, nostockout3mths_mcv, neveravail_mcv,
    vxinstock_opv, nostockout3mths_opv, neveravail_opv,
    vxinstock_pcv, nostockout3mths_pcv, neveravail_pcv,
    vxinstock_rotac, nostockout3mths_rotac,
    neveravail_rotac,fridge_avail, coldchain_system,
    coldchain_system2x_30days, fridge_temp2to8,
    vx_carriers, vx_carriers_icepacks,
    supply_blankimmuncards, supply_tallysheets,
    supply_immunregister,
    #IPC, infrastructure, staff and training
     ipc_sharpsbox, ipc_dispsyringes,
    electricity_gridalways, staff_chw, staff_nurses,
    prov_epitrain_2yrs,
    #Additional predictors
    funct_computer, funct_phone,
    ipc_runningwater, ipc_soap, ipc_disinfectant,
    ipc_pedalwastebin, ipc_alcoholhandrub,
    ipc_guidelines, ip_beds)
```

Now, our data set is ready for imputation. We can save the processed data to our working directory and proceed to imputation part.

```
write.csv(sara2018_formi, file=sara2018_formi.csv)
```

Now, we can proceed to imputation part. The following steps were caried out to conduct imputation on SARA 2018 data set. We start with making predictor matrix excluding fac_id and for variables with missingness.

```
#All variables
    allvars_sara2018 <- names(sara2018_formi)
    #Matrix of all variables
    matrix_sara2018 <- matrix(0, ncol = length(allvars_sara2018 ),
    nrow = length(allvars_sara2018))
    rownames(matrix_sara2018) <- allvars_sara2018
    colnames(matrix_sara2018) <- allvars_sara2018
    #Variables with any missingness in original dataset
    missVars_sara2018 <- names(sara2018_formi)
```

```
    [colSums(is.na(sara2018_formi)) > 0]
    #Variables without any missingness to serve as predictors (imputers)
    predVars <- names(sara2018_formi)
    [colSums(is.na(sara2018_formi))==0]
    #Remove "fac_id" from predictor variables
    vars_impute <- predVars[!predVars %in% c("fac_id")]
    #Set up predictor (imputer) matrix
    imputerVars <- intersect(unique(vars_impute),
    allvars_sara2018)
    imputerMatrix <- matrix_sara2018
    imputerMatrix[,imputerVars] <- 1
    #Variables for imputation
    imputedVars_sara2018 <-intersect(unique(c(allvars_sara2018)),
    missVars_sara2018)imputedVars_sara2018
    imputedMatrix_sara2018 <- matrix_sara2018
    imputedMatrix_sara2018[imputedVars_sara2018,] <- 1
    predictorMatrix_sara2018 <- imputerMatrix
   imputedMatrix_sara2018 diag(predictorMatrix_sara2018)<- 0
```

In this study, MICE algorithm [3] was adopted to compute missingness of key indicators of readiness domain from SARA 2018 and SARA 2018 survey. The following sections present the steps followed during implementation of MICE algorithm in R software.

3.2 MICE for SARA 2018

```
#setting up methods
imp0 <- mice(data=sara2018_formi,maxit=0)
meth = imp0$method
sara2018_mice <- mice(data=sara2018_formi, m=10,
maxit=100,  method=meth, predictorMatrix=predictorMatrix_sara2018, print=
sara2018_mice_long <- mice::complete(sara2018_mice,
action="long", include=TRUE)
```

The following code is used for computing and comparing indices for original data versus imputation.

```
    ###Original data inputs
sara2018_processed[is.na(sara2018_processed)] <-0
sara2018index_org <- sara2018_processed %>%
    mutate(aux_coldchain=ifelse(fridge_avail==1 &
    coldchain_system==1 & coldchain_system2x_30days==1 &
    fridge_temp2to8==1,1,0),
    aux_power_coldchain=ifelse(fridge_avail==1 &
    electricity_gridalways==1,1,0),
    vxcarrier_withicepacks=ifelse(vx_carriers==1 &
     vx_carriers_icepacks==1,1,0),
    supply_tallyregister=ifelse(supply_tallysheets==1 |
    supply_immunregister==1,1,0),
    staff_epi=prov_epitrain_2yrs,
    fac_index=(any_child_ri_guidelines + staff_epi +
```

```
    vxcarrier_withicepacks + ipc_dispsyringes +
    ipc_sharpsbox + supply_blankimmuncards +
    supply_tallyregister + fridge_avail + coldchain_system
    + fridge_temp2to8 + aux_coldchain +
    aux_power_coldchain)/12,
    fac_index=ifelse(ri_services==0 |
    fac_freq_any==0,NA,fac_index),
    ## Replacing with NA if no RI or no facility-based #services
    source="SARA 2018",
    type="Original (missing as 0s)")
    #Imputed data
    sara2018index_imputed <- sara2018_mice_long %>%
    filter(.imp !=0) %>% ##Filter out non-imputed data
    mutate(aux_coldchain=ifelse(fridge_avail==1 &
    coldchain_system==1 & coldchain_system2x_30days==1 &
    fridge_temp2to8==1,1,0), aux_power_coldchain=ifelse(fridge_avail==1 &
    electricity_gridalways==1,1,0),
    vxcarrier_withicepacks=ifelse(vx_carriers==1 &
    vx_carriers_icepacks==1,1,0),
    supply_tallyregister=ifelse(supply_tallysheets==1 |
    supply_immunregister==1,1,0),
    staff_epi=prov_epitrain_2yrs,
    fac_index=(any_child_ri_guidelines + staff_epi +
    vxcarrier_withicepacks + ipc_dispsyringes + ipc_sharpsbox +
    supply_blankimmuncards + supply_tallyregister +
    fridge_avail + coldchain_system +
    fridge_temp2to8 + aux_coldchain +
    aux_power_coldchain)/12,
    fac_index=ifelse(ri_services==0 |
    fac_freq_any==0,NA,fac_index),
    ## Replacing with NA if no RI or no facility-based services
    source="SARA 2018", type="Imputed")
    #Taking the average across imputations
    sara2018index_mivalues <- sara2018index_imputed  %>%
   group_by(fac_id) %>%
   summarise(fac_index_imp=mean(fac_index),
   fac_index_sd=sd(fac_index))
   sara2018index_compare <- merge(sara2018index_org,
    sara2018index_mivalues, by="fac_id")
    # save the imputed data
write.csv(sara2018_mice_long,"sara_imputed_ETH2018.csv",row.names=FALSE)
```

The above pre-processed data is then used to plot scatter diagram showing imputed index scores (average of 100 iterations of 10 imputations each) against the original index scores (missingness hard coded as 0s). The code and the results are presented below

```
(ggplot(sara2018index_compare, aes(x=fac_index*100,
y=fac_index_imp*100)) +
geom_abline(a=0, b=1) +
geom_jitter(height=0.7, width=0.7, size=6, alpha=0.8,
aes(colour=fac_index_sd*100)) +
scale_colour_viridis(option="B", end=0.8, name="SD for MI") +
```

```
theme_bw() +
ggtitle("Comparing facility index scores (12-indicator scores):
SARA 2018") +
ylab("Imputed index scores (average of 100 iterations of 10
imputations each)") +
xlab("Original index scores (missingness hardcoded as 0s)") +
theme(
  legend.position="right",
  legend.title=element_text(size=11),
  axis.title.y=element_text(size=13),
  axis.title.x=element_text(size=13),
  axis.text.x=element_text(size=11),
  axis.text.y=element_text(size=11),
  panel.grid.major=element_blank(),
  panel.grid.minor=element_blank()))
```

We can save the above plot in any format using 'ggsave' function.

```
ggsave("eth_sara2018_facindex_10mi100it.pdf", width=15,
  height=8)
```

The result is displayed under the following scatter plot diagram. The diagram shows imputed index scores (average of 100 iterations of 10 imputations each) on the y axis and original index scores (missingness coded as 0s) on the x-axis.

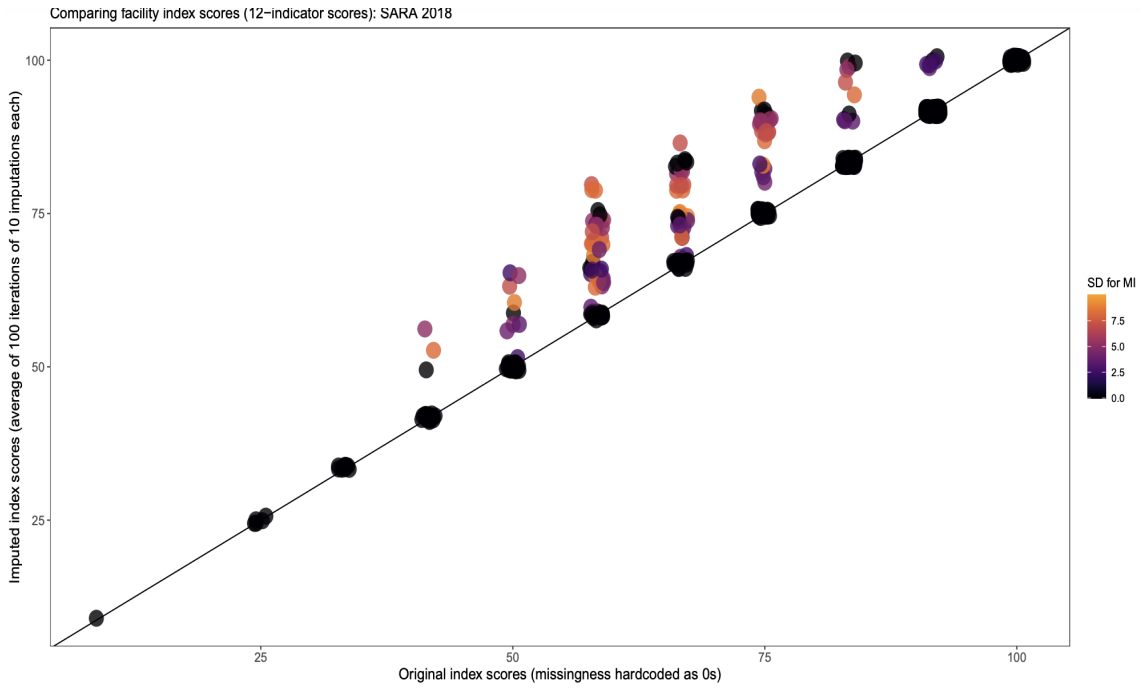

Figure 3.1: Scatter plot: Imputed index scores vs original index scores from SARA 2018 data

Chapter 4

Application of Machine Learning Model

As stated under the introduction section, the overall objective of this analysis is to quantify the relative contribution of the three drivers (intent to vaccine, community access and facility readiness) of vaccine coverage in Ethiopia. To achieve this objective, we compiled and analyzed data from different sources to produce harmonized geospatial estimates of the three drivers, link estimates of the three drivers to available household surveys, estimate the predicted relationship between the probability that a child is vaccinated and levels of the three drivers using machine learning methods, quantify the relationship between remaining variation and sociod-emographic factors. The following sections present the overall steps carried out in fitting machine ML model for quantifying the effects of the three drivers of vaccine and production of geospatial estimates of the drivers.

We started with loading the already installed packages and installing the missing packages which are required to run the ML model.

```
 #Load libraries
libs <- c(data.table, sf,tidyverse, pbapply, survey,
               mgcv, xgboost, pROC, pdp, SHAPforxgboost, lavaan, foreign,
               dplyr)

for(l in libs){
  if(!require(l,character.only = TRUE, quietly = TRUE)){
    message(sprintf('Did not have the required package <<
    %s >> installed. Downloading now ... ',l))
    install.packages(l)
  }
  library(l, character.only = TRUE, quietly = TRUE)
}
#Clear workspace
rm(list=ls())
```

We then set our working directory and load the source file using *source()* function in R (note that the source a file that we want to source must be in a format that

R can understand, such as R script. In this case, we source our source file named *func_model.r* which is located under 03_*code* folder under our working directory).

```
    # set working directory
setwd("/Desktop/model_decomp")
# load model code
source("03_code/func_model.r")
```

We then defined sub-directories under our working directory for locating data files (needed for the script) and output results/plots as follows:

```
  # location of data files needed for script
  work_root <- "02_data"
  # location to output results/plots
 out_dir <- "04_outputs"
```

Load Data and shape files

We then load Ethiopia regional (admi1) shapefile and load the pre-processed data (SARA 2016, SARA 2018, SPA 2014, and imputed CCEI results) from 02_*data* folder created above.

```
   #Shape file
   shp.admin1 <- read_rds(file.path(work_root,
   "admin_1_shapefile_2018.rds"))
   #Facility readiness data
   df.fac_readiness_full <- fread(file.path(work_root,
   "facility_readiness/2023-06 18_sara_facindices_full_index_ETH.csv"))
df.fac_readiness_full <- unique(df.fac_readiness_full,
by = c("svy", "fac_id"))
```

In the above code, unique() function drops all duplicate rows from the *df.fac_readiness_full* data frame, keeping only the first occurrence of each row.

The we read in subsetted index for SARA 2016, SARA 2018, SPA 2014, and CCEI 2019-2020 (which only includes 3 indicators: fridge available, coldchain system, electricity grid always) using the following code:

```
  df.fac_readiness_subset <- fread(file.path(work_root,
   "facility_readiness/2023-06-05_sara_facindices_cceisubset_index_ETH.csv")
   df.fac_readiness_subset <-
   unique(df.fac_readiness_subset,
   by = c("svy", "fac_id"))
```

Then we load community access data and intent to vaccinate and child level observations data from their respective folders under our working directory:

```
      # read in motor/walk travel times to closest
      #facility with RI services
    df.access <- read_rds(file.path(work_root,
      "community_access/access.rds"))
      # get additional facility info
   df.facilities <- read_rds(file.path(work_root,
```

```
 "community_access/prepped.rds"))
# Intent to vaccinate
  df.intent <- fread(file.path(work_root,
  "intent/predictions.csv"))
  # Child-level observations
  df.dhs <- list.files(file.path(work_root,
  "child_level_observations"), full.names=T,
  pattern="csv") %>% lapply(., fread) %>% rbindlist(.,
  fill=T)
```

4.1 Data Preparation

Process Child Level DHS Data

Drop individuals without latitude or longitude information using *filter()* function from *dplyr* library. In addition, the $year\_shift$ calculates the difference between the survey year and the child's age in years, rounded down to the nearest integer.

```
 #  Drop individuals without lat/long
 df.dhs<-filter(df.dhs, !is.na(lat))
 # Include individuals >= 12 months of age
 df.dhs<-filter(df.dhs, age_month>=12)
# When were kids 0-11 months (based on age/interview month)
   df.dhs[, diff := age_month-11]
   df.dhs[, year_shift := trunc((int_year * 12 + int_month - diff -1)/12)]
```

Then the code below does the follwing:

- Sets the year column to the value of the $year\_shift$ column.
- Recodes the sex_id column to a binary variable indicating whether the child is female (1) or male (0).
- Replaces missing values in the pent3 column with the values from the dpt3 column.
- Replaces missing values in the pent1 column with the values from the dpt1 column.

```
 # Set year as shift year
  df.dhs[, year := year_shift]
  # Recode gender
  df.dhs[, female := sex_id - 1]
  # Replace pent3 with dpt3
  df.dhs[is.na(pent3), pent3 := dpt3]
  df.dhs[is.na(pent1), pent1 := dpt1]
```

Then we fixed admin1 (regional administration column) as follows. The folllwing code first creates a new column called $admin\_1\_clean$ in the df.dhs data frame.

- It then uses the $str\_replace\_all()$ function to replace all occurrences of "-" with a single space in the $admin\_1$ column.

- Uses the *str_replace_all*() function to replace all misspelled occurrences of "addis abeba" in the *admin_1* column.
- Then it uses the *str_replace_all*() function to replace all occurrences of "affar" with "afar" in the *admin_1* column.
- Uses the *str_replace_all*() function to replace all occurrences of "benishangul gumuz" in the a$dmin_1 column$.
- Uses the $str_replace_all()$ function to replace all occurrences of "snnpr" in the *admin_1* column.
- Uses the *str_replace_all*() function to replace all occurrences with "oromia" in the *admin_1* column.
- Sorts the unique values in the *admin_1_clean* column.
- The *str_replace_all*() function is used to replace all occurrences of a pattern with another pattern. In this case, the *str_replace_all*() function is used to replace all occurrences of certain strings with other strings in the $admin_1$ column.
- The *sort()* function is then used to sort the unique values in the *admin_1_clean* column.

```
# Fix admin_1
df.dhs[, admin_1 := admin_1 %>%
    str_replace_all(., "\\-", " ") %>%
    # replace - with a single space
     str_replace_all(., "addis abeba|addis
     abada|^addis$|addis adaba", "addis ababa") %>%
     str_replace_all(., "affar", "afar") %>%
     str_replace_all(., "ben gumz|^benishangul$",
     "benishangul gumuz") %>%
     str_replace_all(., "\\bsnnp\\b", "snnpr") %>%
     str_replace_all(., "oromiya", "oromia")]
     sort(unique(df.dhs$admin_1))
```

We then created *has_motor* column in the df.dhs data frame. The following code first sets the value of the *has_motor* column to 1 if the value of the *has_car_truck* column or the *has_motorcycle_scooter* column is 1. It then sets the value of the *has_motor* column to 0 if the value of the *has_car_truck* column and the *has_motorcycle_scooter* column are both 0 or both 7 (response 7 refers to individuals who don't usually live in the household which is treated as walking time). The *has_car_truck* column and the *has_motorcycle_scooter* column indicate whether the household has a car or truck and a motorcycle or scooter, respectively. The *has_motor* column is a binary variable indicating whether the household has any motorized transportation.

```
df.dhs[has_car_truck == 1 | has_motorcycle_scooter ==
1, has_motor := 1]
df.dhs[(has_car_truck == 0 & has_motorcycle_scooter ==
0) | (has_car_truck == 7 & has_motorcycle_scooter ==
7), has_motor := 0]
```

The respondent education is re coded to match intent education categories. The value "no education", "Primary", "secondary" and "higher" in the *edu_level* column were renamed to "No education", "Primary", "Secondary and above" respectively. Note that the *edu_level* column contains the educational attainment of the household head.

```
# Recode respondent's partner education
df.dhs[partner_edu_level == "no education",
partner_edu_level := "No education"]
df.dhs[partner_edu_level == "primary",
partner_edu_level := "Primary"]
df.dhs[partner_edu_level %in% c("secondary", "higher"),
partner_edu_level := "Secondary and above"]
```

Then we check if the value of the *partner_edu_level* column is missing or is equal to "" or "don't know". If the value of the *partner_edu_level* column is missing or is equal to "" or "don't know", then the value of the *partner_edu_level* column is set to the value of the *edu_level* column.

```
df.dhs[is.na(partner_edu_level) | partner_edu_level
%in% c("", "don't know"), partner_edu_level := edu_level]
```

The religion column which contains the religious affiliation of the household head is recoded as follows:

- New column called *religion_recode* is created first in the df.dhs data frame.
- The value of the *religion_recode* column is set to "Christianity: Orthodox" if the value of the religion column is 1.
- The value of the *religion_recode* column is set to "Christianity: Catholic" if the value of the religion column is 2.
- The value of the *religion_recode* column is set to "Christianity: Non-Catholic" if the value of the religion column is 3.
- The value of the *religion_recode* column is set to "Islam" if the value of the religion column is 4.
- The value of the *religion_recode* column is set to "Other/None" if the value of the religion column is 5, 6, or 96.
- The value of the *religion_recode* column is set to NA if the value of the religion column is 9 or 99.

```
# Recode religion
df.dhs[religion == 1, religion_recode := "Christianity: Orthodox"]
df.dhs[religion == 2, religion_recode := "Christianity: Catholic"]
df.dhs[religion == 3, religion_recode := "Christianity: Non-Catholic"]
df.dhs[religion == 4, religion_recode := "Islam"]
df.dhs[religion %in% c(5, 6, 96), religion_recode := "Other/None"]
df.dhs[religion %in% c(9, 99), religion_recode := NA]
```

We then drop those missing religion as follows:

```
df.dhs <- subset(df.dhs, !is.na(religion_recode))
```

Regarding occupation, we recoded it as follows: The code you have provided does the following:

- We first created a new column called *employment_bin* in the df.dhs data frame.
- The value of the *employment_bin* column is to 0 if the value of the occupation column is "not working", "other", or "don't know".
- The value of the *employment_bin* column is to 1 if the value of the occupation column is not empty and the value of the *employment_bin* column is missing.
- We then created another column *partner_employment_bin* in the df.dhs data frame.
- The value of the *partner_employment_bin* column is to 0 if the value of the *partner_occupation* column is "not working", "other", or "don't know".
- The value of the *partner_employment_bin* column is to 1 if the value of the *partner_occupation* column is not empty and the value of the *partner_employment_bin* column is missing.

```
df.dhs[occupation %in% c("not working", "other",
"don't know"), employment_bin := 0]
df.dhs[occupation != "" & is.na(employment_bin),
employment_bin := 1]
df.dhs[partner_occupation %in% c("not working",
"other", "don't know"),partner_employment_bin := 0]
df.dhs[partner_occupation != "" &
is.na(partner_employment_bin), partner_employment_bin := 1]
```

The urban column which indicates whether the household is located in an urban or rural area, the *location_name* column contains the name of the administrative region where the household is located and the *child_id* column is a unique identifier for each child in the data set. The following creates a new column called rural in the df.dhs data frame. It then sets the value of the rural column to 0 if the value of the urban column is 1 and to 1 if the value of the urban column is 0. It then removes the *location_name* column from the df.dhs data frame and creates a new column called $child_id$ in the df.dhs data frame. Then the value of the *child_id* column is set to the row number of the df.dhs data frame.

```
# Urban/rural
df.dhs[, rural := ifelse(urban, 0, 1)]
# Remove location_name
df.dhs <- df.dhs[, -c("location_name"), with = F]
# create child id
df.dhs[, child_id := .I]
```

The pent1, mcv1, pcv1, and rota1 columns in our data frame indicate whether the child has received the first dose of the pentavalent, measles, pneumococcal conjugate, and rotavirus vaccines, respectively. The pent3, mcv2, pcv3, and rotac columns indicate whether the child has received the third dose of the respective vaccines.

The following code:

- First creates new columns called *pent_drop*, *mcv_drop*, *pcv_drop*, and *rota_drop* in the df.dhs data frame.
- Then it sets the value of the *pent_drop* column to the value of the pent3 column if the value of the pent1 column is 1.
- Sets the value of the *mcv_drop* column to the value of the mcv2 column if the value of the mcv1 column is 1.
- Sets the value of the *pcv_drop* column to the value of the pcv3 column if the value of the pcv1 column is 1.
- Sets the value of the *rota_drop* column to the value of the rotac column if the value of the rota1 column is 1.

```
df.dhs[pent1 == 1, pent_drop := pent3]
df.dhs[mcv1 == 1, mcv_drop := mcv2]
df.dhs[pcv1 == 1, pcv_drop := pcv3]
df.dhs[rota1 == 1, rota_drop := rotac]
# Drop unnecessary columns
drop <- c(names(df.dhs)[grepl("dose_from|_card", names(df.dhs))])
df.dhs <- df.dhs[, -drop, with = F]
```

MERGING DRIVER DOMAINS

Merge on intent

The follwing code is used to align intent variables to match DHS:

- The code first sorts the unique values of the *admin_1* column in the df.intent and df.dhs data frames. The output of the *sort()* function shows that the unique values of the *admin_1* column in the two data frames are the same, except the fact that the *admin_1* column in the df.intent data frame has "hareri" and "beneshangul gumu" while the *admin_1* column in the df.dhs data frame has "harari" and "benishangul-gumuz"
- We changed the values of the *admin_1* column in the df.intent data frame to match the values in the df.dhs data frame.
- The *tolower()* function converts all the characters in the *admin_1* column to lowercase. The *str_replace_all*() function replaces all occurrences of the string "hareri" with the string "harari" and all occurrences of the string "beneshangul gumu" with the string "benishangul gumuz".
- Finally, the code intersects the unique values of the *admin_1* column in the df.dhs and df.intent data frames. The output of the *intersect()* function shows that the common values of the *admin_1* column in the two data frames are the administrative regions of Ethiopia.

```
sort(unique(df.intent$admin_1))
sort(unique(df.dhs$admin_1))
df.intent[, admin_1 := admin_1 %>%
        tolower %>%
        str_replace_all(., "hareri", "harari") %>%
```

```
        str_replace_all(., "beneshangul gumu",
        "benishangul gumuz")]
intersect(unique(df.dhs$admin_1), unique(df.intent$admin_1))
```

The following code first changes the names of the *education_combined* and *religion_combined* columns in the df.intent data frame to *edu_level* and *religion_recode*, respectively. This is done in order to match the column names.

```
# match column names
setnames(df.intent, old = c("education_combined",
"religion_combined"), new = c("edu_level",
"religion_recode"))
```

We then merged the df.dhs and df.intent data frames on the *admin_1*, *edu_level*, and *religion_recode* columns as follows (the all.x=T argument tells R to keep all rows from the df.dhs data frame, even if there are no matching rows in the df.intent data frame).

```
# merge on intent model variables
df.dhs <- merge(df.dhs, df.intent, by = c("admin_1",
"edu_level", "religion_recode"), all.x=T)
```

In the above code, the *merge()* function is used to combine the df.dhs data frame, which contains information about children in Ethiopia, and the df.intent data frame, which contains information about the vaccination intentions of mothers in Ethiopia. The by argument to the merge() function specifies that the df.dhs and df.intent data frames are merged on the *admin_1*, *edu_level*, and *religion_recode* columns.

Merge on community access:

```
# ensure rank is numeric
df.access[, motor_rank := as.numeric(motor_rank)]
df.access[, walk_rank := as.numeric(walk_rank)]
```

The following code then:

- First calculates the minimum travel time to the closest facility with RI services for motor and walking travel.
- It then subsets the data to the facilities with the shortest travel time for each mode of transportation.
- Merges the two data subsets to create a single data frame that contains the information about the closest facility with RI services for both motor and walking travel.
- Merges the *df.access_mins* data frame with the df.dhs data frame on the *cluster_long* and *cluster_lat* columns.
- Assigns the time to the closest facility with RI services based on the household's vehicle ownership status.

```
identify closest facility with RI services
df.access[ri_services == 1, motor_min :=
min(motor_rank), by = c("cluster_lat", "cluster_long")]
```

```
df.access[ri_services == 1, walk_min :=
min(walk_rank), by = c("cluster_lat", "cluster_long")]
## subset to shortest walking or motor travel time
df.access_motor <- df.access[motor_rank == motor_min]
setnames(df.access_motor, c("fac_long", "fac_lat",
"group_id", "source"), paste0("motor_", c("fac_long",
"fac_lat", "group_id", "source")))
df.access_motor <- df.access_motor[, c("cluster_long",
"cluster_lat", paste0("motor_", c("fac_long",
"fac_lat", "group_id", "source")), "time_motor")]
df.access_walk <- df.access[walk_rank == walk_min]
setnames(df.access_walk, c("fac_long", "fac_lat",
"group_id", "source"), paste0("walk_", c("fac_long",
"fac_lat", "group_id", "source")))
df.access_walk <- df.access_walk[, c("cluster_long",
"cluster_lat", paste0("walk_", c("fac_long",
"fac_lat", "group_id", "source")), "time_walk")]
df.access_mins <- merge(df.access_motor,
df.access_walk, by = c("cluster_long", "cluster_lat"), all = T)
# merge on DHS cluster GPS
df <- merge(df.dhs, df.access_mins, by.x = c("lat",
"long"), by.y = c("cluster_lat", "cluster_long"), all.x = T)
# assign community access based on vehicle ownership
df[has_motor == 1, c("time_vehicle_dependent",
"fac_lat", "fac_long", "source", "group_id") := .
(time_motor, motor_fac_lat, motor_fac_long,
motor_source, motor_group_id)]
df[has_motor == 0, c("time_vehicle_dependent",
"fac_lat", "fac_long", "source", "group_id") := .
(time_walk, walk_fac_lat, walk_fac_long, walk_source,
walk_group_id)]
```

In the above code, the *ri_services* column in the df.access data frame indicates whether the facility provides RI services. The *motor_rank* and *walk_rank* columns in the df.access data frame indicate the rank of the facility based on the travel time to the facility. The *min()* function is used to find the minimum value in a column. The by argument to the *min()* function specifies that the minimum value is calculated over the *cluster_lat* and *cluster_long* columns.

The merge() function is used to combine the *df.access_motor* and *df.access_walk* data frames on the *cluster_long* and *cluster_lat* columns. The all.x=T argument to the *merge()* function is used to keep all rows from the *df.access_motor* data frame. The *has_motor* column in the df.dhs data frame indicates whether the household owns a motor vehicle. The *assign()* function is used to assign the time to the closest facility with RI services based on the household's vehicle ownership status.

Merge on facility readiness

In order to merge the data on facility readiness, we followed the following steps using

the below code:

- We started with renaming the fac_index column in the $df.fac\_readiness\_full$ data frame to $fac\_index\_full_equal$ and the fac_index column in the $df.fac\_readiness\_sub$ data frame to $fac\_index\_subset_equal$ using *setnames()* function
- We then allign the survey names in the $df.fac\_readiness\_subset$ data frame by changing the name of the svy column to Cold Chain Assessment 2020 if the value of the svy column is equal to "CCEI".
- Then merge the $df.fac\_readiness\_full$ and $df.fac\_readiness\_subset$ data frames on the fac_id and svy columns.
- New column called fac_id_orig is then created in the merged data frame that contains the original value of the fac_id column.
- The fac_id column is removed from the merged data frame.

```
# rename columns
setnames(df.fac_readiness_full, "fac_index",
"fac_index_full_equal", skip_absent = T)
setnames(df.fac_readiness_subset, "fac_index",
"fac_index_subset_equal", skip_absent = T)
# align survey names
df.fac_readiness_subset[svy == "CCEI", svy := "Cold
Chain Assessment 2020"]
# combine different index constructs
df.readiness <- merge(df.fac_readiness_full[,
c("fac_id", "svy", "ri_services",
names(df.fac_readiness_full)[grepl("vxavail",
names(df.fac_readiness_full))],
"fac_index_full_equal"), with = F],
 df.fac_readiness_subset[, c("fac_id", "svy",
 "fac_index_subset_equal"), with = F],
                  by = c("fac_id", "svy"))
df.readiness <- df.readiness[, fac_id_orig := fac_id][,
-c("fac_id"), with = F]
```

We then merge facility readiness scores onto grouped facility list using the below code:

```
df.facs_readiness <- merge(df.facilities[,
c("fac_id_orig", names(df.facilities)
[names(df.facilities) %ni% names(df.readiness)]),
  with = F],  df.readiness, by.x = c("fac_id_orig",
  "source"), by.y = c("fac_id_orig", "svy"), all.x = T)
```

The above code:

- Merges the df.facilities and df.readiness data frames on the fac_id_orig and svy columns.
- The all.x = T argument to the *merge()* function is used to keep all rows from the df.facilities data frame, even if there are no matching rows in the df.readiness data frame.

- The $names(df.facilities)[names(df.facilities)\%ni\%names(df.readiness)]$ expression returns a vector of the column names in the df.facilities data frame that are not in the df.readiness data frame.

We then drop facilities that do not have a corresponding index score using the code below:

```
df.facs_readiness <- subset(df.facs_readiness,
!is.na(fac_index_full_equal))
```

Then the readiness scores were merged onto combined DHS, intent, and access table using the following code:

```
df <- merge(df[, -c("source"), with = F],
    df.facs_readiness[, c("group_id", "source",
    names(df.facs_readiness)[names(df.facs_readiness)
    %ni% names(df)]), with = F],
        by.x = c("fac_lat", "fac_long", "group_id"),
        by.y = c("group_lat", "group_long",
        "group_id"),
        all = T)
```

Final Cleanup

```
# drop those with no vehicle info
df %>% filter(!is.na(time_vehicle_dependent))
# remove facilities that did not map onto a child
df %>% filter(!is.na(child_id))
# remove children that mapped onto facilities without scores
df %>% filter(!is.na(fac_index_full_equal))
# check for children that fall outside of Ethiopia shapefile
df.outside_eth <- df[is.na(fac_long)]
# keep children whose birth year is within +/- 3 years
#of facility readiness score year
df %>% filter(!is.na(abs(year_data - year) <= 3))
```

When individuals map onto facilities with multiple readiness scores, the average average score is taken using the below code:

```
df[, c("readiness_full", "readiness_subset") := .
(mean(fac_index_full_equal, na.rm = T),
mean(fac_index_subset_equal, na.rm = T)), by =
"child_id"]
df <- unique(df, by = "child_id")
```

Reshape the data frame in to long format on vaccine, the following code is used which:

- Creates a vector called vacc.cols that contains the names of the columns in the df data frame that represent vaccination information.
- Creates a vector called id.cols that contains the names of the columns in the df data frame that are not vaccination information.

- The setdiff() function is used to find the difference between the names of all the columns in the df data frame and the names of the columns in the vacc.cols vector.

```
vacc.cols <- c("pent1", "pent2", "pent3", "pent_drop",
              "mcv1", "mcv2", "mcv_drop",
              "pcv1", "pcv2", "pcv3", "pcv_drop",
              "rota1", "rota2", "rotac", "rota_drop")
              id.cols <- setdiff(names(df), vacc.cols)
```

Using the above information,the following code:

- Creates a long format data frame called df.dhs.l from the df data frame.
- The melt() function is used to melt the data frame. The id.cols vector is used to specify the columns that should be kept in the original order.
- The variable.name argument is used to specify the name of the column that will contain the names of the vaccine doses. The value.name argument is used to specify the name of the column that will contain the vaccination coverage data.
- The code then removes the numeric prefixes from the *vaccine_dose* column and assigns the resulting string to the vaccine column.

```
df.dhs.l <- df[, c(id.cols, vacc.cols), with=F] %>%
  data.table::melt(id.vars=id.cols,
 variable.name="vaccine_dose", value.name="coverage")
 df.dhs.l[, vaccine := gsub("_drop", "", gsub("[0-9]",
 "", vaccine_dose))]
```

Then the following code subsets the df.dhs.l data frame to rows where the *vaccine_dose* column is equal to "pent3" and the coverage column is not missing.

```
df <- df.dhs.l[vaccine_dose == "pent3" &
!is.na(coverage)]
```

4.2 Fitting Machine Learning Model

We first create a vector called vaccs that contains the names of two vaccines: pent3 and pent1.

```
vaccs <- c("pent3", "pent1")
```

Then we transform and set access and readiness as follows

- First we create a new column called access that ranges from 0 to 1, where 0 indicates the worst access and 1 indicates the best access. The *time_vehicle_dependent* column indicates the time to the closest facility with RI services for children who do not have a motor vehicle. The *log()* function is used to transform the *time_vehicle_dependent* column to a logarithmic scale. The *scales::rescale()* function is then used to transform the logarithmic scale to a scale from 0 to 1.
- Then we created a new column called readiness that indicates the overall readiness of the facility to provide RI services. The *readiness_full* column

is a composite index that measures the readiness of the facility to provide RI services.

- The intent column is also created which indicates the predicted vaccination intention of the mother.

```
# Transform and set access/readiness
df[, access := 1 -
scales::rescale(log(time_vehicle_dependent+0.001), c(0, 1))]
df[, readiness := readiness_full]
df[, intent := pred_intent_all]
```

Note that the access and readiness columns are continuous variables that can be used to measure the impact of access and readiness on vaccination coverage.

4.2.1 Visualization of the domain average by region

To visualize the average of the three domains (rediness, intent and access) by region, the following variables were first created using the code below:

- We first created a new data frame called $plot_dt$ that contains the mean values of the access, intent, and readiness columns for each administrative region. The $str\_replace\_all()$ function is used to replace "benishangul gumuz" with "beneshangul gumu", "harari" with "hareri", "S N N P" with "SNNPR"
- The $str\_to\_title()$ function is used to convert the administrative region names to title case.
- We then merged the $plot\_dt$ data frame with the shp.admin1 data frame (shape file) on the $ADM1\_NAME$ column. The by.x argument specifies that the $ADM1\_NAME$ column in the $plot\_dt$ data frame should be merged with the $ADM1\_NAME$ column in the $shp.admin1$ data frame. The by.y argument specifies that the admin1 column in the $plot\_dt$ data frame should be merged with the $ADM1\_NAME$ column in the $shp.admin1$ data frame. The $all.x = T$ argument is used to keep all rows from the $plot\_dt$ data frame.

```
plot_dt <- df[, .(access =mean(time_vehicle_dependent),
intent = mean(intent), readiness = mean(readiness)), .(admin1)]
plot_dt[, admin1 := admin1 %>%
      str_replace_all(., "benishangul gumuz",
      "beneshangul gumu") %>%
      str_replace_all(., "harari", "hareri") %>%
      str_to_title() %>%
      str_replace_all(., "S N N P", "SNNPR")]
      plot_dt <- merge(shp.admin1, plot_dt, by.x =
      "ADM1_NAME", by.y = "admin1", all.x = T)
```

Then we created a choropleth map of Ethiopia, where the fill color of each administrative region represents the mean vaccination intention for that region by using the following code. The $geom\_sf()$ function is used to create a choropleth map, the *aes()* function specifies that the fill color of each polygon should be determined by the value of the intent column. The $scale\_fill\_gradientn()$ function specifies the color scheme for the choropleth map. The *labs()* function specifies the title and the

legend for the plot. The *theme_void()* function removes the background and grid lines from the plot and the *theme()* function specifies the alignment of the plot title.

```
p1 <- ggplot(plot_dt)+ geom_sf(aes(fill = intent*100),
color = "black", show.legend = TRUE)+
scale_fill_gradientn(colors = c("#005159", "#4cb6c1",
"#99d5db", "white") %>% rev())+
labs(title = "Intent", fill = "%")+ theme_void()+
theme(plot.title = element_text(hjust = 0.5))
```

In this same manner, the following code creates a choropleth map of Ethiopia, where the fill color of each administrative region represents the mean facility readiness score for that region. The darker the color, the higher the mean facility readiness score. The code is similar to the code for the p1 plot above, with the only difference being that the fill aesthetic is now set to the readiness column.

```
p2 <- ggplot(plot_dt)+ geom_sf(aes(fill =
readiness*100), color = "black", show.legend = TRUE)+
scale_fill_gradientn(colors = c("#005159", "#4cb6c1",
"#99d5db", "white") %>% rev())+
labs(title = "Facility Readiness", fill = "Score")+
theme_void()+ theme(plot.title = element_text(hjust = 0.5))
```

In the same manner, the below code creates a choropleth map of Ethiopia, where the fill color of each administrative region represents the mean time to the closest facility with RI services. The darker the color, the longer the mean time to the closest facility. The code is similar to the code for the p1 and p2 plots, with the only difference being that the fill aesthetic is now set to the access column.

```
p3 <- ggplot(plot_dt)+ geom_sf(aes(fill = access),
color = "black", show.legend = TRUE)+ scale_fill_gradientn(colors = c("#0
"#99d5db", "white") %>% rev())+  labs(title = "Access",
fill = "Mins")+ theme_void()+
theme(plot.title = element_text(hjust = 0.5))
```

Then the following code is used to combine the three plots (p1, p2, p3) into a single grid. The *nrow=1* argument specifies that the plots should be arranged in a single row.

```
cowplot::plot_grid(p1, p2, p3, nrow = 1)
```

The visualization result of the above code is displayed below:

Figure 4.1: Distribution of the average intent, readiness and access scores by region

4.2.2 Running the Machine Learning Model

We then fit XGBoost model. In order to do that, we started with setting the order of the x-axis names to intent, readiness, and access and the y-axis to coverage as follows:

```
## Organize x names
cols.base <- c("access", "readiness", "intent")
y <- "coverage"
```

We then set the seed for the random number generator to 1 to ensure that the results of the random sampling are reproducible. We then use the *createDataPartition*() function to create a training set and a test set from the caret package in R. The *createDataPartition*() function randomly partitions the data into two sets, such that the training set contains 80% of the data and the test set contains 20% of the data. The y argument specifies the target variable. The indexes object contains the row indices for the training set and the test set. The training set will be used to train the ML model, and the test set will be used to evaluate the performance of the model. The code is presented below:

```
## Run XGB
set.seed(1)
#install.packages("caret")
library(caret)
indexes <- createDataPartition(df[[y]], times = 1,p =
0.8,list = FALSE)
```

We then fit XGBoost model for with access, readiness, intent

```
# Fit with access, readiness, intent
```

```
library(xgboost)
train1 <- xgb.DMatrix(data=df[indexes, c(cols.base),
with=F] %>%  as.matrix, label= df[indexes][[y]])
test1 <- xgb.DMatrix(data=df[-indexes, c(cols.base),
with=F] %>%  as.matrix, label= df[-indexes][[y]])
library(data.table)
fit1 <- fit.xgb(train1, n=250, nfolds=5,
objective="binary:logistic", eval_metric="auc",
minimize_eval=0)
```

The above code first loads the xgboost package which is an implementation of the XGBoost algorithm. It then converts the training data into an xgb.DMatrix object. The *xgb.DMatrix* object is a special data structure that is used by the XGBoost algorithm. Then it converts the test data into an xgb.DMatrix object abd fits an XGBoost model to the training data. The *fit.xgb()* function is a custom function that created to fit an XGBoost model. The *n=250* argument specifies the number of trees in the model. The *nfolds=5* argument specifies the number of folds for cross-validation. The objective argument specifies the loss function to be used while the *eval_metric* argument specifies the evaluation metric to be used. The *minimize_eval* argument specifies whether to minimize or maximize the evaluation metric. The fit1 object contains the fitted XGBoost model which can then be used to make predictions on new data. Runing the code may take few minutes depending on the performance of our machine.

The following code saves the fitted XGBoost model to a file called *fit_dtp3_base_intent_all_3years_readiness.rds* and the data used to train the XGBoost model to a file called *fit_dtp3_base_intent_all_3years_readiness_data.csv* in the *out_dir* directory. The *export()* function from the *rio* package is used. The code is as follows:

```
export(fit1, file.path(out_dir,
"/fit_dtp3_base_intent_all_3years_readiness.rds"))
export(df, file.path(out_dir,
"/fit_dtp3_base_intent_all_3years_readiness_data.csv"))
```

Getting Shapley Values

The first stage Shapley values are obtained as follows:

- We first calculate the SHAP values for the XGBoost model using the *shap.prep()* function from the SHAP package. The *xgb_model* argument specifies the fitted XGBoost model. The *X_train* argument specifies the training data.
- We then create a summary plot of the SHAP values using the *shap.plot.summary()* function from the SHAP package.
- A dependence plot of the SHAP values for the access variable is then created using the *shap.plot.dependence()* function from the SHAP package. The *color_feature* argument specifies the variable to use for coloring the points.

```
## First stage SHAP Values
shap_long <- shap.prep(xgb_model = fit1$best, X_train =
```

```
df[, c(cols.base), with=F] %>% as.matrix)
shap.plot.summary(shap_long)
shap.plot.dependence(data_long = shap_long, x =
'access', y = 'access', color_feature = 'access')
```

The above step produces the following plot:

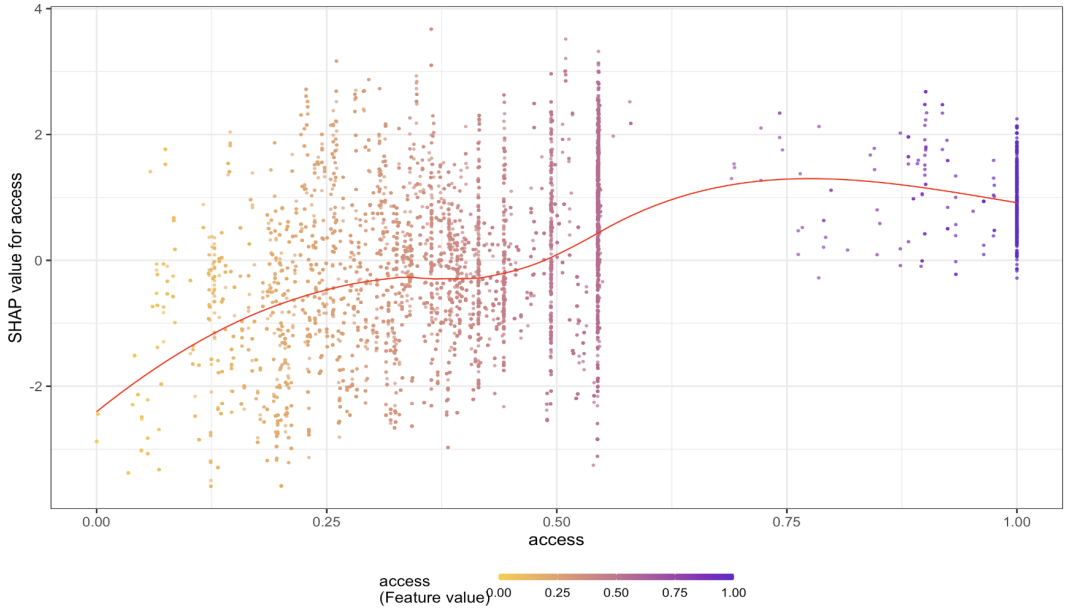

Figure 4.2: Shapley Values of Access Variable

- Using the below code, we obtain the following result for readiness variable:

```
shap.plot.dependence(data_long = shap_long, x =
'readiness', y = 'readiness', color_feature = 'readiness')
```

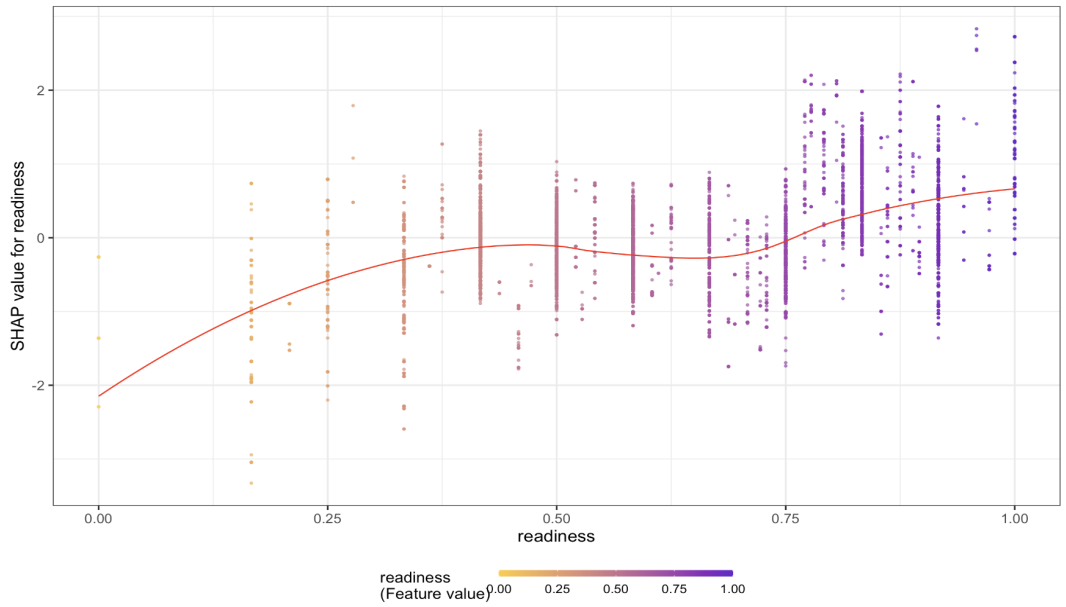

Figure 4.3: Shapley Values of Readiness Variable

- Again repeating the above steps for intent variable and using the code below, we obtain the following plot

```
shap.plot.dependence(data_long = shap_long, x =
'intent', y = 'intent', color_feature = 'intent')
```

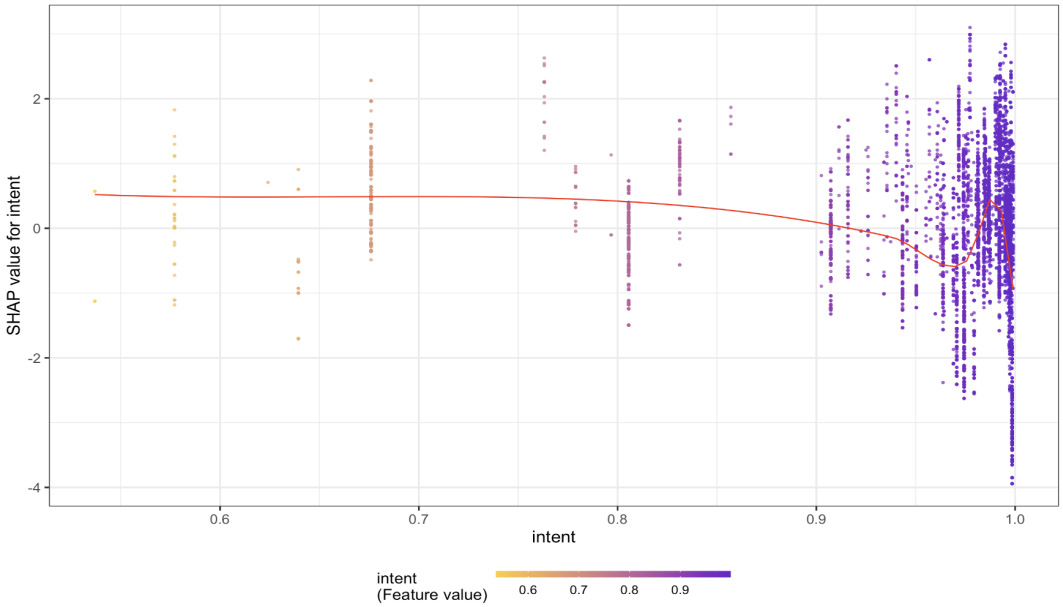

Figure 4.4: Shapley Values of Intent Variable

The SHAP values are a measure of the importance of each feature ML model. The dependence plots show the SHAP values for each feature as a function of the value of the feature. The plots show that the access variable has the greatest impact on the predicted vaccination coverage. The readiness variable also has a significant impact, while intent variable has a relatively small impact on the predicted vaccination coverage.

4.2.3 Demographic Effects

In order to quantify the effects of demographic variables on the likelihood of vaccination, the study applied logistic regression model. We started with creating a vector called predictors that contains the names of the variables that will be used as predictors and converts the year variable to an integer using the below code:

```
## data diagnostics
predictors <- c("sex_id", "edu_level", "wealth_index_dhs",
"religion_recode", "admin_1", "year")
df[, year := as.integer(year)]
```

Then using the predict() function from the xgboost package to, we predict the vaccination coverage for the test data using the code below:

```
pred1 <- predict(fit1$best, newdata=df[, c(cols.base), with=F]
%>% as.matrix)
df[, pred1 := pred1]
```

In the above code, the $fit1\$best$ argument specifies the fitted XGBoost model from the previous section while the newdata argument specifies the test data. The code stores the predicted vaccination coverage in the pred1 object. We then add a new column called pred1 to the df data frame. The pred1 column contains the predicted

vaccination coverage for each observation.

We then fit a generalized linear mixed model (GLMM) to the data using the follwing code:

```
fit1.glm.stage2 <- glmer(formula = dpt3 ~ offset(pred1) +
as.factor(sex_id) + edu_level + as.factor(wealth_index_dhs) +
(1|year_shift) + (1|admin_1), family = "binomial", data = df)
```

The *glmer()* function from the *lme4* package is used to fit a generalized linear mixed model (GLMM). The formula argument specifies the model formula. The *offset(pred1)* term specifies that the predicted vaccination coverage from the XGBoost model is used as an offset in the GLMM where as the *as.factor()* function is used to convert the categorical variables to factors. The $(1|year\_shift)$ and $(1|admin\_1)$ terms specify that the year and administrative region are random effects in the model.

We then check for multicollinearity of the above using:

```
library(performance)
check_collinearity(fit1.glm.stage2)
```

The above code resulted in the follwing output:

```
                    Term                        VIF   VIF 95% CI   Increased SE Tolerance
                    as.factor(sex_id)          1.00   [1.00, Inf]   1.00         1.00
                    edu_level                  1.13   [1.10, 1.16]  1.06         0.89
 as.factor(wealth_index_dhs)  1.13   [1.10, 1.16]  1.06         0.89
```

We then check for coefficients that make sense using their respective odds ration (>10 is probably too large). The following code is used to check for coefficients. We used the *plot_model*() function from the *sjPlot* package to plot the results of the fitted GLMM model.

```
sjPlot::plot_model(fit1.glm.stage2)
```

The following result is obtained after running the code:

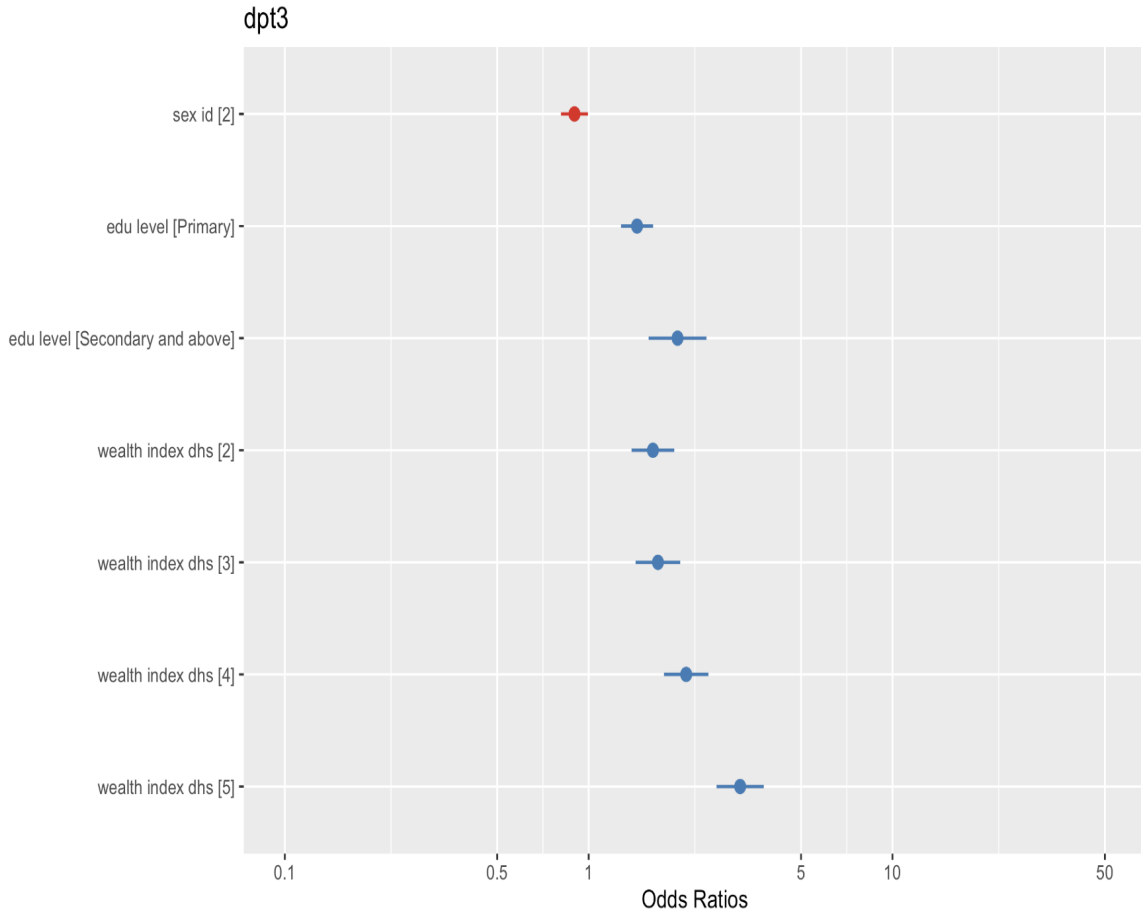

Figure 4.5: Odds Ratio for Coefficients

Visualization of Odds Ratios

We create plot model object for stage 2 model using the below code:

```
p.m2 <- plot_model(fit1.glm.stage2)
p.m2.admin1 <- plot_model(fit1.glm.stage2, type="re")[[1]]
p.m2.year <- plot_model(fit1.glm.stage2, type="re")[[2]]
```

Group headings and reference values were then added to our stage to plot model:

```
p.m2$data <- p.m2$data %>% as.data.table
p.m2.year$data <- p.m2.year$data %>% as.data.table
```

We then add terms to our stage two

```
p.stage2 <- rbind(data.table(term="SEX (CHILD)",
estimate=NA, facet="(Intercept)"),
data.table(term="Male (Ref)", estimate=1, std.error=0.1,
conf.low=1, conf.high=1, xpos=12, group="pos", facet="
(Intercept)"),  p.m2$data[term=="as.factor(sex_id)2"] %>%
mutate(term="Female"),
data.table(term="MATERNAL EDUCATION", estimate=NA, facet="
(Intercept)"),
data.table(term="Less than primary (Ref)", estimate=1,
std.error=0.1, conf.low=1, conf.high=1, xpos=12,
```

```
group="pos", facet="(Intercept)"),
p.m2$data[grepl("edu_level", term)] %>%
mutate(term=c("Primary", "Secondary+")),
data.table(term="DHS WEALTH INDEX", estimate=NA),
data.table(term="1st Quintile (Ref)", estimate=1,
std.error=0.1, conf.low=1, conf.high=1, xpos=12,
group="pos"),
p.m2$data[grepl("wealth", term)] %>% mutate(term=c("2nd
Quintile", "3rd Quintile", "4th Quintile", "5thQuintile")),
data.table(term="REGION", estimate=NA, facet="
(Intercept)"), p.m2.year$data, data.table(term="BIRTH
YEAR", estimate=NA, facet="(Intercept)"), p.m2.admin1$data
%>% mutate(term=str_to_title(term)), fill=T)
```

Then the following code will factor the term variable in the p.stage2 data frame, and then reverse the order of the levels. This means that the highest level will be at the bottom, and the lowest level will be at the top.

```
p.stage2$term <- p.stage2$term %>% factor(., levels=.,
labels=., ordered=T) %>% fct_rev
```

The odds ration is then visualized using the ggplot function as follows:

```
p.effects <- ggplot(data = p.stage2, aes(y = term, x =
estimate, color = group, fill = group)) +
geom_pointrange(aes(xmin = conf.low, xmax = conf.high)) +
geom_vline(aes(xintercept = 1), linetype = "dashed") +
scale_color_brewer(palette = "Set1") +
scale_x_log10(breaks = c(0.5, 1, 2, 3, 4), labels =
c("0.5", "1", "2", "3", "4")) + scale_y_discrete(labels=c("SEX(CHILD)"=
expression(bold("SEX (CHILD)")), "Male (Ref)", "Female",
"MATERNAL EDUCATION"=expression(bold("MATERNAL EDUCATION")),
"Primary (Ref)",  "Secondary+", "DHS WEALTH
INDEX"=expression(bold("DHS WEALTH INDEX")),
"1st Quintile (Ref)", "2nd Quintile", "3rd Quintile",
"4th Quintile", "5th Quintile", "REGION"=expression(bold("REGION")),
p.m2.year$data %>% dplyr::select(term), "BIRTH
YEAR"=expression(bold("BIRTH YEAR")),
p.m2.admin1$data %>% mutate(term=str_to_title(term)) %>%
dplyr::select(term))) + labs(title="", x="Odds of
Coverage") + coord_cartesian(xlim = c(-1, 5)) +
theme_sjplot2() + theme(legend.position = "none",
axis.title.y = element_blank(), axis.line.x =
element_line(color = NA), axis.line.y = element_line(color = NA))
```

Running the above code resulted in the following result showing the odds ratio of stage2 variables:

Pie Chart for

The following section presents the steps carried out to obtain pie chart for explaining proportion of variance of likelihood of vaccination explained by each feature (intent, access, readiness, child sex, maternal education, year and region). We first fit logistic

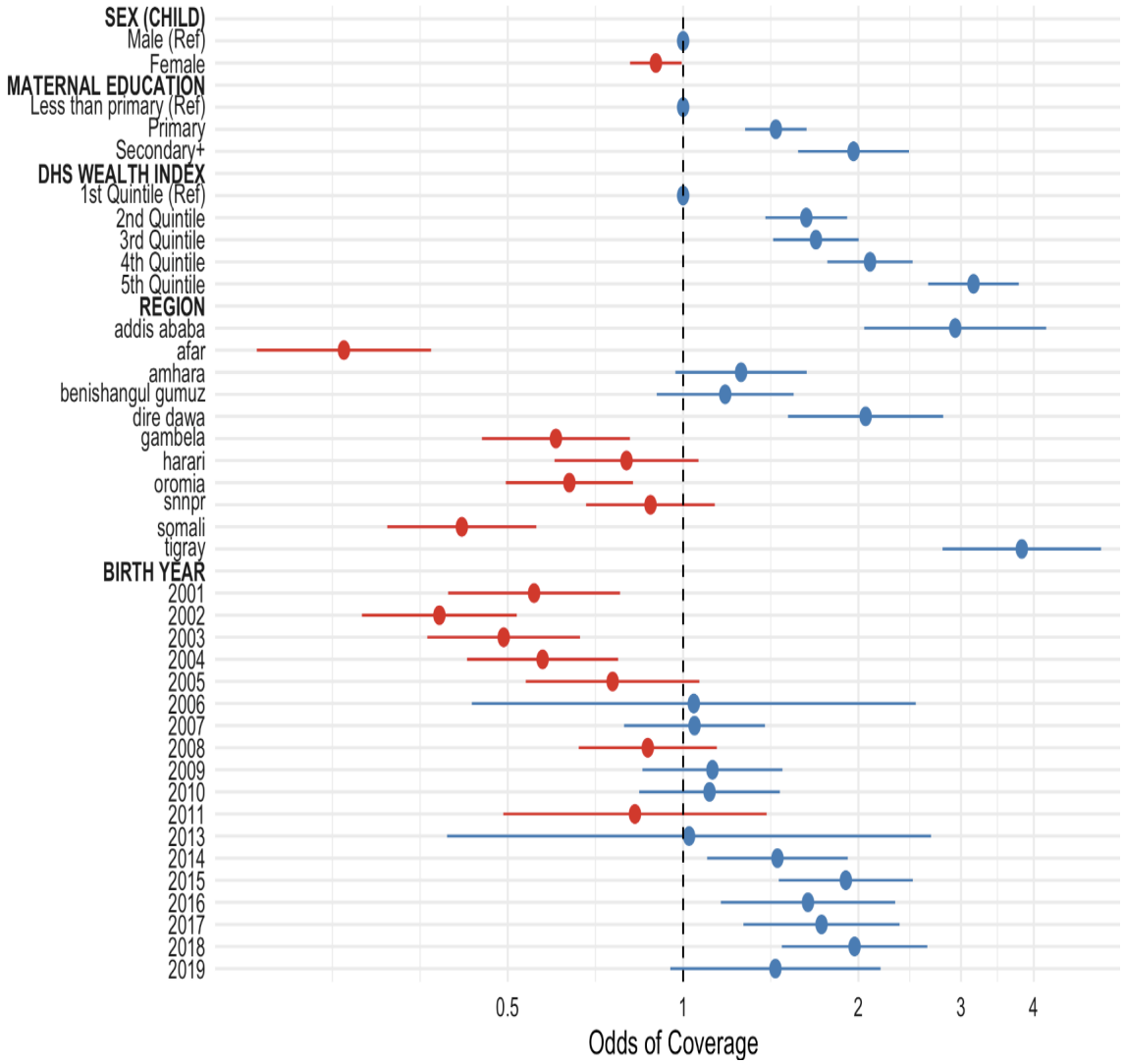

Figure 4.6: The distribution of odds of vaccination by demographic predictors

regression model taking pred1 as the predictor variable and dpt3 as the response variable (0 in the formula is used for excluding an intercept term in the model). We then calculate the pseudo R-squared value for the model to see how well the model fits the data.

```
m1 <- glm(dpt3 ~ 0 + pred1, data=df, family=binomial)
pr1 <- 1 - m1$deviance/m1$null.deviance
```

The following code then creates a vector of covariate (pred1, sex_id, edu_level, year, and admin_1). It then creates a new variable called pred2 which is the mean of the dpt3 variable, minus 1, within each group of the covariates. This line fits the logistic regression model with pred2 as the predictor variable and dpt3 as the response variable. Finally the pseudo R-squared value for the model are calculated by subtracting the null deviance from the deviance and then dividing by the null deviance

```
## Stage 2 - Perfect prediction Pseudo R2
covs <- c("pred1", "sex_id", "edu_level", "year", "admin_1")
df[, pred2 := mean(as.integer(dpt3)-1, na.rm=T), by=covs]
```

```
m2 <- glm(dpt3 ~ 0 + pred2, data=df, family=binomial)
pr2 <- 1 - m2$deviance/m1$null.deviance
```

Then importance of each feature are calculated using the following code:

```
shap1 <- shap_long
shap1.l <- shap1
shap1.l <- shap1.l[variable%in%fit1$best$feature_names]
hap1.l[, group := variable]
shap1.l[, global := mean(abs(value)), by=group]
shap1.l[, var := var(value), by=group]
shap1.agg <- shap1.l[, .(group, global, var)] %>% unique
shap1.agg[, p := var/sum(var)]
```

In the above code:

- The importance of each feature are calculated using Shapley values. The *shap_long* object contains the Shapley values for each feature, and the fit1$best$feature_names vector contains the names of the best features.
- The shap1[variable%in%fit1$best$feature_names], subsets the shap_long object to only include the rows where the variable column is in the fit1$best$feature_names vector.
- shap1.l[, group := variable], creates a new column called group that contains the names of the features.
- shap1.l[, global := mean(abs(value)), by=group], calculates the mean absolute value of the Shapley values for each feature as a measure of the overall importance of the feature.
- shap1.l[, var := var(value), by=group], calculates the variance of the Shapley values for each feature (a measure of how spread out the Shapley values are for each feature)
- shap1.agg <- shap1.l[, .(group, global, var)] %>% unique, creates a new data frame called shap1.agg that contains the group, global, and var columns. The *unique()* function removes any duplicate rows from the data frame.
- shap1.agg[, p := var/sum(var)], calculates the proportion of variance explained by each feature. This is a measure of the relative importance of each feature.
- The output of the code is a data frame that contains the group column (names of the features), global column (mean absolute value of the Shapley values for each feature), var column (variance of the Shapley values for each feature), and p column (proportion of variance explained by each feature).

```
shap1 <- shap_long
shap1.l <- shap1
shap1.l <- shap1.l[variable%in%fit1$best$feature_names]
shap1.l[, group := variable]
shap1.l[, global := mean(abs(value)), by=group]
shap1.l[, var := var(value), by=group]
shap1.agg <- shap1.l[, .(group, global, var)] %>% unique
shap1.agg[, p := var/sum(var)]
```

The product of the proportion of variance explained by each feature and the pseudo R-squared value for the model is calculated and saved as a new column called p_r2 under *shap1.agg* data frame as follows:

```
## As a proportion of pseudo-R2
shap1.agg[, p_r2 := p*pr1]
```

Then the above calculated values are appended on shap1.agg data frame and unexplained variations are also calculated using the code below:

```
# Append on stage 2 and calculate unexplained
shap1.agg <- shap1.agg %>% rbind(
data.table(group="Stage 2", p_r2 =pr2-pr1),
data.table(group="Unexplained", p_r2 = 1 - pr2), fill=T)
shap1.agg[, disp := c("Intent to Vaccinate", "Community
Access", "Facility Readiness", "Additional Factors
(Child's Sex, \nMaternal Edu, Wealth, Region, Year)",
"Unexplained")]
shap1.agg[, disp := factor(disp, levels=c("Intent to
Vaccinate", "Community Access", "Facility Readiness",
"Additional Factors (Child's Sex, \nMaternal Edu, Wealth,
Region, Year)", "Unexplained"), labels=c("Intent to
Vaccinate", "Community Access", "Facility Readiness",
"Additional Factors (Child's Sex, \nMaternal Edu, Wealth,
Region, Year)", "Unexplained"), ordered=T)]
shap1.agg <- shap1.agg[order(disp)]
```

Then the cumulative probability (pcum) is calculated and labels (labs) are added to the above data frame shap1_agg as follows:

```
shap1.agg[, pcum := 1 - (cumsum(p_r2) - 0.5*p_r2)]
shap1.agg[, labs := paste0(round(shap1.agg$p_r2*100, 1), "%")]
```

Then the pie chart showing the variations on likelihood of vaccination explained by each of the predictors is obtained using the below code:

```
## Pie chart
shap1.agg %>% ggplot(aes(y=p_r2,x="", fill=disp))
+  geom_bar(stat="identity", position="stack") +
 coord_polar("y", start=0, clip="off", direction=-1) +
 scale_fill_manual(values=c( "#00416b", "#0098a7",
 "#ff6400","#ffa266", "#0a6bd1",  "grey90")) +
scale_y_continuous(breaks=shap1.agg$pcum, labels=
shap1.agg$labs)+  theme_minimal()+ labs(fill="") +
theme( axis.title.x = element_blank(), axis.title.y =
element_blank(), panel.border = element_blank(),
panel.grid=element_blank(), axis.ticks = element_blank(),
plot.title=element_text(size=14),
axis.text.y=element_text(size=20), legend.position="bottom")
+  guides(fill=guide_legend(nrow=2, byrow=TRUE))
```

The following pie chart is obtained from the above code:

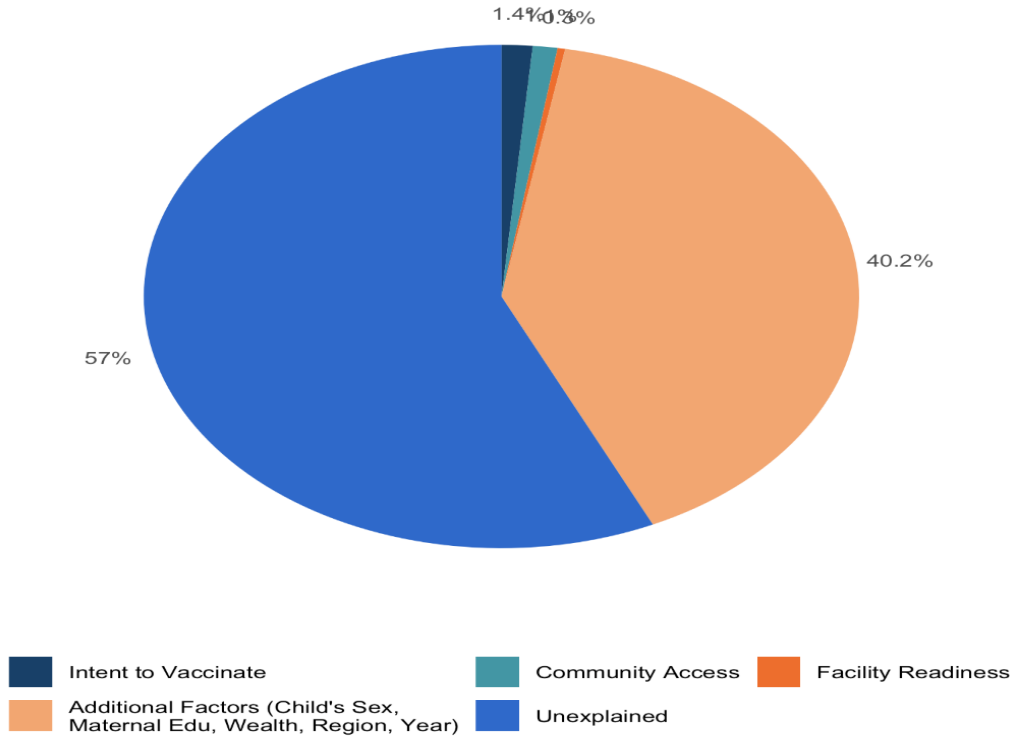

Figure 4.7: Explained variations in the likelihood of vaccination by different predictors

Bibliography

[1] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[2] R. D. C. Team, "R: A language and environment for statistical computing," *(No Title)*, 2010.

[3] S. Van Buuren and C. G. Oudshoorn, "Multivariate imputation by chained equations," 2000.