Exploratory Data Analysis

1 Personal Information

Name: Alexandra de Nooy
Student ID: 14581728
Email: alexandra.de.nooy@student.uva.nl or denooy.alex@gmail.com
Submitted on: 22/03/2023
Github link: https://github.com/adenooy2/MSc-Thesis.git

2 Data Context

This exploratory data analysis is conducted usinf R and RStudio. There are two main sets of data to be considered, linked to the two main sections of the project.

The first set of data represents the baseline or standard of care output runs produced by the existing patient pathway model. For the baseline there are 15 data output files (in csv format). Multiple files had been generated to account for model stochasticity, with the results of each simulation being based on a different set of random probabilities. The data files have a consistent structure and represent the population of indiviudals who move through the TB diagnostic patient pathway. In this, each column represents either a patient disease status or a point in the patient pathway that the indiviudal may or may not have reached.

The second data set consists of TB burden estimates for Kenya produced by the World Heath Organization as well as the accompanying data dictionary. The estimates cover a range of data variables and their estimated values between the years 2000 and 2022. Several key variables include estimates on TB incidence (new cases), notifications (diagnoses) and deaths. Estimates are also provided for different groups of individuals (for example HIV postive patients) and for different tpes of TB.

3 Data Description: Baseline TB model

3.1 Load baseline data

```
# Set path to baseline data
baseline_path = "/Users/adenooy/Library/CloudStorage/OneDrive-Personal/UVA/Thesis/MSc-Thesis/data/static

# Determine how many files
files = list.files(baseline_path)
num_files = length(files)
print(paste("Number of baseline files: ", num_files, sep = ""))
```

```
## [1] "Number of baseline files: 20"
```

3.2 Explore variables and format of one baseline file

Each baseline data file represents a population of individuals (one per row) and various columns representing the different states or points in the patient pathway reached by each individual.

3.2.1 File Structure

```
b_data = read_excel(paste(baseline_path, files[1], sep = ""))

print(paste("Each data file consists of ", dim(b_data)[1],
    " rows and ", dim(b_data)[2], " columns", sep = ""))
```

```
## [1] "Each data file consists of 10000 rows and 35 columns"
```

```
# List of column names
colnames(b_data)
```

```
##  [1] "hiv"                          "rnum"
##  [3] "tb_status"                    "tb_present"
##  [5] "rif_status"                   "num_visits"
##  [7] "patient_time"                 "tb_seek_care"
##  [9] "do_triage"                    "tb_screened"
## [11] "sens_screen"                  "spec_screen"
## [13] "screen_result"                "do_confirmatory"
## [15] "tb_triaged"                   "sens_triage"
## [17] "spec_triage"                  "tb_triage_result"
## [19] "tb_confirmatory_offered"      "patient_referred_for_sample"
## [21] "patient_reached_sample_site"  "conf_test"
## [23] "spec_conf"                    "sens_conf"
## [25] "rif_sens"                     "rif_spec"
## [27] "conf_sample_provided"         "conf_initial_sample_provided"
## [29] "conf_sample_status"           "conf_sample_tested"
## [31] "conf_sample_referred"         "conf_sample_result"
## [33] "patient_conf_result_received" "conf_res_same_encounter"
## [35] "emp_notification"
```

```
print(b_data[1:3, ])
```

```
## # A tibble: 3 x 35
##     hiv  rnum tb_sta~1 tb_pr~2 rif_s~3 num_v~4 patie~5 tb_se~6 do_tr~7
##   <dbl> <dbl> <chr>      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     1 0.705 eptb           1       0       2       5       1       1
## 2     1 0.394 eptb           1       0       3       8       1       1
## 3     1 0.968 eptb           1       0       3       8       1       1
## # ... with 26 more variables: tb_screened <dbl>, sens_screen <dbl>,
## #   spec_screen <dbl>, screen_result <lgl>, do_confirmatory <dbl>,
## #   tb_triaged <dbl>, sens_triage <dbl>, spec_triage <dbl>,
## #   tb_triage_result <lgl>, tb_confirmatory_offered <dbl>,
## #   patient_referred_for_sample <dbl>,
## #   patient_reached_sample_site <dbl>, conf_test <chr>,
## #   spec_conf <dbl>, sens_conf <dbl>, rif_sens <dbl>, ...
```

```r
# Type of each column
sapply(b_data, class)
```

```
##                          hiv                         rnum 
##                    "numeric"                    "numeric" 
##                    tb_status                   tb_present 
##                  "character"                    "numeric" 
##                   rif_status                   num_visits 
##                    "numeric"                    "numeric" 
##                 patient_time                 tb_seek_care 
##                    "numeric"                    "numeric" 
##                    do_triage                  tb_screened 
##                    "numeric"                    "numeric" 
##                  sens_screen                  spec_screen 
##                    "numeric"                    "numeric" 
##                screen_result              do_confirmatory 
##                    "logical"                    "numeric" 
##                   tb_triaged                  sens_triage 
##                    "numeric"                    "numeric" 
##                  spec_triage             tb_triage_result 
##                    "numeric"                    "logical" 
##      tb_confirmatory_offered  patient_referred_for_sample 
##                    "numeric"                    "numeric" 
##  patient_reached_sample_site                    conf_test 
##                    "numeric"                  "character" 
##                    spec_conf                    sens_conf 
##                    "numeric"                    "numeric" 
##                     rif_sens                     rif_spec 
##                    "numeric"                    "numeric" 
##         conf_sample_provided conf_initial_sample_provided 
##                    "numeric"                    "numeric" 
##           conf_sample_status           conf_sample_tested 
##                    "numeric"                    "numeric" 
##         conf_sample_referred           conf_sample_result 
##                    "numeric"                    "numeric" 
## patient_conf_result_received      conf_res_same_encounter 
##                    "numeric"                    "numeric" 
##             emp_notification 
##                    "numeric"
```

Notably there are many columns in the dataframe listed as numeric, but only have values 0-1

3.2.2 Missing Data

The code below highlights only a few columns with missing data. These columns all relate to the result of a TB test (screen_result, tb_triage_result, conf_sample_result, patient_conf_result_received). In the design of the baseline model, these missing values are not as a result of data being incorrcetly recorded or collected, but rather represent a status themselves - that is, no result was availble at that point in the patient pathway

```r
# count the missing values by column wise
print("Count of missing values by column wise")
```

```
## [1] "Count of missing values by column wise"
```

```
sapply(b_data, function(x) sum(is.na(x)))
```

```
##                          hiv                         rnum
##                            0                            0
##                    tb_status                   tb_present
##                            0                            0
##                   rif_status                   num_visits
##                            0                            0
##                 patient_time                 tb_seek_care
##                            0                            0
##                    do_triage                  tb_screened
##                            0                            0
##                  sens_screen                  spec_screen
##                            0                            0
##                screen_result              do_confirmatory
##                        10000                            0
##                   tb_triaged                  sens_triage
##                            0                            0
##                  spec_triage             tb_triage_result
##                            0                        10000
##      tb_confirmatory_offered  patient_referred_for_sample
##                            0                            0
##  patient_reached_sample_site                    conf_test
##                            0                            0
##                    spec_conf                    sens_conf
##                            0                            0
##                     rif_sens                     rif_spec
##                            0                            0
##         conf_sample_provided conf_initial_sample_provided
##                            0                            0
##           conf_sample_status           conf_sample_tested
##                            0                            0
##         conf_sample_referred           conf_sample_result
##                            0                         2330
## patient_conf_result_received      conf_res_same_encounter
##                         2945                            0
##             emp_notification
##                            0
```

3.2.3 HIV and TB summary

```
# Group by hiv and TB status and count
hiv_tb_counts = b_data %>%
    group_by(tb_status, hiv) %>%
    count()
hiv_tb_counts$hiv_status = "hiv_pos"
hiv_tb_counts$hiv_status[hiv_tb_counts$hiv == 0] = "hiv_neg"
hiv_tb_counts$hiv = NULL
hiv_tb_counts = hiv_tb_counts %>%
    spread(hiv_status, n)

print(hiv_tb_counts)
```

```
## # A tibble: 3 x 3
## # Groups:   tb_status [3]
##   tb_status   hiv_neg hiv_pos
##   <chr>         <int>   <int>
## 1 eptb             77      24
## 2 ptb             741     213
## 3 tb_negative    8376     569
```

```
# Summarise counts
print(paste("Total HIV positive: ", sum(hiv_tb_counts$hiv_pos),
    sep = ""))
```

```
## [1] "Total HIV positive: 806"
```

```
print(paste("Total HIV Negative: ", sum(hiv_tb_counts$hiv_neg),
    sep = ""))
```

```
## [1] "Total HIV Negative: 9194"
```

```
print(paste("Total EPTB: ", hiv_tb_counts$hiv_neg[hiv_tb_counts$tb_status ==
    "eptb"] + hiv_tb_counts$hiv_pos[hiv_tb_counts$tb_status ==
    "eptb"], sep = ""))
```

```
## [1] "Total EPTB: 101"
```

```
print(paste("Total PTB: ", hiv_tb_counts$hiv_neg[hiv_tb_counts$tb_status ==
    "ptb"] + hiv_tb_counts$hiv_pos[hiv_tb_counts$tb_status ==
    "ptb"], sep = ""))
```

```
## [1] "Total PTB: 954"
```

```
print(paste("Total TB positive: ", hiv_tb_counts$hiv_neg[hiv_tb_counts$tb_status ==
    "eptb"] + hiv_tb_counts$hiv_pos[hiv_tb_counts$tb_status ==
    "eptb"] + hiv_tb_counts$hiv_neg[hiv_tb_counts$tb_status ==
    "ptb"] + hiv_tb_counts$hiv_pos[hiv_tb_counts$tb_status ==
    "ptb"], sep = ""))
```

```
## [1] "Total TB positive: 1055"
```

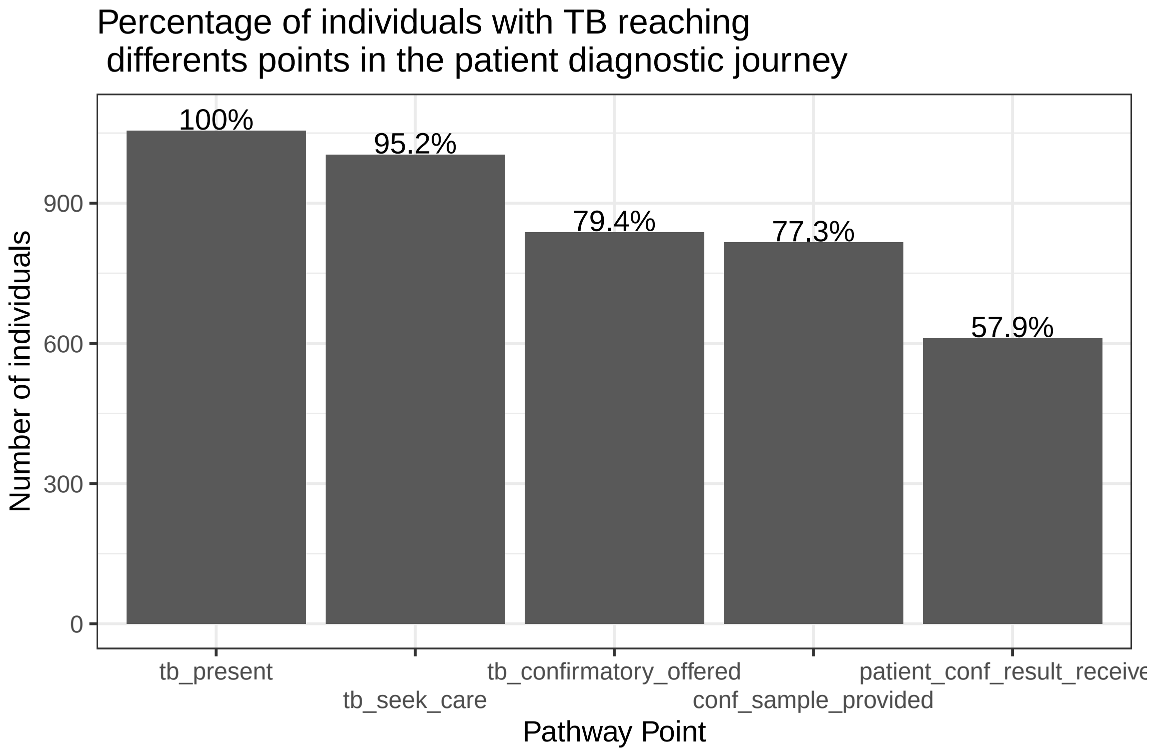
```
print(paste("Total TB Negative: ", hiv_tb_counts$hiv_neg[hiv_tb_counts$tb_status ==
    "tb_negative"] + hiv_tb_counts$hiv_pos[hiv_tb_counts$tb_status ==
    "tb_negative"], sep = ""))
```

```
## [1] "Total TB Negative: 8945"
```

3.2.4 TB patient pathway summary

Given the format of the model, it is useful to summarise the number of individuals reaching key points in the patient pathway. An initial analysis is conducted only for those people with TB. Based on descriptions of the model, key variables representing some some of these points are: tb_present, tb_seek_care,tb_confirmatory_offered,conf_sample_provided,patient_conf_result_received. Summarising these variables will provide an indiciatio of how many individuals with TB reach each point in the pathway and are diagnoses corrcetly.

```
# Filter to only those with TB
b_data_tb_path = b_data %>%
    filter(tb_present == 1) %>%
    select(tb_present, tb_seek_care, tb_confirmatory_offered,
        conf_sample_provided, , patient_conf_result_received)

# Count pathway points
tb_path_counts = data.frame(colSums(b_data_tb_path, na.rm = TRUE))
colnames(tb_path_counts)[1] = "count"
tb_path_counts$variable = rownames(tb_path_counts)

# Determine percentages
tb_path_counts$perc = round(tb_path_counts$count/max(tb_path_counts$count) *
    100, 1)

# Plot tB cascade

ggplot(tb_path_counts, aes(x = reorder(variable, -count),
    y = count)) + geom_bar(stat = "identity") + theme_bw() +
    xlab("Pathway Point") + ylab("Number of individuals") +
    labs(title = "Percentage of individuals with TB reaching \n differents points in the patient diagnos
    geom_text(aes(x = reorder(variable, -count), y = count +
        25, label = paste(perc, "%", sep = ""))) + scale_x_discrete(guide = guide_axis(n.dodge = 2))
```



4 Data Description: WHO Tuberculosis Data

4.1 Load and merge WHO TB burden data and data dictionary

```
# Path to directory
basePath = "/Users/adenooy/Library/CloudStorage/OneDrive-Personal/UVA/Thesis/MSc-Thesis/"

# Load data dictionary
datadict = read.csv(paste(basePath, "data/dynamic/TB_data_dictionary_2024-01-30.csv",
    sep = ""))
colnames(datadict)
```

```
## [1] "variable_name" "dataset"       "code_list"     "definition"
```

```
print(datadict[1:3, ])
```

```
##     variable_name dataset code_list
## 1 budget_cpp_dstb  Budget
## 2  budget_cpp_mdr  Budget
## 3  budget_cpp_tpt  Budget
##
## 1 Average cost of drugs budgeted per patient for drug-susceptible TB treatment, excluding buffer sto
## 2              Average cost of drugs budgeted per patient for MDR-TB treatment, excluding buffer sto
## 3      Average cost of drugs budgeted per patient for  TB preventive treatment, excluding buffer sto
```

```
# Load TB data
tb_estimates = read_excel(paste(basePath, "data/dynamic/kenya_tb_burden.xlsx",
    sep = ""))
colnames(tb_estimates)
```

```
##  [1] "country"                  "iso2"
##  [3] "iso3"                     "iso_numeric"
##  [5] "g_whoregion"              "year"
##  [7] "e_pop_num"                "e_inc_100k"
##  [9] "e_inc_100k_lo"            "e_inc_100k_hi"
## [11] "e_inc_num"                "e_inc_num_lo"
## [13] "e_inc_num_hi"             "e_tbhiv_prct"
## [15] "e_tbhiv_prct_lo"          "e_tbhiv_prct_hi"
## [17] "e_inc_tbhiv_100k"         "e_inc_tbhiv_100k_lo"
## [19] "e_inc_tbhiv_100k_hi"      "e_inc_tbhiv_num"
## [21] "e_inc_tbhiv_num_lo"       "e_inc_tbhiv_num_hi"
## [23] "e_mort_exc_tbhiv_100k"    "e_mort_exc_tbhiv_100k_lo"
## [25] "e_mort_exc_tbhiv_100k_hi" "e_mort_exc_tbhiv_num"
## [27] "e_mort_exc_tbhiv_num_lo"  "e_mort_exc_tbhiv_num_hi"
## [29] "e_mort_tbhiv_100k"        "e_mort_tbhiv_100k_lo"
## [31] "e_mort_tbhiv_100k_hi"     "e_mort_tbhiv_num"
## [33] "e_mort_tbhiv_num_lo"      "e_mort_tbhiv_num_hi"
## [35] "e_mort_100k"              "e_mort_100k_lo"
## [37] "e_mort_100k_hi"           "e_mort_num"
## [39] "e_mort_num_lo"            "e_mort_num_hi"
## [41] "cfr"                      "cfr_lo"
## [43] "cfr_hi"                   "cfr_pct"
## [45] "cfr_pct_lo"               "cfr_pct_hi"
## [47] "c_newinc_100k"            "c_cdr"
## [49] "c_cdr_lo"                 "c_cdr_hi"
```

```
print(tb_estimates[1:3, ])
```

```
## # A tibble: 3 x 50
##   country iso2  iso3  iso_nume~1 g_who~2  year e_pop~3 e_inc~4 e_inc~5
##   <chr>   <chr> <chr>      <dbl> <chr>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 Kenya   KE    KEN          404 AFR      2000  3.09e7     451     182
## 2 Kenya   KE    KEN          404 AFR      2001  3.18e7     499     178
## 3 Kenya   KE    KEN          404 AFR      2002  3.28e7     534     174
## # ... with 41 more variables: e_inc_100k_hi <dbl>, e_inc_num <dbl>,
## #   e_inc_num_lo <dbl>, e_inc_num_hi <dbl>, e_tbhiv_prct <dbl>,
## #   e_tbhiv_prct_lo <dbl>, e_tbhiv_prct_hi <dbl>,
## #   e_inc_tbhiv_100k <dbl>, e_inc_tbhiv_100k_lo <dbl>,
## #   e_inc_tbhiv_100k_hi <dbl>, e_inc_tbhiv_num <dbl>,
## #   e_inc_tbhiv_num_lo <dbl>, e_inc_tbhiv_num_hi <dbl>,
## #   e_mort_exc_tbhiv_100k <dbl>, e_mort_exc_tbhiv_100k_lo <dbl>, ...
```

```
# Merge tb data with data dictionary
tbData = tb_estimates %>%
    gather("variable_name", "value", 7:50) %>%
    left_join(datadict)
```

```
## Joining, by = "variable_name"
```

```r
# remove unnecessary regional columns, blank code_list
# column
tbData = subset(tbData, select = -c(iso2, iso3, iso_numeric,
    g_whoregion, code_list))
print(tbData[1:5, ])
```

```
## # A tibble: 5 x 6
##   country  year variable_name    value dataset   definition
##   <chr>   <dbl> <chr>            <dbl> <chr>     <chr>
## 1 Kenya    2000 e_pop_num     30851606 Estimates Estimated total popu~
## 2 Kenya    2001 e_pop_num     31800343 Estimates Estimated total popu~
## 3 Kenya    2002 e_pop_num     32779823 Estimates Estimated total popu~
## 4 Kenya    2003 e_pop_num     33767122 Estimates Estimated total popu~
## 5 Kenya    2004 e_pop_num     34791836 Estimates Estimated total popu~
```

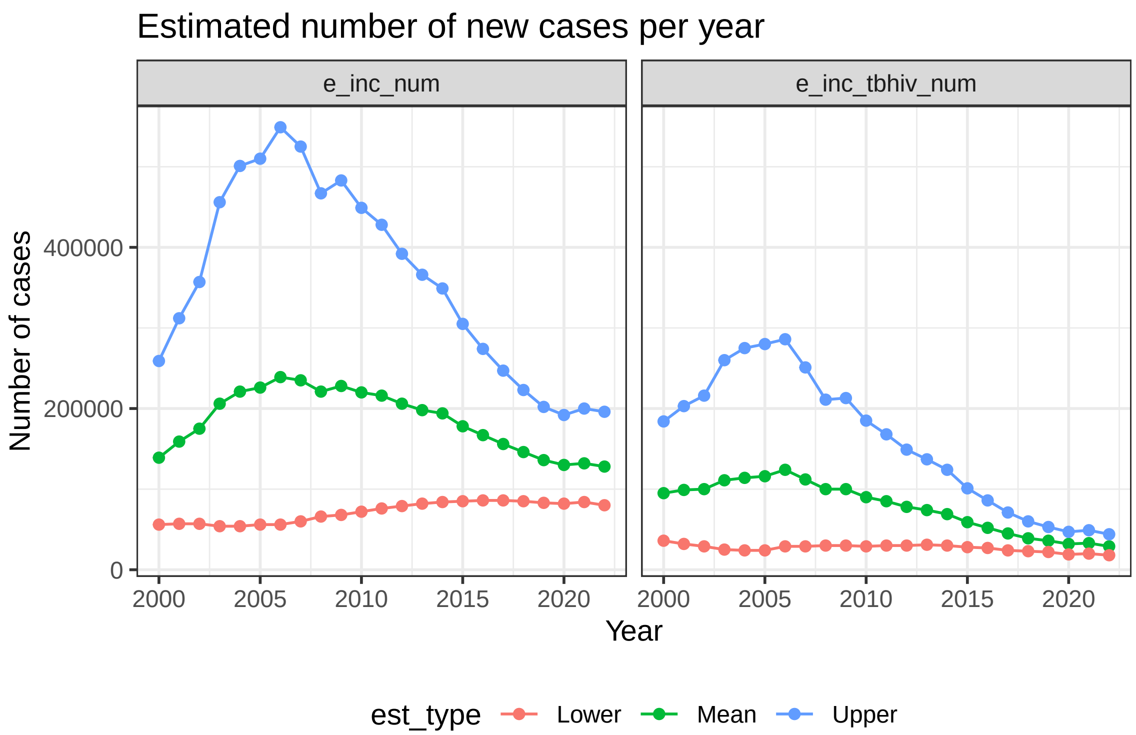
4.2 Exploring new incident infections (all infections and HIV)

Incident infections are the number of estimated people being infected with and acquiring active TB each year. The number of new infections is an estimate and is different from the number of reported cases or diagnoses - which is reliant on the identification, testing and treating of people with TB. This data represents a key element in the transmisison model and it is important in understanding the past dynamics of TB in kenya and provides an idea on the current trend.

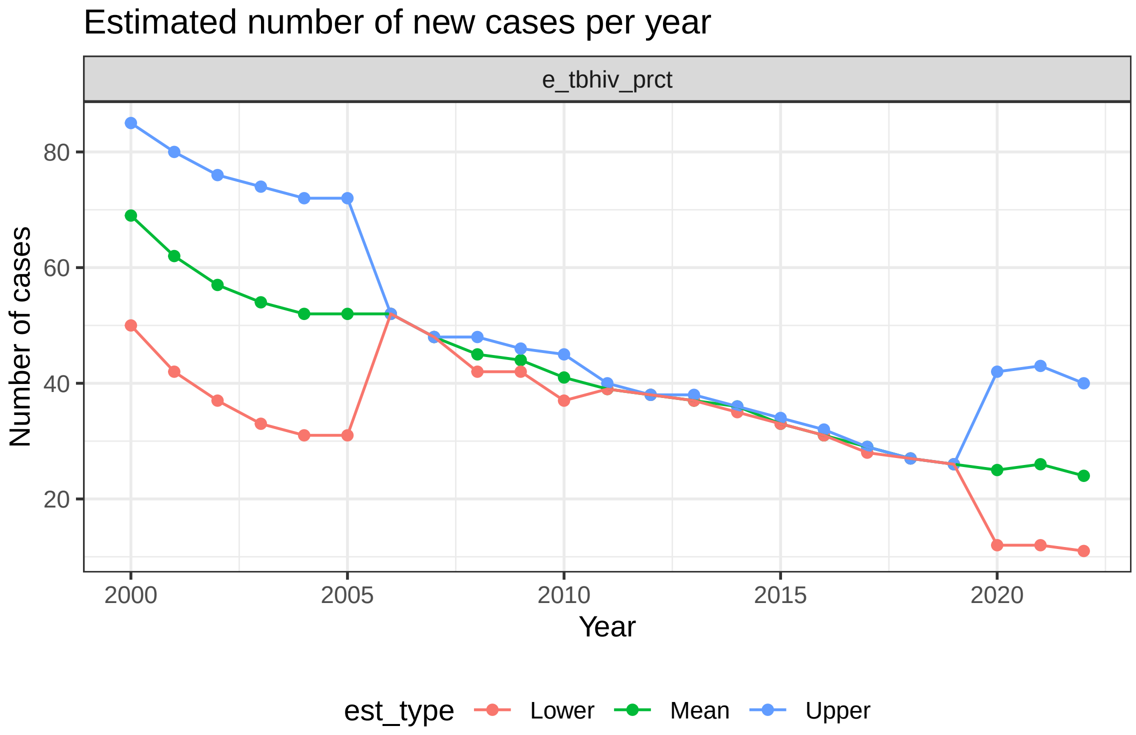
HIV is an important factor to consider, given that Kenya has relatively high HIV/TB coinfection and because HIV impacts the likelihood of contracting TB, becoming infectious or of becoming severely ill.

```r
# select relevant variables related to incidence
inc_data = tbData %>%
    filter(variable_name %in% c("e_inc_num", "e_inc_num_lo",
        "e_inc_num_hi")) %>%
    mutate(var = "e_inc_num")
hiv_inc = tbData %>%
    filter(variable_name %in% c("e_inc_tbhiv_num", "e_inc_tbhiv_num_lo",
        "e_inc_tbhiv_num_hi")) %>%
    mutate(var = "e_inc_tbhiv_num")
hiv_perc_inc = tbData %>%
    filter(variable_name %in% c("e_tbhiv_prct", "e_tbhiv_prct_lo",
        "e_tbhiv_prct_hi")) %>%
    mutate(var = "e_tbhiv_prct")

# label upper, lower and mean estimates
all_inc = rbind(inc_data, hiv_inc)
all_inc$est_type = "Mean"
all_inc$est_type[grepl("_lo", all_inc$variable_name, fixed = TRUE) ==
    TRUE] = "Lower"
all_inc$est_type[grepl("_hi", all_inc$variable_name, fixed = TRUE) ==
    TRUE] = "Upper"

hiv_perc_inc$est_type = "Mean"
hiv_perc_inc$est_type[grepl("_lo", hiv_perc_inc$variable_name,
    fixed = TRUE) == TRUE] = "Lower"
hiv_perc_inc$est_type[grepl("_hi", hiv_perc_inc$variable_name,
    fixed = TRUE) == TRUE] = "Upper"
```

```
# Incident cases (all and HIV)
ggplot(all_inc, aes(x = year, y = value, group = variable_name,
    color = est_type)) + geom_point() + geom_line() + theme_bw() +
    xlab("Year") + ylab("Number of cases") + labs(title = "Estimated number of new cases per year") +
    theme(legend.position = "bottom") + facet_wrap(. ~ var)
```



```
# Percentage of new cases HIV positive
ggplot(hiv_perc_inc, aes(x = year, y = value, group = variable_name,
    color = est_type)) + geom_point() + geom_line() + theme_bw() +
    xlab("Year") + ylab("Number of cases") + labs(title = "Estimated number of new cases per year") +
    theme(legend.position = "bottom") + facet_wrap(. ~ var)
```



4.3 Exploring mean estimates of key factors - population, incidence, case detection, mortality (for whole population)

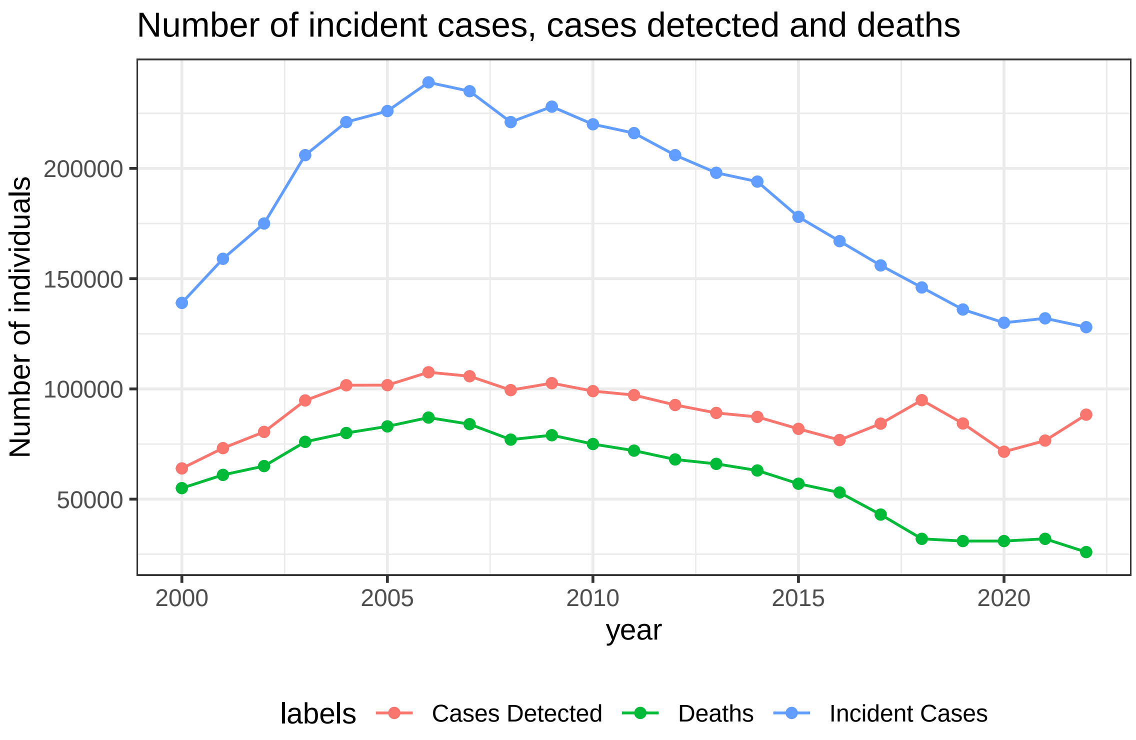
Key definitions from WHO indicator metadata registry and estimate methedology appendix
- Case Detection rate (%) : Proportion of estimated new and relapse TB (incident) cases diagnosed in a year
-Number of deaths: Product of incidence and case fatality rate
-Case fatality rate: risk of death among people with active (incident) TB, adapte dto account fro low covergae/reporting

```
# Collect key factors relevant to understanding the
# dynamics of TB transmission
key_fact = tbData %>%
    filter(variable_name %in% c("e_pop_num", "e_inc_num",
        "e_inc_rr_num", "e_mort_num", "c_cdr", "cfr", "cfr_pct")) %>%
    select(variable_name, year, value) %>%
    spread(variable_name, value)


# Conduct additional caluclations fo rates or use
# rates to estimate numbers
key_fact$calc_cfr_inc = 100 * (key_fact$e_mort_num/key_fact$e_inc_num)
key_fact$calc_case_detect = key_fact$c_cdr/100 * key_fact$e_inc_num
key_fact$calc_cfr_case = 100 * (key_fact$e_mort_num/key_fact$calc_case_detect)
key_fact$deaths_cal = key_fact$cfr * key_fact$e_inc_num

# Add user friendly labels
```

```
variable = c("e_pop_num", "e_inc_num", "e_inc_rr_num", "e_mort_num",
    "c_cdr", "cfr", "cfr_pct", "calc_case_detect")
labels = c("Population", "Incident Cases", "Incident RR Cases",
    "Deaths", "Case Detection Rate", "Case fatality Rate",
    "Case Fatality Rate (%)", "Cases Detected")
labs = data.frame(cbind(variable, labels))

# Plot key factors with absolute numbers (other than
# population)
key_fact_num = key_fact %>%
    select(year, e_inc_num, calc_case_detect, e_mort_num) %>%
    gather("variable", "value", 2:4) %>%
    left_join(labs)
```

```
## Joining, by = "variable"
```

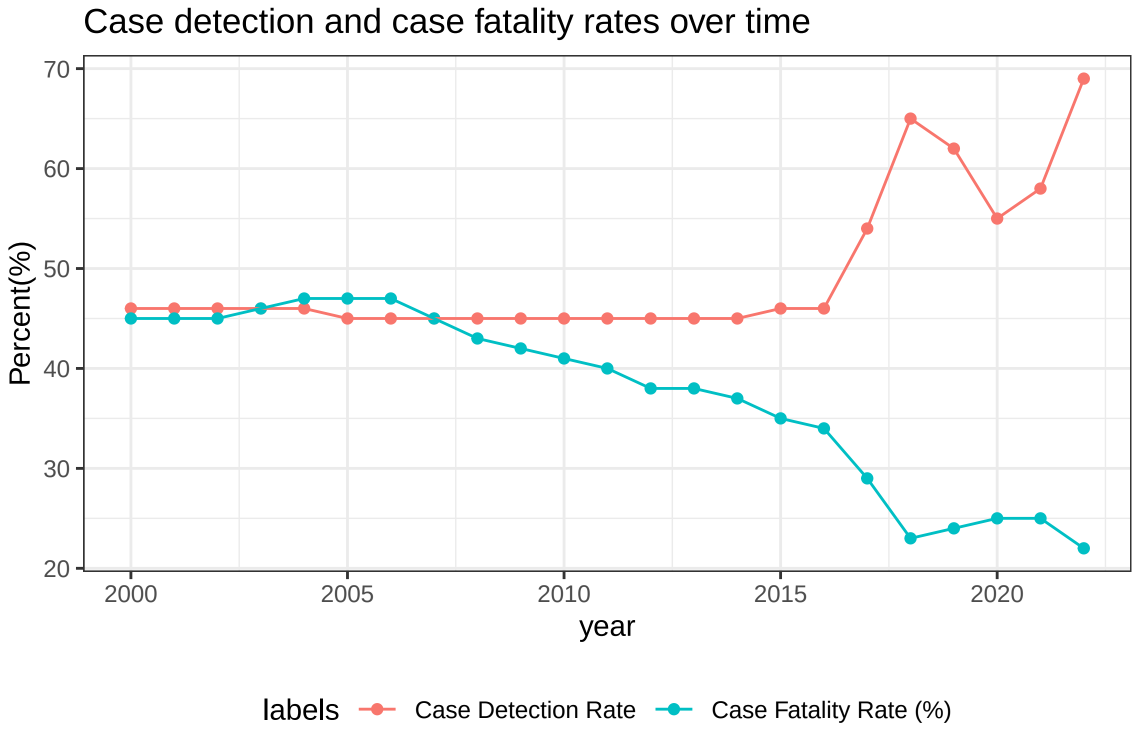
```
ggplot(key_fact_num, aes(x = year, y = value, group = variable,
    colour = labels)) + geom_point() + geom_line() + theme_bw() +
    labs(title = "Number of incident cases, cases detected and deaths") +
    ylab("Number of individuals") + theme(legend.position = "bottom")
```



```
# Plot key factors which are rates
key_fact_rate = key_fact %>%
    select(year, c_cdr, cfr_pct) %>%
    gather("variable", "value", 2:3) %>%
    left_join(labs)
```

```
## Joining, by = "variable"
```

```
ggplot(key_fact_rate, aes(x = year, y = value, group = variable,
    colour = labels)) + geom_point() + geom_line() + theme_bw() +
    labs(title = "Case detection and case fatality rates over time") +
    ylab("Percent(%)") + theme(legend.position = "bottom")
```



4.4 Set up calibration data for transmission model

The transmission model will be calibrated against data from this set. In this case incidence data over time will be the key comparator, however, later it will be useful to compare other variables - like cases detected.

```
# Collect incidence data in correct format - key
# variables and wide format
cal_data = all_inc %>%
    select(year, var, est_type, value) %>%
    spread(est_type, value) %>%
    filter(var == "e_inc_num")

# Save data
write.csv(cal_data, "/Users/adenooy/Library/CloudStorage/OneDrive-Personal/UVA/Thesis/MSc-Thesis/data/d
```