World Bank Region Analysis

Sun, 8th Nov 2020

Different teams were tasked with exploring World Bank data consisting of 41 indicators on 217 countries. This report will focus on the Arabian Peninsula region of the dataset, consisting of 15 countries.

> by: *Estrella Spaans, Jack Daoud, Sara Mareike Krause*

The Arabian Peninsula

الْجَزِيرَةُ الْعَرَبِيَّة This region is the largest peninsula globally, with more than 80% of its border touching water. In fact, its connected to four different seas.

Arabic is the dominant language of the region and is the 2nd most difficult language to learn in the world: letters have four different forms depending on their position in a word while there are almost nine different dialects.

Arabia has a handful of aspects that makes it world-renowned. For example, Islam was born in Mecca in Saudi Arabia and it the governing religion of the majority of Arabian Peninsula countries. Furthermore, *Byblos*, one of the oldest cities in the world listed as a UNESCO World Heritage Site in Lebanon.

The Israeli-Palestinian Conflict in the Gaza Strip and West Bank has been going on for centuries and stems from religion. The recent problems revolve around Israel's interference, pushing for the solution to become a two-state region enforcing severe and discriminatory restrictions on Palestinians' human rights.

Both, Syria and Yemen currently experience civil war. A continuous fight against ISIS's activities, a terroristic Islamic activation group and previous wars still affect different indicators in the region.

Figure 1.1 Arabian Peninsula Countries

---

1. Data Preparation

1. **all_regions** (The data with all the countries and indicators)
2. **arabian_peninsula** (Arabian Peninsula country indicator data)
3. **meta_series_indicators** (The meta data that describes more about the indicators)
4. **meta_country** (Additional data about the countries)

In [ ]:
```
## Data Preperation

##############################################################################
## 1. IMPORTANT PACKAGES NEEDED ##
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

##############################################################################
## 2. SETTING DISPLAY CONDITIONS ##

# Display all columns of datasets
pd.set_option('display.max_columns', None)

# Display all floats in 2 decimal spaces
pd.options.display.float_format = "{:,.2f}".format

##############################################################################
## 3. IMPORTING FILE AND CREATE DATAFRAMES ##

# Creating variable for filepath.
file = "./_datasets/original.xlsx"

#Creating a DataFrame for our meta series sheet
meta_series = pd.read_excel(io         = file,
                            sheet_name = "Series - Metadata",
                            header     = 0)

# Creating a DataFrame for region data sheet
All_Regions = pd.read_excel(io         = file,
                            sheet_name = "Data",
                            header     = 0)

# Creating a DataFrame for our meta country sheet
meta_country = pd.read_excel(io          = file,
                             sheet_name = "Country - Metadata",
                             header     = 0)

##############################################################################
## 4. CHECKING COLUMN NAMES BETWEEN META_SERIES & ALL_REGIONS #

# Composing the columns names to rows from all_regions data.
transposted_all_regions = All_Regions.transpose()

# Reset the index in order to make the index a column.
all_columns_all_regions = transposted_all_regions.reset_index()
columns_all_regions = all_columns_all_regions.loc[ :,  ['index']]
columns_all_regions = columns_all_regions[(columns_all_regions['index'] != 'Country Code Total') & (columns_all_regions['index'] != 'Country Name') & (columns_all_regions['index'] != 'Hult Region')& (columns_all_regions['index'] != 'Cool Name')]

# Checking which ones are in the columns are matching in both DataFrames
columns_all_regions['index'].isin(meta_series['Indicator Name'])

# Assign boolean outcomes of matches as a new column.
columns_all_regions['common_columns_meta_series'] = columns_all_regions['index'].isin(meta_series['Indicator Name'])

# Checking which columns names are missing (remove # to run code)
#columns_all_regions[['index','common_columns_meta_series']].sort_values('common_columns_meta_series')

##############################################################################
## 5. ADJUSTING COLUMN NAMES META_SERIES TO MATCH ALL_REGIONS ##

# Show meta_series Data Frame to look up the right column_names.
meta_series

# Creating a dictionary to change column names in meta_series.
new_values_meta_series = {"Self-employed, total (% of total employment) (modeled ILO estimate)":"Self-employed, total (% of total employment)",
"Malaria cases reported":"Reported cases of malaria",
"GDP per unit of energy use (constant 2017 PPP $ per kg of oil equivalent)":"GDP per unit of energy use (constant 2011 PPP $ per kg of oil equivalent)",
"Energy use (kg of oil equivalent) per $1,000 GDP (constant 2017 PPP)":"Energy use (kg of oil equivalent) per $1,000 GDP (constant 2011 PPP)",
"Contributing family workers, female (% of female employment) (modeled ILO estimate)":"Contributing family workers, female (% of female employment)",
"Contributing family workers, male (% of male employment) (modeled ILO estimate)":"Contributing family workers, male (% of male employment)",
"Contributing family workers, total (% of total employment) (modeled ILO estimate)":"Contributing family workers, total (% of total employment)",
"Tuberculosis death rate (per 100,000 people)":"Tuberculosis death rate (per 100,000 people), including HIV",
"GDP per person employed (constant 2017 PPP $)":"GDP per person employed (constant 2011 PPP $)"}

# Updating meta_series with the right descriptions.
updated_meta_series = meta_series.copy()
updated_meta_series = updated_meta_series.replace({"Indicator Name": new_values_meta_series})

# Assign boolean outcomes of matches as a new column(remove # to run code)
#columns_all_regions['common_columns_meta_series'] = columns_all_regions['index'].isin(updated_meta_series['Indicator Name'])

# Checking if all columns names are the same (remove # to run code)
#columns_all_regions[['index','common_columns_meta_series']].sort_values('common_columns_meta_series')

##############################################################################
## 6. SUBSET APPLICABLE META SERIES COLUMNS AND ADD NEW ONES##

# Subsetting the indicators names.
indicators = columns_all_regions.loc[ :,  ['index',]]

# Renaming the column.
indicators.columns = ['Indicator']

# Only keep rows that match based on an inner-join between indicaters and updated_meta desceiptio.
meta_series_indicators = indicators.merge(updated_meta_series, how = 'inner', left_on='Indicator', right_on='Indicator Name').copy()

# Dropping columns that are not needed.
meta_series_indicators = meta_series_indicators.drop(columns = ["Indicator", "Topic", "Code", "Short definition","License Type", "Dataset", "Unit of measure", "Base Period", "General comments","Notes from original source","License URL"], axis = 1)

# Adding a new column for the renamed_column in all_regions.
    #setting the names for the missing column with a list
column_name_missing = ["AIDS deaths",
                "Adj School Enrollment",
                "Adolescent Fertility",
                "ART Coverage",
                "Delivery Care",
                "CO2 Emissions",
                "Family Workers (f)",
                "Family Workers (m)",
                "Family Workers (t)",
                "Employment Ratio (f)",
                "Employment Ratio (m)",
                "Employment Ratio (t)",
                "Energy Usage",
                "Fertility",
                "GDP per Capita",
                "GDP per Energy",
                "GNI per Capita",
                "Measles Immunization",
                "Sanitation Quality",
                "Water Quality",
                "Tuberculosis Cases",
                "BoP Income Share",
                "Internet Users",
                "Life Expectancy",
                "Literacy Rate",
                "Maternal Mortality",
                "Cellular Subscriptions",
                "Infant Mortality",
                "ODA per Capita",
                "Population",
                "Poverty Line Gap",
                "Prenatal Care",
                "HIV Cases",
                "Undernourishment",
                "School Completion",
                "Parliament Seats (f)",
                "Malaria Cases",
                "School Enrollment",
                "Self-Employment",
                "International Trade",
                "Tuberculosis Mortality"]

    #Add a new column at the beginning of the dataframe with the list above
meta_series_indicators.insert(loc=0, column='Column_Name', value= column_name_missing)

column_category = ["health",
                "education",
                "health",
                "health",
                "health",
                "environment",
                "employment",
                "employment",
                "employment",
                "employment",
                "employment",
                "employment",
                "environment",
                "health",
                "economic",
                "economic",
                "economic",
                "health",
                "environment",
                "environment",
                "health",
                "economic",
                "connectivity",
                "health",
                "education",
                "health",
                "connectivity",
                "health",
                "economic",
                "general",
                "economic",
                "health",
                "health",
                "health",
                "education",
                "economic",
                "health",
                "education",
                "employment",
                "economic",
                "health"]

   #Add a new column at the beginning of the dataframe with the list above
meta_series_indicators.insert(loc=3, column='Category', value= column_category)

##############################################################################
## 7. PREPARING ALL_REGIONS DATAFRAME ##
# Output shows which the data types before and after adjustments
# Drop Cool Name field.
All_Regions = All_Regions.drop(["Cool Name"], axis = 1)

#Create list for shorter column names.
column_names = ["Country Code",
                "Country",
                "Region",
                "AIDS deaths",
                "Adj School Enrollment",
                "Adolescent Fertility",
                "ART Coverage",
                "Delivery Care",
                "CO2 Emissions",
                "Family Workers (f)",
                "Family Workers (m)",
                "Family Workers (t)",
                "Employment Ratio (f)",
                "Employment Ratio (m)",
                "Employment Ratio (t)",
                "Energy Usage",
                "Fertility",
                "GDP per Person Employed",
                "GDP per Energy",
                "GNI per Capita",
                "Measles Immunization",
                "Sanitation Quality",
                "Water Quality",
                "Tuberculosis Cases",
                "BoP Income Share",
                "Internet Users",
                "Life Expectancy",
                "Literacy Rate",
                "Maternal Mortality",
                "Cellular Subscriptions",
                "Infant Mortality",
                "ODA per Capita",
                "Population",
                "Poverty Line Gap",
                "Prenatal Care",
                "HIV Cases",
                "Undernourishment",
                "School Completion",
                "Parliament Seats (f)",
                "Malaria Cases",
                "School Enrollment",
                "Self-Employment",
                "International Trade",
                "Tuberculosis Mortality"]

# Assigning column_names list to dataframe columns.
All_Regions.columns = column_names

# Lower case all the column names in order to make referencing easier.
All_Regions.columns = All_Regions.columns.str.lower()

# Subsetting the wanted columns from the Meta Country DataFrame.
income_group = meta_country[["Income Group", "Code"]]

# (inner) Joining the two DataFrames together.
All_Regions = All_Regions.merge(income_group, how = 'inner', left_on='country code', right_on='Code').copy()

# Dropping the joined column
All_Regions = All_Regions.drop(["Code"], axis = 1)

# Assign a copy of data frame to new variable.
all_regions = All_Regions.copy()

# Assinging Country code to index (rule set by the team for analysis)
all_regions = all_regions.set_index('country code')

# Checking the information on data types an non-null values (remove # to run)
#all_regions.info()

# Create dictionary data types to change data types
data_types = {'aids deaths'      : int,
              'life expectancy'  : int,
              'malaria cases'    : int}

# Converting the data types
all_regions = all_regions.convert_dtypes(data_types)

# showing the new overview to check whether data types have changed (remove # to run)
#all_regions.info()

##############################################################################
## 8. CREATING SUBCATEGORY LISTS FOR REFERENCING ##

# Category 01: Connectivity
connectivity = ['country',
                'region',
                'population',
                'internet users',
                'cellular subscriptions']

# Category 02: Economic
economic   =    ['country',
                'population',
                'gdp per person employed',
                'gni per capita',
                'bop income share',
                'oda per capita',
                'poverty line gap',
                'international trade',
                'parliament seats (f)']

# Category 03: Education
education   =  ['country',
                'region',
                'adj school enrollment',
                'school enrollment',
                'school completion',
                'literacy rate']

# Category 04: Employment
employment =    ['country',
                'population',
                'family workers (f)',
                'family workers (m)',
                'family workers (t)',
                'employment ratio (f)',
                'employment ratio (m)',
                'employment ratio (t)',
                'self-employment']

# Category 05: Environment
environment =  ['country',
                'region',
                'population',
                'co2 emissions',
                'energy usage',
                'gdp per energy',
                'sanitation quality',
                'water quality']


# Category 06: Health
health    =   ['country',
                'region',
                'aids deaths',
                'art coverage',
                'measles immunization',
                'tuberculosis cases',
                'tuberculosis mortality',
                'hiv cases',
                'malaria cases',
                'life expectancy',
                'undernourishment',
                'adolescent fertility',
                'fertility',
                'maternal mortality',
                'infant mortality',
                'delivery care',
                'prenatal care']


##############################################################################
## 9. SUBSET ARABIAN PENISULA REGION AS NEW DATAFRAME ##

# Filter the date where region = 'Arabian Peninsula'. This is done with loc subsetting
arabian_peninsula = all_regions.loc[all_regions['region'] == 'Arabian Peninsula'].copy()

# Checking results (remove # to run)
#arabian_peninsula.head(n=5)

# Create variable for adjusted arabian_peninsula after analysis.
arabian_peninsula_adj = arabian_peninsula.copy()

# Checking results (remove # to run)
#arabian_peninsula_adj.head(n=5)
```

2. Exploratory Analysis

2.1 Strategy

Exploring 40 indicators under seven categories (figure 2.1), we identified obscure findings, outliers, and decided how to handle null values. In the following cases, we decided to drop an indicator:

- Unreliable, untrustworthy, or misleading data according to metadata or external research
- More than 30% null values

For considered indicators, null values are imputed with:

- Data from the original data source
- Mean or median of the available distribution
- Other values based on external research

Outliers have been visually identified by:

- Skewness of distribution histogram
- Boxplots
- External research

All flagged null values and outliers can be accessed in this dataset.

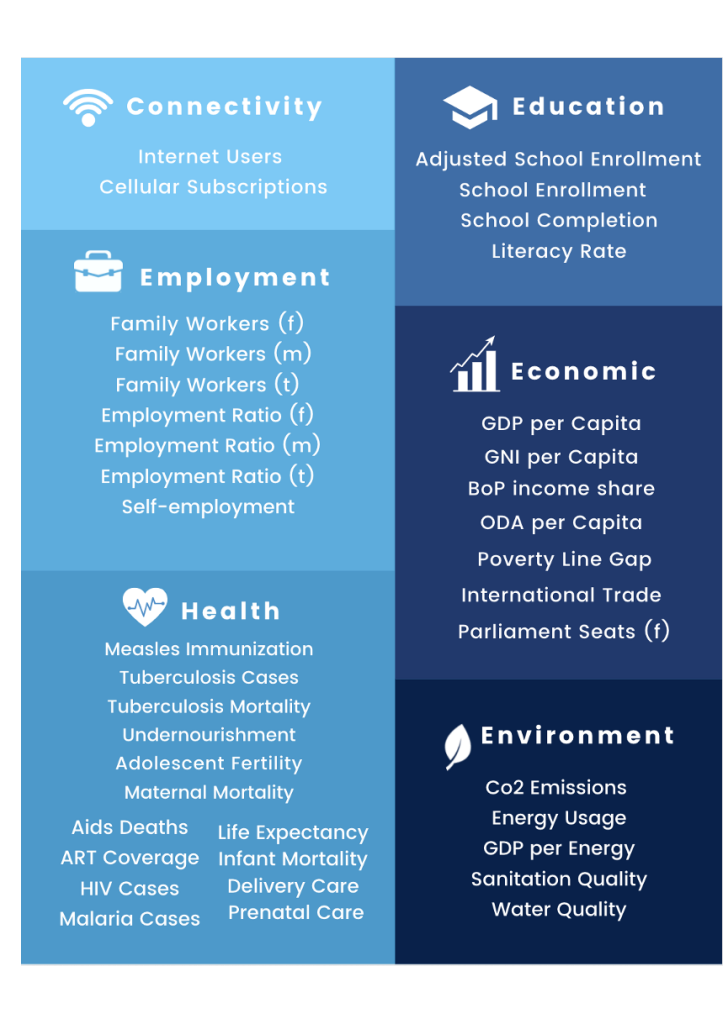

Figure 2.1: Indicators (columns) by Category →

2.2 Example of Analysis

To exemplify the process, the analysis of the indicator *measles immunization* is presented.

1. How many missing values are there? Can the data be trusted?

There are 6.67% missing values. The metadata describes that the data from WHO was gathered from national censuses and nationally representative household surveys. The last census data collection year varies within our region from 1943 to 2017. In this case, the outdated censuses do not reflect an accurate representation of the immunization situation. However, recent data has been published showing that the values in our datasets reflect reports from 2019, i.e. the data reflects the current situation regardless of the census collection year. → *Decision: keep*

2. What are we doing with the missing values?

The distribution histogram (Figure 2.2: Left Graph) without missing values is skewed to the left which normally indicates the median better represents missing values. Trying both, mean and median, the imputation differs less from the original distribution with the mean (Figure 2.2: Middle Graph), which validates mean as imputation method.

3. Are there any outliers?

Two outliers were found in Figure 2.2: Right Graph: Yemen and Iraq have significantly lower values than the rest. As the boxplot did not mark these as outliers, a manual threshold was applied.

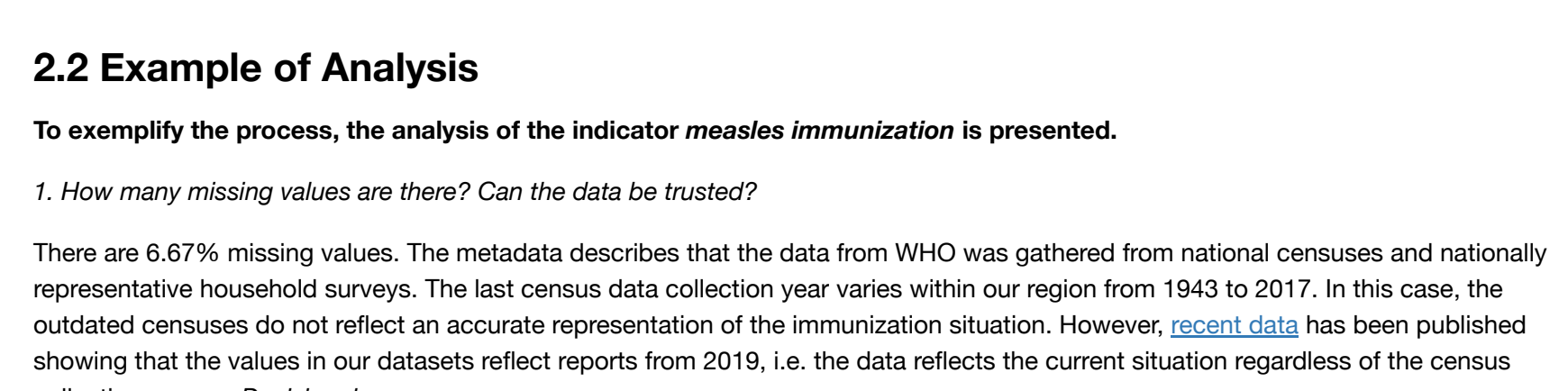

Figure 2.2: Example for Exploratory Analysis

In [ ]:
```
# Plots for measles immunization (change plt.close() to plt.show to display plot)

# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell

# setting figure size
fig, ax = plt.subplots(figsize = [16, 5],
                        sharex = True, # sharing x-axis between visualizations
                        sharey = True) # sharing y-axis between visualizations)


#PLOT 1: Measles immunization Mean / Median check.
plt.subplot(1, 3, 1) #1 row, 3 columns, spot 1

# Histogram
sns.distplot(a = measles_immunization_dropped['measles immunization'],
             bins  = 'fd',
             hist  = True,
             kde   = False,
             rug   = False,
             color = 'gray')

# Titles and labels
plt.title(label = "\nDistribution of Measles Immunization")
plt.xlabel(xlabel = 'Measles Immunization')
plt.ylabel(ylabel = 'Frequency')


# Vertical lines for mean and median
plt.axvline(x = measles_immunization_dropped['measles immunization'].mean(),
            color = 'maroon')

plt.axvline(x = measles_immunization_dropped['measles immunization'].median(),
            color = 'darkorange')

# Legend
plt.legend(labels =  ['mean', 'median'])

#PLOT 2: Measles Imputation NaN Values Distribution
plt.subplot(1, 3, 2) #1 row, 3 columns, spot 2

# Histogram for Tuberculosis mortalitY (without NaN)
sns.distplot(a     = measles_immunization_dropped['measles immunization'],
             bins  = 'fd',
             hist  = True,
             kde   = True, # activating kde
             rug   = False,
             color = 'gray')

# Histogram (with imputation)
sns.distplot(a     = arabian_peninsula_adj['measles immunization'],
             bins  = 'fd',
             hist  = True,
             kde   = True, # activating kde
             rug   = False,
             color = 'deepskyblue')


# Titles, labels, and formatting
plt.title(label   = "\n Measles Immunization Distribution\n(With/without imputation of missing values)"
)
plt.xlabel(xlabel = 'Measles Immunization')
plt.ylabel(ylabel = 'Frequency')

# This adds a legend
plt.legend(labels =  ['original distribution',
                      'imputed (mean) distribution'])

# PLOT 3: Measles Imputation Outliers
plt.subplot(1, 3, 3) #1 row, 3 columns, spot 3

# Developing a boxplot for Measles Immunization
sns.boxplot(x      = 'measles immunization',  # x-variable
            y      = None,      # optional y-variable
            hue    = None,      # optional categorical feature
            orient = 'h',       # horizontal or vertical
            data   = arabian_peninsula_adj, # DataFrame where features exist
            color = "skyblue")

# Adding a line to signify an outlier threshold
plt.axvline(x = 79,
           color = 'red',
            linestyle= '--')

# Formatting and displaying the plot
plt.title(label = '\nMeasles Immunization Outliers')
plt.xlabel(xlabel = 'Measles Immunization')

# If you want to display the plots, please change plt.close() to plt.show()
plt.close()
```

2.3 Overview Analysis

The process above was applied to all indicators.

Figure 2.3 shows a summary of the complete analysis and conclusions.

Please see this PDF for an in-depth analysis for each indicator.

Indicator	Percentage Missing	Action		Imputation		Outlier
		Decision	Reason	Method	Reason	
Internet Users	No	Keep	The original data source correlates with the data	-	-	Oman, Iraq
Cellular Subscription	No			-	-	Kuwait
GNI	6.67%	Keep	Combined sources from the World bank's National Account and OECD are reliable.	Average GNI from Westbank & Gaza	Because those countries have similar economic situations	Qatar
International Trade	6.67%					-
Parliament Seats Hold by women	26.67%	Keep	Numbers are aligned with the Global Gender Gap Report	Actual data from the World Fact Book	Data found reflected with the other values that were already present.	-
Co2 Emissions	No	Keep	Data is accurate and verifiable at its source. There's a discrepancy of almost x3 between the numbers from the source due to different measurement units.	-	-	-
Malaria Cases	66.67%	Keep	Data seems to be accurate, regardless of the missing values due to the fact that some areas are more suitable for malaria.	0	Areas that have NaN values are not suitable for malaria, and therefore, would have 0 cases to report.	Yemen
Tuberculosis Cases	No	Keep	The report and country outlook on the original source shows that the numbers (in the form of the weighted average, matches with the data reported in our data set	-	-	UAE, Westbank/ Gaza
Tuberculosis Mortality	6.67%			Median	Median represents the higher values, which are in the majority in our region.	Yemen
Measles Immunization¶	6.67%	Keep	Data is outdated and based on various census data; however, the average presented in our data does match with the more recent data provided by its original source.	Mean	Represents the overall dataset better due to more variance.	Yemen, Iraq
Life Expectancy	No	Keep	The annual data series data are interpolated data from 5-year period data. The data was taken from six different sources to complement each other to get an indication	-	-	Yemen, Iraq
Fertility Rate	No			-	-	-
GDP per person employed	6.67%	Drop	National Account conventions systems vary in years, therefore not comparable. Many limitations to the reliability of the data.	-	-	-
Oda per Capita	46.67%	Drop	The amount of aid given was not taken into account. Some countries would be reflected as aid givers while they were receivers.	-	-	-
BOP Income Share	80%	Drop	Three countries did file the income share as they are complying with the European reporting standards. Household surveys are also not comparable.	-	-	-
Poverty Line Gap	93.33%	Drop	Only Jordan is reporting on this indicator as the country has a defined poverty reduction strategy	-	-	-
School Enrollment Primary	46.67%	Drop	Values differ with the original average-weighted reported number with 1.5+. The source also classifies the indicators' actual meaning as something different from described in our metadata, rendering untrustworthy.	-	-	-
Adjusted net enrollment rate, primary	46.67%			-	-	-
Completion Rate	40%	Drop	Original data contradiction with the data provided. Data must have been altered at least twice and, therefore, not trustworthy.	-	-	-
Literacy Rate	73.33%	Drop	Census data are collected from different years with at least a gap of more than 50+ years. No other sources were available.	-	-	Lebanon, Iraq
Family Workers	No	Drop	Meta description explains that the data is not usable for international comparison.	-	-	-
Employment Ratio	No			-	-	-
Self-Employment	No			-	-	-
Energy Usage Per GDP	13.33%	Drop	The data is already quite outdated, and the hyperlink to the source of information is not working. On the website itself, indicators cannot be identified. There is also no explicit meta description.	-	-	-
GDP Per Energy	13.33%			-	-	-
Improved Sanitation	No	Drop	Outdated data. More recent data is available on the original source, which reflects the current situation better.	-	-	-
Water Quality	No			-	-	-
Aids deaths	66.67%	Drop	The numbers in the sheets are estimates by UNAIDS. Nothing is known about the collection of data. There should be a positive correlation, which is not found with the estimated numbers	-	-	-
ART coverage	66.67%			-	-	-
HIV Cases	66.67%			-	-	-
Undernourishment	26.67%	Drop	The indicator is dependent on other variables (access to food, the availability of food per person, quality of care), which are not available in our dataset. Therefore, the indicator is hard to interpret.	-	-	-
Maternal Mortality	No	Drop	The reliability of the ratios provided is unknown reliability, therefore the estimates cannot be verified.			
Adolescent Fertility	No	Drop	The original source does not explicitly have data related to Adolescent Fertility. Average-weighted data does not match with the data given in our dataset. There is no info on the collection data.	-	-	-
Prenatal Care	80%	Drop	As described before and plotted in 3.3 Literacy Rate, the data of these indicators are dependent censuses data, which have a lot of variations in the collection years. As there are no other data substitutes, justifying the numbers and direct reasons for missing values is difficult.	-	-	-
Delivery Care	46.67%			-	-	-
Infant Mortality	No			-	-	-

Figure 2.3: Actions Taken by Indicator

2.4 Code

In [ ]:
```
# Flag missing values, subset arabian peninsula, export sheets
##############################################################################

all_regions_flagged = all_regions.copy()

# Loop through each column in all_regions and flag any null values in a new column
for col in all_regions:

    if all_regions[col].isnull().astype(int).sum() > 0:
        all_regions_flagged['m_'+col] = all_regions[col].isnull().astype(int)          # Source: Chase Kusterer

# Sum all null value columns into a missing value sum column
all_regions_flagged['mv_sum'] = all_regions_flagged['m_aids deaths'] + \
                                all_regions_flagged['m_adj school enrollment'] + \
                                all_regions_flagged['m_adolescent fertility'] + \
                                all_regions_flagged['m_art coverage'] + \
                                all_regions_flagged['m_delivery care'] + \
                                all_regions_flagged['m_co2 emissions'] + \
                                all_regions_flagged['m_family workers (f)'] + \
                                all_regions_flagged['m_family workers (m)'] + \
                                all_regions_flagged['m_family workers (t)'] + \
                                all_regions_flagged['m_employment ratio (f)'] + \
                                all_regions_flagged['m_employment ratio (m)'] + \
                                all_regions_flagged['m_employment ratio (t)'] + \
                                all_regions_flagged['m_energy usage'] + \
                                all_regions_flagged['m_fertility'] + \
                                all_regions_flagged['m_gdp per person employed'] + \
                                all_regions_flagged['m_gdp per energy'] + \
                                all_regions_flagged['m_gni per capita'] + \
                                all_regions_flagged['m_measles immunization'] + \
                                all_regions_flagged['m_sanitation quality'] + \
                                all_regions_flagged['m_water quality'] + \
                                all_regions_flagged['m_tuberculosis cases'] + \
                                all_regions_flagged['m_bop income share'] + \
                                all_regions_flagged['m_internet users'] + \
                                all_regions_flagged['m_life expectancy'] + \
                                all_regions_flagged['m_literacy rate'] + \
                                all_regions_flagged['m_maternal mortality'] + \
                                all_regions_flagged['m_cellular subscriptions'] + \
                                all_regions_flagged['m_infant mortality'] + \
                                all_regions_flagged['m_oda per capita'] + \
                                all_regions_flagged['m_poverty line gap'] + \
                                all_regions_flagged['m_prenatal care'] + \
                                all_regions_flagged['m_hiv cases'] + \
                                all_regions_flagged['m_undernourishment'] + \
                                all_regions_flagged['m_school completion'] + \
                                all_regions_flagged['m_parliament seats (f)'] + \
                                all_regions_flagged['m_malaria cases'] + \
                                all_regions_flagged['m_school enrollment'] + \
                                all_regions_flagged['m_self-employment'] + \
                                all_regions_flagged['m_international trade'] + \
                                all_regions_flagged['m_tuberculosis mortality']

all_regions_flagged['mv_perc'] = all_regions_flagged.loc[: , 'mv_sum'] / 40 * 100

##############################################################################
# Subset the Arabian Peninsula region and its categories
arabian_peninsula = all_regions.loc[all_regions['region'] == 'Arabian Peninsula'].copy()

arabian_peninsula_flagged = all_regions_flagged.loc[all_regions['region'] == 'Arabian Peninsula'].copy
()

# Create list of region's country codes for future reference
arabian_peninsula_country_codes = [ 'ARE',
                                    'BHR',
                                    'CYP',
                                    'IRQ',
                                    'ISR',
                                    'JOR',
                                    'KWT',
                                    'LBN',
                                    'OMN',
                                    'PSE',
                                    'QAT',
                                    'SAU',
                                    'SYR',
                                    'TUR',
                                    'YEM']

##############################################################################
# Create copy of arabian peninsula for dropping and imputing during the analysis

arabian_peninsula_adj = arabian_peninsula_flagged.copy()

##############################################################################
# Export flagged & subsetted datasets
all_regions_flagged.to_excel(excel_writer = "./_datasets/saved_datasets/all_regions_flagged.xlsx",
                             sheet_name   = 'all_regions_null',
                             index        = True)

with pd.ExcelWriter("./_datasets/saved_datasets/arabian_peninsula.xlsx") as writer:
    arabian_peninsula.to_excel(writer,
                               sheet_name   = 'arabian_peninsula',
                               index        = True)
    arabian_peninsula_flagged.to_excel(writer,
                                       sheet_name   = 'null_values_flagged',
                                       index        = True)
```

In [ ]:

```python
# Drop Columns per Category
###############################################################################
###############################################################################

### Drop Education Indicators ###
arabian_peninsula_adj.drop(labels  = ['adj school enrollment', 'm_adj school enrollment',
                                      'school enrollment', 'm_school enrollment',
                                      'school completion', 'm_school completion',
                                      'literacy rate', 'm_literacy rate'],
                           axis    = 1,
                           inplace = True)

###############################################################################
###############################################################################

### Drop Environment Indicators ###
arabian_peninsula_adj.drop(labels  = ['energy usage', 'm_energy usage',
                                      'gdp per energy', 'm_gdp per energy',
                                      'water quality', 'm_water quality',
                                      'sanitation quality', 'm_sanitation quality'],
                           axis    = 1,
                           inplace = True)

###############################################################################
###############################################################################

### Drop Economic Indicators ###

arabian_peninsula_adj.drop (labels  = ['gdp per person employed', 'm_gdp per person employed',
                                       'oda per capita', 'm_oda per capita',
                                       'bop income share', 'm_bop income share',
                                       'poverty line gap', 'm_poverty line gap'],
                            axis    = 1,
                            inplace = True)


###############################################################################
###############################################################################

### Drop Health Indicators ###

arabian_peninsula_adj.drop (labels  = ['aids deaths', 'm_aids deaths',
                                       'art coverage', 'm_art coverage',
                                       'hiv cases', 'm_hiv cases',
                                       'undernourishment', 'm_undernourishment',
                                       'adolescent fertility', 'm_adolescent fertility',
                                       'maternal mortality', 'm_maternal mortality',
                                       'infant mortality', 'm_infant mortality',
                                       'delivery care', 'm_delivery care',
                                       'prenatal care', 'm_prenatal care'],
                            axis    = 1,
                            inplace = True)

###############################################################################
###############################################################################

### Drop Health Indicators ###

arabian_peninsula_adj.drop (labels  = ['family workers (f)', 'm_family workers (f)',
                                       'family workers (m)', 'm_family workers (m)',
                                       'family workers (t)', 'm_family workers (t)',
                                       'employment ratio (f)', 'm_employment ratio (f)',
                                       'employment ratio (m)', 'm_employment ratio (m)',
                                       'employment ratio (t)', 'm_employment ratio (t)',
                                       'self-employment', 'm_self-employment'],

                            axis    = 1,
                            inplace = True)

###############################################################################
###############################################################################

# Recalculate mv_sum & mv_perc based on remaining columns

# Sum all null value columns into a missing value sum column
arabian_peninsula_adj['mv_sum']  =  arabian_peninsula_adj['m_co2 emissions'] + \
                                    arabian_peninsula_adj['m_fertility'] + \
                                    arabian_peninsula_adj['m_gni per capita'] + \
                                    arabian_peninsula_adj['m_measles immunization'] + \
                                    arabian_peninsula_adj['m_tuberculosis cases'] + \
                                    arabian_peninsula_adj['m_internet users'] + \
                                    arabian_peninsula_adj['m_life expectancy'] + \
                                    arabian_peninsula_adj['m_cellular subscriptions'] + \
                                    arabian_peninsula_adj['m_parliament seats (f)'] + \
                                    arabian_peninsula_adj['m_malaria cases'] + \
                                    arabian_peninsula_adj['m_international trade'] + \
                                    arabian_peninsula_adj['m_tuberculosis mortality']

arabian_peninsula_adj['mv_perc'] = arabian_peninsula_adj.loc[: , 'mv_sum'] / 12 * 100

## Check adjusted dataframe ##
#arabian_peninsula_adj

###########################
# Export Reduced Dataframe

with pd.ExcelWriter("./_datasets/saved_datasets/arabian_peninsula.xlsx",
                    engine = "openpyxl",
                    mode   = "a") as writer:
    arabian_peninsula_adj.to_excel(writer, sheet_name="dropped indicators")
```

In [ ]:

```python
# Imputation for null values per Category
###############################################################################

### Impute null values for Economic Indicators ###

###############################################################################

########## GNI per capita column #############

## Create data frame without null values for GNI per capita ##

# subset the existing data set
GNI_dropped = arabian_peninsula[economic].loc[ : , ['country', 'population', 'gni per capita']].copy()

# drop the missing rows with dropna
GNI_dropped = GNI_dropped.dropna(axis = 0)


## Impute missing values for GNI per capita ##

# soft coding mean for SYR
GNI_imputation = arabian_peninsula.loc[['PSE', 'YEM'],['gni per capita']].mean()

# filling GNI per capita SYR with GNI_imputation in the arabian_peninsula_adj data frame
arabian_peninsula_adj.fillna(value   = GNI_imputation,
                             inplace = True)


###############################################################################

########## International Trade column #############

## Create data frame without null values for International Trade ##

# subset the existing data set
InternationalTrade_dropped = arabian_peninsula.loc[ : , ['country', 'population', 'international trade'
]].copy()

# drop the missing rows with dropna
InternationalTrade_dropped = InternationalTrade_dropped.dropna(axis = 0)

## Impute missing values for International Trade ##

# soft coding mean for International Trade
IT_imputation = arabian_peninsula.loc[['PSE', 'YEM'],['international trade']].mean()

# filling International trade indicator for SYR with IT_imputation in the arabian_peninsula_adj data fr
ame
arabian_peninsula_adj.fillna(value   = IT_imputation,
                             inplace = True)


###############################################################################

########## Parliament seats (f) column #############

## Create data frame without null values for Parliament Seats (f) ##

# subset the existing data set
parliamentF_dropped = arabian_peninsula.loc[ : , ['country', 'population', 'parliament seats (f)']].cop
y()

# drop the missing rows with dropna
parliamentF_dropped = parliamentF_dropped.dropna(axis = 0)


## Impute missing values for Parliament Seats held by Women ##

# soft coding values
FemParl_OMN_imputation = 2.33   # https://www.ipu.org/parliament/OM
FemParl_QAT_imputation = 9.76   # https://www.ipu.org/parliament/QA
FemParl_SAU_imputation = 19.87  # https://www.ipu.org/parliament/SA

# fill values for OMN, QAT, SAU
arabian_peninsula_adj.loc['OMN','parliament seats (f)']  = FemParl_OMN_imputation
arabian_peninsula_adj.loc['QAT','parliament seats (f)']  = FemParl_QAT_imputation
arabian_peninsula_adj.loc['SAU','parliament seats (f)']  = FemParl_SAU_imputation

###############################################################################

### Impute null values for Health Indicators ###

###############################################################################

########## Malaria Cases #############

## Create data frame without null values for Malaria Cases ##

# subset the existing data set
malariacases_dropped = arabian_peninsula.loc[ : , ['country', 'malaria cases']].copy()

# drop the missing rows with dropna
malariacases_dropped  = malariacases_dropped.dropna(axis = 0)

# Impute value for Malaria
malaria_mortality_imputation = 0

# fill null values for measles immunization.
arabian_peninsula_adj['malaria cases'].fillna( value = malaria_mortality_imputation,
                                               inplace = True)

########## Tuberculosis Mortality #############
# subset the existing data set
tuberculosis_mortality_dropped = arabian_peninsula.loc[ : , ['country', 'tuberculosis mortality']].copy
()

# drop the missing rows with dropna
tuberculosis_mortality_dropped  = tuberculosis_mortality_dropped.dropna(axis = 0)

# Impute value for tuberculosis mortality.
tuberculosis_mortality_imputation =  round(arabian_peninsula['tuberculosis mortality'].median(), ndigit
s = 2)

# fill null values for tuberculosis mortality.
arabian_peninsula_adj['tuberculosis mortality'].fillna( value = tuberculosis_mortality_imputation,
                                                        inplace = True)

########## Measles Immunization #############

# subset the existing data set
measles_immunization_dropped = arabian_peninsula.loc[ : , ['country', 'measles immunization']].copy()

# drop the missing rows with dropna
measles_immunization_dropped  = measles_immunization_dropped.dropna(axis = 0)

# Impute value for measles immunization.
measles_immunization_imputation =  round(arabian_peninsula['measles immunization'].mean(), ndigits = 2)

# fill null values for measles immunization.
arabian_peninsula_adj['measles immunization'].fillna( value = measles_immunization_imputation,
                                                      inplace = True)
```

In [ ]:

```python
# Outlier Flagging per Category
###############################################################################


#create placeholder for outliers
for col in arabian_peninsula_adj:
    if 'm_' in col or\
    'mv_' in col or\
    'region' in col or\
    'country' in col or\
    'Income Group' in col:
        continue
    else:
        arabian_peninsula_adj['o_'+col] = 0

##########################################################################################
##########################################################################################

## Connectivity Indicators ##

#define lower thresholds for outliers
threshold_lower_internet_high = 35           # set according to statistical analysis (boxplot)
threshold_lower_internet_upper_middle = 16   # set according to statistical analysis (boxplot)
threshold_lower_cellular_high = 100          # set according to statistical analysis (boxplot)

#set outlier flags

for index, col in arabian_peninsula_adj.loc[:,:].iterrows():

    #Internet Users Outlier
    if arabian_peninsula_adj.loc[ index , 'internet users'] < threshold_lower_internet_high or\
    arabian_peninsula_adj.loc[ index , 'internet users'] < threshold_lower_internet_upper_middle:
        arabian_peninsula_adj.loc[ index , 'o_internet users'] = 1

    #Cellular Subscriptions Outlier
    elif arabian_peninsula_adj.loc[ index , 'cellular subscriptions'] < threshold_lower_cellular_high:
        arabian_peninsula_adj.loc[ index , 'o_cellular subscriptions'] = 1

##########################################################################################
##########################################################################################

## Economic Indicators ##

#define upper thresholds for outliers
threshold_upper_GNI       = 60000   # set according to statistical analysis (boxplot)
threshold_upper_IntTrade  = 130     # set according to statistical analysis (histogram)
threshold_upper_ParlSeats = 26      # set according to statistical analysis (boxplot)

#define lower thresholds for outliers
threshold_lower_GNI       = 0       # set according to statistical analysis (boxplot)
threshold_lower_IntTrade  = 45      # set according to statistical analysis (boxplot)
threshold_lower_ParlSeats = 0       # set according to statistical analysis (boxplot)

#set outlier flags

for index, col in arabian_peninsula_adj.loc[:,:].iterrows():

    #GNI Outlier
    if arabian_peninsula_adj.loc[ index , 'gni per capita'] > threshold_upper_GNI or\
    arabian_peninsula_adj.loc[ index , 'gni per capita'] < threshold_lower_GNI:
        arabian_peninsula_adj.loc[ index , 'o_gni per capita'] = 1

    #International Trade outlier
    elif arabian_peninsula_adj.loc[ index , 'international trade'] > threshold_upper_IntTrade or\
    arabian_peninsula_adj.loc[ index , 'international trade'] < threshold_lower_IntTrade:
        arabian_peninsula_adj.loc[ index , 'o_international trade'] = 1

    #Parlament seats hold by women outlier
    elif arabian_peninsula_adj.loc[ index , 'parliament seats (f)'] > threshold_upper_ParlSeats or\
    arabian_peninsula_adj.loc[ index , 'parliament seats (f)'] < threshold_lower_ParlSeats:
        arabian_peninsula_adj.loc[ index , 'o_parliament seats (f)'] = 1


##########################################################################################
##########################################################################################

## Health Indicators ##

#define upper thresholds for outliers
threshold_upper_malaria        = 20000   # set according to statistical analysis (boxplot)
threshold_upper_tuberculosis_c = 65      # set according to statistical analysis (boxplot)
threshold_upper_tuberculosis_m = 1.8     # set according to statistical analysis (boxplot)
threshold_upper_measles        = 99      # set according to statistical analysis (boxplot)
threshold_upper_life_exp       = 82      # set according to statistical analysis (boxplot)
threshold_upper_fertility      = 5       # set according to statistical analysis (boxplot)

#define lower thresholds for outliers
threshold_lower_malaria        = 0       # set according to statistical analysis (boxplot)
threshold_lower_tuberculosis_c = 5       # set according to external research (WHO, 2020)
threshold_lower_tuberculosis_m = 0       # set according to statistical analysis (boxplot)
threshold_lower_measles        = 79      # set according to statistical analysis (histogram)
threshold_lower_life_exp       = 72      # set according to statistical analysis (boxplot)
threshold_lower_fertility      = 1.45    # set according to statistical analysis (boxplot)

#set outlier flags

for index, col in arabian_peninsula_adj.loc[:,:].iterrows():

    #malaria cases outlier
    if arabian_peninsula_adj.loc[ index , 'malaria cases'] > threshold_upper_malaria or\
    arabian_peninsula_adj.loc[ index , 'malaria cases'] < threshold_lower_malaria:
        arabian_peninsula_adj.loc[ index , 'o_malaria cases'] = 1

    #tuberculosis cases outlier
    elif arabian_peninsula_adj.loc[ index , 'tuberculosis cases'] > threshold_upper_tuberculosis_c or\
    arabian_peninsula_adj.loc[ index , 'tuberculosis cases'] < threshold_lower_tuberculosis_c:
        arabian_peninsula_adj.loc[ index , 'o_tuberculosis cases'] = 1

    #tuberculosis mortality outlier
    elif arabian_peninsula_adj.loc[ index , 'tuberculosis mortality'] > threshold_upper_tuberculosis_m
or\
    arabian_peninsula_adj.loc[ index , 'tuberculosis mortality'] < threshold_lower_tuberculosis_m:
        arabian_peninsula_adj.loc[ index , 'o_tuberculosis mortality'] = 1


    #measles immunization outlier
    elif arabian_peninsula_adj.loc[ index , 'measles immunization'] > threshold_upper_measles or\
    arabian_peninsula_adj.loc[ index , 'measles immunization'] < threshold_lower_measles:
        arabian_peninsula_adj.loc[ index , 'o_measles immunization'] = 1

    #life expectancy outlier
    elif arabian_peninsula_adj.loc[ index , 'life expectancy'] > threshold_upper_life_exp or\
    arabian_peninsula_adj.loc[ index , 'life expectancy'] < threshold_lower_life_exp:
        arabian_peninsula_adj.loc[ index , 'o_life expectancy'] = 1

    #fertility outlier
    elif arabian_peninsula_adj.loc[ index , 'fertility'] > threshold_upper_fertility or\
    arabian_peninsula_adj.loc[ index , 'fertility'] < threshold_lower_fertility:
        arabian_peninsula_adj.loc[ index , 'o_fertility'] = 1

##########################################################################################
##########################################################################################

# General outlier flagging due to high or low missing values

# instantiate column
arabian_peninsula_adj['o_null values'] = 0

# set flag for PSE, SYR, CYP and TUR
arabian_peninsula_adj.loc['PSE', 'o_null values'] = 1  #high number of missing values, identified by fi
gure 3.1
arabian_peninsula_adj.loc['SYR', 'o_null values'] = 1  #high number of missing values, identified by fi
gure 3.1
arabian_peninsula_adj.loc['CYP', 'o_null values'] = 1  #low number of missing values, identified by fig
ure 3.1
arabian_peninsula_adj.loc['TUR', 'o_null values'] = 1  #low number of missing values, identified by fig
ure 3.1


##########################################################################################
##########################################################################################


# Create two columns for summing outliers and deriving a percentage

# Sum all null value columns into a missing value sum column
arabian_peninsula_adj['o_sum']  =  arabian_peninsula_adj['o_co2 emissions'] + \
                                   arabian_peninsula_adj['o_fertility'] + \
                                   arabian_peninsula_adj['o_gni per capita'] + \
                                   arabian_peninsula_adj['o_measles immunization'] + \
                                   arabian_peninsula_adj['o_tuberculosis cases'] + \
                                   arabian_peninsula_adj['o_internet users'] + \
                                   arabian_peninsula_adj['o_life expectancy'] + \
                                   arabian_peninsula_adj['o_cellular subscriptions'] + \
                                   arabian_peninsula_adj['o_parliament seats (f)'] + \
                                   arabian_peninsula_adj['o_malaria cases'] + \
                                   arabian_peninsula_adj['o_international trade'] + \
                                   arabian_peninsula_adj['o_tuberculosis mortality'] + \
                                   arabian_peninsula_adj['o_null values']

# Calculate percentage of outliers
arabian_peninsula_adj['o_perc'] = arabian_peninsula_adj.loc[: , 'o_sum'] / 13 * 100

###########################
# Export Adjusted Dataframe

with pd.ExcelWriter("./_datasets/saved_datasets/arabian_peninsula.xlsx",
                    engine = "openpyxl",
                    mode   = "a") as writer:
    arabian_peninsula_adj.to_excel(writer, sheet_name="outliers flagged")
```

```python
# Additional Plots used in the PDF (change plt.close() to plt.show to display plot)

# If you want to display the plots, please change plt.close() to plt.show() on line 446, 484, 521
#(plots were first saved individually, then added as as group for the PDF)


# setting figure size
fig, ax = plt.subplots(figsize = [16, 100],
                       sharex = True, # sharing x-axis between visualizations
                       sharey = True) # sharing y-axis between visualizations)

##########################  Connectivity Plots  ############################

# PLOT 1: Internet users
plt.subplot(18, 2, 1) #18 rows, 2 columns, spot 1

# Boxplot for internet users
sns.boxplot(x      = 'internet users',
            y      = 'Income Group',
            orient = 'h',
            color  = 'skyblue',
            data   = arabian_peninsula)

# titles and labels
plt.title(label = "Figure 1:\n Distribution of Internet Users by Income Group",
          pad = 20.0)
plt.xlabel(xlabel = 'Internet Users')
plt.ylabel(ylabel = 'Frequency')


# PLOT 2: cellular subscriptions
plt.subplot(18, 2, 2) #18 rows, 2 columns, spot 2

# Boxplot for cellular subscriptions
sns.boxplot(x      = 'cellular subscriptions',
            y      = 'Income Group',
            orient = 'h',
            color  = 'skyblue',
            data   = arabian_peninsula)

# titles, labels, and formatting
plt.title(label    = "Figure 2:\n Distribution of Cellular Subscriptions by Income Group",
          pad = 20.0)
plt.xlabel(xlabel = 'Cellular Subscriptions')

############################  Economic Plots  ##############################

# PLOT 3: GNI per capita without imputed values
plt.subplot(18, 2, 3) #18 rows, 2 columns, spot 3

# histogram for GNI per capita without imputed values
sns.distplot(a     = GNI_dropped['gni per capita'],
             bins  = 'fd',
             hist  = True,
             kde   = True,
             rug   = False,
             color = 'gray')

# histogram for GNI per capita with imputed values
sns.distplot(a     = arabian_peninsula_adj['gni per capita'],
             bins  = 'fd',
             hist  = True,
             kde   = True,
             rug   = False,
             color = 'deepskyblue')

# titles, labels, and formatting
plt.title(label   = """Figure 4:\nGNI per Capita Distribution\n(With/without imputation of missing valu
es)""",
          pad    = 10.0)
plt.xlabel(xlabel = 'GNI per Capita [USD]')
plt.ylabel(ylabel = 'Frequency')
plt.xlim(0.0, 70000)
#plt.ylim(0.0, 0.00004)

# legend
plt.legend(labels =  ['original distribution',
                      'imputed distribution'])

# PLOT 4: GNI per capita outliers
plt.subplot(18, 2, 4) #19 rows, 2 columns, spot 4

# boxplot 1
sns.boxplot(x      = 'gni per capita',  # x-variable
            y      = None,     # optional y-variable
            hue    = None,     # optional categorical feature
            orient = 'h',      # horizontal or vertical
            data   = arabian_peninsula_adj, # DataFrame where features exist
            color  = 'skyblue')

# formatting
plt.title(label   = "Figure 5:\nGNI per Capita Outliers",
          pad    = 10.0)
plt.xlabel(xlabel = 'GNI per capita [USD]')


# PLOT 5: International Trade without imputed values
plt.subplot(18, 2, 5) #18 rows, 2 columns, spot 5

# histogram for International Trade without imputed values
sns.distplot(a     = InternationalTrade_dropped['international trade'],
             bins  = 'fd',
             hist  = True,
             kde   = True,
             rug   = False,
             color = 'gray')

# histogram for International Trade with imputed values
sns.distplot(a     = arabian_peninsula_adj['international trade'],
             bins  = 'fd',
             hist  = True,
             kde   = True,
             rug   = False,
             color = 'deepskyblue')

# titles, labels, and formatting
plt.title(label   = """Figure 6:\nInternational Trade Distribution\n(With/without imputation of missing
 values)""",
          pad     = 10)
plt.xlabel(xlabel = 'International Trade [% of GDP]')
plt.ylabel(ylabel = 'Frequency')
plt.xlim(40, 150)
plt.ylim(0.0, 0.02)

# legend
plt.legend(labels =  ['original distribution',
                      'imputed distribution'])

# PLOT 6: parliament seats without imputed values
plt.subplot(18, 2, 6) #18 rows, 2 columns, spot 6

# histogram for parliament seats without imputed values
sns.distplot(a     = parliamentF_dropped['parliament seats (f)'],
             bins  = 'fd',
             hist  = True,
             kde   = True,
             rug   = False,
             color = 'gray')


# histogram for parliament seats with imputed values
sns.distplot(a     = arabian_peninsula_adj['parliament seats (f)'],
             bins  = 'fd',
             hist  = True,
             kde   = True,
             rug   = False,
             color = 'deepskyblue')

# titles, labels, and formatting
plt.title(label   = """Figure 7:\nParliament Seats Held by Women Distribution\n(With/without imputation
 of missing values)""",
          pad     = 10)
plt.xlabel(xlabel = 'Percentage of Parliament Seats Held by Women')
plt.ylabel(ylabel = 'Frequency')
plt.xlim(0.0, 40)
plt.ylim(0.0, 0.06)

# legend
plt.legend(labels =  ['original distribution',
                      'imputed distribution'])


############################  Education plot  ##############################

# PLOT 7: Literacy Rate Analysis (boxplot & table on census years)

# Set figure size
plt.subplot(18, 2, 7) #18 rows, 2 columns, spot 7

# Create DF of Census Dates per Country
literacy_rate = arabian_peninsula.loc[:,['country']].copy()

literacy_rate['census date'] = [2010, 2010, 2011, 1997, 2009, 2015, 2011, 1943, 2010, 2017, 2015, 2010,
 2004, 2011, 2004]

literacy_rate.sort_values(by        = 'census date',
                          ascending = False)

# Develop a boxplot for Census Date
sns.boxplot(x       = 'census date',  # x-variable
            y       = None,     # optional y-variable
            hue     = None,     # optional categorical feature
            orient  = 'h',      # horizontal or vertical
            color   = 'skyblue',
            data    = literacy_rate) # DataFrame where features exist

# Format and Display the plot
plt.axvline(x = 2003,
            color = "red",
            linestyle= '--')

plt.title(label   = """Figure 8:\nBoxplot of Census Year per Country in Arabian Peninsula""")
plt.xlabel(xlabel = 'Census Year')


##############################  Health Plots  ##############################

# PLOT 8: Malaria Cases imputation
# Set figure size
plt.subplot(18, 2, 8) #18 rows, 2 columns, spot 8

#plot distribution histogram for table without NaN values
sns.distplot(a = malariacases_dropped['malaria cases'],
             bins  = 2,
             hist  = True,
             kde   = True, # activating kde
             rug   = False,
             color = 'gray')

# titles, labels, and formatting
plt.title(label   = "Figure 12:\nMalaria Cases Distribution\n(With/without imputation of missing value
s)")
plt.xlabel(xlabel = 'Malaria Cases')
plt.ylabel(ylabel = 'Frequency')

# histogram for Tuberculosis mortalitY (with imputation)
sns.distplot(a     = arabian_peninsula_adj['malaria cases'],
             bins  = 2,
             hist  = True,
             kde   = True, # activating kde
             rug   = False,
             color = 'deepskyblue')

# this adds a legend
plt.legend(labels =  ['original distribution',
                      'imputed distribution'])

# PLOT 9: Malaria Cases Outliers
# Set figure size
plt.subplot(18, 2, 9) #18 rows, 2 columns, spot 9


# developing a boxplot for Malaria Cases
sns.boxplot(x      = 'malaria cases',  # x-variable
            y      = None,     # optional y-variable
            hue    = None,     # optional categorical feature
            orient = 'h',      # horizontal or vertical
            data   = arabian_peninsula_adj, # DataFrame where features exist
            color = "skyblue")

# formatting and displaying the plot
plt.title(label = 'Figure 13:\n Malaria Cases Outliers')
plt.xlabel(xlabel = 'Malaria Cases')

# PLOT 10: Tuberculosis Cases Outliers
# setting figure size
plt.subplot(18, 2, 10) #18 rows, 2 columns, spot 10


# Developing a boxplot for Tuberculosis Cases
sns.boxplot(x      = 'tuberculosis cases',  # x-variable
            y      = None,     # optional y-variable
            hue    = None,     # optional categorical feature
            orient = 'h',      # horizontal or vertical
            data   = arabian_peninsula_adj, # DataFrame where features exist
            color = "skyblue")

# Adding a line to signify an outlier threshold
plt.axvline(x = 5,
           color = 'red',
            linestyle= '--')

# Formatting and displaying the plot
plt.title(label = 'Figure 14:\nTuberculosis Cases Outliers')
plt.xlabel(xlabel = 'Tuberculosis Cases')

# PLOT 11: Tuberculosis Mortality Distribution
# setting figure size
plt.subplot(18, 2, 11) #18 rows, 2 columns, spot 11


# histogram for Tuberculosis Mortality Distribution
sns.distplot(a = tuberculosis_mortality_dropped['tuberculosis mortality'],
             bins  = 2,
             hist  = True,
             kde   = False,
             rug   = False,
             color = 'gray')


# titles and labels
plt.title(label = "Figure 15:\nDistribution of Tuberculosis Mortality")
plt.xlabel(xlabel = 'Tuberculosis Mortality')
plt.ylabel(ylabel = 'Frequency')


#vertical lines for mean and median
plt.axvline(x = tuberculosis_mortality_dropped['tuberculosis mortality'].mean(),
            color = 'maroon')

plt.axvline(x = tuberculosis_mortality_dropped['tuberculosis mortality'].median(),
            color = 'darkorange')

#legend
plt.legend(labels =  ['mean', 'median'])


#PLOT 12: Tuberculosis Mortality Distribution Imputed
plt.subplot(18, 2, 12) #18 rows, 2 columns, spot 12

# histogram for Tuberculosis mortalitY (without NaN)
sns.distplot(a     = tuberculosis_mortality_dropped['tuberculosis mortality'],
             bins  = 2,
             hist  = True,
             kde   = True, # activating kde
             rug   = False,
             color = 'gray')


# histogram for Tuberculosis mortalitY (with imputation)
sns.distplot(a     = arabian_peninsula_adj['tuberculosis mortality'],
             bins  = 2,
             hist  = True,
             kde   = True, # activating kde
             rug   = False,
             color = 'deepskyblue')


 # titles, labels, and formatting
plt.title(label   = "Figure 16:\nTuberculosis Mortality Distribution\n(With/without imputation of missi
ng values)")
plt.xlabel(xlabel = 'Tuberculosis Mortality')
plt.ylabel(ylabel = 'Frequency')


# this adds a legend
plt.legend(labels =  ['original distribution',
                      'imputed distribution'])

#PLOT 13: Tuberculosis Mortality Outliers
plt.subplot(18, 2, 13) #18 rows, 2 columns, spot 13

# developing a boxplot for Tuberculosis Mortality
sns.boxplot(x      = 'tuberculosis mortality',  # x-variable
            y      = None,     # optional y-variable
            hue    = None,     # optional categorical feature
            orient = 'h',      # horizontal or vertical
            data   = arabian_peninsula_adj, # DataFrame where features exist
            color = "skyblue")

# formatting and displaying the plot
plt.title(label = 'Figure: 17:\nTuberculosis Mortality Outliers')
plt.xlabel(xlabel = 'Tuberculosis Mortality')

#PLOT 14: Measles immunization Mean / Median check.
plt.subplot(18, 2, 14) #18 rows, 2 columns, spot 14

# Histogram
sns.distplot(a = measles_immunization_dropped['measles immunization'],
             bins  = 'fd',
             hist  = True,
             kde   = False,
             rug   = False,
             color = 'gray')

# Titles and labels
plt.title(label = "Figure 18:\nDistribution of Measles Immunization")
plt.xlabel(xlabel = 'Measles Immunization')
plt.ylabel(ylabel = 'Frequency')


# Vertical lines for mean and median
plt.axvline(x = measles_immunization_dropped['measles immunization'].mean(),
            color = 'maroon')

plt.axvline(x = measles_immunization_dropped['measles immunization'].median(),
            color = 'darkorange')

# Legend
plt.legend(labels =  ['mean', 'median'])

#PLOT 15: Measles Imputation NaN Values Distribution
plt.subplot(18, 2, 15) #18 rows, 2 columns, spot 15

# Histogram for Tuberculosis mortalitY (without NaN)
sns.distplot(a     = measles_immunization_dropped['measles immunization'],
             bins  = 'fd',
             hist  = True,
             kde   = True, # activating kde
             rug   = False,
             color = 'gray')

# Histogram (with imputation)
sns.distplot(a     = arabian_peninsula_adj['measles immunization'],
             bins  = 'fd',
             hist  = True,
             kde   = True, # activating kde
             rug   = False,
             color = 'deepskyblue')


# Titles, labels, and formatting
plt.title(label   = "Figure 19:\n Measles Immunization Distribution\n(With/without imputation of missin
g values)")
plt.xlabel(xlabel = 'Measles Immunization')
plt.ylabel(ylabel = 'Frequency')

# This adds a legend
plt.legend(labels =  ['original distribution',
                      'imputed (mean) distribution'])

# PLOT 16: Measles Imputation Outliers
plt.subplot(18, 2, 16) #18 rows, 2 columns, spot 16

# Developing a boxplot for Measles Immunization
sns.boxplot(x      = 'measles immunization',  # x-variable
            y      = None,     # optional y-variable
            hue    = None,     # optional categorical feature
            orient = 'h',      # horizontal or vertical
            data   = arabian_peninsula_adj, # DataFrame where features exist
            color = "skyblue")

# Adding a line to signify an outlier threshold
plt.axvline(x = 79,
           color = 'red',
            linestyle= '--')

# Formatting and displaying the plot
plt.title(label = 'Figure 20:\nMeasles Immunization Outliers')
plt.xlabel(xlabel = 'Measles Immunization')

# PLOT 17: Life Expectancy Outliers
plt.subplot(18, 2, 17) #18 rows, 2 columns, spot 17

# Developing a boxplot for life expectancy
sns.boxplot(x      = 'life expectancy',  # x-variable
            y      = None,     # optional y-variable
            hue    = None,     # optional categorical feature
            orient = 'h',      # horizontal or vertical
            data   = arabian_peninsula_adj, # DataFrame where features exist
            color = "skyblue")

# Formatting and displaying the plot
plt.title(label = 'Figure 21:\nLife Expectancy Outliers')
plt.xlabel(xlabel = 'Life Expectancy Age')

# PLOT 18: Life Expectancy Outliers
plt.subplot(18, 2, 18) #18 rows, 2 columns, spot 18

# Developing a boxplot for life expectancy
sns.boxplot(x      = 'fertility',  # x-variable
            y      = None,     # optional y-variable
            hue    = None,     # optional categorical feature
            orient = 'h',      # horizontal or vertical
            data   = arabian_peninsula_adj, # DataFrame where features exist
            color = "skyblue")

# Formatting and displaying the plot
plt.title(label = 'Figure 22:\nFertility Outliers')
plt.xlabel(xlabel = 'Fertility Rate')

# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell
plt.close()

# PLOT 19: Correlation Aids Deaths, HIV Cases, ART Coverage compared to Populations.

#Copy the dataset to determine the correlation.
correlation_hiv_aids_coverage = arabian_peninsula.loc[ : , ['hiv cases', 'aids deaths','art coverage',
'population']].copy()

# Dropping null-values (because this cannot be compared)
correlation_hiv_aids_coverage = correlation_hiv_aids_coverage.dropna(axis = 0)

#Convert data to the right percentage.
correlation_hiv_aids_coverage['art coverage'] = correlation_hiv_aids_coverage['art coverage'] /100

# Plot the correlation.
    # Size for the population
marker_size = [10, 15, 20, 25, 30]

sns.lmplot(x          = 'hiv cases',  # x-axis feature
           y          = 'art coverage',  # y-axis feature
           hue        = 'aids deaths',
           legend_out = False,    # formats legend if hue != None
           scatter    = True,     # renders a scatter plot
           fit_reg    = False,    # renders a regression line
           aspect     = 2,        # aspect ratio for plot
           data       = correlation_hiv_aids_coverage,
           scatter_kws={'s':marker_size})# DataFrame where features exist


    # Formatting and displaying the plot
plt.title(label    = 'Figure 10:\nCorrelation HIV Cases, Treatment, Aids Deaths')
plt.xlabel(xlabel  = 'Hiv Cases % of population')
plt.ylabel(ylabel  = 'Art Coverage % of population')

    # Reset size for x axis
plt.xticks([0.09, 0.10, 0.11])


# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell
plt.close()

############################ Environment Plot  #############################

# PLOT 20: CO2 per Top 10 Countries for all_regions

# Set figure size
fig, ax = plt.subplots(figsize=[15, 8])

splot = sns.barplot(data=all_regions.sort_values(by='co2 emissions',
                                                 ascending=False).head(n = 11),
                    x='country',
                    y='co2 emissions',
                    orient='v',
                    color='grey')

for p in splot.patches:
    splot.annotate(format(p.get_height(), '.0f'),
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha='center',
                   va='center',
                   xytext=(0, 6),
                   textcoords='offset points')

# titles, labels, and formatting
plt.title(label="""
Figure 9:\nTop 10 Countries in terms of CO2 Emissions per Capita""")
plt.xlabel(xlabel='Country')
plt.xticks(rotation=30)
splot.spines['right'].set_visible(False)
splot.spines['top'].set_visible(False)
splot.spines['left'].set_visible(False)
splot.axes.yaxis.set_visible(False)

############################################################################

# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell
plt.close()
```

3. Obscure Findings

3.1 Null Values

Figure 3.1 displays the count of null values in the region before dropping or imputations. It is remarkable that PSE and SYR, the countries with a long-lasting history of war, have the most missing values, while CYP and TUR, the two countries which are members/candidates for the EU, have the least missing values. Thus, we flagged these countries as outliers.

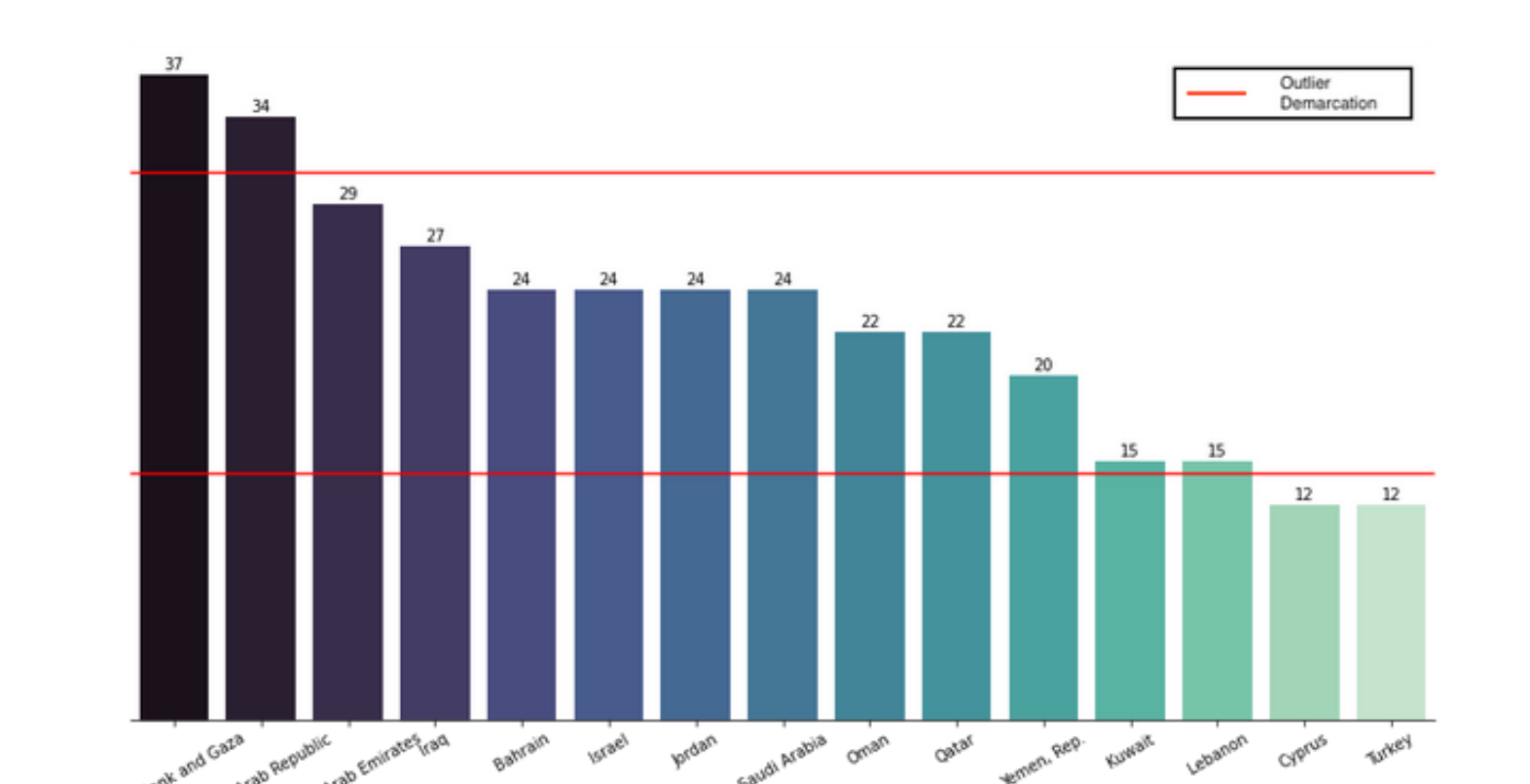

Figure 3.1: Null values per Country in Arabian Peninsula

```python
# Barplot for Null Values per Country (change plt.close() to plt.show to display plot)
############################################################################

# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell

# Set figure size
fig, ax = plt.subplots(figsize=[15, 8])

splot = sns.barplot(data=all_regions_flagged.loc[arabian_peninsula_country_codes, :].sort_values(by='mv
_perc',
                                                                                                  ascend
ing=False),
                    x='country',
                    y='mv_perc',
                    orient='v',
                    palette='mako')

for p in splot.patches:
    splot.annotate(format(p.get_height(), '.0f'),
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha='center',
                   va='center',
                   xytext=(0, 6),
                   textcoords='offset points')

# labels, and formatting
plt.xlabel(xlabel='Country')
plt.xticks(rotation=30)
plt.ylabel(ylabel='Null % of Total Indicators')
plt.axhline(y=31, color='red')
plt.axhline(y=14, color='red')
splot.spines['right'].set_visible(False)
splot.spines['top'].set_visible(False)
splot.spines['left'].set_visible(False)
splot.axes.yaxis.set_visible(False)

# Save plot as image
plt.savefig(
    fname='./_images/jupyter-exports/null-values-country.png')

# If you want to display the plots, please change plt.close() to plt.show()
plt.close()
```

3.2 Accuracy

Many of the organizations that collect data rely on government reported information. Due to instability in many of the countries in the Arabian Peninsula, not much information is available. The majority of data gathered are estimates, or collected from the latest census, which varies from 1963 to 2018, making it incomparable. Furthermore, the Weighted Average aggregation method renders some inaccuracy as the weights are unknown and many of the original and external sources consider the most recent reported values. Figure 3.3 exemplifies the difference between weighted average to more recent data, showing that weighted averages increase outliers.

Figure 3.2: Weighted Average Aggregation Criticism

```python
# WA Aggregation Method Analysis (change plt.close() to plt.show to display plot)

# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell

############################################################################
# Prepare Data

# Import internet_users original data
internet_df = pd.read_csv(filepath_or_buffer = "./_datasets/evidence/internet_users.csv",
                          header             = 2)

# Join Income Group to internet_users
internet_df = internet_df.merge(income_group,
                                how = 'inner',
                                left_on='Country Code',
                                right_on='Code')

# Set index to country code and drop unnecessary columns
internet_df.set_index('Code', inplace = True)
internet_df.drop(['Unnamed: 65',
                  'Country Code'], axis = 1, inplace = True)


############################################################################
# Boxplots of Internet Users

# Set figure size
fig, ax = plt.subplots(figsize = [15, 7])

# Plot the first graph: weighted average
plt.subplot(1, 2, 1)

# Boxplot for internet users
sns.boxplot(x      = 'internet users',
            y      = 'Income Group',
            orient = 'h',
            color  = 'skyblue',
            data   = all_regions.loc[arabian_peninsula_country_codes, :])


# titles and labels
plt.title(label = """WA Aggregation Method (1960 to 2020)""",
          pad = 20.0)
plt.xlabel(xlabel = 'Internet Users')
plt.ylabel(ylabel = 'Income Group')


# Plot the second graph: cellular subscriptions
plt.subplot(1, 2, 2)

# Boxplot for cellular subscriptions
sns.boxplot(x      = '2017',
            y      = 'Income Group',
            orient = 'h',
            color  = 'skyblue',
            data   = internet_df.loc[arabian_peninsula_country_codes, '2017':])


# titles, labels, and formatting
plt.title(label   = """Most Recent Year (2017)""",
          pad = 20.0)
plt.xlabel(xlabel = 'Internet Users')
plt.ylabel(ylabel = 'Income Group')

#compile and display the plot
plt.tight_layout(pad = 5.0)

# If you want to display the plots, please change plt.close() to plt.show()
plt.close()
```

3.3 Correlations

Figure 3.3 suggests three obscure correlations between indicators:

1. The high correlation between GNI per capita and CO2 emissions (0.9) is driven by the significant oil reserves, and the correlated energy production in the region (Al-mulali, 2011).
2. The strong negative correlation between life expectancy and fertility (-0.79) refers to the evolutionary trade-off between reproduction and body maintenance. Fertility comes at the cost of the mother: the process of birthing and raising many children is demanding and can have negative effects on womens' health and lifespan (Liefbroer, et al., 2015).
3. The identified positive correlation between malaria cases and tuberculosis mortality (0.92) cannot be verified by external sources. We assume, the correlation formed due to the high numbers of Yemen in both indicators, a country reported as an outlier within the health section.

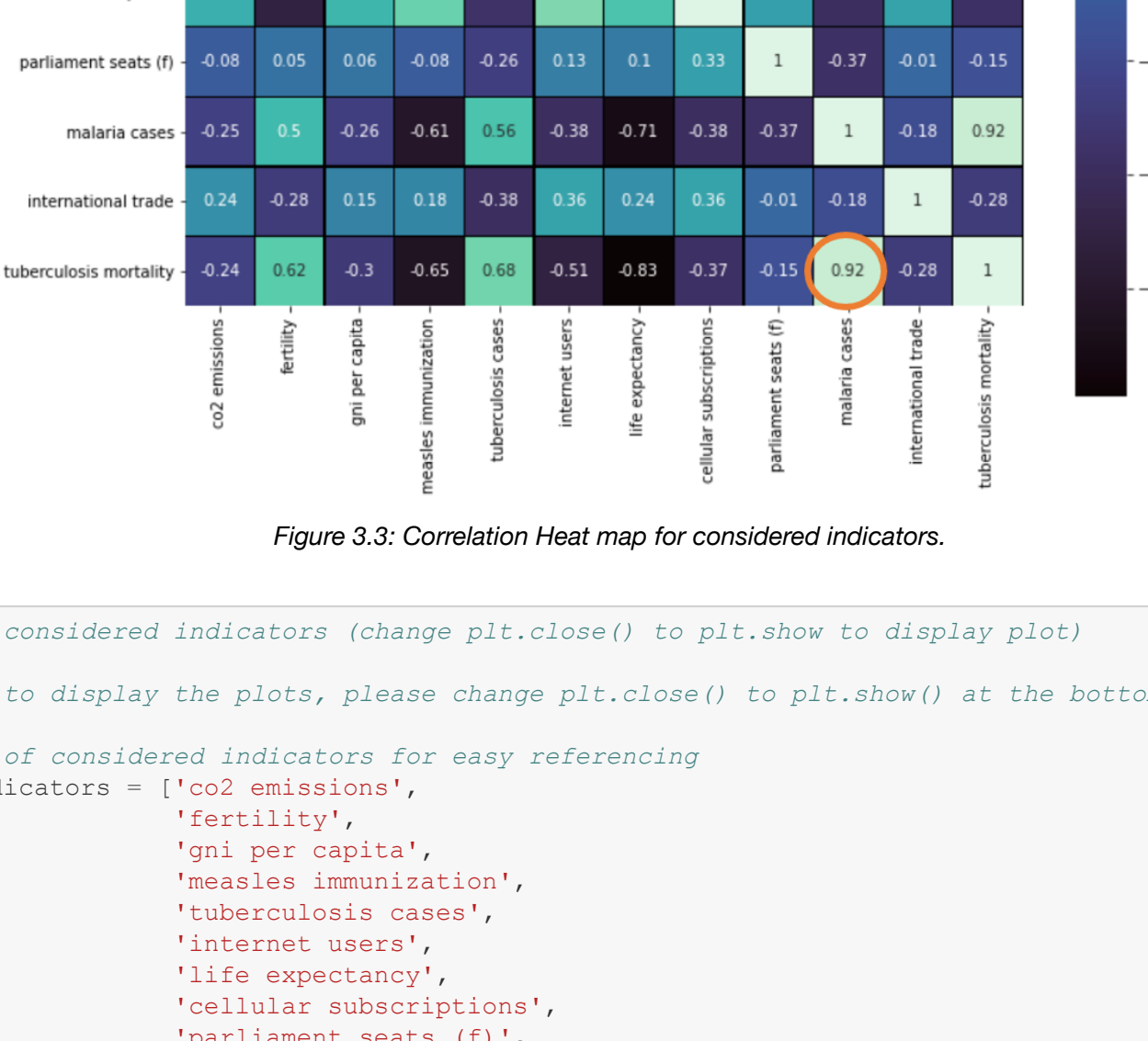

Figure 3.3: Correlation Heat map for considered indicators.

```python
# Heatmap for considered indicators (change plt.close() to plt.show to display plot)

# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell

# Create list of considered indicators for easy referencing
considered_indicators = ['co2 emissions',
                         'fertility',
                         'gni per capita',
                         'measles immunization',
                         'tuberculosis cases',
                         'internet users',
                         'life expectancy',
                         'cellular subscriptions',
                         'parliament seats (f)',
                         'malaria cases',
                         'international trade',
                         'tuberculosis mortality']

# converting correlation matrix into a DataFrame
cons_indicators_corr = arabian_peninsula_adj.loc[:, considered_indicators].corr(method = 'pearson').rou
nd(decimals = 2)

# specifying plot size (making it bigger)
fig, ax = plt.subplots(figsize=(12,12))


# developing a spicy heatmap
sns.heatmap(data       = cons_indicators_corr, # the correlation matrix
            cmap       = 'mako',
            robust     = True,
            square     = True,
            annot      = True,
            linecolor  = 'black',
            linewidths = 0.25)


# title and displaying the plot
plt.title("""
Linear Correlation Heatmap for Considered Indicators
""")

# If you want to display the plots, please change plt.close() to plt.show()
plt.close()
```

4. Representative Country

The considered indicators are used to identify the country which best represents our region. For each indicator, the countries have been ranked according to their difference from the respective indicator mean. Figure 4.1 displays the average ranking.

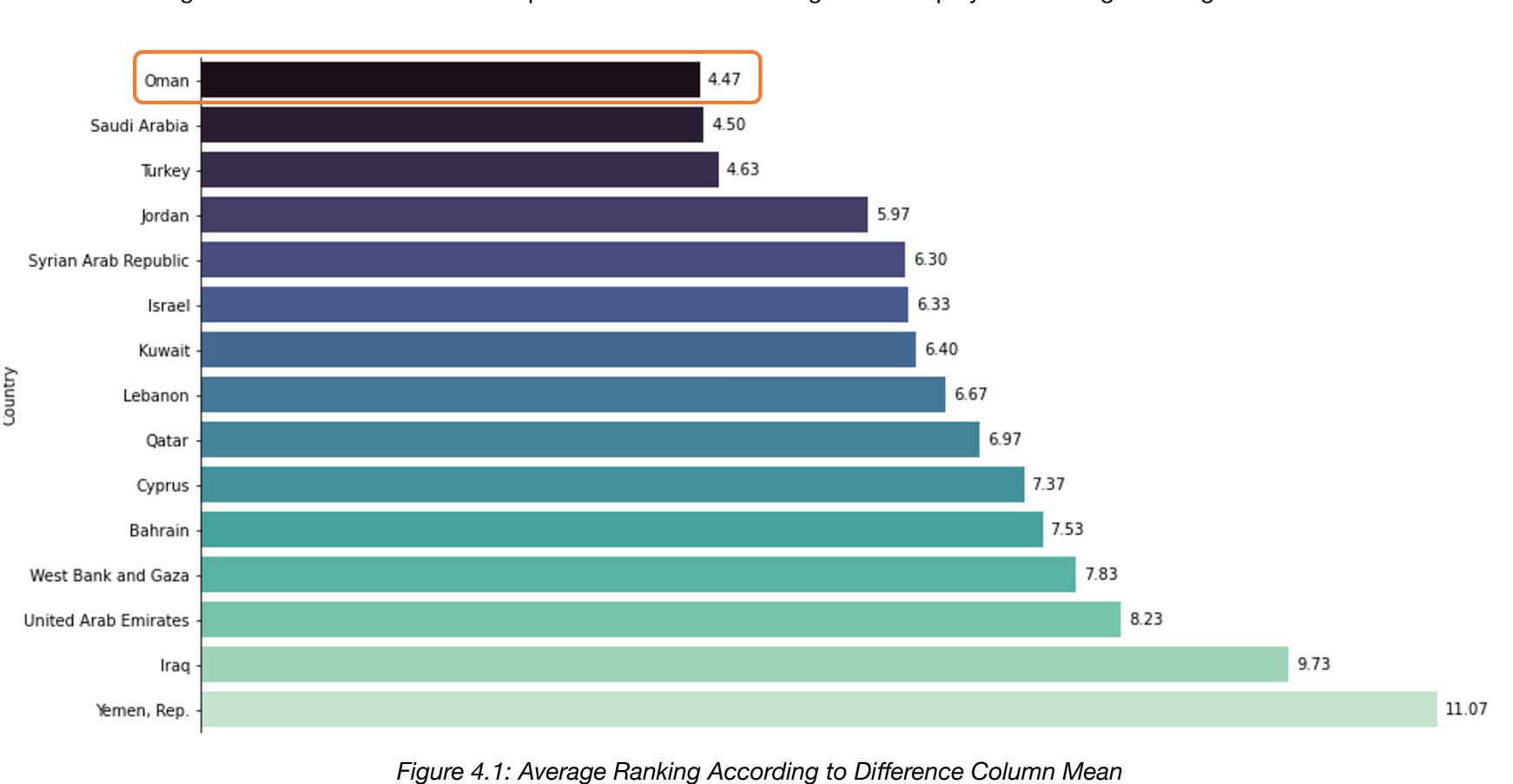

Figure 4.1: Average Ranking According to Difference Column Mean

Oman is the country which best represents the Arabian Peninsula region.

However, we have to consider missing values and outlier flags. Oman has one missing value for the considered indicators: Parliament seats (f). This has been imputed with the value published by the data source. Furthermore, Oman is flagged as an outlier for internet users in high-income countries. As explained in chapter 3.2, this indicator is a weighted average and has its structural flaws. Hence, there is no reason to discard Oman as a representative country.

In [ ]:
```
## Data Preparation

################################################################################
## 1. Import Packages ##
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math

################################################################################
## 2. Import Adjusted Arabian Peninsula Data ##

# Creating variable for filepath.
file = "./_datasets/saved_datasets/arabian_peninsula.xlsx"


# Create a DataFrame from our adjusted data
arabian_peninsula_adj = pd.read_excel(  io         = file,
                                        sheet_name = "outliers flagged",
                                        header     = 0,
                                        index_col  = 0 )

################################################################################
## 3. Set Display Conditions ##

# Display all columns of datasets
pd.set_option('display.max_columns', None)

# Display all floats in 2 decimal spaces
pd.options.display.float_format = "{:,.2f}".format
```

In [ ]:
```
## Identify representing country ##

#loop through the columns of arabian_peninsula_adj to calculate variance to column mean and rank accord
ingly
for col in arabian_peninsula_adj:

    #if column begins with m, mv, o or is region, country or income group, move on (continue)
    if 'm_' in col or\
    'mv_' in col or\
    'o_' in col or\
    'region' in col or\
    'country' in col or\
    'Income Group' in col:
        continue

    else:
        #Create variance column (var_col): Difference between column value and column mean
        arabian_peninsula_adj['var_'+col] = abs(arabian_peninsula_adj.loc[:, col] - arabian_peninsula_a
dj.loc[:, col].mean())

        #Create ranking column per indicator: Ranking of the variation, ascending
        arabian_peninsula_adj['rank_'+col] = arabian_peninsula_adj['var_'+col].rank( ascending = True)


# correct ranking for PSE Parliament Seats (f) which was not imputed
arabian_peninsula_adj.loc['PSE', 'rank_parliament seats (f)'] = 15

# create sum of all the rankings
arabian_peninsula_adj['rank_sum'] = arabian_peninsula_adj['rank_co2 emissions'] +\
                                    arabian_peninsula_adj['rank_fertility'] + \
                                    arabian_peninsula_adj['rank_gni per capita'] + \
                                    arabian_peninsula_adj['rank_measles immunization'] + \
                                    arabian_peninsula_adj['rank_tuberculosis cases'] + \
                                    arabian_peninsula_adj['rank_internet users'] + \
                                    arabian_peninsula_adj['rank_life expectancy'] + \
                                    arabian_peninsula_adj['rank_cellular subscriptions'] + \
                                    arabian_peninsula_adj['rank_population'] + \
                                    arabian_peninsula_adj['rank_parliament seats (f)'] + \
                                    arabian_peninsula_adj['rank_malaria cases'] + \
                                    arabian_peninsula_adj['rank_international trade'] + \
                                    arabian_peninsula_adj['rank_tuberculosis mortality']


arabian_peninsula_adj['rank_avg'] = arabian_peninsula_adj.loc[: , 'rank_sum'] / 15

# for better visualisation: create 'representative candidate' column by ranking of rank_sum
arabian_peninsula_adj['representative candidate']= arabian_peninsula_adj['rank_sum'].rank( ascending =
True)

#arabian_peninsula_adj.loc[:,['country','rank_sum', 'rank_avg', 'representative candidate']].sort_value
s( by = 'rank_sum')
```

In [ ]:
```
## Plot average ranking (change plt.close() to plt.show to display plot)##

# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell

# Set figure size
fig, ax = plt.subplots(figsize=[15, 8])

plt.subplot(1, 1, 1)  # 1st row, 1st column, 1st space

splot = sns.barplot(data=arabian_peninsula_adj.sort_values(by='rank_avg',
                                                           ascending=True),
                    x='rank_avg',
                    y='country',
                    orient='h',
                    palette='mako')

for p in splot.patches:
    splot.annotate("%.2f" % p.get_width(),
                   (p.get_x() + p.get_width(),
                    p.get_y()),
                    xytext=(5, -15),
                    textcoords='offset points')

# labels, and formatting
plt.ylabel(ylabel='Country')
splot.spines['right'].set_visible(False)
splot.spines['top'].set_visible(False)
splot.spines['bottom'].set_visible(False)
splot.axes.xaxis.set_visible(False)

# Save plot as image
#plt.savefig(fname='./_images/jupyter-exports/rep_country_ranking.png')


# If you want to display the plots, please change plt.close() to plt.show()
plt.close()
```

---

5. Differentiators of the region

The top 5 indicators making our region unique have been identified by comparing indicator means to respective values in the world row of the original dataset. Population has not been considered due to outliers like India and China who report the value in billions while everyone else is reporting in millions.

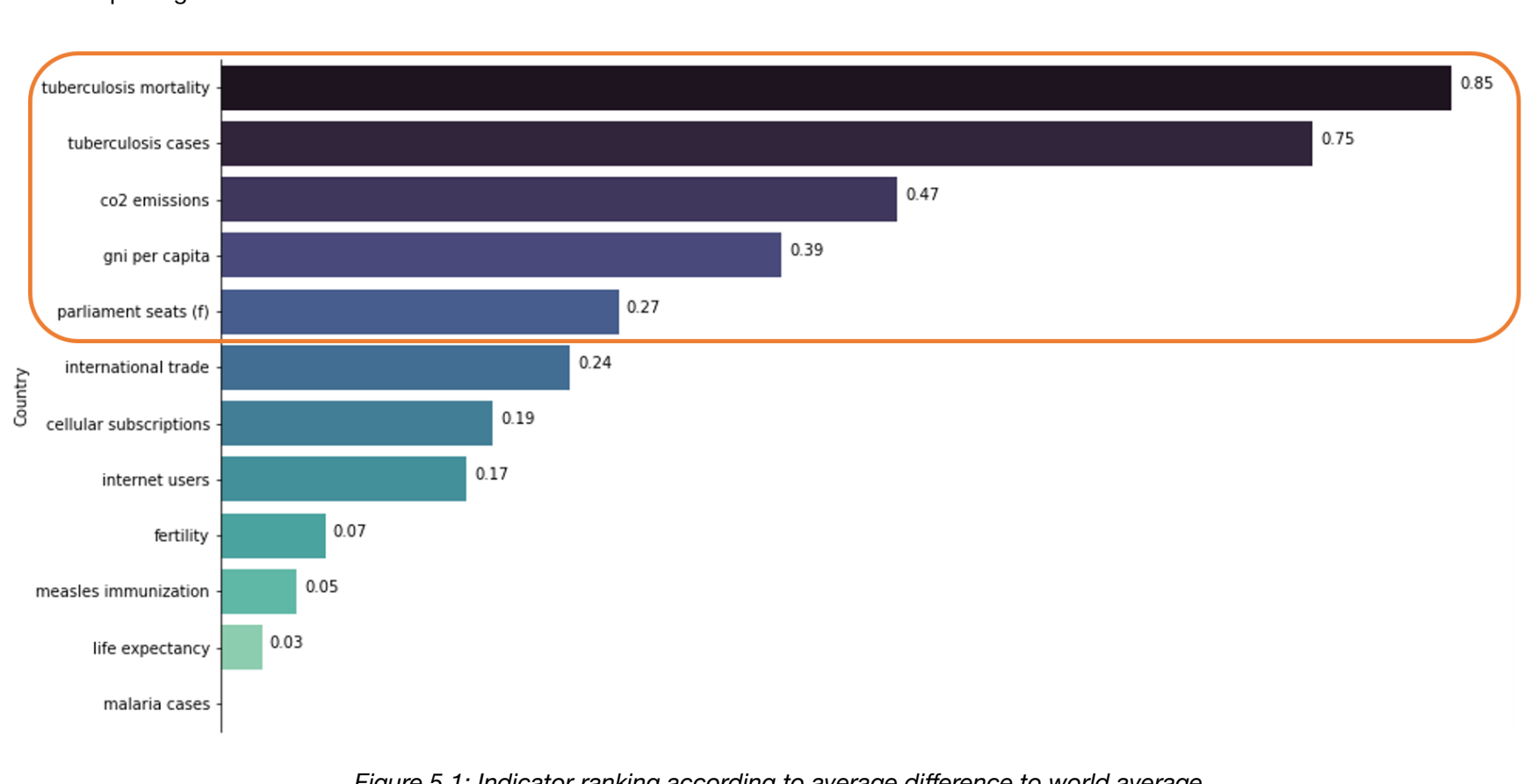

Figure 5.1: Indicator ranking according to average difference to world average

The five indicators with the highest average difference best represent our region (figure 5.1):

- **Tuberculosis Mortality**
- **Tuberculosis Cases**
- **CO2 Emissions**
- **GNI per Capita**
- **Parliament Seats Hold by Women**

Compared to the world, our region has significantly lower tuberculosis cases and mortality. GNI per capita and the correlated CO2 emissions exceed the world average by far due to the vast natural oil reserves (chapter 3.3). Given the fact that 50% of the countries in our region are within the last 15 ranks of the latest Global Gender Gap report, the prevailing low gender equality is displayed by the number of parliament seats held by women, which is below the world average.

In [ ]:
```
# Data Preparation

################################################################################
## 1. Import Packages ##
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math

################################################################################
## 2. Import Adjusted Arabian Peninsula Data ##

# Creating variable for filepath.
file = "./_datasets/saved_datasets/arabian_peninsula.xlsx"
original = "./_datasets/saved_datasets/all_regions_flagged.xlsx"


# Import adjusted data (arabian_peninsula_adj)
arabian_peninsula_adj = pd.read_excel(io          = file,
                                      sheet_name = "outliers flagged",
                                      header     = 0,
                                      index_col  = 0)

# Import all regions data (all_regions_flagged)
all_regions = pd.read_excel(io         = original,
                            sheet_name = "all_regions_null",
                            header     = 0,
                            index_col  = 0)


################################################################################
## 3. Set Display Conditions ##

# Display all columns of datasets
pd.set_option('display.max_columns', None)

# Display all floats in 2 decimal spaces
pd.options.display.float_format = "{:,.2f}".format


################################################################################
## 4. Prepare Dataframe for Arabian Peninsula vs World ##

# Create list of considered indicators for easy referencing
considered_indicators = ['co2 emissions',
                         'fertility',
                         'gni per capita',
                         'measles immunization',
                         'tuberculosis cases',
                         'internet users',
                         'life expectancy',
                         'cellular subscriptions',
                         'parliament seats (f)',
                         'malaria cases',
                         'international trade',
                         'tuberculosis mortality']

# Instantiate dataframe from arabian_peninsula_adj
arabia_vs_world = arabian_peninsula_adj.loc[:, :'Income Group'].copy().drop(['country',
                                                                             'region',
                                                                             'population',
                                                                             'Income Group'],
                                                                            axis = 1)

# Calculate mean for each considered indicator for arabian_peninsula
arabia_vs_world.loc['AP'] = arabian_peninsula_adj.mean()

# Append World row from all_regions to arabia_vs_world
arabia_vs_world.loc['WLD'] = all_regions.loc['WLD', considered_indicators]

# Display only Arabian Peninsula and World Figures
arabia_vs_world = arabia_vs_world.loc[['AP', 'WLD'], :]

# transpose rows and columns of arabia_vs_world
arabia_vs_world = arabia_vs_world.transpose( copy = True)

# set column names for the transposed data frame
arabia_vs_world.columns = ["AP", "WLD"]

# add the indicator names as a column
arabia_vs_world.insert (loc = 0, column = 'considered indicators', value = considered_indicators)
```

In [ ]:
```
## Calculate difference to world and rank indicators ##

# Calculate the average distance between world and region
arabia_vs_world['avg difference'] = abs((arabia_vs_world.loc[:, "AP"] - arabia_vs_world.loc[: , "WLD"
])/(arabia_vs_world.loc[:, "AP"] + arabia_vs_world.loc[: , "WLD"]))
arabia_vs_world['rank']     = arabia_vs_world['avg difference'].rank(ascending = False)

# Display result
#arabia_vs_world.loc[:,:].sort_values (by = 'rank')
```

In [ ]:
```
## Plot indicator ranking (change plt.close() to plt.show to display plot)##

# If you want to display the plots, please change plt.close() to plt.show() at the bottom of the cell

# Set figure size
fig, ax = plt.subplots(figsize=[15, 8])

plt.subplot(1, 1, 1)  # 1st row, 1st column, 1st space

splot = sns.barplot(data=arabia_vs_world.sort_values(by='rank',
                                                     ascending=True),
                    x='avg difference',
                    y= 'considered indicators',
                    orient='h',
                    palette='mako')

for p in splot.patches:
    splot.annotate("%.2f" % p.get_width(),
                   (p.get_x() + p.get_width(),
                    p.get_y()),
                    xytext=(5, -15),
                    textcoords='offset points')

# labels, and formatting
plt.ylabel(ylabel='Country')
splot.spines['right'].set_visible(False)
splot.spines['top'].set_visible(False)
splot.spines['bottom'].set_visible(False)
splot.axes.xaxis.set_visible(False)

# Save plot as image
#plt.savefig(fname='./_images/jupyter-exports/indicator_rank.png')


# If you want to display the plots, please change plt.close() to plt.show()
plt.close()
```

6. Conclusion

Our region shines in diversity. Some countries face on-going civil war, others are economically prosperous due to their natural oil reserves.

We explored 41 indicators and determined that only 12 revealed meaningful insights (chapter 2). This was due to a variety of obscure findings regarding null values, data accuracy, and indicator correlations (chapter 3).

Afterwards, we determined Oman as the best representative of the Arabian Peninsula based on a statistical analysis revolving around each indicator mean (chapter 4).

Lastly, we compared & contrasted the mean of each indicator for both the Arabian Peninsula and the world. We identified 5 differentiating indicators for the Arabian Peninsula (chapter 5):

1. **Tuberculosis mortality**
2. **Tuberculosis cases**
3. **CO2 emissions**
4. **GNI per capita**
5. **Parliament seats held by women**

These indicators should be closely monitored by the countries of the region to develop future strategies for the triple bottom line: *social, environmental, economic sustainability*.

In [ ]: