


```
# 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',
                                     'top income share', 'm_top 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 = "w") as writer:
    arabian_peninsula_adj.to_excel(writer, sheet_name="dropped indicators")

In [ ]:

# 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 frame
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)']].copy()

# 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
PenParl_OMN_imputation = 2.33 # https://www.ipu.org/parliament/QA
PenParl_QAT_imputation = 9.76 # https://www.ipu.org/parliament/QA
PenParl_SAU_imputation = 19.87 # https://www.ipu.org/parliament/QA

# fill values for OMN, QAT, SAU
arabian_peninsula_adj.loc['OMN', 'parliament seats (f)'] = PenParl_OMN_imputation
arabian_peninsula_adj.loc['QAT', 'parliament seats (f)'] = PenParl_QAT_imputation
arabian_peninsula_adj.loc['SAU', 'parliament seats (f)'] = PenParl_SAU_imputation

#####

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

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

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

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

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

# Impute value for Malaria
malaria_mortality_imputation = 0

# fill null values for malaria in arabian_peninsula_adj
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(), ndigits = 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 [ ]:

# 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_internet_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

    # Parliament seats held 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 figure 3.1
arabian_peninsula_adj.loc['SYR', 'o_null values'] = 1 #high number of missing values, identified by figure 3.1
arabian_peninsula_adj.loc['CYP', 'o_null values'] = 1 #low number of missing values, identified by figure 3.1
arabian_peninsula_adj.loc['TUR', 'o_null values'] = 1 #low number of missing values, identified by figure 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_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_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 = "w") as writer:
    arabian_peninsula_adj.to_excel(writer, sheet_name="outliers flagged")
```



```
In [ ] : # Additional Plot, saved 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 values)""",
          pad = 10.0)
plt.xlabel(xlabel = 'GNI per Capita [USD]')
plt.ylabel(ylabel = 'Frequency')
plt.xlim(0.0, 70000)
plt.ylim(0.0, 0.0004)

# 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(0.0, 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 = 'parliament_f_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)

# 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 imputon
# 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 = 'mariacases_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 values)""",
          pad = 10)
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:\nMalaria 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
            aspect = '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 missing values)""",
          pad = 10)
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 immunisation'),
             bins = 1,
             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 missing values)""",
          pad = 10)
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 Population.
#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', # formats legend if hue != None
            legend_out = False,
            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([0.05, 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))

sp1ot = 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 sp1ot.patches:
    sp1ot.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)
plt.axline(y=31, color='red')
plt.xlabel(ylabel='Total CO2 Emissions')
plt.spines('right').set_visible(False)
plt.spines('top').set_visible(False)
plt.spines('left').set_visible(False)
plt.spines('bottom').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()
```

```
In [ ] : # Boxplot 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))

sp1ot = sns.barplot(data=all_regions_flagged.loc[arabian_peninsula_country_codes, :].sort_values(by='mv_perc',
                                                    ascending=False),
                    x='country',
                    y='mv_perc',
                    orient='v',
                    palette='mako')

for p in sp1ot.patches:
    sp1ot.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.xlabel(ylabel='Null % of Total Indicators')
plt.axline(y=14, color='red')
plt.spines('right').set_visible(False)
plt.spines('top').set_visible(False)
plt.spines('left').set_visible(False)
plt.spines('bottom').set_visible(False)

# Save plot as image
plt.savefig('./_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 incompatible. 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 3.3: Correlation Heat map for considered indicators.

3.3 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.

```
In [ ] : # 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']

# Comparing correlation matrix into a DataFrame
cons_indicators_corr = arabian_peninsula_adj.loc[:, considered_indicators].corr(method = 'pearson').round(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,
            linewidth = '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.

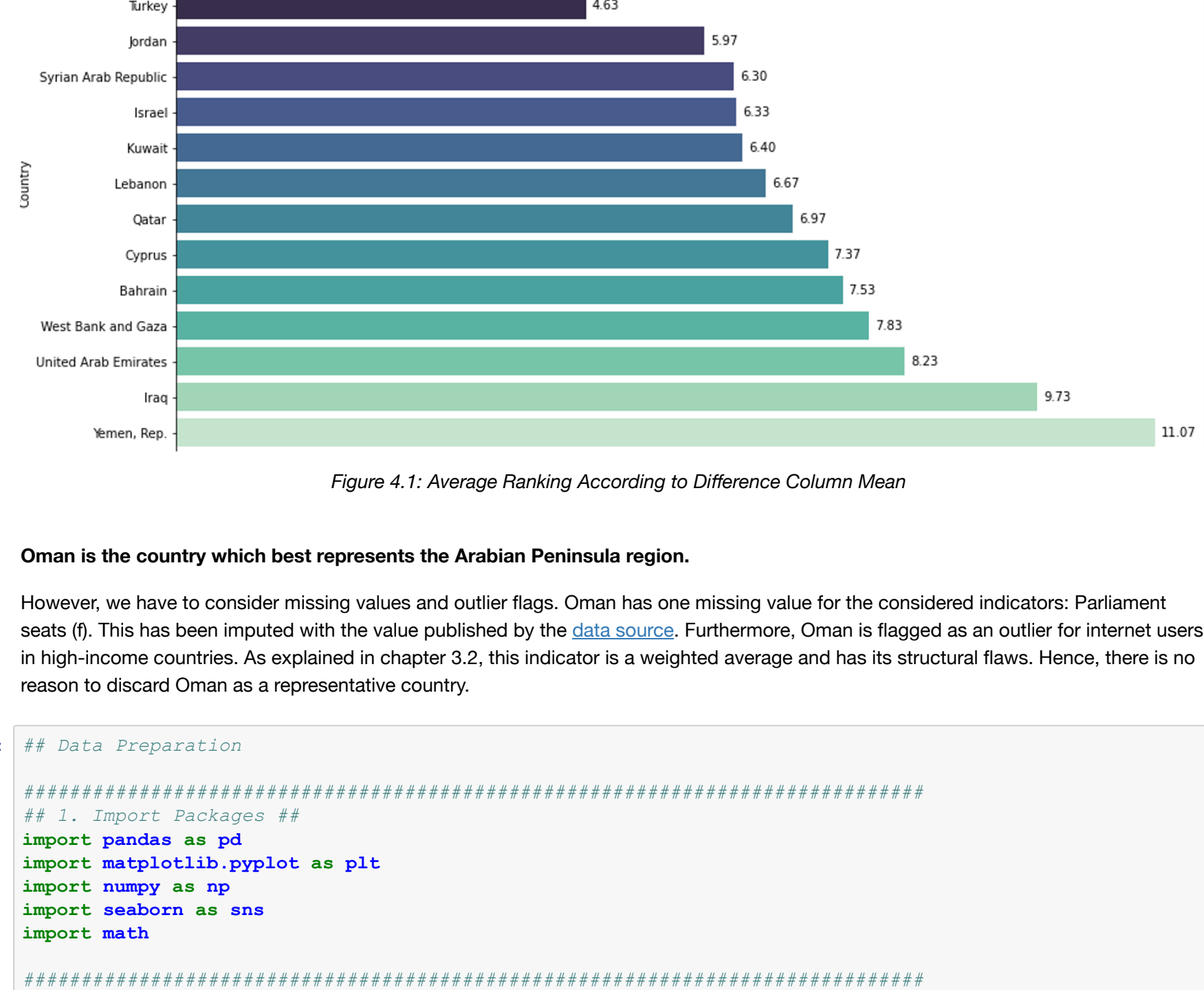


Figure 4.1: Average Ranking According to Difference Column Mean

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
inply
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['rank_' + 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['rank_' + col].rank(ascending = True)

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

# create sum of all the rankings
arabian_peninsula_adj['rank_sum'] = arabian_peninsula_adj['rank_co2 emissions'] + \
    arabian_peninsula_adj['rank_gni per capita'] + \
    arabian_peninsula_adj['rank_fertility'] + \
    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_valu
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(filename="../images/jupyter-exports/top_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 results 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 Held 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
# 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 considered 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(filename="../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 [ ]:
```