Global Affairs Canada - Compliance Report Documentation

Release 1.0

Rory Scott

Dec 20, 2017

CONTENTS:

CHAPTER

ONE

INTRODUCTION

This initial report will give a comprehensive review of the 'compliance' of GAC's IATI activities according to the well established aspects of IATI data quality, namely:

- Schema validity: a simple test of adherence to the IATI standard XML syntax
- Adherence to IATI rulesets: test all activities for adherence to the IATI Standard machine readable rulesets.
- A deeper view of the fields published with respect to the use cases 3.1.1-6 detailed below.

CHAPTER

TWO

SETUP AND DATA ACQUISITION

This section may be of interest on a technical level, but isn't informative with regards to the analytical scope of this report. In short, it sets up the notebook and then processes the given IATI files and aggregates them for analysis. Some changes will need to be made here if a user or practitioner wishes to supply local IATI XML files rather than using the ones pulled from the IATI Registry.

2.1 Imports and Housekeeping

The following value is **''True''**, the notebook attempts to get files and validate them over the network. If **''False''**, it assumes that these processes have been run before, meaning the required files would be in the raw and intermediate data folders and it can defer to them instead.

```
In [303]: run_from_scratch = False
```

There are several libraries required by Python to conduct the analysis and visualisation in this notebook, which are all imported below.

This notebook uses a three part files system to store data:

- **''raw''**: this folder should contain all and only the IATI XML of interest. The initial use of this notebook is to analyse live IATI data, hosted at specific URLs. This notebook can be adapted to use only local files, and instructions for this are given in-line.
- **''intermediate''**: this folder is used as a store of data which has been processed and might be of interest, is used multiple times, or is useful to have in case it is desirable to run the notebook off-line.
- **''final''**: this folder contains any data appendices. For instance, an Excel workbook is built over the course of this notebook, with data tables that might be of interest included.

```
In [304]: # Import Python libraries
          import json
          import re
          import pandas as pd
          import requests as rq
          from bokeh.charts import Histogram, output_notebook, show
          from datetime import datetime

          # set directories
          RAW = "../data/raw/"
          INTERMEDIATE = "../data/intermediate/"
          FINAL = "../data/final/"
          REGISTRY_ID = "gac-amc"
          # Create and Excel Writer to output sheets throughout the analysis
```

```
appendix_1_filepath = FINAL + 'Compliance-Report-Data.xlsx'
pd_writer = pd.ExcelWriter(appendix_1_filepath)

output_notebook()
```

2.2 Getting Original IATI Files

Retrieving the IATI files from supplied URLs and saving them to the 'raw' data folder. The cell below can be skipped if unpublished / local files are being used. Just save all and only the IATI XML files of interest into the 'raw' data folder.

```
In [305]: # List of all XML urls to pull and merge
          registry_files = [
              "http://w05.international.gc.ca/projectbrowser-banqueprojets"
                  "/iita-iati/dfatd-maecd_activit_status_2_3.xml",
              "http://w05.international.gc.ca/projectbrowser-banqueprojets"
                  "/iita-iati/dfatd-maecd_activit_status_4.xml"
          ]

          if run_from_scratch:
              for registry_file in registry_files:

                  # split off after the last dash to create the file name, so http://www.abc/def.xml
                  registry_xml_name = re.search(r'[^/]*.xml', registry_file).group(0)
                  output_path = RAW + registry_xml_name

                  request = rq.get(registry_file)

                  with open(output_path, "wb") as out_file:
                      out_file.write(request.content)

                  print("{}: file written to {}".format(registry_xml_name, output_path))
          else:
              print('skipped - to run, change \'run_from_scratch\' to true...')
```

```
skipped - to run, change 'run_from_scratch' to true...
```

2.3 Merging IATI Files

```
In [306]: # Show IATI files available
          import os
          import lxml.etree as ET

          file_names = [RAW + name for name in os.listdir(RAW) if name.endswith(".xml")]
          for name in file_names:
              print(name)
```

```
../data/raw/dfatd-maecd_activit_status_2_3.xml
../data/raw/dfatd-maecd_activit_status_4.xml
```

```
In [307]: # This cell takes all of the XML IATI files in
          # the 'raw' directory and merges them into one file

          combined_filepath = INTERMEDIATE + "combined.xml"

          print("\nCombining {} IATI files \n".format(len(file_names)))
```

```
        # Start with the first file
        big_iati = ET.parse(file_names[0]).getroot()

        # Start a dictionary to keep track of the additions
        merge_log = {file_names[0]: len(big_iati.getchildren())}

        # Iterate through the 2nd through last file and
        # insert their activtities to into the first
        # and update the dictionary
        for xml_file in file_names[1:]:
            data = ET.parse(xml_file).getroot()
            merge_log[xml_file] = len(data.getchildren())
            big_iati.extend(data.getchildren())

        # Print a small report on the merging
        print("Files Merged: ")
        for file, activity_count in merge_log.items():
            print("|-> {} activities from {}".format(activity_count, file))
        print("|--> {} in total".format(len(big_iati.getchildren())))

        with open(combined_filepath, "wb") as out_file:
            out_file.write(ET.tostring(big_iati, encoding='utf8', pretty_print=True))
```

```
Combining 2 IATI files

Files Merged:
|-> 1210 activities from ../data/raw/dfatd-maecd_activit_status_2_3.xml
|-> 2751 activities from ../data/raw/dfatd-maecd_activit_status_4.xml
|--> 3961 in total
```

CHAPTER

THREE

INITIAL VALIDATION

This step uses the CoVE api to validate the combined XML file made above, which yields two sets of outcomes: the traditional schema validation, and validation against the IATI Rulesets. Using this api, both are returned in structured JSON, which has been used here to create succinct and opinionated tables, for example, by using a pivot table to see how many different rules have been broken, before attempting to list them all.

First, we send the file to CoVE and wait for it's response:

```
In [308]: import requests as rq

          json_validation_filepath = INTERMEDIATE + 'validation.json'

          if run_from_scratch:
              url = 'http://localhost:8000/api_test'
              files = {'file': open(INTERMEDIATE + "combined.xml", 'rb')}
              r = rq.post(url, files=files, data={"name": "combined.xml"})

              print("CoVE validation was successful.") if r.ok else print(
                  "Something went wrong.")

              validation_json = r.json()

              with open(json_validation_filepath, "w") as out_file:
                  json.dump(validation_json, out_file)

              print('Validation JSON file has been written to {}.'.format(
                  json_validation_filepath))

          else:
              validation_json = json.load(open(json_validation_filepath, 'r'))
              print('skipped - to run, change \'run_from_scratch\' to true...')
```

```
skipped - to run, change 'run_from_scratch' to true...
```

Now, let's take a look at the data we received back.

```
In [309]: ruleset_table = pd.DataFrame(data=validation_json['ruleset_errors'])
          schema_table = pd.DataFrame(data=validation_json['validation_errors'])

          print("CoVE has found {} schema errors, and {} ruleset errors".format(
              len(schema_table), len(ruleset_table)))
```

```
CoVE has found 3477 schema errors, and 30 ruleset errors
```

3.1 Schema Validation

Before looking at all of the specific validation errors, let's use a pivot table to uncover how many types of errors there are:

The numbers you see under 'path' and 'value' are counts, which allows this function to serve as a count of the number of schema violations associated with each 'description' (each schema rule).

```
In [310]: schema_table.pivot_table(index='description', aggfunc='count')
```

```
Out[310]: path  value
          description
          'document-link', attribute 'url' is not a valid...     2      2
          'result': Missing child element(s), expected is...  3475   3475
```

Only two types are found. The first two shown below indicate that two activities have invalid URIs. The greater issue which affects more than 85% (3470 / 3961 * 100) of the activities supplied is the lack of an indicator in the results element. The first five rows of the raw table can be seen here, and the whole file has been saved to Appendix 1 under the tab 'Schema Violations'.

```
In [311]: schema_table.to_excel(pd_writer, "Schema Violations")
          schema_table.head() # show the first five rows
```

```
Out[311]: description  \
          0  'document-link', attribute 'url' is not a vali...
          1  'document-link', attribute 'url' is not a vali...
          2  'result': Missing child element(s), expected i...
          3  'result': Missing child element(s), expected i...
          4  'result': Missing child element(s), expected i...

                                              path  \
          0  iati-activity/1513/document-link/2/@url
          1  iati-activity/1679/document-link/3/@url
          2                  iati-activity/0/result
          3                  iati-activity/1/result
          4                iati-activity/5/result/0

                                                 value
          0  http://ttp://www.snclavalin.com/fr/index.aspx
          1  http://ttp://www.snclavalin.com/fr/index.aspx
          2
          3
          4
```

If we look at the first result element, we can indeed see that it doesn't contain an indicator element:

```
In [312]: print(
              ET.tostring(big_iati.find('iati-activity/result'),
                          pretty_print=True).decode())
```

```
<result type="2">
  <title>
    <narrative xml:lang="en">Results Achieved</narrative>
    <narrative xml:lang="fr">R&#233;sultats atteints</narrative>
  </title>
  <description>
    <narrative xml:lang="en">Results as of March 31, 2011 include: the Fund delivered 62 initiatives
    <narrative xml:lang="fr">Parmi les r&#233;sultats obtenus jusqu&#8217;au 31 mars 2011 : Le Fonds
  </description>
```

```
</result>
```

3.2 Ruleset Validation

There are 30 ruleset violations in total:

```
In [391]: len(ruleset_table)
```

```
Out[391]: 30
```

Looking at the first five rows of the ruleset violations isn't particularly informative:

```
In [313]: ruleset_table.head()
```

```
Out[313]: id                                                message  \
          0  CA-3-A034764001  `(recipient-country|recipient-region)/@percent...
          1  CA-3-A035272001  `(recipient-country|recipient-region)/@percent...
          2  CA-3-A035470001  `(recipient-country|recipient-region)/@percent...
          3  CA-3-D002423002  `(recipient-country|recipient-region)/@percent...
          4  CA-3-D004492001  `(recipient-country|recipient-region)/@percent...

                                                      path  \
          0  /iati-activities/iati-activity[215]/recipient-...
          1  /iati-activities/iati-activity[292]/recipient-...
          2  /iati-activities/iati-activity[331]/recipient-...
          3  /iati-activities/iati-activity[952]/recipient-...
          4  /iati-activities/iati-activity[1178]/recipient...

                                                      rule
          0  recipient-country/@percentage and recipient-re...
          1  recipient-country/@percentage and recipient-re...
          2  recipient-country/@percentage and recipient-re...
          3  recipient-country/@percentage and recipient-re...
          4  recipient-country/@percentage and recipient-re...
```

However, by again using a pivot table and structuring the output first by the type of rule broken, then the specifics of the violation, and then the related activity, we can see a clearer picture:

N.B. The table below has also been saved to Appendix 1 under the tab 'Ruleset Violations by Rule'.

```
In [314]: ruleset_validation_by_rule = ruleset_table.pivot_table(
              index=['rule', 'message', 'id'], aggfunc='count')

          ruleset_validation_by_rule.to_excel(pd_writer, "Ruleset Violations by Rule")

          ruleset_validation_by_rule
```

```
Out[314]: path
          rule                                                    message
          activity-date[@type="2"]/@iso-date must be befo... Start date (2007-03-27) must be before e
                                                             Start date (2008-03-20) must be before e
                                                             Start date (2009-01-13) must be before e
                                                             Start date (2009-03-26) must be before e
                                                             Start date (2009-03-27) must be before e


                                                             Start date (2010-03-24) must be before e
```

```
                                                                Start date (2010-03-25) must be before e
                                                                Start date (2011-03-30) must be before e
                                                                Start date (2013-01-02) must be before e
                                                                Start date (2014-03-14) must be before e
                                                                Start date (2014-03-28) must be before e
activity-date[date @type="1"] or activity-date[... Neither activity-date[@type="1"] nor act

either sector or transaction/sector must be pre... Neither sector nor transaction/sector ha

recipient-country/@percentage and recipient-reg... `(recipient-country|recipient-region)/@p
```

```
In [ ]: print("")
```

Here we can see that most of the ruleset violations have been to do with dates - either activities starting and ending on the same day, or not including a start date, although there are also some ruleset violations regarding sectors and recipient countries.

CHAPTER

FOUR

ASSESSING COMPLIANCE AND COVERAGE

This section assesses the compliance of GAC's data with the IATI 2.02 standard, with a specific focus on the use and coverage of all elements of the standard (version 2.02) and adherence to rules. There is an initial high level look at timeliness and comprehensiveness (coverage), followed by a deeper analysis informed by 6 specified use-cases use cases in the subsections below.

To summarise the following few sections, the IATI data published by Global Affairs Canada performs very well on the general metrics used for compliance. The timeliness and recency of its data is exemplary, and in general the comprehensive use of IATI standard fields puts it at the top five IATI publishers, according to the IATI Dashboard.

This is described in more detail in the next few sections. It is important to note, however, that just evaluating the proportion of activities with certain fields does not elaborate on how useful the inclusion of various fields is, hence the *use case driven approach below* is required to analyse in more depth.

4.1 Initial Evaluation at Compliance and Coverage

(Deliverable 3.1)

This section utilises the data available on the IATI Dashboard Publishing Statistics page, filtering and interpreting to evaluate Timeliness and Coverage. As we see in the *sub-sections below* however, the dashboard doesn't evaluate the combination of fields required to assess some specific use cases, hence the need for more detailed analysis.

4.1.1 Timeliness

Both the frequency of publication and the recency of GAC's IATI data is exemplary, achieving the highest designation from the IATI Dasboard's metrics:

```
In [378]: timeliness = pd.read_csv(
              "http://dashboard.iatistandard.org/timeliness_frequency.csv")

          timeliness[timeliness['Publisher Registry Id'] == 'gac-amc']
```

```
Out[378]: Publisher Name Publisher Registry Id  \
          5  Canada - Global Affairs Canada | Affaires mond...               gac-amc

             2016-12  2017-01  2017-02  2017-03  2017-04  2017-05  2017-06  2017-07  \
          5        1        1       11       20        2        1        3       16

             2017-08  2017-09  2017-10  2017-11 Frequency
          5       13       17       12       13   Monthly
```

Although the frequency varies a lot by season, there has been consistent publication at least once every month, achieving a score of 'Monthly' which is the highest available score on the IATI Dashboard.

```
In [379]: timeliness = pd.read_csv(
              "http://dashboard.iatistandard.org/timeliness_timelag.csv")

          timeliness[timeliness['Publisher Registry Id'] == 'gac-amc']
```

```
Out[379]: Publisher Name Publisher Registry Id  \
          11  Canada - Global Affairs Canada | Affaires mond...               gac-amc

              2016-12  2017-01  2017-02  2017-03  2017-04  2017-05  2017-06  2017-07  \
          11      145      122      149      361       46      152       94       74

              2017-08  2017-09  2017-10  2017-11   Time lag
          11       91      105       86       98  One month
```

Again, with the Time Lag measurement, which shows the number of transactions dated in each month, this data achieves the highest measure. This means that the data is very up to date by the standards of the IATI Dashboard.

4.1.2 Forward Looking Data

The table below shows that GAC has gradual decline in the proportion of activities with budgets over the course of the next three years. This gives an average of 78%, putting GAC at the 37th position on this metric when compared to all IATI publishers, according to the IATI Dashboard.

```
In [389]: forwardlooking = pd.read_csv(
              "http://dashboard.iatistandard.org/forwardlooking.csv")

          forwardlooking = forwardlooking[
              forwardlooking.columns.drop(
                  list(forwardlooking.filter(regex='Current activities ')))]

          forwardlooking[forwardlooking['Publisher Registry Id'] == 'gac-amc']
```

```
Out[389]: Publisher Name Publisher Registry Id  \
          88  Canada - Global Affairs Canada | Affaires mond...               gac-amc

             Percentage of current activities with budgets (2017)  \
          88                                                 95

             Percentage of current activities with budgets (2018)  \
          88                                                 80

             Percentage of current activities with budgets (2019)
          88                                                 61
```

4.1.3 Coverage Of IATI Standard Elements

The tables below show the percentage of GAC's activities which include the fields listed in the headings. They are lifted from the IATI Dashboard.

GAC has 100% coverage of the core IATI Elements:

```
In [380]: comprehensiveness_core = pd.read_csv(
              "http://dashboard.iatistandard.org/comprehensiveness_core.csv")

          comprehensiveness_core = comprehensiveness_core[
              comprehensiveness_core.columns.drop(
                  list(comprehensiveness_core.filter(regex='with valid data')))]
```

```
In [380]: comprehensiveness_core[comprehensiveness_core['Publisher Registry Id'] == 'gac-amc']

Out[380]: Publisher Name Publisher Registry Id  \
          88  Canada - Global Affairs Canada | Affaires mond...               gac-amc

             Version (with any data) Reporting-Org (with any data)  \
          88                     100                           100

             Iati-identifier (with any data) Participating Organisation (with any data)  \
          88                             100                                          100

             Title (with any data) Description (with any data) Status (with any data)  \
          88                   100                         100                    100

             Activity Date (with any data) Sector (with any data)  \
          88                           100                    100

             Country or Region (with any data)  Average (with any data)
          88                               100                      100
```

This is nearly true of the financials, though there is a slight dip in disbursements and expenditure transactions. The IATI Dashboard doesn't not consider `planned-disbursements` which have been considered *below*.

```
In [381]: comprehensiveness_financials = pd.read_csv(
              "http://dashboard.iatistandard.org/comprehensiveness_financials.csv")

          comprehensiveness_financials = comprehensiveness_financials[
              comprehensiveness_financials.columns.drop(
                  list(comprehensiveness_financials.filter(regex='with valid data')))]

          comprehensiveness_financials[comprehensiveness_financials['Publisher Registry Id'] == 'gac-

Out[381]: Publisher Name Publisher Registry Id  \
          88  Canada - Global Affairs Canada | Affaires mond...               gac-amc

             Transaction - Commitment (with any data)  \
          88                                      100

             Transaction - Disbursement or Expenditure (with any data)  \
          88                                                     87

             Transaction - Traceability (with any data) Budget (with any data)  \
          88                                        100                     100

              Average (with any data)
          88                       97
```

With 'Value Added' fields, there is a more pronounced drop. The most significant field here is the `result/indicator`. As seen in the section on *schema validation* above, this causes the majority of GAC's activities to be invalid.

```
In [382]: comprehensiveness_valueadded = pd.read_csv(
              "http://dashboard.iatistandard.org/comprehensiveness_valueadded.csv")

          comprehensiveness_valueadded = comprehensiveness_valueadded[
              comprehensiveness_valueadded.columns.drop(
                  list(comprehensiveness_valueadded.filter(regex='with valid data')))]

          comprehensiveness_valueadded[comprehensiveness_valueadded['Publisher Registry Id'] == 'gac-
```

```
Out[382]: Publisher Name Publisher Registry Id  \
          88  Canada - Global Affairs Canada | Affaires mond...               gac-amc

             Contacts (with any data) Location Details (with any data)  \
          88                      100                                95

             Geographic Coordinates (with any data) DAC Sectors (with any data)  \
          88                                     95                          100

             Capital Spend (with any data) Activity Documents (with any data)  \
          88                           23                                100

             Aid Type (with any data) Recipient Language (with any data)  \
          88                      100                                55

             Result/ Indicator (with any data)  Average (with any data)
          88                                  0                        74
```

```
In [383]: summary_stats = pd.read_csv(
              "http://dashboard.iatistandard.org/summary_stats.csv")

          summary_stats = summary_stats[
              summary_stats.columns.drop(
                  list(summary_stats.filter(regex='with valid data')))]

          summary_stats[summary_stats['Publisher Registry Id'] == 'gac-amc']
```

```
Out[383]: Publisher Name Publisher Registry Id  \
          88  Canada - Global Affairs Canada | Affaires mond...               gac-amc

             Publisher Type  Timeliness  Forward looking  Comprehensive  Score  \
          88     Government         100               78             92     90

              Coverage  Coverage-adjusted score
          88       100                       90
```

4.2 Detailed Analysis Method

Each of the deliverables below corresponds to a use case, around which compliance and coverage is framed. These are reflected in the 'Use Case' section which opens each of them.

The analytical approaches employed have varied by deliverable, and have generally been grounded in initial exploratory analysis and built depending on the findings.

All of the code is visible, so scrutiny on the methods is encouraged.

4.3 Deliverable 3.1.1

4.3.1 Use Case

Identify projects in specific countries, with specific partners, with specific types of partners (eg multilateral organisations, CSOs, private sector)

This section analyses the existence and coverage of the relevant fields: recipient countries, participating organisations, participating organisation (type and role).

First, let's extract some information about the fields we're interested in, namely recipient countries and participating organisations.

The table below extracted below shows, for each activity, how many instances of each field there are. This will gives us a first pass, and allows us to make more opinionated analyses subsequently.

Note: Although recipient regions aren't the focus of this deliverable, they've been included to help analysis of recipient country below. Please also note that this table has been included in the Appendix Workbook

---

```
In [384]: locations_and_partners = pd.DataFrame(
              columns=[
                  'iati-identifier',
                  'activity-status',
                  'recipient-country-count',
                  'recipient-region-count',
                  'participating-organisation-count'
              ],
              data=[[
                  activity.find('iati-identifier').text,
                  activity.find('activity-status').get('code'),
                  len(activity.findall('recipient-country')),
                  len(activity.findall('recipient-region')),
                  len(activity.findall('participating-org'))
              ] for activity in big_iati.findall('iati-activity')])

          locations_and_partners.to_excel(pd_writer, "Locations and Partners")

          locations_and_partners.head()
```

```
Out[384]: iati-identifier activity-status  recipient-country-count  \
          0  CA-3-A031268001               3                      12
          1  CA-3-A031470001               2                       1
          2  CA-3-A031708001               2                       1
          3  CA-3-A031708003               3                       1
          4  CA-3-A031717001               3                       1

             recipient-region-count  participating-organisation-count
          0                       0                                 3
          1                       0                                 3
          2                       0                                 3
          3                       0                                 3
          4                       0                                 3
```

Now, we can use the 'describe' cuntion to analyse this distribution

```
In [385]: locations_and_partners.describe()
```

```
Out[385]: recipient-country-count  recipient-region-count  \
          count              3961.000000             3961.00000
          mean                  2.696289                0.44307
          std                   9.298740                1.15179
          min                   0.000000                0.00000
          25%                   1.000000                0.00000
          50%                   1.000000                0.00000
          75%                   1.000000                0.00000
          max                 148.000000                5.00000

                 participating-organisation-count
          count                            3961.0
```

```
mean                                3.0
std                                 0.0
min                                 3.0
25%                                 3.0
50%                                 3.0
75%                                 3.0
max                                 3.0
```

The key things to observe here are as follows:

- There are exactly three participating organisation elements in every element (max = 3; min = 3). This will allow a more opinionated analysis below.
- Although the mean number of recipient countries provided is around 3, there is a lot of variation, and there are some with none, and at least one with 0 and one with 148. Again, this guides analysis below.
- The two middle quartiles are constituted entirely of activities with one recipient country.

4.3.2 Recipient Countries

To get more of a sense of the distribution of recipient country coverage, let's use a histogram.

```
In [317]: p1 = Histogram( locations_and_partners, 'recipient-country-count',
                  title = "Histogram of Recipient Country Counts", bins=150)

          show(p1)
```

Here we can see that indeed, the vast majority of activities have one recipient country. However, over 500 do not.

These might have a recipient region associated, so let's see if there are any activities which have neither by filtering the above table to include only rows which have none of either:

```
In [318]: locations_and_partners[
              (locations_and_partners['recipient-country-count'] == 0) &
              (locations_and_partners['recipient-region-count'] == 0)].head()

Out[318]: iati-identifier activity-status  recipient-country-count  \
          146  CA-3-A033944001               2                       0
          856  CA-3-D002114001               3                       0

               recipient-region-count  participating-organisation-count
          146                       0                                 3
          856                       0                                 3
```

As we can see, there are in fact only two activities which have neither element.

Let's also look at the activities which have a very high number of recipient countries, using 50 arbitrarily:

```
In [319]: locations_and_partners[locations_and_partners['recipient-country-count'] > 50]

Out[319]: iati-identifier activity-status  recipient-country-count  \
          358   CA-3-A035580001               3                     133
          564   CA-3-D000114001               2                      66
          666   CA-3-D000514001               2                     132
          667   CA-3-D000514002               3                     132
          731   CA-3-D000942001               2                     133
          931   CA-3-D002306001               2                     115
          959   CA-3-D002455001               2                     138
          1652  CA-3-A033536001               4                      53
          1653  CA-3-A033536002               4                      53
          1740  CA-3-A033975001               4                      52
          1741  CA-3-A033975002               4                      52
```

```
2142  CA-3-D000360001              4                      148
2304  CA-3-D000731001              4                      132
2484  CA-3-M012575002              4                       87
2504  CA-3-M012660001              4                       53
2596  CA-3-M012870001              4                       53
2597  CA-3-M012870002              4                       53
2628  CA-3-M012912001              4                       53
2631  CA-3-M012918002              4                      122
2920  CA-3-M013279001              4                      143
2943  CA-3-M013303001              4                      130
3050  CA-3-M013420001              4                      138
3058  CA-3-M013432001              4                      130
3352  CA-3-M013784001              4                      132
3480  CA-3-S061266PRG              4                       56

      recipient-region-count  participating-organisation-count
358                        0                                 3
564                        0                                 3
666                        0                                 3
667                        0                                 3
731                        0                                 3
931                        0                                 3
959                        0                                 3
1652                       0                                 3
1653                       0                                 3
1740                       1                                 3
1741                       1                                 3
2142                       0                                 3
2304                       0                                 3
2484                       0                                 3
2504                       0                                 3
2596                       0                                 3
2597                       0                                 3
2628                       0                                 3
2631                       0                                 3
2920                       1                                 3
2943                       0                                 3
3050                       0                                 3
3058                       0                                 3
3352                       0                                 3
3480                       0                                 3
```

Although it is very possible that these activities are legitimately benefiting each between 50 and 150 countries, this does make any kind of detailed analysis more difficult.

4.3.3 Participating Organisations

To look in more detail, let's create a new table of all of the participating organisation details:

```
In [320]: detailed_participating_orgs = pd.DataFrame(
              columns=['iati-identifier', 'ref', 'name', 'type', 'role'],
              data=[[
                  participating_org.getparent().find('iati-identifier').text,
                  participating_org.get('ref'),
                  participating_org.find('narrative').text,
                  participating_org.get('type'),
                  participating_org.get('role')
              ]
```

```
                for participating_org in big_iati.findall(
                    'iati-activity/participating-org')])

        detailed_participating_orgs.to_excel(pd_writer, "Participating Organisations")

        detailed_participating_orgs.head()
```

Out[320]:
```
  iati-identifier   ref                                                name  \
0  CA-3-A031268001    CA                                              Canada
1  CA-3-A031268001  CA-1          Canadian International Development Agency
2  CA-3-A031268001  None  Public Works and Government Services Canada - ...
3  CA-3-A031470001    CA                                              Canada
4  CA-3-A031470001  CA-1          Canadian International Development Agency

  type role
0   10    1
1   10    3
2   10    4
3   10    1
4   10    3
```

Again, looking at the first five rows, this table isn't particularly informative, and we know that currently a row row for each activity. To give a clearer picture, let's reformat this table to show the number of `reporting-org` elements given, broken down by the organisation role, and then type.

In [321]:
```
detailed_participating_orgs.pivot_table(
    index=['role', 'type'], aggfunc='count')
```

Out[321]:
```
          iati-identifier  name   ref
role type
1    10              3961  3961  3961
3    10              3961  3961  3961
4    10               446   446    63
     21               455   455   375
     22              1225  1225  1049
     30                 6     6     4
     40              1405  1405  1234
     70               135   135   116
```

For every `participating-org` given a role of either Funding or Extending, all of the fields have been provided.

Because there are more types of organisation which have played an 'Implementing' role, let's collapse them down:

In [322]:
```
detailed_participating_orgs.pivot_table(index=['role'], aggfunc='count')
```

Out[322]:
```
     iati-identifier  name   ref  type
role
1               3961  3961  3961  3961
3               3961  3961  3961  3961
4               3961  3961  2985  3672
```

Looking at the 'ref' and 'type' values for the bottom row, it can be seen that 976 (24.64%) Implementing organisation declarations have no identifying reference, and 289 (7.30%) have no type declared. the identifiers for these activities can be found by filtering the 'Participating Organisations' tab of Appendix 1.

4.4 Deliverable 3.1.2

4.4.1 Use Case

Identify local partners by relating the project's implementing Organisation Identifier and the beneficiary country

The aim of this section is to analyse the structure of the IATI activities and their fields to assess the feasibility of this kind of inference.

4.4.2 Existence of Required Fields

The most obvious impediment to this inference is the lack of either a recipient country code, or an implementing organisation's reference.

```
In [323]: implementers_without_ref = big_iati.xpath(
              "iati-activity[participating-org[@role='4' and not(@ref)]]")

          activities_without_country = big_iati.xpath(
              "iati-activity[not(recipient-country)]")

          total_non_inference = implementers_without_ref + activities_without_country

          print(
              "Generated Text: \n\nOf the the {} activities published, {} ({:.2%}) do not include a r
              .format(
                  len(big_iati), len(activities_without_country),
                  (len(activities_without_country) / len(big_iati))))

          print(
              "All of the {} activities analysed include an implementing organisation.\n"
              "Of those however, {} ({:.2%}) of them do not include a reference.\n"
              .format(
                  len(big_iati), len(implementers_without_ref),
                  (len(implementers_without_ref) / len(big_iati))))

          print(
              "Of the {} activities identified in the above procedures, {} ({:.2%} of the all activit
              .format(
                  len(total_non_inference), len(set(total_non_inference)),
                  len(set(total_non_inference)) / len(big_iati)))
```

```
Generated Text:

Of the the 3961 activities published, 572 (14.44%) do not include a recipient country.

All of the 3961 activities analysed include an implementing organisation.
Of those however, 976 (24.64%) of them do not include a reference.

Of the 1548 activities identified in the above procedures, 1439 (36.33% of the all activities) are ur
```

This is to say that for the set of activities analysed in this report, it is impossible to make this inference for over a third of them just in virtue of the necessary fields not being present.

4.4.3 How the Required Fields are Related

Another way of assessing the possibility of this inference is to look structurally at the way these elements are used together. To begin with we make a table of all of the recipient countries declared in activities, along with the corresponding implementing organisation information published in their parent activity.

Let's look at what the fields look like, starting with the recipient country field. Here are three arbitrarily chosen recipient country elements:

```
In [324]: for xml_recipient_country in big_iati.findall(
                  'iati-activity/recipient-country')[520:523]:

              print(ET.tostring(xml_recipient_country, pretty_print=True).decode())
```

```
<recipient-country percentage="100" code="SN"/>

<recipient-country percentage="100" code="HT"/>

<recipient-country percentage="100" code="HT"/>
```

Let's now look at the implementing participating organisation element in the dataset, which includes all of the available fields:

```
In [325]: print(
              ET.tostring(
                  big_iati.xpath("iati-activity/participating-org[@role='4' and @ref]")[3],
                  pretty_print=True).decode())
```

```
<participating-org role="4" ref="CA-CRA_ACR-3118929157" type="22">
  <narrative xml:lang="en">FIT - Foundation for International Training </narrative>
  <narrative xml:lang="fr">FIT - Foundation for International Training </narrative>
</participating-org>
```

Every row in the table below corresponds to a single declaration of a recipient country, so there is a lot of duplication of activity identifiers and implementing organisations, but this is necessary for subsequent analysis.

Again, here are the first five rows:

```
In [326]: implementers_concise = pd.DataFrame(
              columns=[
                  'iati-identifier', 'implementing-org-name', 'implementing-org-ref',
                  'recipient-country-code'
              ],
              data=[[
                  country.getparent().find('iati-identifier').text,
                  country.getparent().xpath("participating-org[@role='4']")[0].find(
                      'narrative').text,
                  country.getparent().xpath("participating-org[@role='4']")[0].get(
                      'ref'),
                  country.get('code')
              ] for country in big_iati.findall('iati-activity/recipient-country')])

          implementers_concise.head()
```

Out[326]:

	iati-identifier	implementing-org-name	\
0	CA-3-A031268001	Public Works and Government Services Canada - ...	
1	CA-3-A031268001	Public Works and Government Services Canada - ...	
2	CA-3-A031268001	Public Works and Government Services Canada - ...	
3	CA-3-A031268001	Public Works and Government Services Canada - ...	
4	CA-3-A031268001	Public Works and Government Services Canada - ...	

```
  implementing-org-ref recipient-country-code
0                 None                      AG
1                 None                      AI
2                 None                      BZ
3                 None                      DM
4                 None                      GD
```

Again, on its own this table doesn't pain a clear picture, but we can manipulate it to see the number of activities, unique names, an unique countries associated with each `implementing-org`. This list is very long, but this time looking at the first five rows gives more insight:

```
In [327]: unique_countries_per_organisation = implementers_concise.pivot_table(
              index=['implementing-org-ref'], aggfunc=lambda x: len(x.unique()))

          unique_countries_per_organisation.columns = [
              'unique-iati-identifier-count', 'unique-implementing-org-name-count',
              'unique-recipient-country-code-count'
          ]

          unique_countries_per_organisation.head()
```

```
Out[327]: unique-iati-identifier-count  \
          implementing-org-ref
          21009                                          1
          21016                                         80
          21018                                         41
          21020                                          1
          21023                                          1

                                unique-implementing-org-name-count  \
          implementing-org-ref
          21009                                                  1
          21016                                                  3
          21018                                                  3
          21020                                                  1
          21023                                                  1

                                unique-recipient-country-code-count
          implementing-org-ref
          21009                                                  52
          21016                                                  70
          21018                                                  39
          21020                                                  41
          21023                                                   5
```

If the consider the second row of this table, this tells us that the reference '21016' has been used in 80 different activities, alongside 3 different names, and has been the implementing organisation for 70 distinct recipient countries.

This organisation is the ICRC - the three organisation names which have been used are as follows:

```
In [328]: for name in set(implementers_concise[implementers_concise[
                  'implementing-org-ref'] == '21016']['implementing-org-name']):
              print(name)
```

```
ICRC - International Committee of the Red Cross
Red Cross International Aid Trust of Canada
International Committee of the Red Cross (ICRC) Appeals via the Canadian Red Cross Society (CRCS)
```

Following the same approach used above, let's look at the distribution of these counts:

```
In [329]: unique_countries_per_organisation.describe()
```

```
Out[329]: unique-iati-identifier-count  unique-implementing-org-name-count  \
          count                      392.000000                          392.000000
          mean                         6.433673                            1.130102
          std                         19.229418                            0.590558
          min                          1.000000                            1.000000
          25%                          1.000000                            1.000000
          50%                          2.000000                            1.000000
          75%                          4.000000                            1.000000
          max                        240.000000                            9.000000

                 unique-recipient-country-code-count
          count                           392.000000
          mean                             10.836735
          std                              20.409713
          min                               1.000000
          25%                               1.000000
          50%                               4.000000
          75%                              10.000000
          max                             149.000000
```

Looking at the recipient country code figures, we can see that the mean is 10 distinct countries, and the median is 4, with a maximum of 149. This indicates a similar distribution as before, so let's confirm with another histogram:

```
In [330]: p2 = Histogram(unique_countries_per_organisation,
                         'unique-recipient-country-code-count',
                         title = "Histogram of Unique Recipient Countries by Implementing Organisatio
                         bins=150)

          show(p2)
```

```
In [331]: num_orgs_with_one_recipient_country = len(
              unique_countries_per_organisation[pd.to_numeric(
                  unique_countries_per_organisation[
                      'unique-recipient-country-code-count'] == 1)])

          num_orgs_with_more_than_one_recipient_country = len(
              unique_countries_per_organisation[pd.to_numeric(
                  unique_countries_per_organisation['unique-recipient-country-code-count']
                  > 1)])

          print(
              "Generated Text: \n\nOf the {} implementing organisations, {} are associated with "
              "one recipient country only, and {} ({:.2%}) are asocciated with more than one."
              .format(
                  len(unique_countries_per_organisation),
                  num_orgs_with_one_recipient_country,
                  num_orgs_with_more_than_one_recipient_country,
                  num_orgs_with_more_than_one_recipient_country /
                  len(unique_countries_per_organisation)))
```

```
Generated Text:

Of the 392 implementing organisations, 109 are associated with one recipient country only, and 283 (
```

Exactly how this effects a data user's ability to make the inference in question is not clear.

For one of the 109 organisations with only one recipient country associated, the hope is that they are a local organisation themselves. This can't be guaranteed without a more extensive study, but regardless, it the search for local implementers would be easier for organisations which are only associated with one recipient country.

For one of the 283 organisations with more than one associated recipient country, there are two possibilities:

- It is a federated organisation which has easily identifiable country offices and channels of contact.

In this case, for the ~75% of recipient country uses which can be associated with the reference of an implementing organisation, in theory a user could identify the local actors by contacting the country office for that organisation, though it would depend on that user's initiative to find out if any more organisation details.

- It is an intermediate-tier organisation which disburses funds to local organisations as subcontractors.

In this case, for the same set as above, a user would have to find out through other means. It would be possible, but would likely require information from the implementing organisation directly and would be laborious.

4.5 Deliverable 3.1.3

4.5.1 Use Case

Determine at least the first step in the delivery chain, both through the Organization Identifier and the partner's Activity Identifier

This section analyses the existence and coverage of the relevant fields: participating organisations and transaction / receiver organisations, as well as provider/receiver activity ids. Additional focus will look at other-identifier (type A9)

4.5.2 Other Identifier

In light of its separation from the transaction element, it's worth briefly looking at the other identifier first given that it doesn't require much analysis in light of this use case:

```
In [332]: activities_with_other_identifier = big_iati.xpath(
              "./iati-activity[other-identifier]")
          other_identifiers = big_iati.xpath("./iati-activity/other-identifier/@type")

          print(
              "Generated Text: \n\nThere are {} activities which include an other-identifier field, "
              "which have {} unique value among them: {}".format(
                  len(activities_with_other_identifier), len(set(other_identifiers)),
                  set(other_identifiers)))
```

```
Generated Text:

There are 3280 activities which include an other-identifier field, which have 1 unique value among th
```

In virtue of the fact that this code designates a CRS identifier, it follows that none of these uses of the `other-identifier` element can help with this inference.

4.5.3 Transactions

4.5.3.1 Transaction Provider Organisation

```
In [333]: activities_with_provider_org_transactions = big_iati.xpath(
              "./iati-activity[transaction[provider-org]]")

          print(
              "There are {} activities which contain a transaction that includes a provider-org eleme
              format(len(activities_with_provider_org_transactions)))
```

```
There are 0 activities which contain a transaction that includes a provider-org element.
```

In virtue of the above, there cannot be any uses of the `provider-org/@ref` or the `provider-org/@provider-activity-id` attributes in evaluating the first step of the delivery chain.

4.5.3.2 Transaction Receiver Organisation

Before conducting any analysis, a dataframe of transactions is retrieved from the IATI files. The first five rows of which shown below:

```
In [334]: transaction_df = pd.DataFrame(
              columns=[
                  'iati-identifier', 'value', 'activity-status', 'ref', 'humanitarian',
                  'transaction-type', 'transaction-date', 'receiver-org-ref',
                  'receiver-org-activity-id'
              ],
              data=[[
                  transaction.getparent().find('iati-identifier').text,
                  float(transaction.find('value').text),
                  transaction.getparent().find('activity-status').get('code'),
                  transaction.get('ref'),
                  transaction.get('humanitarian'),
                  transaction.find('transaction-type').get('code'),
                  transaction.find('transaction-date').get('iso-date'),
                  transaction.find('receiver-org').get('ref')
                  if transaction.find('receiver-org') is not None else None,
                  transaction.find('receiver-org').get('receiver-activity-id')
                  if transaction.find('receiver-org') is not None else None
              ] for transaction in big_iati.xpath("iati-activity/transaction")])

          transaction_df.to_excel(pd_writer, "Transactions")

          transaction_df.head()
```

```
Out[334]: iati-identifier        value activity-status   ref humanitarian  \
          0  CA-3-A031268001  17933718.13               3  None         None
          1  CA-3-A031268001    500000.00               3  None         None
          2  CA-3-A031268001    101000.00               3  None         None
          3  CA-3-A031268001     41000.00               3  None         None
          4  CA-3-A031268001    250000.00               3  None         None

            transaction-type transaction-date receiver-org-ref receiver-org-activity-id
          0                2       2003-04-07             None                       None
          1                3       2003-09-18             None                       None
          2                3       2003-12-05             None                       None
          3                3       2004-02-19             None                       None
          4                3       2004-04-05             None                       None
```

Firstly, as we have the information to hand, we can see that no transactions have `@ref` identifier or `@humanitarian` flag:

```
In [335]: transaction_df[[
              'value', 'activity-status', 'ref', 'humanitarian', 'transaction-type',
              'transaction-date'
          ]].describe(exclude=[float])
```

```
Out[335]: activity-status  ref  humanitarian transaction-type transaction-date
          count            23637  0.0          0.0            23637            23637
          unique               3  0.0          0.0                2             3408
          top                  4  NaN          NaN                3       2015-03-31
          freq             15168  NaN          NaN            19656              282
```

Secondly, we can select only the disbursements, as befits an analysis of receiver organisation data:

```
In [336]: transaction_df[transaction_df['transaction-type'] == '3'][[
              'activity-status', 'transaction-type', 'receiver-org-ref',
              'receiver-org-activity-id'
          ]].describe()
```

```
Out[336]: activity-status transaction-type receiver-org-ref  \
          count             19656            19656            14125
          unique                3                1              398
          top                   4                3            41140
          freq              12414            19656              511

                  receiver-org-activity-id
          count                        0.0
          unique                       0.0
          top                          NaN
          freq                         NaN
```

As we can see from this, there no usage of the `@receiver-org-activity-id`, but 14,125 (71.86%) transactions of the total 19,656 disbursements do include a `@receiver-org-ref`.

4.6 Deliverable 3.1.4

4.6.1 Use Case

Identify joint funding and determine the lead donor/implementing agency

This section considers the existence and coverage of the relevant fields: participating organisations, transaction / receiver organisations, and related activity.

Looking at the *above section on participating organisations* we can see that there is full coverage of participating organisations with regard to funding and and extending, but that 24.64% of declarations don't include a `@ref` attribute, and 7.30% don't include a `@type` declaration. Both of these could make it more difficult to establish the exact organisation and whether or not it has joint funded an activity.

Similarly, looking at the section above *on transaction receiver-orgs*, we see that although there is no prospect of tracing locating a participating organisation's activity, references are available for the recipients of 71.86% of transactions. Although it would be laborious to do, it is possible that in all of these cases, the implementing organisation could be identified.

However, because there are no details on the *providing organisations*, transactions are assumed not to include any details of lead donors which aren't already available in the activity-level participating organisation details.

Let's now consider the related activity fields. First, let's find how many activities have included a related activity element:

```
In [337]: len(big_iati.xpath("./iati-activity[related-activity]"))
```

```
Out[337]: 607
```

Now let's what types have been included:

```
In [338]: set(big_iati.xpath("./iati-activity/related-activity/@type"))
```

```
Out[338]: {'3'}
```

As there are only references to sibling activities, but none to co-funded or third party activities, it follows that of these 607 activities none can be used to identify joint-funded activities.

4.7 Deliverable 3.1.5

4.7.1 Use Case

Determine the geographic area(s) benefiting from the project and, where relevant, the actual location of the project activities

This section considers the existence and coverage of the relevant fields - recipient country & region at the activity and transaction level, as well as locations (national and subnational).

4.7.2 Activity Level Recipient Locations

With regards to recipient country and region at the activity level, by referring to *Deliverable 3.1.1 above* it can be seen that there is nearly 100% coverage of recipient locations as either countries or regions, though the usability of those elements is hindered by the existence of many activities with more than one recipient country. To recap the number of activities with recipient countries and regions:

```
In [339]: print("Of {} activities, {} ({:.2%}) include at least one recipient country"
                .format(
                    len(big_iati),
                    len(big_iati.xpath("iati-activity[recipient-country]")),
                    len(big_iati.xpath("iati-activity[recipient-country]")) / len(big_iati)))
```

```
Of 3961 activities, 3389 (85.56%) include at least one recipient country
```

```
In [340]: print("Of {} activities, {} ({:.2%}) include at least one recipient region"
                .format(
                    len(big_iati),
                    len(big_iati.xpath("iati-activity[recipient-region]")),
                    len(big_iati.xpath("iati-activity[recipient-region]")) / len(big_iati)))
```

```
Of 3961 activities, 616 (15.55%) include at least one recipient region
```

```
In [341]: var = len(
              set(
                  big_iati.xpath("iati-activity[recipient-region]") +
                  big_iati.xpath("iati-activity[recipient-country]")))

          print(
              "Together, excluding any duplicates, these {} activites comprise {:.2%} of the total."
              format(var, var / len(big_iati)))
```

```
Together, excluding any duplicates, these 3959 activites comprise 99.95% of the total.
```

4.7.3 Transaction Level Recipient Locations

```
In [342]: len(big_iati.xpath("iati-activity[transaction[recipient-country]]"))
```

```
Out[342]: 0
```

```
In [343]: len(big_iati.xpath("iati-activity[transaction[recipient-country]]"))
```

```
Out[343]: 0
```

In the current data, there are no location details at transaction level.

4.7.4 Locations Elements

```
In [344]: print("{} of the {} ({:.2%}) published activities contain location elements.".
              format(
                  len(big_iati.xpath("iati-activity[location]")),
                  len(big_iati),
                  len(big_iati.xpath("iati-activity[location]")) / len(big_iati)
              ))
```

```
3730 of the 3961 (94.17%) published activities contain location elements.
```

These can be viewed in more details, as the recipient countries were above. Again, here is every unique declaration of a location at the activity level, clipped at 5 rows.

Note: there are multiple rows per activity, as above.

```
In [345]: detailed_locations = pd.DataFrame(
              columns=[
                  'iati-identifier',
                  'activity-status',
                  'location-reach-code',
                  'location-id-code',
                  'location-id-vocabulary',
                  'location-point-srs',
                  'location-point-pos'
              ],
              data=[[
                  location.getparent().find('iati-identifier').text,
                  location.getparent().find('activity-status').get('code'),
                  location.find('location-reach').get('code'),
                  location.find('location-id').get('code'),
                  location.find('location-id').get('vocabulary'),
                  location.find('point').get('srsName'),
                  location.find('point/pos').text
              ] for location in big_iati.findall('iati-activity/location')])

          detailed_locations.to_excel(pd_writer, "Location Elements")

          detailed_locations.head()
```

```
Out[345]: iati-identifier activity-status location-reach-code location-id-code  \
          0  CA-3-A031268001               3                   1          3378644
          1  CA-3-A031268001               3                   1          3383330
          2  CA-3-A031268001               3                   1          3435910
          3  CA-3-A031268001               3                   1          3439389
          4  CA-3-A031268001               3                   1          3441575

            location-id-vocabulary                          location-point-srs  \
          0                     G1  http://www.opengis.net/def/crs/EPSG/0/4326
          1                     G1  http://www.opengis.net/def/crs/EPSG/0/4326
          2                     G1  http://www.opengis.net/def/crs/EPSG/0/4326
          3                     G1  http://www.opengis.net/def/crs/EPSG/0/4326
          4                     G1  http://www.opengis.net/def/crs/EPSG/0/4326

              location-point-pos
          0    6.80448 -58.15527
          1    5.86638 -55.16682
          2  -34.61315 -58.37723
```

```
3  -25.30066 -57.63591
4  -34.90328 -56.18816
```

Again, this isn't very useful by itself, so let's group by the activity and analyse the counts of each location field:

```
In [346]: location_elements_pivot = detailed_locations.groupby('iati-identifier').agg(['count'])

          location_elements_pivot.describe()
```

```
Out[346]: activity-status location-reach-code location-id-code  \
                        count              count            count
          count     3730.000000        3730.000000      3730.000000
          mean         3.150670           3.150670         3.150670
          std          7.870249           7.870249         7.870249
          min          1.000000           1.000000         1.000000
          25%          1.000000           1.000000         1.000000
          50%          1.000000           1.000000         1.000000
          75%          2.000000           2.000000         2.000000
          max        191.000000         191.000000       191.000000

                location-id-vocabulary location-point-srs location-point-pos
                                 count              count              count
          count             3730.000000        3730.000000        3730.000000
          mean                 3.150670           3.150670           3.150670
          std                  7.870249           7.870249           7.870249
          min                  1.000000           1.000000           1.000000
          25%                  1.000000           1.000000           1.000000
          50%                  1.000000           1.000000           1.000000
          75%                  2.000000           2.000000           2.000000
          max                191.000000         191.000000         191.000000
```

Here all of the fields are used uniformly, and the distribution looks similar to recipient countries above, which represents the same challenge for usability when there are many locations to a given activity. Especially given that there is no way of associating those locations with the recipient-country elements, or the activities transactions.

4.8 Deliverable 3.1.6

4.8.1 Use Case

Verify that the financial data of a project "adds up" (e.g. comparing commitments to budgets and disbursements)

This section considers the existence and coverage of the relevant fields - budgets, commitments and transactions (broken down by type).

4.8.2 Budgets

```
In [347]: print("{} of the {} ({:.2%}) published activities contain budgets.".
                format(
                    len(big_iati.xpath("iati-activity[budget]")),
                    len(big_iati),
                    len(big_iati.xpath("iati-activity[budget]")) / len(big_iati)
                ))
```

```
3952 of the 3961 (99.77%) published activities contain budgets.
```

4.8.3 Transactions

```
In [348]: print(
              "The following transaction types are incuded in the analysed activities: {}".
              format(
                  set(
                      big_iati.xpath(
                          "iati-activity/transaction/transaction-type/@code"))))
```

```
The following transaction types are incuded in the analysed activities: {'2', '3'}
```

These types correspond to commitment and disbursement respectively (see here).

```
In [349]: print("{} of the {} ({:.2%}) published activities contain commitments.".
                format(
                    len(big_iati.xpath("iati-activity[transaction[transaction-type[@code='2']]]")),
                    len(big_iati),
                    len(big_iati.xpath("iati-activity[transaction[transaction-type[@code='2']]]")) /
                ))
```

```
3952 of the 3961 (99.77%) published activities contain commitments.
```

```
In [350]: print("{} of the {} ({:.2%}) published activities contain commitments.".
                format(
                    len(big_iati.xpath("iati-activity[transaction[transaction-type[@code='3']]]")),
                    len(big_iati),
                    len(big_iati.xpath("iati-activity[transaction[transaction-type[@code='3']]]")) /
                ))
```

```
3455 of the 3961 (87.23%) published activities contain commitments.
```

From the above, it's clear that there is good coverage on the considered financial elements, with commitments being the lowest at just under 90% coverage.

Without going into much more detail, this gives a prima facie indication of the number of activities for which 'adding' up the financials is possible, and it is possible that the drop in coverage for commitments is due to the timing of activities. Without a deeper analysis, we can't be certain of this at present.

4.8.4 Planned Disbursements

```
In [351]: print("{} of the {} ({:.2%}) published activities planned disbursements.".
                format(
                    len(big_iati.xpath("iati-activity[planned-disbursement]")),
                    len(big_iati),
                    len(big_iati.xpath("iati-activity[planned-disbursement]")) / len(big_iati)
                ))
```

```
1162 of the 3961 (29.34%) published activities planned disbursements.
```

Planned disbursements are an exception, it is possible that this is just a representation of the lower proportion of forward-looking activities.

4.8.5 In Combination

```
In [352]: activities_with_budgets_commitments_disbursements = big_iati.xpath(
              "iati-activity[transaction[transaction-type[@code='2']] and "
              "transaction[transaction-type[@code='3']] and "
              "budget]")

          print("{} of the {} ({:.2%}) published activities budgets, commitments, and disbursements."
```

```
                format(
                    len(activities_with_budgets_commitments_disbursements),
                    len(big_iati),
                    len(activities_with_budgets_commitments_disbursements) / len(big_iati)
                ))
```

```
3455 of the 3961 (87.23%) published activities budgets, commitments, and disbursements.
```

Disregarding planned disbursements, we can see that there is good coverage of activities which include each of the financial elements considered.

CHAPTER

FIVE

APPENDIX 1: DATA WORKBOOK

Throughout this report various analytical tables have been stored in separate sheets in the workbook found in the 'final' data folder under 'Compliance Report Data.xlsx'. The following cell saves the file:

```
In [353]: pd_writer.save()
```

```
In [354]: %%bash

          cp ../data/final/Compliance-Report-Data.xlsx _static/
```

Download Appendix One

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search