7th Framework Programme
ENV.2010.4.1.2-2
Integrating new data visualisation approaches of earth Systems into GEOSS development

Project Nr: 265178

QUAlity aware VIsualisation for the Global Earth Observation system of systems

Deliverable D3.3
Quality elicitation component for continuous and discrete valued variables

Version 1.2

Due date of deliverable: 31/10/2013
Actual submission date: 31/01/2014

Document control page				
Title	3.3 Quality elicitation component for continuous and discrete valued variables			
Creator	DC_AST			
Editor	DC_AST			
Description	Quality elicitation component for continuous and discrete valued variables			
Publisher	GeoViQua Consortium			
Contributors	GeoViQua Partners			
Type	Text			
Format	MS-Word			
Language	EN-GB			
Creation date	10/10/2013			
Version number	1.2			
Version date	31/01/2014			
Last modified by				
Rights	Copyright © 2014, GeoViQua Consortium			
Dissemination level		CO (confidential, only for members of the consortium)		
	X	PU (public)		
		PP (restricted to other programme participants)		
		RE (restricted to a group specified by the consortium)		
	When restricted, access granted to:			
Nature	X	R (report)		
		P (prototype)		
		D (demonstrator)		
		O (other)		
Review status		Draft	*Where applicable:*	
		WP leader accepted		Accepted by the PTB
		PMB quality controlled		Accepted by the PTB as public document
	X	Coordinator accepted		
Action requested		to be revised by all GeoViQua partners		
		for approval of the WP leader		
		for approval of the PMB		
	X	for approval of the Project Coordinator		
		for approval of the PTB		
Requested deadline				

Revision history			
Version	Date	Modified by	Comments
0.1	8-10-2013	DC_AST	Created the basic content of the deliverable
0.2	20-01-2014	AST / S&T	Added in GECA section
0.3	28-01-2014	AST	Added section on implementation
0.4	29-01-2014	AST	Refined implementation
1.0	31-01-2014	AST	Proofread, polished for submission
1.1	31-01-2014	AST	Final edits
1.2	31-01-2014	JM_CREAF	Coordinator acceptance

Institution	Contributors
S&T	Joost Smeets
AST	DC_AST Dan Cornford, Chris Lush
JM_CREAF	Joan Masó

Table of Contents

Table of Figures

1. Introduction

The core aim of WP3 was to address the issue of the computation of quantitative quality indicators that were deemed to be important within GeoViQua. The target was to integrate the calculation with the GeoViQua quality model, using UncertML to represent the statistically meaningful Quality Indicators (QIs) where appropriate, and using the GeoViQua Quality Model (GVQM) to capture the processing steps and other relevant metadata that would be captured as part of the QI computation process.

Where possible we attempted to build on and reuse existing proven technology, which we believe is both efficient, but also helps with the uptake and integration of the resulting systems. Issues with recruitment meant that some of the more mathematical elements of the problem were addressed at the beginning of the project, with more operation concerns being addressed toward the end of the project. A key challenge we addressed, above that described in the Description of Work was the integration of the QI computation engine with the GVQM. While the scope of the work on the computation of the quality indicators was reduced, this more thorough integration with the GVQM was felt to provide a better return on time investment.

Computing QIs for properties in datasets with unknown quality is a very challenging issue. This needs to consider various aspects of the science of Earth Observation (EO) and constraints on available information. In general, to compute a QI, which provides a statistical summary of the relation of the observed property to the "true" value, it is necessary to know the "true" value of the quantity of interest, which we will refer to in this document as the target data (or observation or property). Rarely will the "true" value be known, however there will often be reference observations available, often explicitly collected to calibrate or validate the observing system, which are of sufficiently good quality that they can approximate the "truth". A general approach to the computation of QI's follows a pattern similar to that shown in Figure 1.

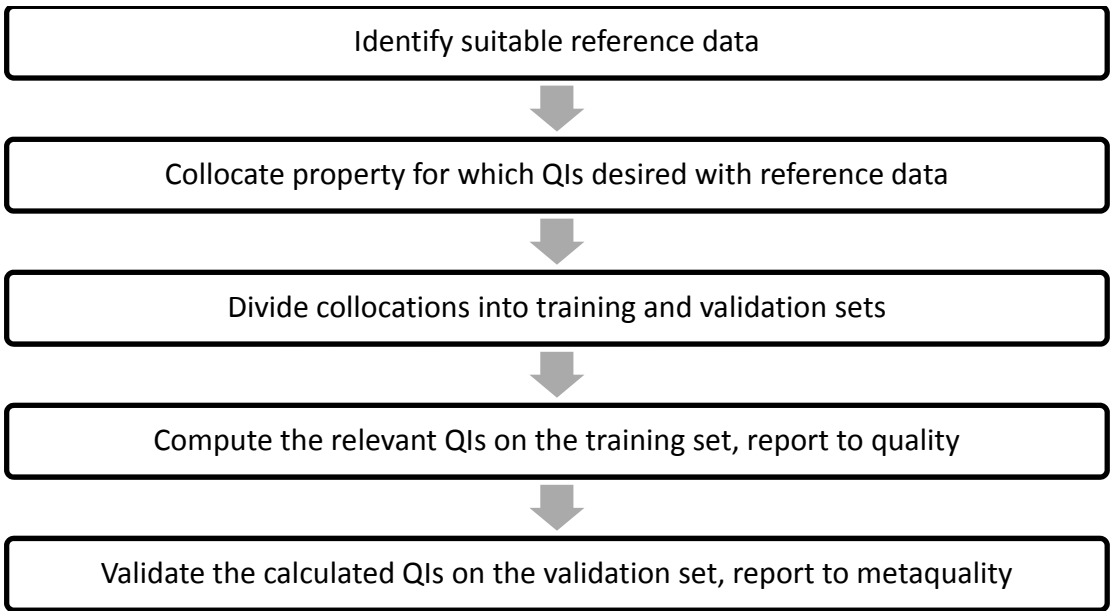

Figure 1. A simplified overview of the process of computing QIs for Earth observation data.

The steps are defined in more detail below:

1. Identify and select appropriate reference observations. This requires knowledge of the observation process and property being observed. This step would usually be carried out by a domain scientist, often within the producer organisation of the dataset under consideration. Such reference observations will typically be collected as part of a Calibration or Validation (Cal/Val) exercise in the commissioning phase of the observation system. Occasionally it is possible to use observations of opportunity as reference data, possibly from fixed professional observing networks. Ideally data used as reference data should be subject to rigorous quality control and in the perfect world will have sufficiently small errors that it can be treated as "truth".
2. Collocate the reference data and the data for which the QIs should be calculated. This is discussed in Section 0. Rarely will the reference data be available at exactly the same location and time at which the target data is available. Thus it is necessary to collocate the data, finding the best matches for the target data, with the available reference data. Best will mean the closest matches in space and time, where this is possible. With remotely sensed EO data, it is very common that the target data has very extensive coverage in space and time, depending on the sampling properties of the instrument and the platform on which it is mounted, but the reference data is sparse in space (but often dense in time), since it is typically obtained from a ground based sensor. Thus collocation involves identifying matches of the target and reference observations that are deemed to be sufficiently close in space and time to be compared. The definition of sufficiently close will depend on the scales of variation of the property being observed (in space and time) and the sampling footprint (in space and time) of the instrument being used to make the observations. This is a very complex issue, and is discussed further later. It can be necessary to interpolate the target data in space and time, in which case care should be taken to ensure the interpolation adds as little error as possible.
3. Divide the collocated data into training and validation sets. The training set, composed of the collocated pairs of target and reference data should ideally be selected in a systematic, yet randomised manner. The systematic aspect should ensure good spatial coverage, and as wide a coverage of other environmental factors as possible (for example different meteorological conditions, such as cloud type and coverage, often have impacts on some types of remote sensing). However it makes sense to retain a degree of randomisation so that the validation set can be selected to also give reasonable coverage.
4. Compute QIs. In the GeoViQua scenarios we consider two main types of data; ratio valued (continuous) data, and categorical data. The two data types require quite different QIs to be computed. For ratio, or continuous valued results there is a very wide range of QIs that can be considered. Most informatively a complete description of the probability distribution of the errors on the target (or observed) values could be provided. Ideally this would be computed using a flexible model, such as a mixture distribution, however operationally it is really only common to fit a normal, or occasionally log-normal distribution to the errors.

The result of fitting such a model can be included in the GVQM using UncertML[1] for the encoding. When fitting a probability distribution it is important the probabilistic model this implies is validated, considering both reliability and resolution (see later). Fitting a probability distribution requires strong assumptions to be made, and if the user is not happy to make such assumptions, then it might be preferable to consider a range of statistics describing the error, such as the mean (error), the standard deviation (of the error), the quantiles (of the error distribution) and so on. Again UncertML can be used to encode these statistics in a GVQM document. If the user wishes to use a metric that is not a widely adopted statistical description of the error with respect to the reference data then the more inclusive QualityML model[2] can be adopted. For discrete valued (categorical) data such as are commonly found in, for example, land-cover classification the natural QIs are either discrete distributions representing misclassification rates, or the statistics describing these, that is confusion matrices, both of which are supported in UncertML.

5. Validate QIs. Having computed the QIs derived from the error with respect to the reference data, especially when fitting something like a probability distribution it is important to validate the QIs. Validation is a form of meta-quality, that is quality information about the quality information, and is informative on the level of trust which a user can place in the provided QIs. Meta-quality should also include provenance information on the derivation of the QIs. For probability distributions, the validation of their fit is not trivial. The reliability of a probabilistic model describes how well the probability judgements it implies translate into the observed data, for example if the probability of a value exceeding 2.0 is 0.1, does this really happen 10% of the time in an independent test set? The resolution of a probabilistic model is the sharpness of the distribution – how tight are the prediction intervals. Put simply the goal of probabilistic modelling is to make the tightest possible predictions, subject to being calibrated. So we validate the distributional fit to the errors using a series of diagnostic plots, and compute some appropriate measures of fit. These are then partially encoded in the GVQM so that the user can trace the judgements and consider for themselves whether they trust the computed QIs.

Once these steps are completed the summary of the process must be translated into XML that follows the GVQM so that the QIs can be shared and used in both data set discovery and use. This is achieved within the Quality Emitter tool, which can either produce an XML fragment describing the quality information or inject this into an existing GVQM document to add the relevant quality information, including meta-quality. In this work we primarily focus on the Quality Emitter tool, which computes QIs on at the data set level which are most useful for data set discovery, however we also consider mechanism for computing QIs at the individual observation (or pixel) level, which are most useful when further using the data.

The Quality Emitter tool forms part of the overall GeoViQua architecture as shown in Figure 2. It forms part of the system that provides metadata, in this case it really should be considered as a tool to enrich the metadata with quantitative quality information conforming to the GVQM.

[1] http://www.uncertml.org/
[2] http://qualityml.geoviqua.org/

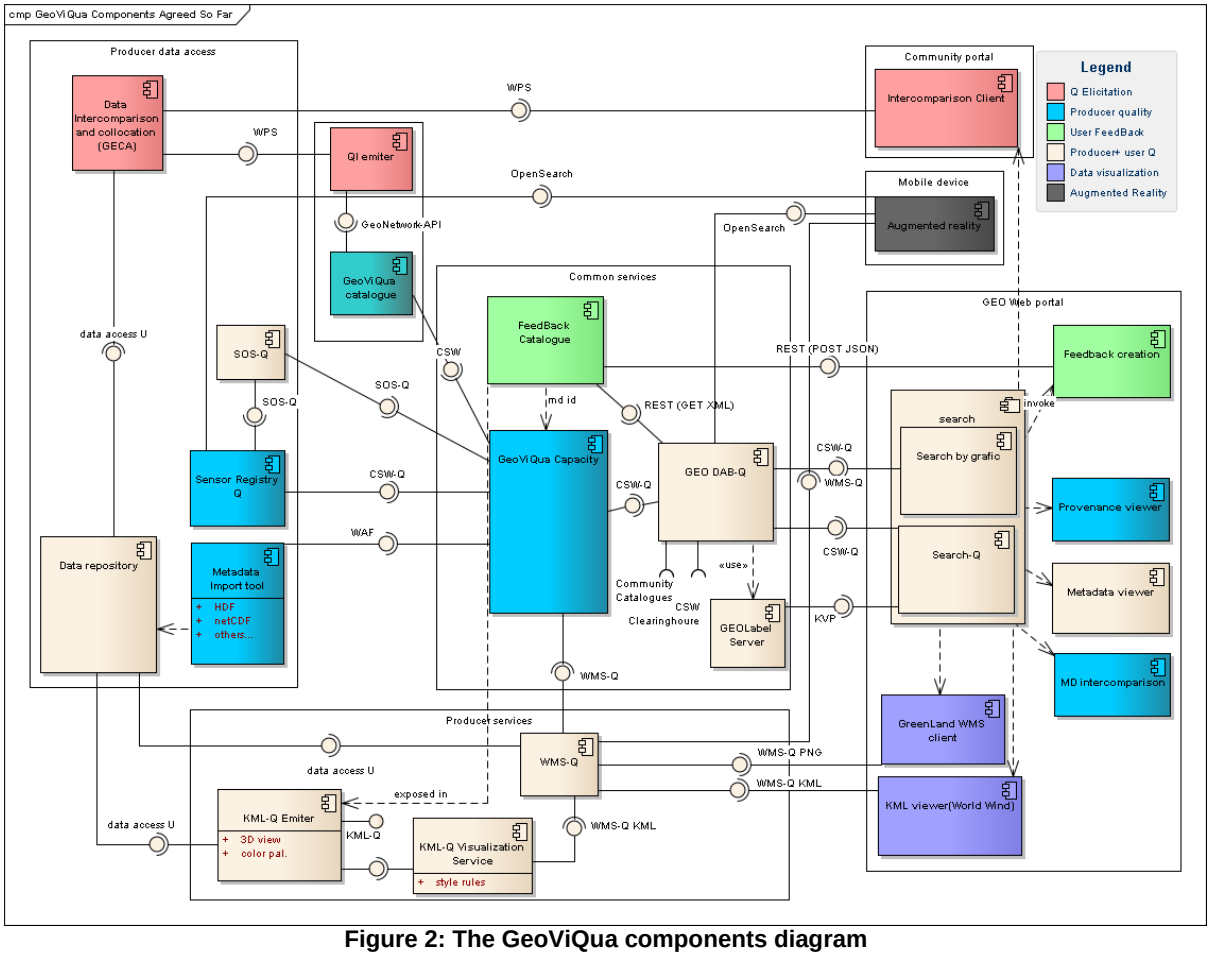

Figure 2: The GeoViQua components diagram

2. Semi-automatic collocation including EO and in-situ (CAL/VAL) data

In order to compute the quality of a given data set in a quantitative manner it is necessary to know the 'true' value of the variable being measured at the give location and time. In general knowledge of the 'true' value is never available to us, for a variety of reasons, and we are forced to consider instead the use of 'reference values', which are of known quality, and ideally synonymous with the 'true' value.

Having identified suitable reference values, the next task is to collocate, in space and time, the observed values with the reference values. The existing GECA software fulfils this role in a tested and mature way, so the decision was taken to build on the existing system, exposing GECA as a service, to permit integration into GEOSS.

2.1. Overview of issues

One of the main issues for collocation and thus intercomparison is being able to access the actual dataset data to obtain measurement time, longitude, latitude, potentially altitude, and crucially the value for each measurement within the configured intercomparison scope. Several aspects complicate the retrieval of such data, such as:

1. Access restrictions and user policies for using the data
2. The lack of standardization in how dataset data is organized (i.e. in files, as a service, in databases)
3. The lack of standardization in file formats used for describing data and associated metadata, for example HDF(-EOS), netCDF, binary formats, SAFE formats, or the correlative data format used by the GECA end user toolbox.
4. The processing time and resources needed for finding potentially matching data for intercomparison within large datasets.

All these aspects need to be addressed before intercomparison can occur, which requires effort. In addition these aspects should be hidden from the user, who expects a clear user interface with parameter selection (GeoViQua deliverable D2.1, D2.2). The exposure of the actual data by dataset providers within GEOSS/GeoViQua is outside the scope of this study. For GeoViQua, ad-hoc interfaces to the relevant ESA (subsets of) datasets were implemented, to which unique dataset Ids were coupled that can be selected by the user.

Once the data sets have been identified and read into the collocation tool, collocation is relatively simple, based on obtaining acceptable matches in space and time, with acceptability thresholds defined by the user.

2.2. GECA

During the course of the GeoViQua project it was decided to integrate the GECA end user toolbox as a Web Processing Service (WPS). The WPS standard [OGC, 2006] provides a natural interface for encapsulating processing servers, and integrates naturally with the GEOSS architecture, as part of the Open Geospatial Consortium service stack.

In order for the intercomparison to be useful, users must be able to configure the analysis, and as such, the intercomparison processing must be run each time the user specifies a configuration. The user can configure the comparison using the following parameters:

- ID of dataset 1 – The first dataset for the intercomparison.
- ID of dataset2 – The second dataset for the intercomparison.
- type – The quantity of interest used in the intercomparison (e.g. ozone concentration, methane)
- quantity – The quantity used in the intercomparison (e.g. parts per million, total column density, volume mixing ratio)
- startTime, endTime – a referenced dataset might have a wide coverage over time. These parameters are used to restrict the intercomparison to the provided times
- boundingBox – similar to the temporal extent, a dataset might have a wide spatial coverage. This parameter is used to restrict the intercomparison to the spatial area defined by provided bounding box
- comparisonType – This parameter acts as a switch between “intercomparison” and “collocation”
- deltaTime (optional) – The maximum time offset between both datasets in hours deltaPosition (optional) – The maximum special offset between both datasets in meters
- reportType – the type of report generated

The prototype has been tested using the test data from the Air Quality Pilot Case, illustrated in Figure 3.

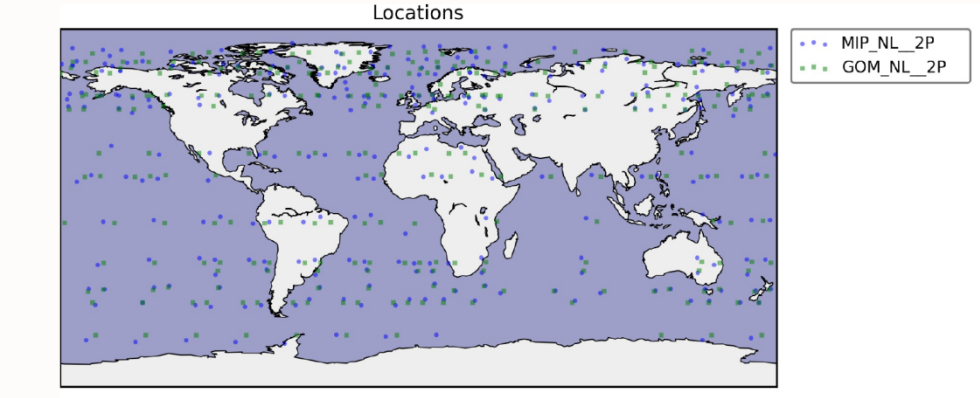

Figure 3. Intercomparison using GECA end user toolbox: the figure shows measurements from two different instruments on ENVISAT (GOMOS, MIPAS) that are close both in time and geo-location.

A WPS provides a more generic interface for the intercomparison processing and can return results potentially including temporal information and altitude information.

2.3. GECA as a service

GECA has been exposed as a Web Processing Service (WPS) to facilitate easy access to collocation tools within GeoViQua. The architecture is shown in Figure 4. In essence the WPS process wraps the core GECA functionality, which permits collocation of two data sources, one of which will typically, but not exclusively be a reference data set. It can also be useful to intercompare two data sets, as shown in GeoViQua deliverable D7.3, Section 4.3, D7.4, Section 4.

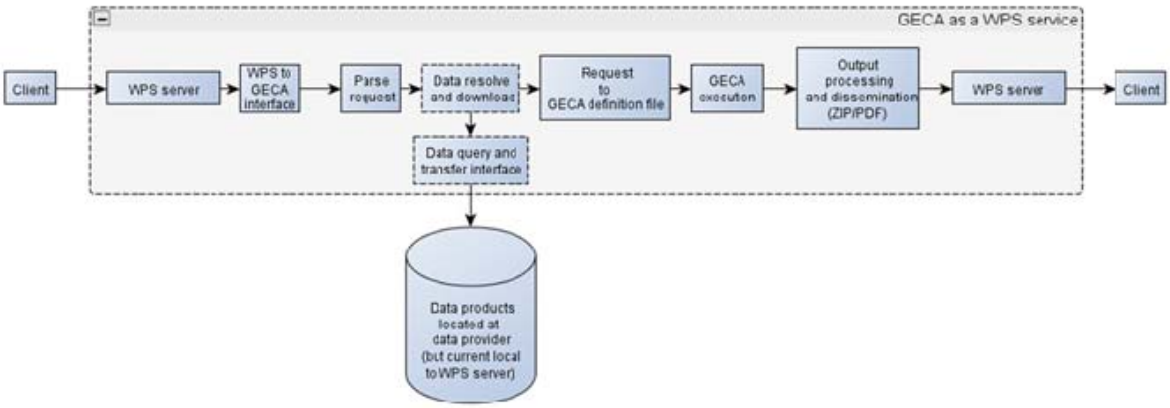

Figure 4. The overall architecture of GECA as a WPS.

GECA as a service enables users to search and find reference data for a dataset (collocation) and to perform an initial assessment of the agreement between dataset thematic values. However a main challenge in providing intercomparison to users is the implementation of the access to the actual dataset data, which is hampered by the absence of widely used standards in dataset organization, file formats, user restriction policies, and interfaces for querying large datasets for individual measurements in terms of measurement time, geolocation, thematic value, and potentially altitude. One solution would be to provide the GECA end user toolbox and its WPS encapsulation as a tool to data providers, such that they can expose intercomparison for their own dataset as a service. This however requires considerable implementation/configuration effort from the dataset provider. We believe that further work is required to define effective mechanisms for enabling timely intercomparison while also providing remote access to very large volumes of Earth observation data.

The GECA end user toolbox yields multiple types of results for the intercomparison, such as tabular data describing the matching measurement samples for both datasets, and statistics on (difference in) values between the two data sets. In addition, the GECA end user toolbox yields pdf reports with plots and tables that are tailored to a specific intercomparison type. Such outputs need to be provided back to the user. The approach chosen within GeoViQua was to supply the user with a dynamic URL that presents intercomparison progress during processing and a link to a zip-file containing the results after intercomparison finishes. In this way users can inspect the results in the report, and perform additional processing on the actual results-file. Note that the standardization of result format was out of scope for this study. The tools developed in GeoViQua to compute quality indicators, described below in Section 3 are able to open and read the resulting zip files and process the results to compute a relevant set of quality indicators for the target data set, to validate these, and to create and insert GeoViQua compliant metadata (GeoViQua deliverable D6.1) representing the outcomes of the collocation and QI computation, including provenance information.

The study has also lead to development of a simple intercomparison client that enables user to configure and monitor the intercomparison processing, and can serve as a basis for future implementations.

3. Computation and validation of quality indicators for continuously-valued data

When computing quality indicators for continuously valued data there are two main approaches that should be considered. The first option, which is generally to be preferred is to compute quality indicators as part of the initial data capture / retrieval methodology. For example if using a remote sensing instrument to retrieve total ozone thickness (for an atmospheric column) it really should be a formal part of the processing of the raw satellite data to produce quality information about the retrieved ozone thickness, which is derived from some inversion or modelling process applied to the raw remote sensing measurements. Including a thorough uncertainty analysis as part of the retrieval / measurement process has several advantages:

1. It is possible to include physical knowledge of the processes and measurement physics in the computation of the QIs.
2. Since the QIs are computed as part of the retrieval or processing they can be computed for every retrieval individually, with the possibility that quality information can be provided per pixel (which could be subsequently averaged to provide data set level quality information if desired).
3. Validation of the QIs would be a natural part of the model fitting / inverse problem that typically defines the retrieval process, and validates not just the QIs but also the retrieval process itself in a very natural manner.

However it is not always possible to undertake a full probabilistic retrieval, or compute per pixel / observation quality indicators, indeed in many existing data sets there are very few quantitative QIs available at either pixel or data set level. The most widely populated QIs at dataset level are often relatively less useful for actually using the data, for example completeness, or lineage, although they can play a key role in data set discovery. Computing QIs post-hoc, that is after the data set has been captured, processed and made available implies that it is only possible to compute data set level summary information on the observation quality. If the quality could be related to a secondary known factor, for example pixel cloudiness, then it might be possible to post-hoc compute individual QIs for each observation / pixel, but this would be a practical challenge and would require dedicated processing / information for each observation type processed. It is far more likely such additional information would be available at data set creation (i.e. the initial retrieval / processing) and per pixel QIs should be computed as part of this step.

When computing post-hoc QIs the focus should be on data set level QIs, since the statistics of multiple comparisons with reference data of known quality can be used to learn about the post-hoc quality of data sets, making some reasonable assumptions about the stationary of errors within the data set. Note that in practice these stationarity

assumptions will be incorrect, however without making further strong assumptions there can be no progress made. Ideally one should attempt to validate the assumptions, exploring whether the residuals of the observed minus reference values are indeed from a single population, or whether there is any evidence of some sort of systematic variation. This sort of check is very challenging to formalise and is best achieved by visualisation of the residuals and their distribution in space and time, which is not covered in the Quality Emitter tool.

3.1. Key qualitative quality indicators

For continuously valued variables the most common quality indicators are probably the bias (mean error) and standard deviation (of the error). To compute the quality indicators is relatively simple in practice. The main challenge is the collocation and matching of the observed data (x_o) with the reference data (x_r). The residual is then simply defined as $r = x_o - x_r$ and it is the statistics of this residual which can be computed. The following key summaries are computed:

- the best approximating normal distribution using moment based estimators (equivalent to maximum likelihood estimators) – this in essence computes the sample mean and (unbiased) variance of the residuals in the training set;
- the mean residual, often also called the bias, computed as the expected value of the residuals in the training set;
- the median of the residuals;
- the standard deviation of the residuals, computed in the usual manner;
- the correlation of the observed and reference values;
- the skewness and kurtosis (that is the third and fourth centred moments) of the distribution of the residuals;
- the quantiles of the distribution, computed by interpolating the sample based, non-parametric cumulative distribution function at levels [0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99].

These summaries were chosen to provide a sufficiently complete description of the error (residual) distribution for most downstream processing. In particular the estimates of the quantiles of the distribution are relatively flexible, and encode a lot of information that can be later used for calculation.

3.2. Validation of quality indicators

While the computation of quality indicators is important, it is also important to consider their validation, and capturing the lineage of the computation. Validation of the statistical characterisation of the errors is done by splitting the collocated dataset into training and validation sets. The QIs are computed on the training set, but the validation set is withheld and used to assess the degree of agreement of the statistics computed on the training set with the validation set. A good agreement gives confidence that the summary statistics capture the errors in the overall data set. This information is presented as a set of additional metrics in the Quality Emitter. These metrics alone are helpful, but to best judge the degree of reliability it is very useful to provide a set of summary plots. The Quality Emitter provides several plots for the user including:

- Reference vs observed plot, which is a simple scatterplot, but shows the +/- 1 standard deviation confidence intervals as well for observed versus predicted values in the validation set. The more reliable the observations the closer this graph should plot to the 1:1 line. The closer the points are to the 1:1 line the smaller the error bars should be too.
- Standard score plot (for errors from reference). The standard score, sometimes called the z-score, computes the observed value minus the reference value, scaled by the standard deviation. If the distribution of the errors is Normal these values should follow a standard Normal distribution, with 95% of the errors falling within the +/- 2 interval on the y-axis. The x-axis is simply a random ordering of the data in the validation set.
- Mean residual from reference shows two plots - the first is a histogram showing the distribution of the errors (observed – reference), and the associated fit to a Normal distribution, in the quantile – quantile (QQ) plot on the right.
- Reliability diagram, which shows the degree to which the Normal approximation to the error distribution learnt on the training set is followed in the validation set. The reliability diagram splits the distribution into several classes, and compares the theoretical probability of being in that class, with the frequency that the residual (error) falls in that class in the validation set. The points should ideally fall along the 1:1 line.
- Coverage diagram, which is similar to the reliability diagram, but shows the proportion of the time the errors are within a given symmetric confidence interval versus the theoretical probability assuming the errors follow the Normal distribution learnt in the training set.

These plots are meant to provide additional information about the structure of the errors in the observed values compared to the reference values. It is not possible to provide all this visual information in a GVQM XML document, so instead the web link to the persistently stored graphs and associated metrics is also embedded in the GVQM document, which of course includes all the QIs represented as UncertML.

4. Quality indicators for categorical variables

When it comes to categorical variables there are fewer quality indicators that make sense to compute and are useful in practice. The main metric of use is the confusion matrix, which shows the frequencies with which the observed classes map to the reference classes. A good example is illustrated in the classification of land-use data (GeoViQua D7.3), which is briefly described here. In this example we pursue the goal of providing per pixel level information on quality.

Figure 5. The Ebre study region.

This work considers data derived from the problem of classifying pixel level Landsat data from the Ebre delta in Spain (Figure 1Figure 5), to determine the degree of flooding in a rice growing region. This is important because management practices have environmental impacts and attract government subsidies. Our focus is on producing an accurate classification, but also a reliable probabilistic prediction of the classification uncertainty. This will then be encoded in UncertML and provided to the users within the GVQM.

The data consists of radiometrically corrected, georeferenced Landsat data from several times across 3 years. There is also some labelled ‘ground truth’ data collected from across the time period at specific locations – this provides our reference data. The land cover is classified into 8 different classes which primarily reflect the extent of flooding in the fields, which is critical in this context. The data set is split into two parts, one for learning, and the other for validation. Figure 6 shows the projection of the data onto the leading three principal components of the training set – showing there is reasonable, but not perfect, class separation. We applied a range of Bayesian classifiers, from simple Linear Discriminant Analysis to a complex variational Gaussian process classifier. All performed reasonably on the independent test set, with the best classifier being an equally weighted committee, which combines the probabilistic classifiers.

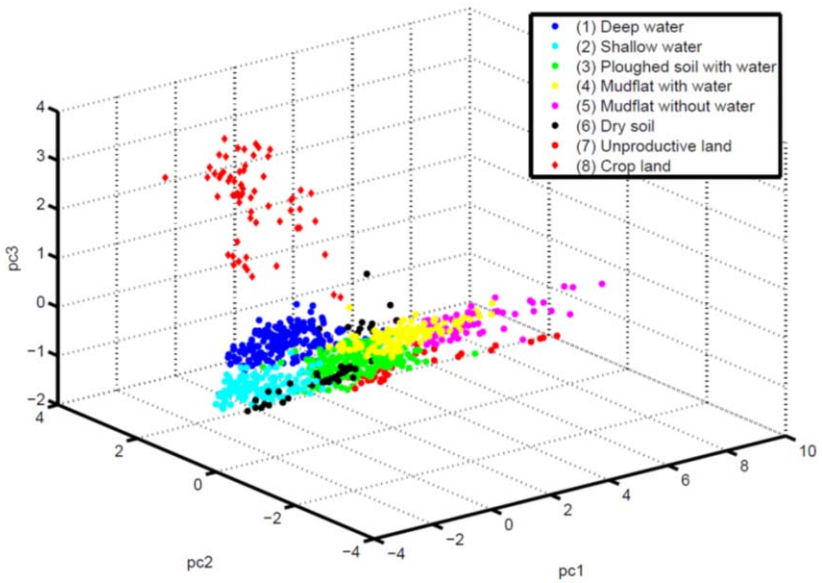

Figure 6. A projection of the training data showing the lack of clear separation between classes.

Typically validation of classification methods proceeds by looking at confusion matrices (Figure 7, left) for an independent validation data set. These can be summarised in various ways, for example the widely used kappa statistic. However this is only half the story; for probabilistic methods Receiver Operating Characteristic (ROC) curves (Figure 7, right) are useful, but are most relevant for algorithm comparison.

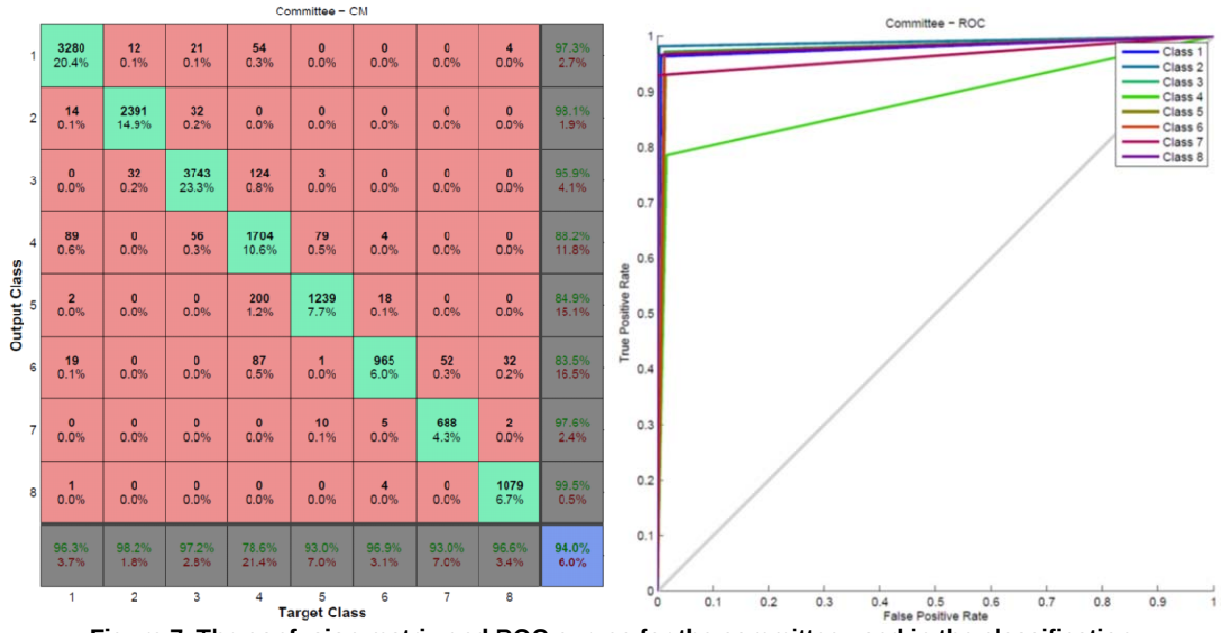

Figure 7. The confusion matrix and ROC curves for the committee used in the classification.

We advocate the use of reliability diagrams (Figure 8) which show the predicted probability of class membership against the frequency that the predictions are correct. For a statistically reliable method the points should plot on a one to one line. They are also useful diagnostics, helping to identify why a probabilistic method is performing badly.

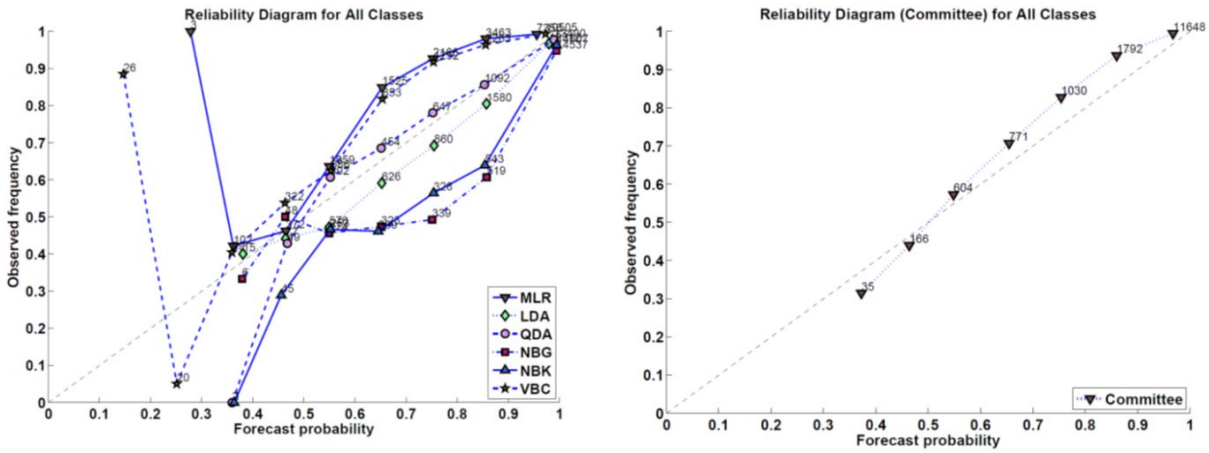

Figure 8. Reliability diagrams for the different methods (left) and the committee (right).

Having validated the probabilistic classifiers, we applied these to the complete images, as shown below in Figure 9. Figure 9 shows the most probable class in each pixel, and Figure 10 shows the entropy ($\sum p_i \log[p_i]$). Figure 10 clearly shows that some locations are far more reliably predicted (low entropy) than others (high entropy). Users of this data would ideally be provided with information on the pixel level uncertainty since this could impact their decision. The next step would be to propagate this uncertainty to produce an estimate of the probability a field was flooded for the period required, and use this to inform subsidy payment.

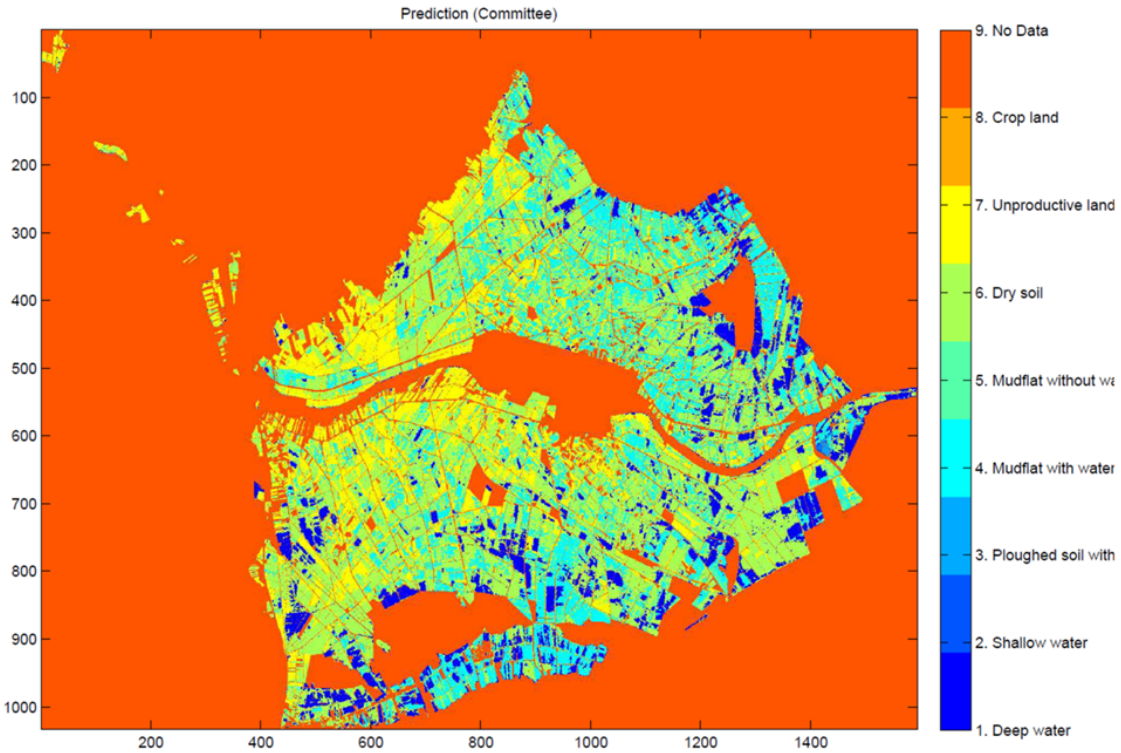

Figure 9. The most probable class, as computed using the committee model.

The next challenge is to convey this uncertainty information to the users. They are likely to be interested in two scenarios:

- discovery - identify a suitable data set with sufficiently small uncertainty;
- use - propagate the uncertainty through some workflow.

At the discovery level summary information is most useful at the data set or product level and might be an overall summary, or a confusion matrix. At the use level the per-pixel probability of each class is most useful – all this can be provided using the GVQM and UncertML. There remain interesting questions about mixed pixels, which complicate the interpretation of the probabilities, but are not dealt with here.

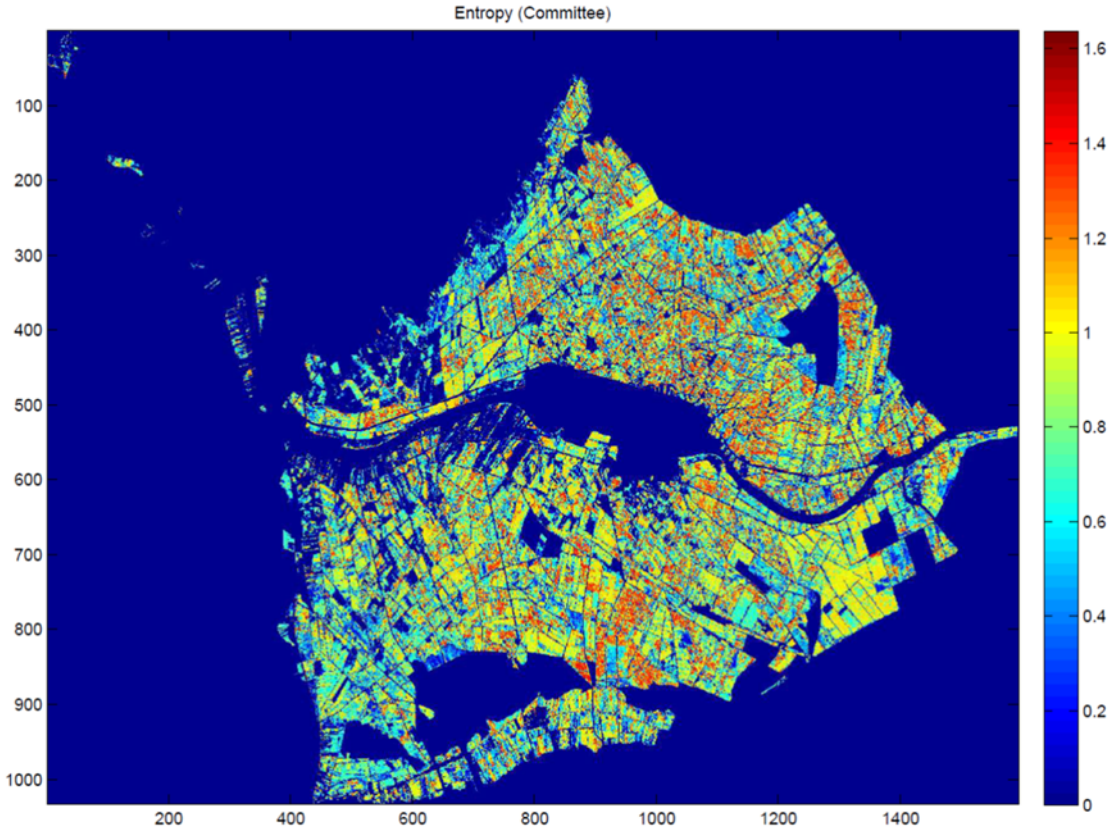

Figure 10. The entropy of the predictions. Large values show pixels in which the classifier is not confident.

Overall this case study, and others conducted in GeoViQua has shown that computing quality indicators for categorical variables is best achieved at the same time as the data is being processed to create the data sets. This cannot be realised as easily by post hoc computation, as is provided in the Quality Emitter, and should really be part of the data producer workflow and responsibility.

5. Integration with GeoViQua architecture

To integrate the computation of QIs (for values derived from GECA as a service or a user's own intercomparison methods) within GeoViQua's architecture, it was proposed that a set of existing software components – a web application with service-oriented architecture called "Emulatorization" – created in-house at Aston could be extended to support the elicitation and validation of such QIs. Emulatorization was developed to build and validate emulators in the Model Web as part of the UncertWeb project, and consisted of two major software components: a backend API that exposed – as a web service – all major data processing and calculation operations involved during the emulator building process, and a frontend web-based tool that guided a user through each step of this process in an intuitive manner. Central to the API's computational tasks was the remote execution of a number of functions written in MATLAB that performed operations such as sensitivity analysis and validation.

5.1. The Quality Emitter

Emulatorization had numerous characteristics of a platform that would be ideal for computing & expressing QIs, having already solved a number of communication and integration issues that would have been encountered during development of a similar tool – for example, interoperable remote execution of the MATLAB functions that compute the QIs described in Section 3 and processing their outputs.

The overall architecture is shown on the following page.

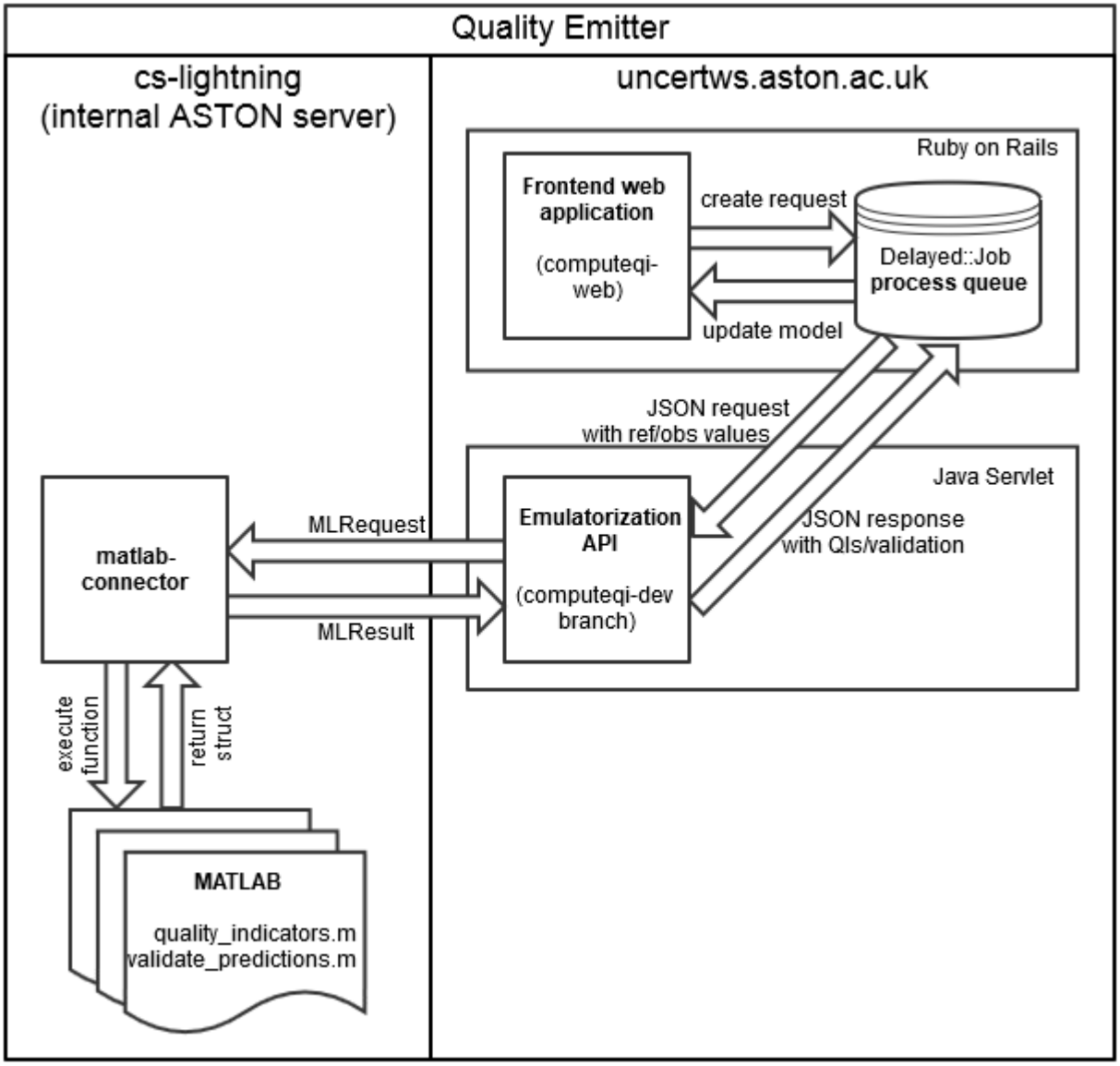

Figure 11. Quality emitter components diagram

The "Emulatorization" backend and frontend components were branched off and a new software lifecycle was undertaken to develop the "Quality Emitter" that adds another operation to its backend API to compute QIs and a separate, heavily-modified version of the frontend tool to leverage this new operation and provide GeoViQua-related functionality of added value such as the automatic generation of relevant GVQ metadata.

5.2. Backend API

The backend API is comprised of a number of Java classes and libraries which facilitate communication between the frontend tool that parses the user-submitted dataset values and the MATLAB functions which compute the QIs for said inputs, exposed as a web service through the use of Java Servlet technology. The format used for data-interchange is JSON, with the Google Gson library converting a valid Quality Emitter API request received through HTTP into a suitable Java object. This object's type is then evaluated at runtime to determine the operation that was requested, with its

parameters extracted and suitable functionality classes called to pre-process and perform the required computational tasks. Once completed, a new response object is created to encapsulate the result, serialised to JSON and returned to the client.

The majority of the underlying computational work done by the API is offloaded to a dedicated internal Aston server which runs several instances of MATLAB. To query these instances in an interoperable manner, we employ a library called "matlab-connector" which abstracts MATLAB data types, enabling the API to instead communicate with MATLAB using Java objects. This could be classed as an individual component in the Quality Emitter's service-oriented architecture since, in theory, one could use the matlab-connector library to construct the request to call the MATLAB functions directly and bypass the backend API. However, this assumes that the integrator has their own server from which they can run MATLAB and the matlab-connector since the computational server is currently not accessible outside Aston's network.

In addition to the provision of new MATLAB functions, The Quality Emitter extends Emulatorization's backend API by implementing a new operation called `QualityIndicators`. The accepted parameters are two scalar arrays containing reference and observed values, a learning percentage which defines the fraction of data to be used for learning with the remainder reserved for validation, and a map of metrics to compute. Each array is then randomised before being split into learning and validation sets and transposed into matrices, ready to be provided as input to the compute QIs MATLAB function alongside a MATLAB structure specifying the requested quality metrics. The function is then called remotely using the matlab-connector, parsing the returned structure containing the computed QIs, and the observed values previously reserved for validation are fitted to a normal distribution. A second MATLAB function is then invoked to validate the observed values against reference values, returning a structure containing additional validation metrics. Both the computed QIs and validation metrics are then serialised to JSON and returned as a response.

```
// Request
// Reference and observed arrays have been truncated to 8 items
{
    "type": "QualityIndicatorsRequest",
    "reference": [
        9, 9.5, 9.6, 9.1, 9.3, 8.7, 8.3, 9.9
    ],
    "observed": [
        9.452057779, 9.651580275, 9.364018434, 9.494989945,
        9.750378571, 9.673729738, 9.64152102, 9.973302737
    ],
    "learningPercentage": 80,
    "metrics": {
        "distribution": [
            "normal"
        ],
        "statistics": [
            "correlation", "mean", "stdev", "skewness",
            "kurtosis", "median", "quantiles"
        ]
```

```
    }
}

// Response
// All arrays have been truncated to a maximum of 2 items
{
    "meanBias": 2.297406538446182,
    "meanMAE": 47.65149181729646,
    "meanRMSE": 59.00776870951006,
    "meanCorrelation": 0.03911552285925779,
    "brierScore": 0.06935385542168675,
    "vsObservedMeanPlotData": {
        "x": [
            429.10292867889194, 327.3501782998139
        ],
        "y": [
            361.8724413184152, 347.00744376690864
        ],
        "yRange": [
            [
                308.86969673075697, 414.87518590607345
            ]
        ]
    },
    "standardScorePlotData": {
        "x": [
            1, 2
        ],
        "y": [
            -1.2684340760748352, 0.35012153972486415
        ]
    },
    "meanResidualHistogramData": {
        "x": [
            -146.75917398545081, -136.96012571704594
        ],
        "y": [
            1, 2
        ]
    },
    "meanResidualQQPlotData": {
        "x": [
            -128.46560447906145, -120.2595583007155
        ],
        "y": [
            -136.2545637574698, -119.71864675004863
        ]
    },
    "rankHistogramData": {
        "x": [
            1, 2
        ],
        "y": [
            3, 6
        ]
```

```
    },
    "reliabilityDiagramData": {
        "x": [
            0, 0.03634669151910519
        "y": [
            0, 0.02982292637465051
        ],
        "n": [
            0, 1073
        ]
    },
    "coveragePlotData": {
        "x": [
            20, 22
        ],
        "y": [
            18.87550200803213, 21.285140562248998
        ]
    },
    "learningPercentage": 80,
    "qualityIndicators": {
        "distribution": {
            "normal": {
                "mean": -6.4685738702154385,
                "variance": 3440.4626278112414
            }
        },
        "statistics": {
            "stdev": {
                "value": 58.65545693122884
            },
            "median": {
                "value": -38.75216878625429
            },
            "kurtosis": {
                "value": 2.3523426789211177
            },
            "quantiles": {
                "values": [
                    -127.04931862811792, -104.37990159487559
                ],
                "levels": [
                    0.01, 0.05
                ]
            },
            "mean": {
                "value": -6.4685738702154385
            },
            "correlation": {
                "value": 0.02247379751463892
            },
            "skewness": {
                "value": -0.048858559857585825
            }
        }
```

```
    }
}
```

Listing 1: A QualityIndicators request and response.

Also included in the API response are various plot data, enabling integrating applications to draw visual aids such as a rank histogram or reliability diagram for assessing the reliability of the validation. The Quality Emitter web application draws these plots as part of the QIs report it generates.

The decoupled nature of the API means that there is scope to integrate the computation of quality indicators into other GeoViQua applications that are independent of the frontend tool.

5.3. Frontend Web Application

The frontend tool was developed using the Ruby on Rails framework, originally chosen for its Model-View-Controller (MVC) architecture, RESTful principles and over 60,000 third-party libraries (called “gems”) which allowed for a number of features such as asynchronous queue processing or the parsing of NetCDF to be seamlessly integrated into the application.

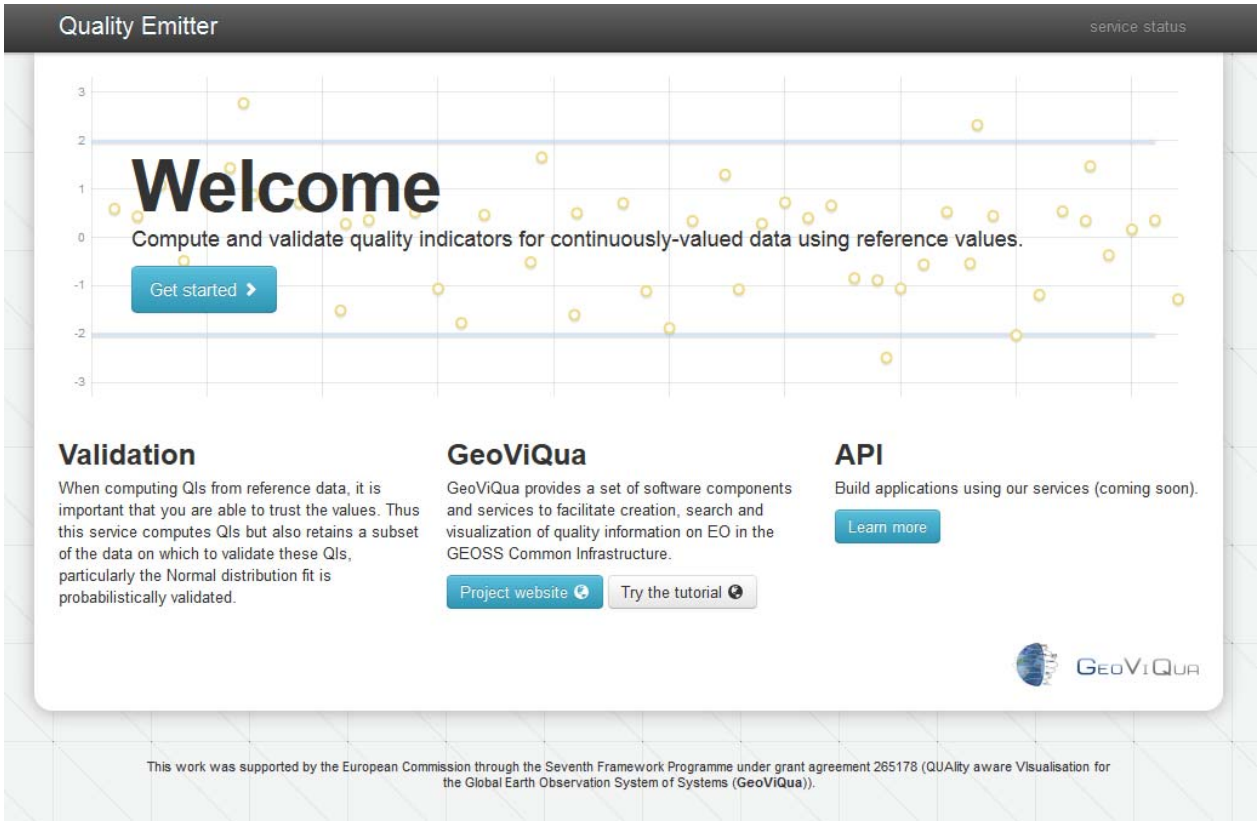

Figure 12. Homepage of the Quality Emitter web application.

The original Emulatorization application was architected in such a way that each stage of building an emulator was represented as a model, in accordance with MVC principles, and that each model class had to implement two methods: one to generate a JSON

request to send to the API, and another to handle in the incoming response. Each model contained fields that represented the parameters of the request and the results from the response. Each model that interacted with the API would be classed as "remotable", with each class including a `Remotable` module that introduced fields and methods that gave the model characteristics of an API request (such as process start time or its current status). Furthermore, a `RemoteableController` was written that utilised Ruby's metaprogramming features to generically handle the process of creating remotable objects. This allowed for the inclusion of Delayed::Job, a database-backed asynchronous queue system for Ruby, to execute API requests on separate daemonised processes since computational-heavy requests may have taken several minutes to complete. When a new model was created the `RemoteableController` would enqueue the remotable object as a `RemoteJob`, which in turn call the remotable's generate method to create the necessary JSON request and send it to the Emulatorization API. If the job completed successfully then the remoteable's handle method would be called to parse the API response and store the result in the model.

The Quality Emitter utilises the same underlying architecture to create and process requests for computing QIs to the API in an asynchronous manner. It currently implements a single model to enable users to create a new `QualityIndicators` request by providing them with a straight-forward two-step process to import the reference and observed values from which they want to compute QIs, enqueuing this request and waiting for a response from the API, automatically displaying the results in an interactive report with a publically accessible URL.

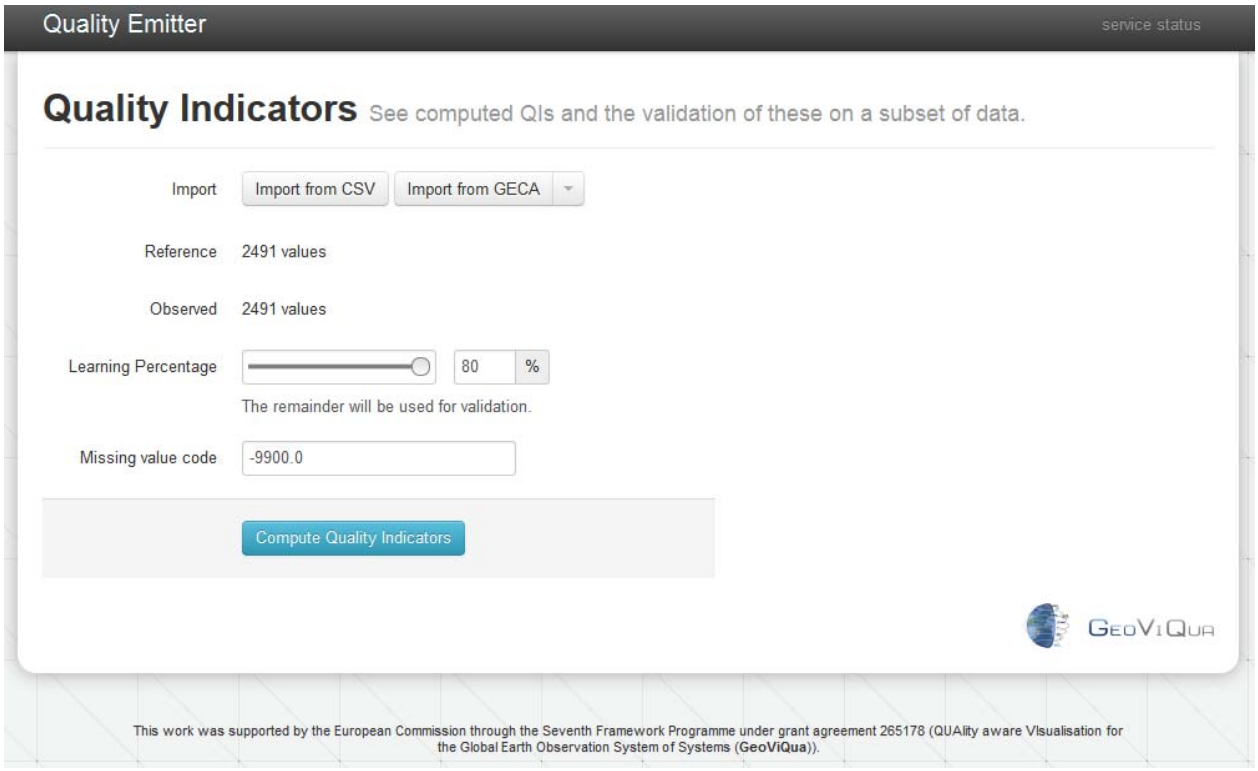

Figure 13. Creating a new report.

The process of computing QIs using the frontend tool starts by creating a new report, as shown in Figure 13, by clicking on the "Get started" button on the homepage (Figure 12). Here the user is able to import reference and observed values from two sources: CSV or GECA report ZIP files; define the percentage of data to be used for learning by interacting with the slider or typing a number manually, and enter a missing value code.

When a user clicks on either of the import buttons a file dialog opens, allowing the user to select the file containing their reference and/or observed values. This file is then uploaded asynchronously using a JavaScript library called Plupload, where it is processed by the `UploadsController`. This controller will check the mimetype of the uploaded file and parse it accordingly. For CSV files the parsing is done using a standard Ruby library function, however for ZIP files containing a GECA report the process is more involved, requiring the use of two gems: nokogiri for reading the ZIP file and ruby-netcdf for parsing NetCDF. Every GECA report ZIP contains a "report_definition.json" file which describes the GECA process and its configured parameters, as well as the pathnames to the NetCDF files used in the intercomparison. To process a GECA report ZIP, the report definition JSON is initially parsed to determine the main variable and the datasets available. Each dataset is then iterated over, writing each NetCDF to a temporary file and reading the values of the main variable's first dimension into a hash, which is then transposed to produce a single column of values for each dataset.

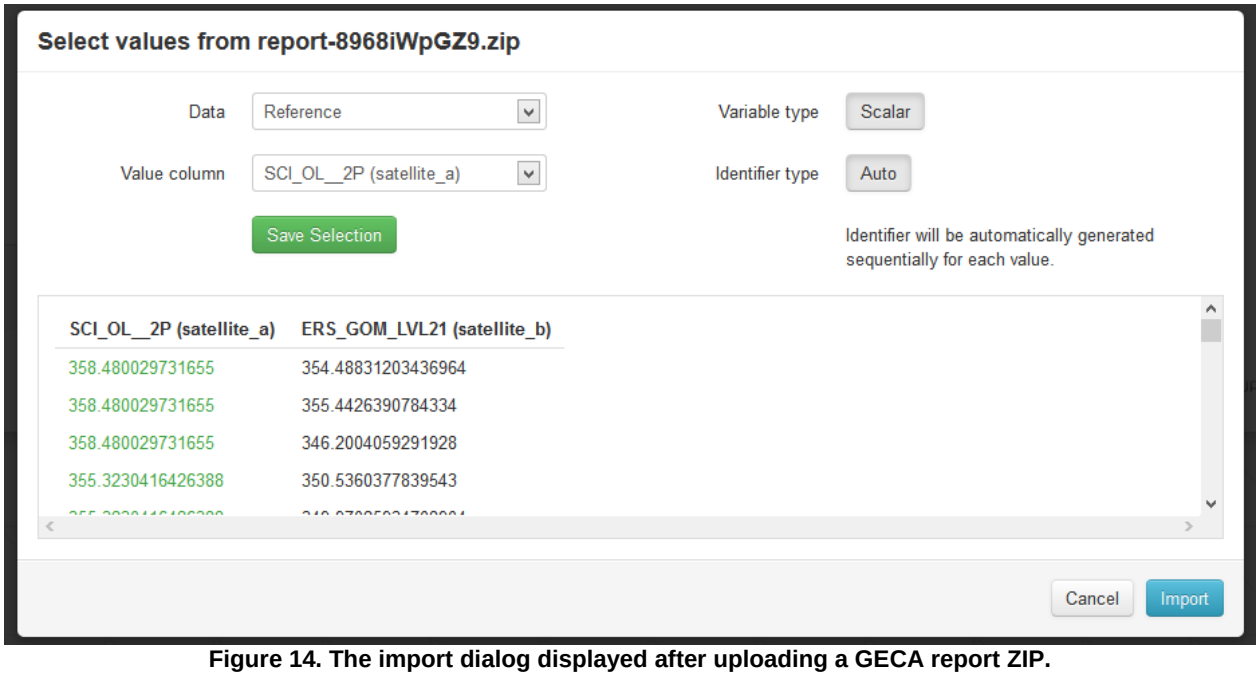

Figure 14. The import dialog displayed after uploading a GECA report ZIP.

The purpose of this process is to send the values contained within the uploaded file back to the browser and display an import dialog that allows the user to select their reference or observed data. In Figure 14, a GECA report ZIP has been uploaded and the user has saved the selection of the first value column as the one containing reference data. When pressing the "Import" button, the saved selections are added to the total pool of reference and observed values, and in this scenario 2491 values would have been added to the pool of reference values to be included in the API request. This

process is repeatable, allowing multiple CSVs or ZIPs to be imported sequentially, each time adding user-selected value columns to the pool.

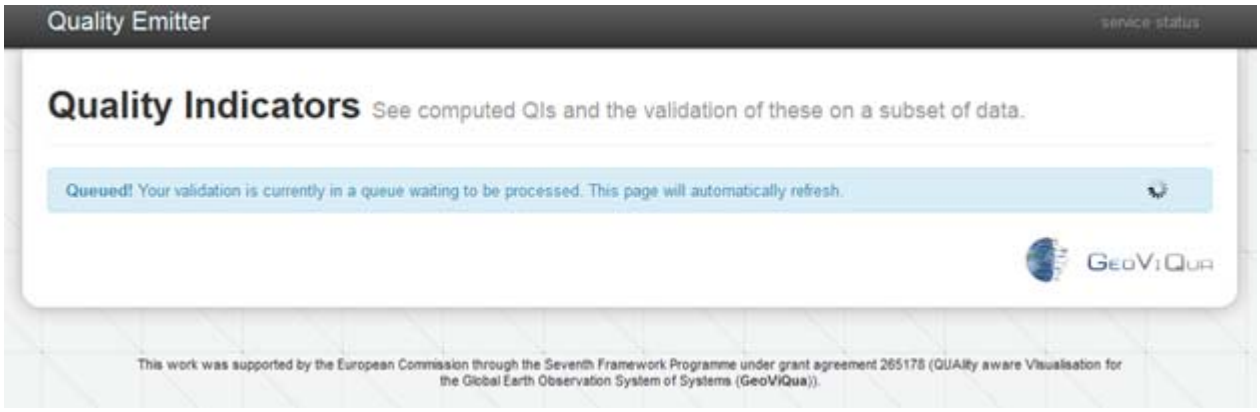

Figure 15. The job is added to the queue, ready to be processed.

Once the user has imported their desired reference & observed values and clicks the "Compute Quality Indicators" button, a new remotable is created containing the imported values and is enqueued for processing. A feature originally implemented in Emulatorization is a polling mechanism to keep the user informed of progress. A small amount of client-side JavaScript requests the remotable resource every 2.5 seconds in JavaScript format rather than the default HTML format, displaying either a progress message or the quality indicators report without the need to refresh the page. If an error occurs whilst processing the job, an error message is returned with an option to re-submit the API request.

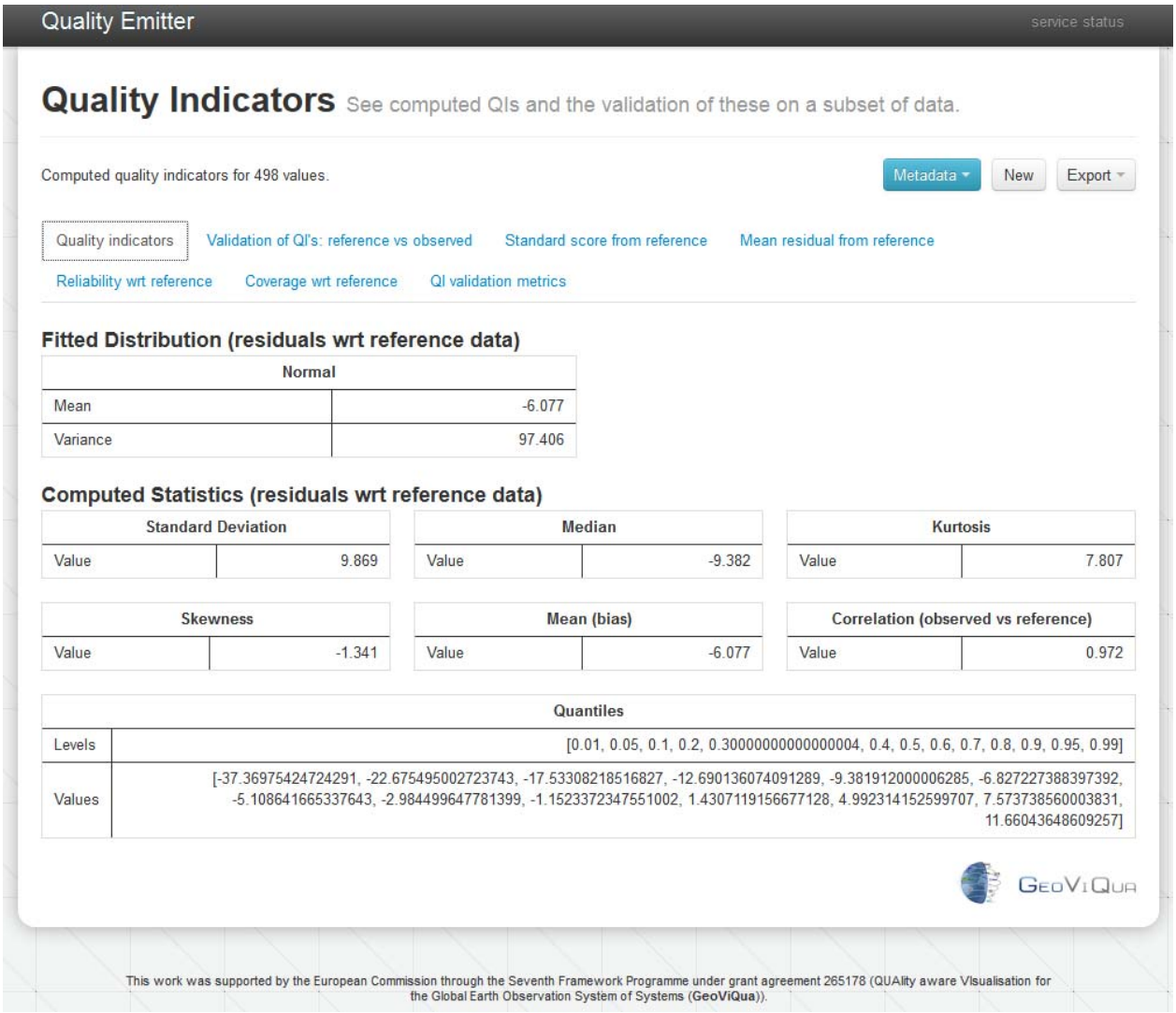

Figure 16. A quality indicator report.

When the API has returned a successful response the remotable job is flagged accordingly and the model updated to contain the returned QIs and their validation metrics. Since the client-side JavaScript is still polling the status of the request, the server-side application now renders the default "view" of the model, a mixture of HTML and embedded Ruby, to display these values across a number of tabs. For example, the first tab (Figure 16) contains a number of tables holding the values of the various QIs that have been computed.

The frontend tool also automatically draw plots and histograms from the validation data – such as the standard score shown in Figure 17 – by using the Flot JavaScript library. Each plot is written as self-contained function that's called when a tab is clicked, with the JavaScript fetching the model as JSON object (by appending ".json" to the end of the report URL) to parse the data required. Since Flot draws to an HTML5 canvas each plot or histogram is able to be interacted with. For example, a user can mouse-over a plotted point to see its X and Y value (Figure 18) or zoom & pan to better view error bars when many values have been plotted.

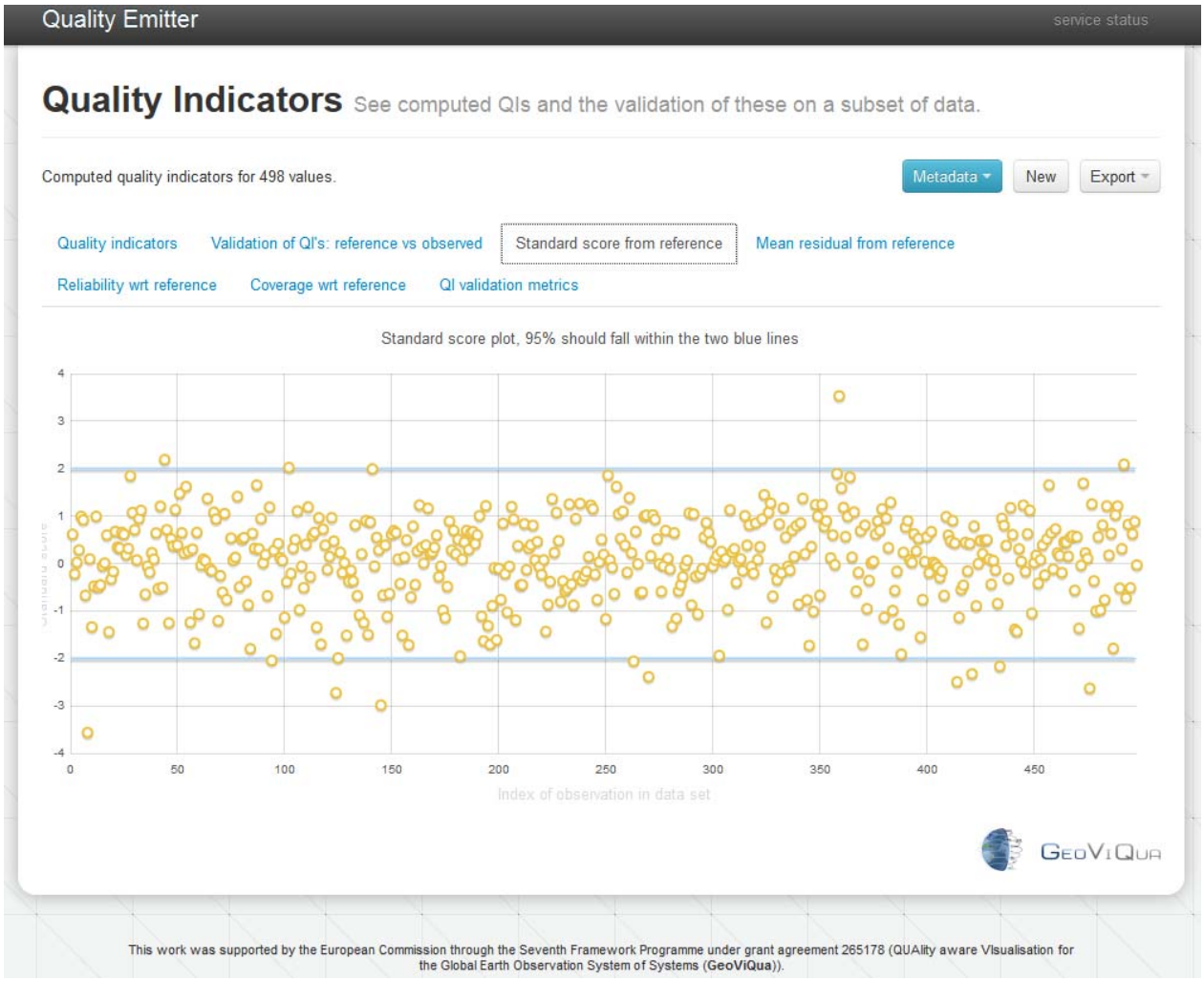

Figure 17. Standard score plot embedded in the report.

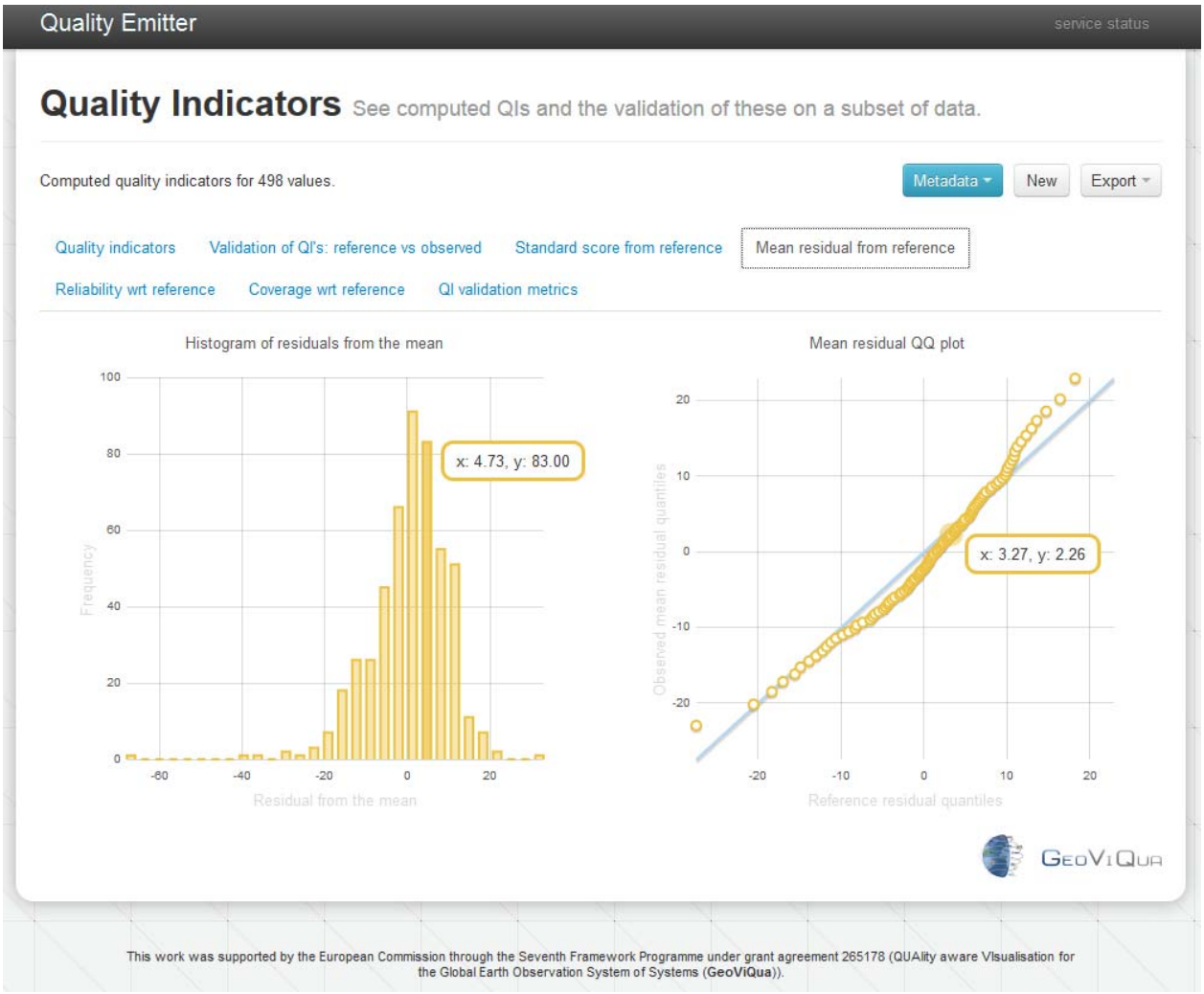

Figure 18. Composition of the "mean residual from reference" tab.

When a user clicks the "Metadata" button at the top of a quality indicators report, they can access GeoViQua-compliant metadata that encodes the computed QIs and the associated meta-quality as a series of quality reports under a gvq:dataQualityInfo element. This metadata is automatically generated by rendering an Embedded Ruby partial containing XML and Ruby expressions that extract the validation metrics from the model.

```
<un:NormalDistribution>
    <un:mean>
        <%= data['distribution']['normal']['mean'] %>
    </un:mean>
    <un:variance>
        <%= data['distribution']['normal']['variance'] %>
    </un:variance>
</un:NormalDistribution>
```

Listing 2: Excerpt from _metadata.xml.erb that creates an UncertML NormalDistribution element.

This partial is then rendered as part of an HTML view (Figure 19) that formats the metadata using the SyntaxHighlighter JavaScript library. An XML view is also available that inserts the QIs metadata into a blank GeoViQua metadata record.

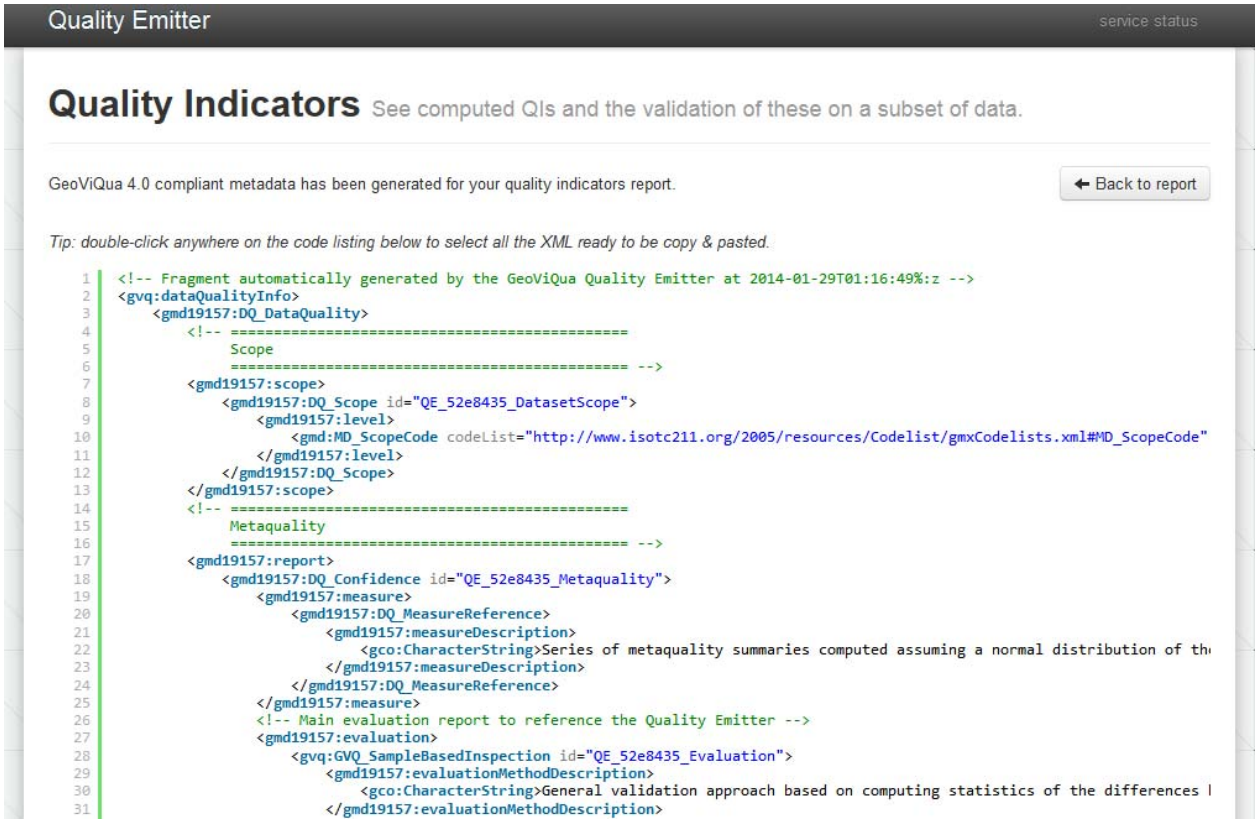

Figure 19. Automatic generation of GeoViQua metadata describing the report.

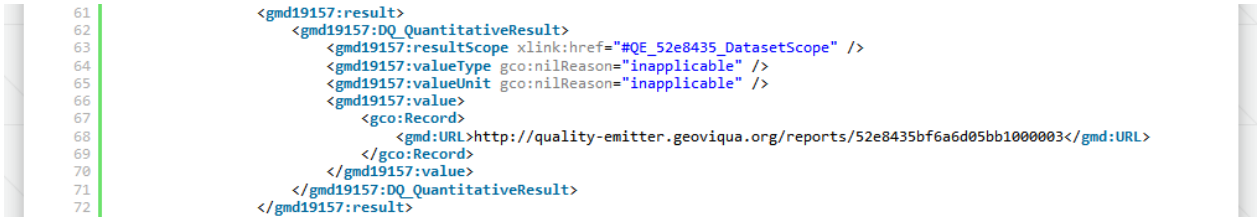

Figure 20. gco:Record element directly linking back to the report within its metaquality.

A DQ_Confidence element describing the metaquality is included, providing a description of the evaluation method, the percentage of data used for learning vs validation, as well as a DQ_QuantitativeResult that references the URL of the quality report (Figure 20).

```
<!-- ================================================
     Quality indicator report: Kurtosis
     ================================================ -->
<gmd19157:report>
    <gmd19157:DQ_QuantitativeAttributeAccuracy id="QE_52e8435_Validation4">
        <gmd19157:dateTime>
            <gco:DateTime>2014-01-28T23:55:10+00:00</gco:DateTime>
        </gmd19157:dateTime>
        <gmd19157:evaluation xlink:href="#QE_52e8435_Evaluation" />
        <gmd19157:result>
            <gmd19157:DQ_QuantitativeResult>
                <gmd19157:resultScope xlink:href="#QE_52e8435_DatasetScope" />
                <gmd19157:valueType>
                    <gco:RecordType xlink:href="http://www.uncertml.org/statistics/kurtosis">Kurtosis</gco:RecordType>
                </gmd19157:valueType>
                <gmd19157:valueUnit gco:nilReason="missing" />
                <gmd19157:value>
                    <gco:Record>
                        <un:Kurtosis>
                            <un:values>7.806554231019095</un:values>
                        </un:Kurtosis>
                    </gco:Record>
                </gmd19157:value>
            </gmd19157:DQ_QuantitativeResult>
        </gmd19157:result>
    </gmd19157:DQ_QuantitativeAttributeAccuracy>
</gmd19157:report>
```

Figure 21. Quality report on the computed kurtosis.

Each validation metric is included as an individual quality report, encoded into a DQ_QuantitativeAttributeAccuracy element using UncertML (Figure 21). Lineage is also present by adding the Quality Emitter as a process step (Figure 22).

```
<!-- ================================================
     Lineage
     ================================================ -->
<gmd19157:lineage>
    <gmd19157:LI_Lineage id="QE_52e8435_ToolReference">
        <gmd19157:processStep>
            <gmd19157:LI_ProcessStep>
                <gmd19157:description>
                    <gco:CharacterString>
                        The following quality reports were automatically generated by the GeoViQua Quality Emitter, a
                    </gco:CharacterString>
                </gmd19157:description>
                <gmd19157:source>
                    <gmd19157:LI_Source id="QE_52e8435_ToolSource">
                        <gmd19157:description>
                            <gco:CharacterString>
                                A service that computes a range of quantitative quality indicators at the dataset leve
                            </gco:CharacterString>
                        </gmd19157:description>
                        <gmd19157:sourceCitation>
                            <gmd:CI_Citation id="QE_52e8435_ToolCitation">
                                <gmd:title>
                                    <gco:CharacterString>Quality Emitter</gco:CharacterString>
                                </gmd:title>
                                <gmd:date gco:nilReason="inapplicable" />
                                <gmd:citedResponsibleParty>
                                    <gmd:CI_ResponsibleParty>
                                        <gmd:organisationName>
                                            <gco:CharacterString>GeoViQua</gco:CharacterString>
                                        </gmd:organisationName>
```

Figure 22. Lineage describing the party responsible for generating the quality reports.

When clicking the “Metadata” button the user is also given the option to insert the generated data quality fragment into an existing metadata record. A simple modal window with a file dialog appears (Figure 13) so the user can upload their metadata record to the Ruby web application. The model’s controller then internally renders the XML partial and inserts the fragment into the document after the last gvq:dataQualityInfo element or last element with the gmd namespace, where it is then sent back to the browser as XML, allowing the user to manually inspect the changes and save the modified document.

Figure 23. Modal dialog to insert generated metadata into an existing metadata record.

5.4. Linking GECA, QIcompute and GeoNetwork

GECA and the Quality Emitter are linked due to the acceptance of a GECA report ZIP as input into the frontend tool, which is automatically processed to extract the collocated values for the user to select reference and/or observed values from. However, this link could be strengthened through tighter integration with GECA as a service. One such method of integration would be utilising the 52North GECA JavaScript client library to implement an additional modal window, where the user could perform a collocation on two datasets before selecting the values without having to use an external website. However, more user-friendly methods for selecting datasets may have to be explored as the current iteration of the client library requires hand-written XML to identify resolvable datasets.

As the Quality Emitter enables users to upload metadata records so that the generated quality reports and meta-quality for the QIs can be inserted, there is scope for broadening the range of methods available for disseminating Quality Emitter metadata. Integration with GeoNetwork is a prime example given the popularity of the open-source geospatial catalogue and the development of a GeoViQua schema plugin. An API is available that would allow the Quality Emitter application to authenticate with a GeoNetwork instance using user-supplied details, download a metadata record, use the same method that inserts the metadata into a user-uploaded document to modify it, and then upload the record back into the catalogue.

6. Summary and recommendations

The work has constructed a system to semi-automatically create and populate GVQM compliant metadata based on the principal of using reference data to compute quantitative quality indicators. The system itself extends the existing GECA system to provide the collocation of the reference values with the observed values for which quality metadata is desired. A new Quality Emitter component is then used to estimate a range of quality indicators, and validate these, and then create and insert the relevant GVQM metadata into an existing metadata document. At present the system supports continuously valued scalar measurements and can compute summary information at the data set level. We have also shown how to compute quantitative quality information at the pixel level for discretely valued data, such as land-use categories. The services to support these processes are online and all code developed in GeoViQua is open source, and available from the GeoViQua GitHub account[3]. The open source nature of the software means it is possible for other users to extend the software and provide a wider range of QIs, or support a wider range of data types.

Several lessons have been learnt in the construction of these services and in the computation of the QIs at both pixel and data set level. Several of these lessons are already understood in some communities, but bear repeating here.

1. Different activities require different QIs, and at different scopes.
   1.1. For data set discovery and selection quantitative QIs are most useful at a data set level, so the user can compare data sets without looking at the detail of the pixel level information, although it can be important to know pixel level information is available.
   1.2. For data set use pixel / observation level quality information is most easily used in subsequent processing. At this level some indication of the probability distribution of the errors on the data is most useful, but in the absence of this meaningful statistical summaries (for example the bias and standard deviation) are useful but require downstream users to make assumptions about the distribution of errors if they are to use this information in an optimal manner. If the specification of a full distribution is challenging then providing a set of quantiles is a reasonable non-parametric alternative.
   1.3. For discrete valued data, such as land use classes, the most complete description of quality information is a per pixel multinomial distribution, giving the probability of the pixel being in each of the possible classes. Users might find a summary measure, such as the entropy in the pixel a useful guide as to when a given class assignment can be trusted.
   1.4. It is possible to aggregate pixel level quality information to the data set level.
2. Computing quantitative QIs requires reference data, which can be used to estimate errors on the observed values. In some settings, for example when validation campaigns are undertaken there is a good chance that reliable reference data will be available (this might often be referred to as “ground truth” data). Ideally this reference data should be used together with the observational data to improve the retrieval method for the observational data to better characterise the per observation / pixel uncertainty.

[3] https://github.com/GeoViQua/

2.1. In general it is likely that for complex retrieval algorithms that are typical in remote sensing, for example the classification methods used to produce land use / land cover data products it makes sense to integrate the determination of quality information with the retrieval. A post-hoc estimate of pixel level quality information will require significant additional information, but at the time of retrieval most methods are capable of providing information on uncertainty as part of their operation.
2.2. Where reference data is not of sufficiently high quality to be considered the true value, which is quite common since many of the Earth system attributes are quite challenging to directly observe, more work is required to define how to include uncertainty in reference values when computing QIs.

3. Meta-quality is important. This can take two forms.
3.1. Lineage information that describes the data and algorithms used to compute the QIs are essential. Users need to be able to reproduce the analysis if necessary and understand any assumptions made, and their impact on the quality statements.
3.2. Ideally some form of formal statistical validation is available, for example providing reliability diagrams or validation statistics that give the users confidence in the quantitative QIs provided. It is suggested that graphical summaries, as provided in the Quality Emitter, are often most informative and convey more information than simple summary statistics or scores.

Providing reliable and meaningful quality information remains a key concern. Users do not feel comfortable without quality information, and ideally quantitative quality information if they are to propagate this into their workflows. Producing quality information is best achieved by data set producers, although we have shown it is possible to post-hoc compute QIs for existing data sets, at least at the data set level. This can certainly help in discovery, and can provide a level of trust in the data. Ideally quantitative quality information should characterise the uncertainty in the data and be validated, allowing trusted propagation in downstream applications. This remains an area that requires further work moving forward, both in producing the quality information and using it effectively.

References

[D2.1] GeoViQua Deliverable D2.1 *User Requirements for GeoViQua*, Version 1.0, 2012

[D2.2] GeoViQua Deliverable D2.2 *System Requirements for GeoViQua*, Version 1.0, 2012

[D3.1] GeoViQua Deliverable D3.1 *Metadata extraction quality component*, Version 1.3, 2012

[D6.1] GeoViQua Deliverable D6.1 *Best practice document for quality encodings*, Version 3.1, 2012

[D7.3] GeoViQua Deliverable D7.3 *Data Quality Parameterisation for the GeoViQua pilot case studies*, Version 1.0, 2013

[D7.4] GeoViQua Deliverable D7.4 *GCI pilot case perspective recommendations for enabling quality aware visualisation and search*, Version 3.1, 2013

[GECA, 2009] Proc. 'Fringe 2009 Workshop', Frascati, Italy, 30 November – 4 December 2009 (ESA SP-677, March 2010) GECA: ESA'S NEXT GENERATION VALIDATION DATA CENTRE. Y.J. Meijer, T. Fehr, R.M. Koopman, A. Pellegrini, G. Busswell, I. Williams, M. De Mazière, S. Niemeijer, R. van Deelen, 2009. http://earth.eo.esa.int/workshops/fringe09/YaskaMeijer.pd

[OGC, 2005] OGC Web Processing Service standard version 1.0.0 OGC 05-007r7 http://portal.opengeospatial.org/files/?artifact_id=24151