Strengthening of R in support of spatial data infrastructures management

Emmanuel Blondel (Independent, Consultant)
Introduction

- Spatial Data Infrastructure (SDI) ~ set of components:
  - Data repository: spatial database, shapefiles, etc.
  - Geographic Data Server: one or more datastores exposing data resources on the web
  - Metadata Catalogue: set of metadata sheets describing data resources

- Programmatic tools for Spatial Data Infrastructure (SDI) management?
  - Two types of tools:
    - standard-related (eg ISO, OGC, OpenDAP, EML)
      → Need to familiarize with standards
      → versioning / long lifecycle, stable, reproducible, sustainable?
    - software-related (eg GeoServer, GeoNetwork, Thredds)
      → Need to familiarize with software
      → versioning?, lifecycle?, stable… or not, reproducible (as long as the APIs do not change suddenly), sustainable?
  - Existing robust tools in other languages, eg Apache SIS / GeoToolkit for metadata,
    geoserver-manager for data publication,
    pygeometra, gsconfig ….
  - … but hard to adopt for many data managers: reserved to a GIS IT specialized community
A good reason to be scared by GIS standards driving Spatial Data Infrastructures: There is a lot of XML (generally)

- Example 1 - ISO 19115 (Dataset metadata)
  - XML
  - The “end product” (less scaring)

- Example 2 - ISO 19110 (Data structure definition)
  - XML
  - The “end product” (less scaring)

Even more scaring when we wonder how we could manage these kind of files with R?
Introduction

- Make tools for SDI management available in R for a targeting a wider data management community: **beyond IT community**
- Complementary to **spatial data handling** tools already available
  - sf, sp, rgeos, rgdal, raster, etc.
- Tools for:
  - **ISO/OGC standard geographic metadata handling**: write, read, validate, convert from other metadata formats (eg Ecological Metadata Language - EML, NetCDF-CF) → `geometa`
  - **OGC Web-Services (OWS) interaction**: including data and metadata services, with binding to `sf` (for data) and `geometa` (for metadata) → `ows4R` package
  - **Software-specific API interaction**: including data and metadata services, eg GeoServer API → `geosapi` package, GeoNetwork API → `geonapi` package
  - **Spatial Data Infrastructure orchestration** → `geoflow` initiative
geometa – Reading and Writing ISO/OGC Geographic Metadata

- Build an API in R for **writing**, **reading** and **validating** metadata sheets following ISO/TC211 and OGC metadata standards

- References
  - ISO standards (some also OGC standards):
    - ISO 19115 (Dataset metadata),
    - ISO 19119 (Service metadata),
    - ISO 19136 (Geographic Markup Language - GML 3.2.1)
    - ISO 19110 (Feature Catalog),
    - ISO 19139 (XML Implementation)
  - ISO 19139 profiles defined to answer specific community needs
    - SeaDataNet CDI/CSR (EU), AS/NZS (Australia/New Zealand)
  - Existing tools in other programming languages
    - Java:
      - GeoAPI / Apache SIS / GeoToolKit essentially;
      - GeoTools in a less extent;
    - Python: pygeometra
Object-Oriented R model (using R6 classes)

Model based on ISO / OGC schemas:
  - ISO 19115 (Dataset metadata),
  - ISO 19119 (Service metadata),
  - ISO 19136 (Geographic Markup Language - GML 3.2.1)
  - ISO 19110 (Feature Catalog),
  - ISO 19139 (XML Implementation)

1 schema element in the standard = 1 class in geometa
geometa – Recent developments

Thanks to financial support of consortium

- Support of multi-language ISO/OGC metadata
- Add INSPIRE metadata validator
- Reach full coverage of standards:
  - ISO 19115-1 (Dataset metadata)
  - ISO 19115-2 (Extension for imagery and gridded datasets)
- Provide a (first) generic converter with other metadata standards
  - From/To EML (with EML and emld packages)
  - From NetCDF-CF conventions (with ncdf4 package)
geometa - How it works?

- All classes inherit from a superclass `ISOAbstractObject` that provides generic functions to deal with geometa objects
- Main functions inherited for all objects

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>encode(...)</code></td>
<td>Writes the geometa object in the equivalent XML (ISO 19139) metadata sheet</td>
</tr>
<tr>
<td><code>decode(xml = xml)</code></td>
<td>Reads a XML (ISO 19139) metadata element into a geometa object</td>
</tr>
<tr>
<td><code>validate()</code></td>
<td>Tests the compliance of the XML produced according to ISO 19139 schemas. By default, this method is triggered with <code>encode(...)</code></td>
</tr>
<tr>
<td><code>save(file = file)</code></td>
<td>Saves the geometa object in the equivalent XML (ISO19139) as file</td>
</tr>
</tbody>
</table>

- Main function `readISO19139` to read geographic metadata from file or url
geometa – How it works - Basic metadata

- Class ISOMetadata (dataset metadata) the starting point...

```r
# create ISOMetadata object
md = ISOMetadata$new()

# metadata identifier
md$setFileIdentifier("my-metadata-identifier")

# parent metadata identifier
md$setParentIdentifier("my-parent-metadata-identifier")

# charset
md$setCharacterSet("utf8")

# metadata language
md$setLanguage("eng")

# print (object summary)
md
```
**geometa – How it works – Basic metadata**

- **Metadata sheet summary (print)**
  - Allows to check the metadata in creation

```xml
<ISOMetadata>
  ....|-- fileIdentifier: my-metadata-identifier
  ....|-- language <ISOLanguage>: eng {English}
   .......|-- value: English
  ....|-- characterSet <ISOCharacterSet>: utf8 {8-bit variable size UCS Transfer Format, based on ISO/IEC 10646}
   .......|-- value: utf8
  ....|-- parentIdentifier: my-parent-metadata-identifier
  ....|-- hierarchyLevel <ISOHierarchyLevel>: dataset {information applies to the dataset}
   .......|-- value: dataset
```
geometa - How it works - codelists

- geometa manages all ISO/OGC standard codelists (loaded together with ISO/OGC schemas when loading the package)
- The list of available codelists can be obtained:

```r
getISOStdCodelists()
```

- The elements of a codelist can be obtained with the method `values()`

```r
ISODateType$values(labels = TRUE)
```

- For *setter* methods, both codelist item or code (string) can be used:

```r
md$setHierarchyLevel(ISOHierarchyLevel$new(value = "series"))
md$setHierarchyLevel("series")
```
For all textual properties of an object, a `locales` argument can be used.
By default, `encode()` will test XML compliance with schemas. Results will appear as R message and comments in XML footer.
md$encode(inspire = TRUE)  # or md$save("metadata.xml", inspire = TRUE)

- With INSPIRE option, INSPIRE compliance results will be added as comments in XML footer.

```r
[geometa][WARN] Element '{http://www.isotc211.org/2005/gmd}dateStamp': This element is not expected. Expected is one of {http://www.isotc211.org/2005/gmd}hierarchyLevel, {http://www.isotc211.org/2005/gmd}hierarchyLevelName, {http://www.isotc211.org/2005/gmd}contact) at line 8.
[geometa][WARN] Object 'ISOMetadata' is INVALID according to ISO 19139 XML schema!
[geometa][INFO] Sending metadata file to INSPIRE metadata validation web-service...
[geometa][INFO] INSPIRE metadata validation test done!
```

*useR! 2019 – Toulouse, France, 11th July 2019*
**geometa – How it works - More?**

- Convert from / to other metadata standards [*EXPERIMENTAL*]
  - From / To EML (EML/emld packages)
    ```r
    #from geometa to emld
    emld_obj = as(md, "emld")
    ```
  - From NetCDF-CF - Climate & Forecast Conventions (ncdf4 package)
    ```r
    #from ncdf4 to geometa
    nc =
    ncdf4::nc_open("http://gsics.eumetsat.int/thredds/dodsC/DemoLevel1B25Km/W_XX-EUMETSAT-Darmstadt,SURFACE+SATELLITE,METOPA+ASCAT_C_EUMP_20131231231800_37368_eps_o_125_l1.nc")
    md_cf = as(nc, "ISOMetadata")
    ```
geometa – How it works - Read an ISO 19139 file

Reading come interesting when we want to update metadata, local or fetched from a remote metadata catalogue:

```r
#metadata example file
mdfile <- system.file("extdata/examples", "metadata.xml", 
                      package = "geometa")

#read it in geometa
md <- readISO19139(mdfile)
```
• Consolidate converter for moving from one metadata standard to another
• Enrich conversion rules ("mappings") liaising with communities:
  ○ With Ecological Metadata Language (EML)
  ○ With NetCDF-CF Conventions
• Provide functions to connect easily to web controlled vocabularies for easier metadata production
• Extend the coverage of native GML support to foster interoperability with OGC web-services through R.
ows4R – Project and Principle

• “OGC Web-Services for R”
• Objective: To provide an interface in R for using OGC web-services, and associated standards. These standards are designed to provide a common way to access and manage geographic (meta)data on the web, such as:
  ○ Catalogue Service for the Web (CSW) → geographic metadata
  ○ Web Feature Service (WFS) → vector data
  ○ Web Coverage Service (WCS) → raster data
  ○ etc.
• Analog to Python “OWSLib”
• Object-Oriented R model (using R6 classes)
### OGC standards coverage

<table>
<thead>
<tr>
<th>Standard</th>
<th>Description</th>
<th>Supported versions</th>
<th>Unsupported versions</th>
<th>Supported R bindings</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGC Filter</td>
<td>Filter Encoding</td>
<td>1.1.0</td>
<td>2.0</td>
<td>ongoing</td>
<td></td>
</tr>
<tr>
<td>OGC Common</td>
<td>Web Service Common</td>
<td>1.1, 2.0</td>
<td></td>
<td>ongoing</td>
<td></td>
</tr>
<tr>
<td>OGC CSW</td>
<td>Catalogue Service</td>
<td>2.0.2</td>
<td>3.0.0</td>
<td>geometa (ISO 19115 / 19119 / 19110 / 19139 XML)</td>
<td>ongoing / seeking sponsors</td>
</tr>
<tr>
<td>OGC WFS</td>
<td>Web Feature Service</td>
<td>1.0.0, 1.1.0, 2.0.0</td>
<td></td>
<td>sf (OGC Simple Feature)</td>
<td>ongoing</td>
</tr>
<tr>
<td>OGC WCS</td>
<td>Web Coverage Service</td>
<td>1.1.0, 1.1.1, 2.0.1</td>
<td>1.0.0</td>
<td>raster</td>
<td>Not yet released - under investigation / seek sponsors</td>
</tr>
</tbody>
</table>
ows4R – Metadata services

- Interact with a CSW-compliant metadata catalogue (e.g. GeoNetwork)

```r
fao_csw <- CSWClient$new(
    serviceVersion = "2.0.2",
    logger = "INFO"
)
```

- Search metadata

```r
cons <- CSWConstraint$new(cqlText = "dc:identifier like '%firms%'")
q <- CSWQuery$new(constraint = cons)
records <- fao_csw$getRecords(
    query = q, outputSchema = "http://www.isotc211.org/2005/gmd")
```

- Get metadata

```r
record <- fao_csw$getRecordById("fao-species-map-tth", outputSchema = "http://www.isotc211.org/2005/gmd")
```
## ows4R – Metadata services

- Operations for metadata transactions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>insertRecord(...)</code></td>
<td>Push a geometa object into a CSW catalogue</td>
</tr>
<tr>
<td><code>updateRecord(...)</code></td>
<td>Updates an existing ISO 19139 record on CSW with a new geometa object. Batch/Selective update based on filters is possible.</td>
</tr>
<tr>
<td><code>deleteRecord(...)</code></td>
<td>Deletes a record on CSW</td>
</tr>
</tbody>
</table>
Vector Data retrieval interacting with an OGC Web Feature Service (WFS)

```r
fao_wfs <- WFSClient$ new(
  serviceVersion = "1.0.0",
  logger = "INFO"
)
```

List all feature types (GIS data “layers”)

```r
fao_wfs$getFeatureTypes(pretty = TRUE)
```

Outputs mapped with `sf` (Simple Features) package. Optional WFS vendor parameters given as arguments (eg CQL_FILTER)

```r
tth = fao_wfs$getFeatures("species:SPECIES_DIST_TTH")
```
**ows4R – Perspectives**

- Support any OGC standard service not yet supported in R
- Priority list, depending on resources
  - **Web Coverage Service support** to manage grid data. 
    Ongoing promising experiments for fetching multi-dimensional raster / grid arrays, with tests on GeoServer and Rasdaman WCS implementation, starting managing outputs with `raster` package (raster / stack objects).

  Objective: To offer a **standard**, reproducible and sustainable way to access and query raster/imagery data in R from the web

  - **WFS Transactional mode** (Push/Update/Delete spatial data through standard protocol)
  - **CSW 3.0 support**
We can now manage geographic metadata with R, interacting with web metadata catalogues… but OGC standards are huge and quite… “indigeste” for newbies… This deserves some simplification...

The geoflow initiative is an attempt to help data managers to manage their SDI in easy and reproducible way

Project page: https://github.com/eblondel/geoflow

Current development status:

- Available Github. First release on CRAN planned for 2019
- Ongoing applications with Food & Agriculture Organization of the United Nations (UN-FAO), the French IRD (UMR Marbec), and French INRA (UMR Dynafor / Zones Ateliers).
geoflow - Objectives

● **Orchestrate**
  ○ Upstream data processings (data qualification, spatial, statistics, etc.)
  ○ Metadata creation / publication / update
  ○ Dataset publication (upload, enabling of spatial data OGC services)

● **Automate**
  ○ Avoid using complex web forms and manual metadata editing
  ○ Avoid repetitive tasks: eg enter same contacts for multiple metadata sheets
  ○ Foster proper discovery of datasets over the web with automated referencing with controlled vocabularies (eg Taxonomy).
  ○ Set-up multiple entry points for data discovery & access from a mutualized (meta)data source

● **Facilitate the implementation of a Data Management Plan (DMP)**
  ○ Set of predefined actions
  ○ Possibility for data managers to plug their own tools
    ■ In-house data sources and repository (eg. PostgreSQL database)
    ■ In-house tools/APIs

● **Foster FAIR Principles (Findable, Accessible, Interoperable, Reusable)**
geoflow - How it works

- A simplified metadata model...
  - Two “tables”:
    - Dataset metadata (1 row = 1 dataset)
    - Contacts (Directory)
  - Contacts, Title, Abstract, Subjects/Keywords, spatial/temporal extent, etc

- Managed through various formats ...
  - Metadata: CSV, Excel, Google Sheets
  - Contacts: CSV, Excel, Google Sheets (+ LDAP on wishlist)

- Various web tools where to push (meta)data...
  - General tools: Zenodo (EU e-infrastructure), others on wishlist (Dataverse, CKAN)
  - Specific tools:
    - Spatialized (meta)data: GeoNetwork, GeoServer
    - Others on wishlist (eg GBIF for biodiversity)
geoflow - How it works

● One configuration file (json) where we declare:
  ○ (Meta)data sources
    ■ Main “entities” (1 entity = 1 dataset = 1 dataset metadata)
    ■ Contacts
  ○ Target tools e.g zenodo, geonetwork, geoserver, etc.
  ○ Actions:
    ■ Create a deposit on Zenodo with DOI attribution
    ■ Create an ISO/OGC metadata
    ■ Publish ISO/OGC metadata sheet (Geonetwork or other)
    ■ Publish a shapefile on Geoserver
    ■ Etc…

● A single R code line: `executeWorkflow("config.json")`
geoflow - In summary
geoflow - Examples of SDI managed with R

- **Fisheries Global Information System** - FIGIS (FAO)

- **Global Tuna Atlas** (FAO / IRD / BlueBridge)
Thanks for your attention