# Reproducible geospatial data science: Exploratory Data Analysis using collaborative analysis environments

**Alber Sánchez[1], Lubia Vinhas[1], Gilberto Ribeiro de Queiroz[1], Rolf Simoes[1], Vitor Gomes[1], Luiz Fernando F. G. de Assis[1], Eduardo Llapa, Gilberto Câmara[1]**

[1]National Institute for Space Research (INPE)
Av. dos Astronautas 1758 – 12227-010
São José dos Campos – SP – Brazil

{alber.ipia, lubia.vinhas, gilberto.queiroz}@inpe.br

{rolf.simoes, gilberto.camara}@inpe.br

vitor@ieav.cta.br

{luizffga, edullapa}@dpi.inpe.br

***Abstract.*** *The answers to current our planet's problems could be hidden in gigabytes of satellite imagery of the last 40 years, but scientists lack the means for processing such amount of data. To answer this challenge, we are building a scientific platform for handling big Earth observation data. We organized decades of satellite images into data cubes in order to put together data and analysis. Our platform allows to scale-up analysis to larger areas and longer periods of time. However, we need to provide scientists with tools and mechanisms to test and refine their routines before interacting with the Big data hosted in our platform.*
*We believe that web services along collaborative analysis environments fit the hypothesis-test pattern followed by researchers while writing scientific computer code. Web services enable us to embed our platform's data and algorithms into collaborative analysis environments such as Jupyter notebooks.*
*To make our case, we prepared a Jupyter notebook where Earth observation scientists can interact with our platform through web services and the analytic capabilities of the programming language Python.*

***Resumo.*** *As respostas aos problemas globais atuais podem estar ocultas em gigabytes de imagens de satélite de observação da Terra adquiridas nos últimos 40 anos, mas nem sempre os cientistas possuem os meios para processá-las e transformá-las em informação. Para responder a esse desafio, estamos construindo uma plataforma científica para processar grandes volumes de dados de observação da Terra. Para isso, nós organizamos décadas de imagens de satélite em cubos de dados, a fim de juntar dados e análises. Nossa plataforma, está sendo concebida para permitir a anĺise de grandes ŕeas com dados de longos períodos de tempo mais longos. No entanto, precisamos fornecer aos cientistas ferramentas e mecanismos para testar e refinar suas rotinas antes de interagir com os dados hospedados em nossa plataforma.*
*Acreditamos que os serviços Web e os ambientes de análise colaborativos encaixam com o padrão de hipótese-teste seguido pelos pesquisadores. Os serviços*

*da Web nos permitem incorporar os dados e algoritmos da nossa plataforma em ambientes de análise colaborativa, como os Jupyter notebooks.*
*Para testar nossa hipótese, nós preparamos um Jupyter notebook onde cientistas da observaço da Terra podem interagir com a nossa plataforma através de serviços web e as capacidades analíticas da linguagem de programação Python.*

## 1. Introduction

Earth observation scientist are unable to use all the available images in their analyses because processing such volume of data demands large hardware resources, new software tools, and sound analysis techniques. These issues and requirements associated to large amounts of data are commonly addressed as the *data deluge* or *big data* [Bell et al. 2009, Boyd and Crawford 2012, Li et al. 2016]. Besides, the current satellite image distribution model is based on files. These files have their own formats and access interfaces. This distribution model had led to problems such as data duplication and the inability to track the files used or required for each analysis. The data used for Earth Observation analysis are either unavailable or just too large for independent result validation which in turn, boosts the scientific reproducibility crisis [Baker 2016, Nature 2016]. For these reasons, we are putting together data and analysis by means of a platform for handling big geospatial data. We are using our platform to research land use and land cover change.

As the amount of data increases, it is more efficient to move the algorithms to the data than the other way around [Borthakur 2007]. However, the conditions and mechanisms by which scientists move their algorithms to our platform is unknown; we would like scientist to focus on analysis and to forget about data structures and computing scalability.

We acknowledge how troublesome is the process of writing computerized scientific analysis routines and we are committed to make easier for scientists to scale up their analysis from the desktop to our platform. We believe the best moment to make our data and analysis available to scientist is at the earliest stages of their analysis. This approach can diminish the amount of rework implied while scaling up analysis.

Unfortunately, each scientist writes analysis routines on its own way. However, it is known they keep notebooks with descriptions, data and results of their experiments. Apart from this, Donald Knuth introduced *literate programming* as a way to develop, document, and publish scientific algorithms relying in both natural and machine language. Furthermore, Jim Gray proposed *Overlay Journals* as means to share, manage, and improved scientists' notebooks [Knuth 1984, Gray 2009]. These ideas are being taken to the web in the form of electronic scientific notebooks which are on-line, collaborative documents that mix code, data, descriptions, and tables to summarize the results of scientific research[Pérez and Granger 2007].

We believe that web services along collaborative analysis environments fit the hypothesis-test pattern followed by researchers while writing scientific computer code. Web services enable us to embed our platform's data and algorithms into collaborative analysis environments which are electronic approximations to the scientists' notebooks and laboratory journals.

In this paper, we examine how our platform can be integrated into the analysis workflow of Earth observation data. To achieve this, we briefly introduce our computing platform and its web services(section 2 and 3). Then, we describe analysis environments and how the into into the scientists' workflow (section 4). Finally, we test our approach by setting up Jupyter notebook — a collaborative analysis environment — in which we mixture the web services provided by our platform and the analysis analytical tools provided by the Python programming language.

## 2. The e-sensing platform

The *e-sensing* [1] project aims to build a platform for handling big geospatial data in order to help scientists to research land use and land cover change. We are organizing decades of satellite images into cubes — tridimensional space-time arrays —- inside our platform and finding the best way to put together data and analysis. The *e-sensing project* is ran by the Brazilian National Institute for Space Research (INPE).

The main requirements to these platforms are *analytical scaling*, *software reuse*, *collaborative work*, and *replication*. Analytical scaling is about allowing users to move their data and code between platforms of increasing processing capacities with little or no modifications at all. Software reuse means the platform must be able to use code from different origins. Collaborative work and replication are about enabling scientists to share and replicate their results [Câmara et al. 2016, Stonebraker et al. 2009]. We are addressing the software reuse, collaborative work, and replication by using open source and open access software and data. For example, inside our platform, we are only using open source software and open access data provided by NASA. But in this document we are addressing only the first step in the analytical scaling requirement.

Our platform is hosting an array database with both MODIS and LANDSAT images. We have been classifying time series of vegetation indexes of the Amazon forest into classes of Land Use and Land Cover Change (LUCC). In post-processing stages, we analyze the trajectories of LUCC over time [Assis et al. 2016, Camara et al. 2016, Lu et al. 2016, Maciel et al. 2017, Maus et al. 2016]. But the data workflow inside our platform relies on a mixture of technologies such as scripting languages (R, Python, Bash), distributed storage (SciDB, Hadoop), and operating system tools. As a result, it is hard for scientists to reproduce our results or to run their own [Câmara et al. 2016]. As mentioned earlier, we chose web services as the way to expose our platform computing capabilities while hiding its internal complexities.

On the other hand, the *CEOS Data Cube Platform* (CEOS-ODC) is a platform for storing, accessing, and managing metadata of remotely sensed data. CEOS-ODC is build on top of the *Australian Geoscience Data Cube*. Both platforms — *e-sensing* and CEOS-ODC — are interested in processing large amounts of satellite imagery and using open source tools. However, they use different type of analysis and architectures. While *e-sensing* is focused on time series analysis, the analysis supported by CEOS-ODC puts spatial before temporal analysis. Regarding architectures, *e-sensing* is built on top of array databases while CEOS-ODC is built around the programming language python and data files; this difference is subtle but important since databases are independent of program-

[1] e-sensing project `http://www.esensing.org/`

ming languages. As a consequence, the *e-sensing* platform is able to run analysis written in different languages while CEOS-ODC is constrained to python scripts [CEOS 2016].

## 3. A web service for retrieving time series

Sharing and re-using computer resources has been important since the 90s because writing software is error-prone and high performance hardware is expensive. Nowadays, *Web services* are the most common way to address this matter. Web services are the standardized way to access software and data over the World Wide Web independently of operating systems and programming languages. Through them, scientists can access the data and algorithms available in our platform and at the same time, web services hide complexities — such as mixed technologies, and distributed storage — behind an uniform interface.

The Web Time Series Service (WTSS) retrieves time series of Earth Observation data for specific locations. WTSS reduces the gap between data and remote-sensing time-series clients through a simple JSON representation. Traditionally, assembling time series of Earth Observation imagery is a time-consuming task because users need to sequentially open several image files, extract some pixels, and then store them. Instead, WTSS connects to an multidimensional array database and makes temporal queries on behalf of the client. WTSS exposes three main operations *list_coverages*, *describe_coverage*, and *time_series*. *list_coverages* returns a JSON list of the available coverages in the service. *describe_coverage* retrieves metadata of a specific coverage. Finally, the *time_series* operation retrieves specific time series [Vinhas et al. 2016]. WTSS implementation is publicly available on-line [2].

Moreover, WTSS has clients for the QGIS software and for the scripting languages R and Python. These WTSS clients enable scientists to access our data from on-line analysis environments.

## 4. Interactive and collaborative analysis environments

Literate programming is an style of coding software in which programs are treated as pieces of literature. That is, natural and machine languages are weaved together into a document where thought order prevails over code optimizations. Its goal is to create programs easier to understand and maintain and to achieve this, literate programming makes explicit the reasoning behind the code [Knuth 1984].

Note how literate programming fits the way scientists analyses their data. Once data is collected, scientists make research questions, then formulate hypotheses for later testing them on the data. The question making and hypothesis formulating is better described using natural language while data processing and hypothesis testing are automated using code.

The modern realization of literate programming are the on-line analysis environments. Using modern technologies, they add collaboration and interactivity to the traditional scientific notebooks and laboratory journals. Some examples are the *R*[3] and Jupyter[4] notebooks. It is worth noticing that R notebooks are focused in *R* while Jupyter

[2]e-sensing code repository `https://github.com/e-sensing/`

[3]R Notebooks `http://rmarkdown.rstudio.com/r_notebooks.html`

[4]The Jupyter Notebook `https://ipython.org/notebook.html`

notebooks support various programming languages. For this reason, we preferred the latter in this paper.

Statistical data analysis is crucial to science. From the computing perspective, the most popular and powerful computing tools for statistical analysis are R and Python. R is a computing environment designed for statistical analysis while Python is a general purpose programming language focused on readability and extensibility. Both support numerical processing, statistical data structures; the former natively while the latter trough code libraries such as SciPy [Ihaka 1998, Jones et al. 01 , OGrady 2016]. Both R and Python are supported by large communities of users coming from either the field of statistics or computer science. In this paper we preferred python because most of the author come from computer science field.

IPython adds facilities to Python for scientific computing. IPython has an interactive command with tailor-made features for scientists, such as code completion, plotting, and parallel and distributed processing. These characteristics are taken to the web in the form of Jupyter notebooks [Kluyver et al. 2016]. For example, the data and algorithms regarding the recent astronomic discovery of gravitational waves are available as Jupyter notebooks [Dal Canton et al. 2014, Usman et al. 2016, Nitz et al. 2017].

## 5. Analysis of time series of vegetation indexes

To test our approach, we setup up a Jupyter notebook for the exploratory analysis of time series of vegetation indexes. The time series are provided through a WTSS server attached to a cube hosted in the e-sensing platform. Our notebook is publicly available[5]. In this notebook, we mix the web services provided by our platform and the analysis analytical tools provided by the Python programming language. Our notebook presents three common jobs regarding time series of vegetation indexes: Exploratory analysis, filtering or smoothing, and classification. Figure 1 is an screen-shot of our notebook running on a web browser.

In the exploratory analysis, we get the data and then plot the time series and its location on a map. Figure 2 shows how to retrieve MODIS data into a data frame which is a table-like data structure.

Once the time series is formated as a data frame, it is simple to apply on it functions that receive and return data frame's columns as parameters. In this way, we smoothed our time series using the Kalman filter, the Fourier decomposition and the Whittaker smoother. The Kalman filter is well known in aeronautics while Fourier and Whitaker are known as good estimators of vegetation phenology [Atkinson et al. 2012, Grewal and Andrews 2010]. For example, Figure 1 shows the code and the application the Whitaker smoother to time series of vegetation indexes in a web browser.

The last example in our Jupyter notebook is classification. We used Dynamic Time Warping (DTW) to classify time series of vegetation indexes [Berndt and Clifford 1994]. DTW is an algorithm that computes a similarity measure — a distance — between two time series. Given a set of time series of known land coverages (the patterns), we compute the DTW distances to a time series of an unknown land cover (the samples). The samples

[5]Python for Data Science in Earth Observation Analysis `http://github.com/e-sensing/wgiss-py-webinar`

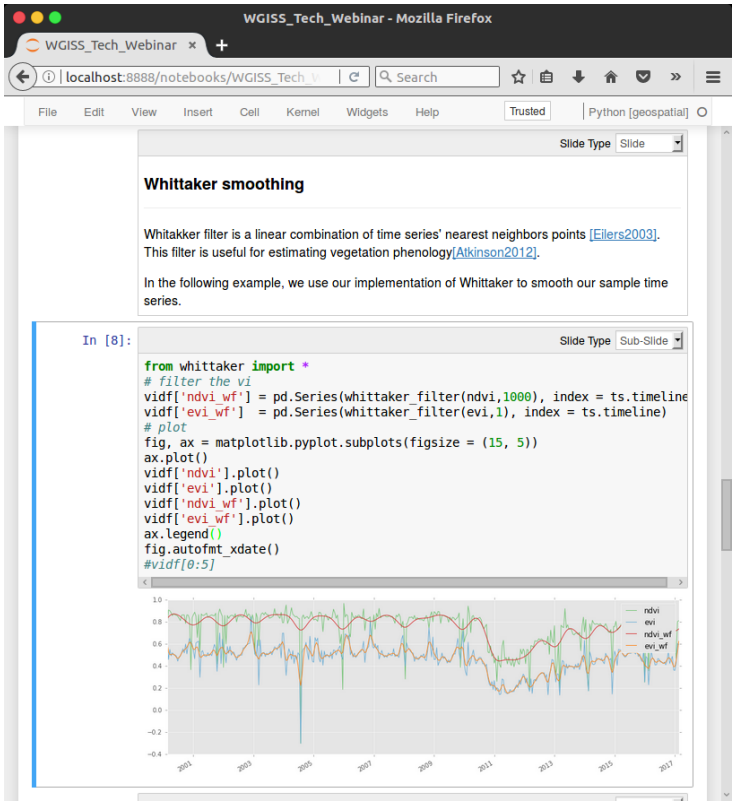


**Figure 1. An on-line analysis environment for time series of Earth observation data. This environment displays a textual description of the Whitaker smoother along its Python implementation and its results when applied to a time series of vegetation indexes.**

are assigned to the labels of the patterns with the shortest DTW distance.

We prepared a set of pattern time series corresponding to the land covers *cerrado* and *forest*. We also collected a set of sample points from which we know the latitude, the longitude and the land cover over a specific time interval; then we retrieved the time series of these points using WTSS. Figure 3 shows the time series of both pattern and samples. Figure 4 shows the code required to read the prepared files, retrieve the time series and to do the classification.

In summary, we joined data and analysis environments in order to plot, filter, and classify time series of Earth observation data by means of Jupyter notebooks and web services. This approach is flexible as users can use the same data and web services over different programming languages and analysis environments. For example, we setup another notebook using *R*, which is an statistical programming language. We do not describe this *R* notebook here because of lack of room, but the code is available on-line. [6]

[6]e-Sensing: Big Earth observation data analytics for land use and land cover change information `https://github.com/e-sensing/SITS_R_notebook`

```
import pandas as pd
from wtss import wtss
from tsmap import *
w = wtss("http://www.dpi.inpe.br/tws")
latitude  = -14.919100049
longitude = -59.11781088
ts = w.time_series("mod13q1_512", ("ndvi", "evi"), \
  latitude, longitude)
ndvi = pd.Series(ts["ndvi"], index = ts.timeline) * \
  cv_scheme['attributes']['ndvi']['scale_factor']
evi  = pd.Series(ts["evi"],  index = ts.timeline) * \
  cv_scheme['attributes']['evi']['scale_factor']
vidf = pd.DataFrame({'ndvi': ndvi, 'evi': evi})
```

**Figure 2. Get a time series into a Python pandas data frame.**

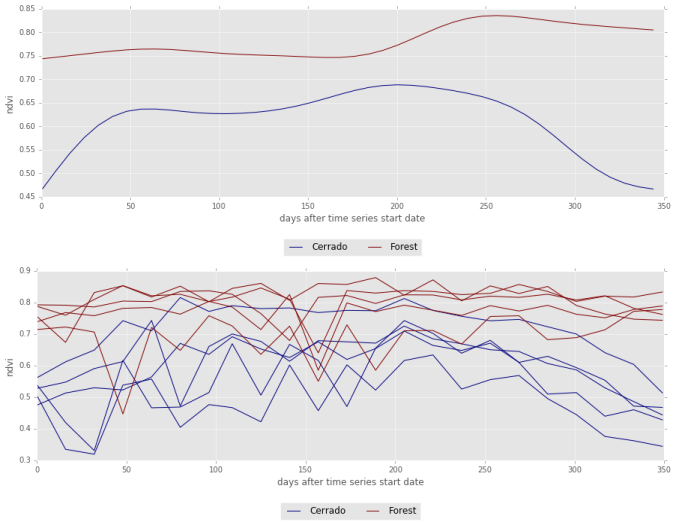


**Figure 3. Patterns (top) and samples (bottom) of NDVI time series for classification.**

```
from dtw import *
from tools import *
patterns_ts = pd.read_json("examples/patterns.json", orient='records')
patterns_ts["timeline"] = pd.to_datetime(patterns_ts["timeline"])
samples = pd.read_csv("examples/samples.csv")
samples_ts = wtss_get_time_series(samples)
classification = classifier_1nn(patterns_ts, samples_ts)
```

**Figure 4. Python code for classifying time series using Dynamic Time Warping.**

## 6. Conclusions

In this paper, we discussed how literate programming is being taking to the Web as interactive and collaborative analysis environments. We also showed how this environments are enhanced with web services and how both — environments and services —-help scientists to prepare their analysis routines. We set up a Jupyter notebook in which we analyzed data retrieved by the Web Time Series Service. In this way, we showed how to display, filter, smooth and classify time series of vegetation indexes. This is a convenient for scientists not only to interact with time series of Earth observation data but also to prepare their analysis routines before running them on big Earth observation data platforms such as *e-sensing*.

Web services close the gap between big Earth observation data and analysis tools by means of collaborative environments for small amounts of data. As the amount of data to be processed increases, it is better to send the analysis routine to the data which is an ongoing effort at the *e-sensing* project.

Finally, we would like to remark that the aforementioned the Jupyter notebook, the Web Time Series Service, and the analysis routine are available on-line to everyone at `http://github.com/e-sensing/wgiss-py-webinar`.

## 7. Acknowledgements

The authors are supported by the São Paulo Research Foundation (FAPESP) e-science program (grant 2014-08398-6). Gilberto Camara is also supported by CNPq (grant 312151-2014-4).

## References

Assis, L. F., Ribeiro, G., Ferreira, K. R., Vinhas, L., Llapa, E., Sanchez, A., Maus, V., and Camara, G. (2016). Big data streaming for remote sensing time series analytics using MapReduce. In *Proceedings of the XVII Brazilian Symposium on GeoInformatics*, number November.

Atkinson, P. M., Jeganathan, C., Dash, J., and Atzberger, C. (2012). Inter-comparison of four models for smoothing satellite sensor time-series data to estimate vegetation phenology. *Remote Sensing of Environment*, 123:400–417.

Baker, M. (2016). Is there a reproducibility crisis? *Nature*, 533(7604):452–454.

Bell, G., Hey, T., and Szalay, A. (2009). Computer science. Beyond the data deluge. *Science (New York, N.Y.)*, 323(5919):1297–1298.

Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In Fayyad, U. M. and Uthurusamy, R., editors, *KDD Workshop*, pages 359–370. AAAI Press.

Borthakur, D. (2007). The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 11(2007):21.

Boyd, D. and Crawford, K. (2012). Critical Questions for Big Data. *Information, Communication & Society*, 15(5):662–679.

Câmara, G., Assis, L. F., Ribeiro, G., Ferreira, K. R., Llapa, E., and Vinhas, L. (2016). Big earth observation data analytics: matching requirements to system architectures. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 1–6, Burlingname, CA, USA. ACM.

Camara, G., Maciel, A., Maus, V., Vinhas, L., and Sanchez, A. (2016). Using dynamic geospatial ontologies to support information extraction from big earth observation data sets. In *Ninth International Conference on Geographic Information Science (GI-Science 2016)*, Montreal, CA. AAG.

CEOS (2016). The CEOS Data Cube. Three-year work plan 2016-2018.

Dal Canton, T. et al. (2014). Implementing a search for aligned-spin neutron star-black hole systems with advanced ground based gravitational wave detectors. *Phys. Rev.*, D90(8):082004.

Gray, J. (2009). Jim gray on escience: A transformed scientific method. *The fourth paradigm: Data-intensive scientific discovery*, 1.

Grewal, M. and Andrews, A. (2010). Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives. *IEEE Control Systems Magazine*, 30(3):69–78.

Ihaka, R. (1998). R: Past and future history. *Computing Science and Statistics*, 392396.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Online; accessed 2011/11/09].

Kluyver, T., Ragan-kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C. (2016). Jupyter Notebooks—a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90.

Knuth, D. E. (1984). Literate programming. *The Computer Journal*, 27(2):97–111.

Li, S., Dragicevic, S., Castro, F. A., Sester, M., Winter, S., Coltekin, A., Pettit, C., Jiang, B., Haworth, J., Stein, A., and Cheng, T. (2016). Geospatial big data handling theory and methods: A review and research challenges. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115:119–133.

Lu, M., Pebesma, E., Sanchez, A., and Verbesselt, J. (2016). Spatio-temporal change detection from multidimensional arrays: Detecting deforestation from MODIS time series. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:227–236.

Maciel, A. M., Vinhas, L., Câmara, G., Maus, V. W., and Assis, L. F. F. G. (2017). STILF - A spatiotemporal interval logic formalism for reasoning about events in remote sensing data. In *Proceedings...*, pages 4558–4565, São José dos Campos. Brazilian Symposium on Remote Sensing, 18. (SBSR), National Institute for Space Research (INPE).

Maus, V., Camara, G., Cartaxo, R., Sanchez, A., Ramos, F. M., and de Queiroz, G. R. (2016). A time-weighted dynamic time warping method for land-use and land-cover mapping. 9(8):3729 – 3739.

Nature (2016). Reality check on reproducibility. *Nature*, 533(7604):437–437.

Nitz, A., Harry, I., Brown, D., Biwer, C. M., Willis, J., Canton, T. D., Pekowsky, L., Dent, T., Williamson, A. R., Capano, C., De, S., Cabero, M., Machenschalk, B., Kumar, P., Reyes, S., Massinger, T., Lenon, A., Fairhurst, S., Nielsen, A., shasvath, Pannarale, F., Singer, L., Macleod, D., Babak, S., Gabbard, H., Veitch, J., Sugar, C., Zertuche, L. M., Couvares, P., and Bockelman, B. (2017). ligo-cbc/pycbc: O2 production release 19.

OGrady, S. (2016). The redmonk programming language rankings: January 2016. 2016. *URL: http://redmonk. com/sogrady/2015/07/01/language-rankings-6-15/(visited on 2017/11/09)*.

Pérez, F. and Granger, B. E. (2007). IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29.

Stonebraker, M., Becla, J., DeWitt, D. J., Lim, K.-t., Maier, D., Ratzesberger, O., and Zdonik, S. B. (2009). Requirements for Science Data Bases and SciDB. In {*CIDR*} *2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*.

Usman, S. A. et al. (2016). The PyCBC search for gravitational waves from compact binary coalescence. *Class. Quant. Grav.*, 33(21):215004.

Vinhas, L., de Queiroz, G. R., Ferreira, K. R., and Câmara, G. (2016). Web services for big earth observation data. In *GeoInfo*, pages 166–177.