ICASAR Manual

Matthew Gaddes and Andrew Hooper

December 17, 2019

1 Installation

In addition to common packages such as numpy, ICASAR also requires t-SNE and HDBSCAN. t-SNE is distributed as part of the scikit-learn package, whilst HDBSCAN is available as its own package.

`package-list.txt` can be used to recreate the Python environment used to make the figures in this manual, but as HDBSCAN is not available from the default conda channels, it must be installed separately. E.g. to create an environment named icasar:

```
conda create -name icasar -file package-list.txt
```

and, to install HDBSCAN from Conda Forge:

```
conda install -c conda-forge hdbscan
```

Note that interactive plotting is configued to work when using Spyder (a common IDE for use with Python), and may not work if ICASAR is run in a different manner.

2 Introduction

ICASAR applies spatial independent component analysis (sICA) to interferograms in a robust manner with the aim of retrieving the latent sources that combined to form them. Applying sICA to InSAR data that covers volcanic centres is discussed in Ebmeier (2016), and Gaddes et al. (2018), whilst the ICASO algorithm from which the ICASAR algorithm was derived is described in Himberg et al. (2004).

The ICASAR algorithm is discussed fully in Gaddes et al. (2019), but is summarised here. It performs sICA many times with both bootstrapped samples of the data and different initiations of the un-mixing matrix. This produces a suite of many repeated sources, which we project onto a 2D hyperplane using t-SNE and cluster using HDBSCAN. The source most representative of each cluster is then found, and a simple inversion performed to calculate the time course for each source (i.e. how strongly they are used to create each interferogram). If you use this software in your research, please cite it as Gaddes et al. (2019).

3 Organising Data

To recover spatially independent sources, each interferogram is considered a variable, and each pixel an observation. In the style of most ICA literature, we therefore organise each interferogram as a row vector. For a time series of i interferograms each of p pixels, this produces an input matrix of size $i \times p$.

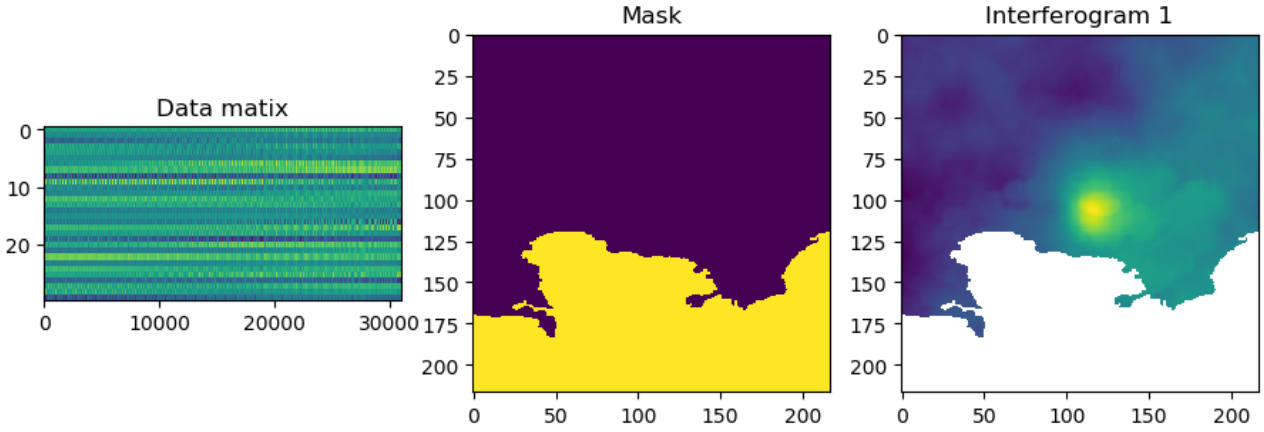

Figure 1: Left: a data matrix for a time series of 30 interferograms each of 31032 pixels. Centre: The mask that converts row vectors into 2D images. Right: Interferogram one.

The data do not need to be mean centred (i.e. the mean of each row set to 0), but the outputs (spatial sources and time courses) return mean centred mixtures. Should a user wish to reconstruct the time series using a selection of the recovered sources, the mean centering process should be performed by the user in order for its reverse to be performed on the reconstructed time series.

Masked arrays are used to convert between row vectors and 2D images. Figure 1 shows a data matrix, its associated *pixel_mask*, and the effect of applying the *pixel_mask* to one of the row vectors.

4 Inputs and Tunable parameters

Setting the number of sources to recover *n_comp* is generally the most important parameter to set correctly. The bootstrapping parameters, HDBSCAN parameters, and t-SNE parameters are somewhat interlinked, as by changing the total number of sources recovered (i.e. changing the number of times the ICA algorithm is run) the number of points per cluster may need fine tuning for the clustering to work well. The t-SNE parameters do not impact on the final sources recovered, and only impact the visualisation.

4.1 Inputs

4.1.1 phUnw

The unwrapped phase for each pixel throughout the time series. Organised as per section 3.

4.1.2 mask

The mask to convert row vectors into 2D images. As described in section 3

4.1.3 figures

Boolean operator to control whether figures are plotted.

4.1.4 scatter_zoom

Controls the size of the source previews in the 2D scatter plots (such as figure 3).

4.2 Tunable parameters

4.2.1 n_comp

This controls the number of sources that are recovered by principal component analysis (PCA) during whitening. Generally best set to slightly higher than the number of sources through to have generated the data (discussed fully in Gaddes et al. (2018)). If it's too high, too much noise will be present for the ICA algorithm to work well, and it will struggle to converge. If it's too low, important signals may be discarded with the higher principal components.

4.2.2 bootstrapping_param

($n\_bootsrap, n\_non\_bootstrap$)This parameter sets the number of times the ICA algorithm is performed on either bootstrapped or non-bootstrapped samples of the data. e.g. $(200, 0)$ for 200 bootstrapped samples, and no non-bootstrapped samples; $(0, 20)$ 0 bootstrapped samples, and 20 non-bootstrapped samples; and $(100, 40)$ for 100 bootstrapped and 40 non-bootstrapped. Note that the latter case is not advised as clustering will struggle with the different densities of the two types of recovered sources. Default: $(200, 0)$

4.2.3 hdbscan_param

($min\_cluster\_size, min\_samples$) These parameters control the clustering of the recovered sources, and are discussed in more detail in Mcinnes et al. (2017). $min\_cluster\_size$ sets the smallest collection of points that can be considered a cluster. $min\_samples$ sets how conservative the clustering is. With larger values, more points will be considered noise. Default: $(35, 10)$

4.2.4 tsne_param

($perplexity, early\_exaggeration$) These parameters control the 2D manifold representation of the data, and are discussed in more detail in Maaten and Hinton (2008). $perplexity$ should lie in the range $5 - 50$. Smaller values emphasise the local structure of the data, whilst larger values emphasise the larger structures in the data. $early\_exageration$ controls the space between clusters. Default: (30,12)

4.2.5 ica_param

($tolerance, max\_iterations$) These parameters control the FastICA algorithm. $tolerance$ sets the threshold at which changes must be below for the algorithm to be considered to have converged (i.e. with smaller values, it will take longer for the algorithm to converge). $max\_iterations$ controls the number of iterations at which the algorithm is considered to have failed to converge. Default: $(1e-2, 150)$

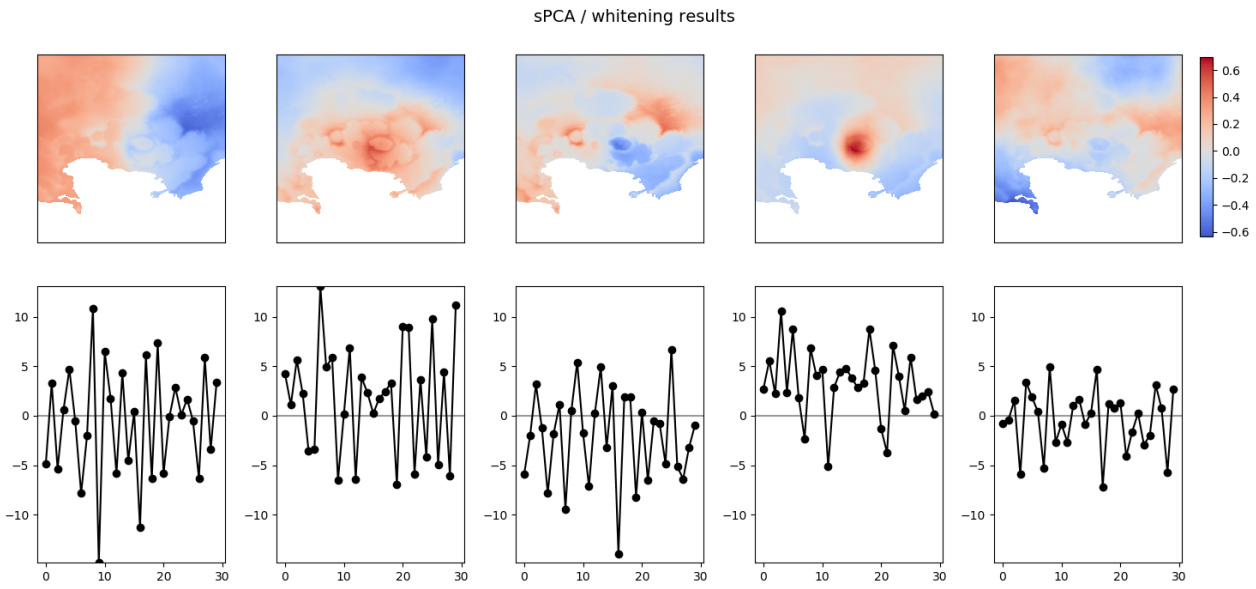

Figure 2: Results of applying PCA to the data. Note that despite only three sources having been used to generate the data, their signals are seen in all of the first five components.

4.2.6 source_order

Controls whether the chosen sources are ordered by their cluster's quality index (I_q), or by the strength of their time course.

5 Synthetic Example

The synthetic example contains three sources stored as row vectors in S_synth. In order, these are inflation modelled as a Mogi source, a topographically correlated atmospheric phase screen (APS), and an east-west phase gradient. Each source has an associated time course which determines how strongly the spatial pattern features in each interferogram, and are stored as the columns of A_dc. We also introduce noise in the form of a turbulent APS, and 30 of these are stored in N_dc. These are mixed together $(A\_dc \times S\_synth) + N\_dc$ to form a daisy chain of interferograms. We wish to recover the three sources and their time courses.

Figure 2 shows the results of applying PCA to the data. It can be worthwhile to explore changing the number of components (n_comp) that PCA recovers, as too few will omit important signals, whilst too few will incorporate too much noise. Figure 3 shows the results of the 200 bootstrapped runs of sICA. This can be explored through hovering over each point to reveal its spatial map. Figure 4 shows how the clusters found by HDBSCAN grew, and is generally of less importance than the other figures. Figure 5 shows the sources chosen as most representative of each of the clusters identified by HDBSCAN, and their time course. In this case, they are ordered by the cluster quality index.

Bibliography

Ebmeier, S.K. (2016). Application of Independent Component Analysis to multi-temporal InSAR data with volcanic case studies. *Journal of Geophysical Research: Solid Earth*, pp. 8970–8986. **URL:** *http://doi.wiley.com/10.1002/2016JB013765*

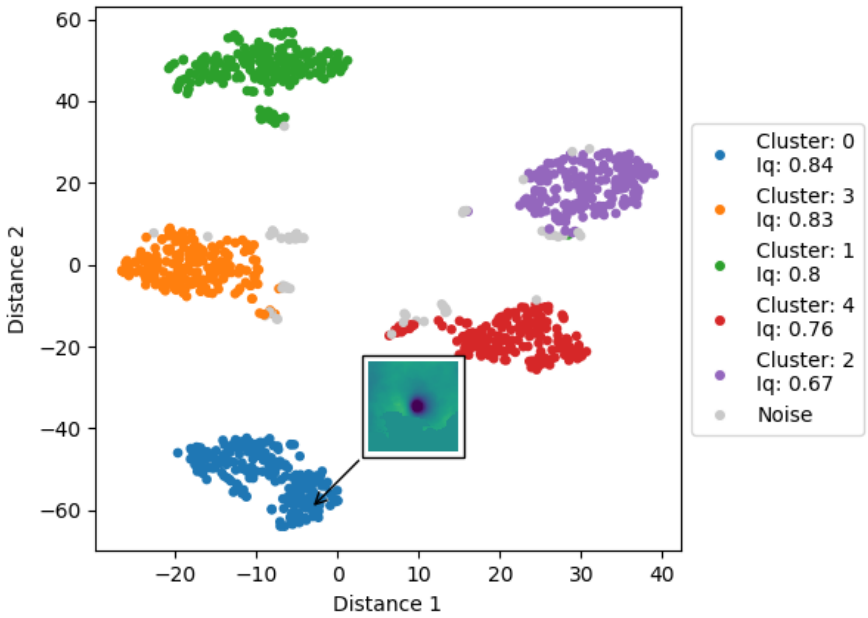

Figure 3: Results of the 200 sICA runs, with the cursor hovering over one point to show the source recovered. The clusters are generally compact and isolated, and the 2D manifold learned by t-SNE is in agreement with the clustering performed by HDBSCAN.

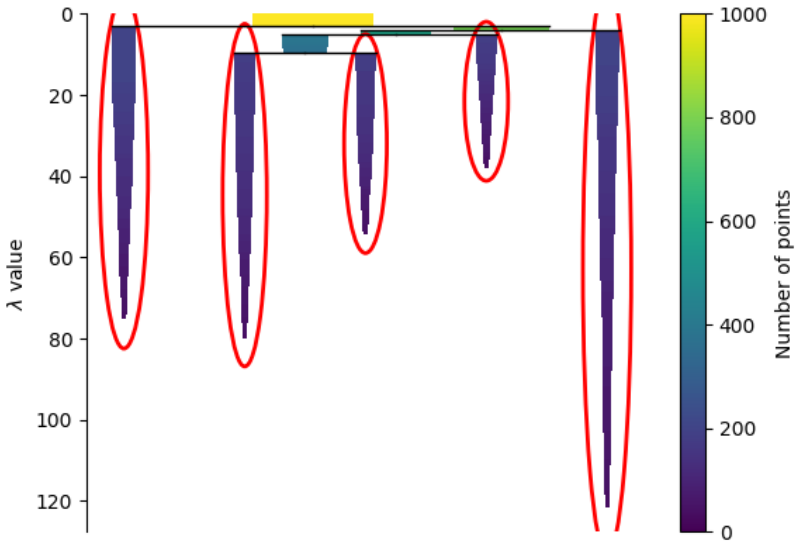

Figure 4: HDBSCAN cluster history, showing how the clusters grow and merge. The red circles show the chosen clusters (i.e. the height that the tree was cut at).

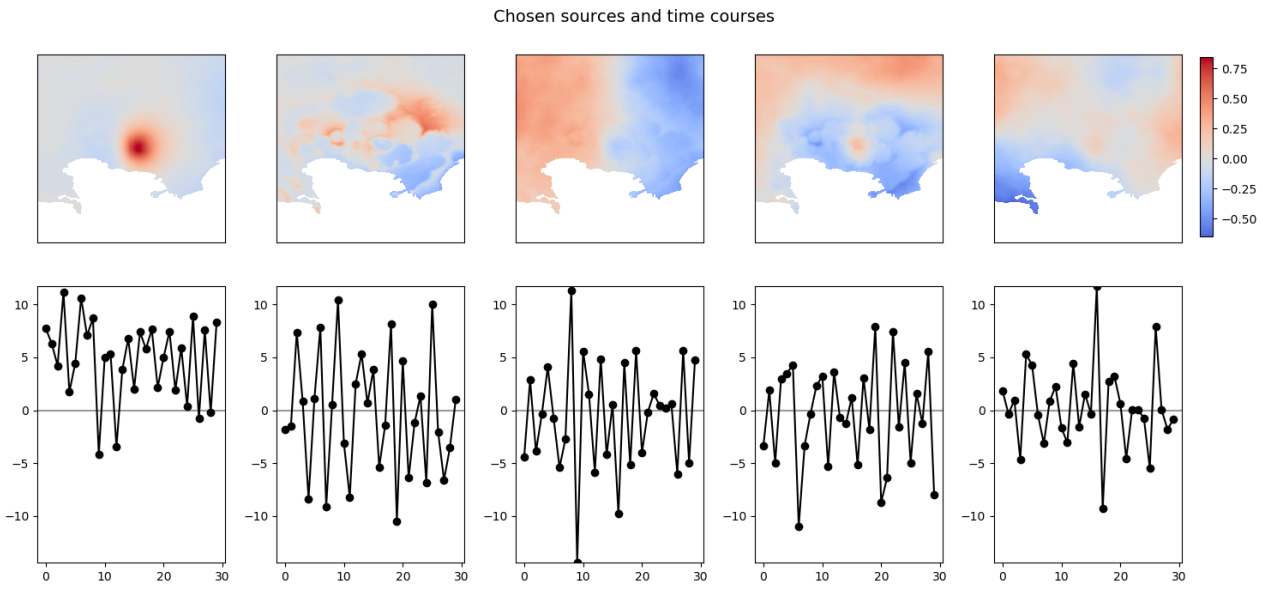

Figure 5: Chosen sources and time courses, ordered by the cluster quality index. The first three broadly correspond to the synthetic sources, whilst the fourth captures a small amount of the topographically correlated APS, and the fifth an aspect of the turbulent APS.

Gaddes, M., Hooper, A., Bagnardi, M. (2019). Using machine learning to automatically detect volcanic unrest in a time series of interferograms. *Journal of Geophysical Research: Solid Earth*.

Gaddes, M., Hooper, A., Bagnardi, M., Inman, H., Albino, F. (2018). Blind signal separation methods for InSAR: The potential to automatically detect and monitor signals of volcanic deformation. *Journal of Geophysical Research: Solid Earth*.

Himberg, J., Hyva, A., Esposito, F. (2004). Validating the independent components of neuroimaging time series via clustering and visualization. *NeuroImage*, **22**, pp. 1214–1222.

Maaten, L.V.D., Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, **9**, pp. 2579–2605.

Mcinnes, L., Healy, J., Astels, S. (2017). hdbscan : Hierarchical density based clustering. *The Journal of Open Source Software*, **2**, pp. 11–12.