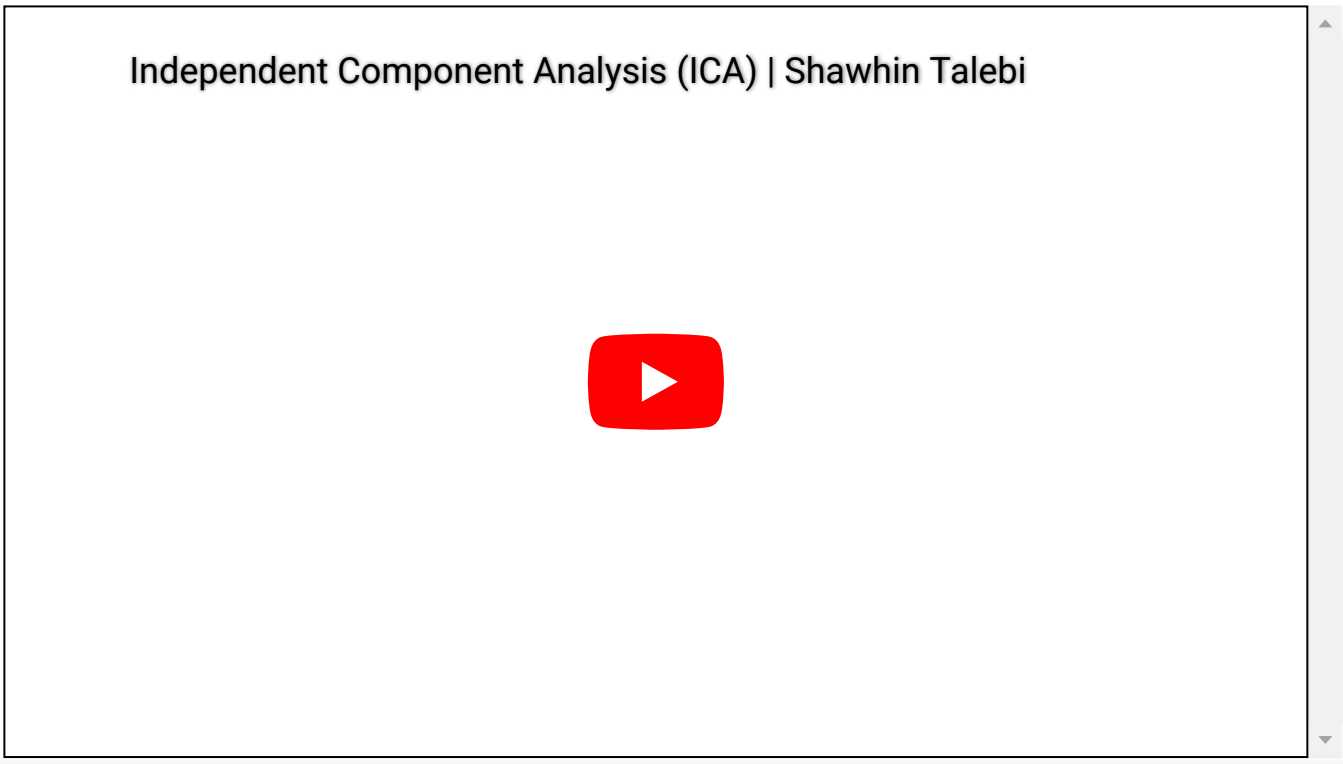
Shawhin Talebi Follow

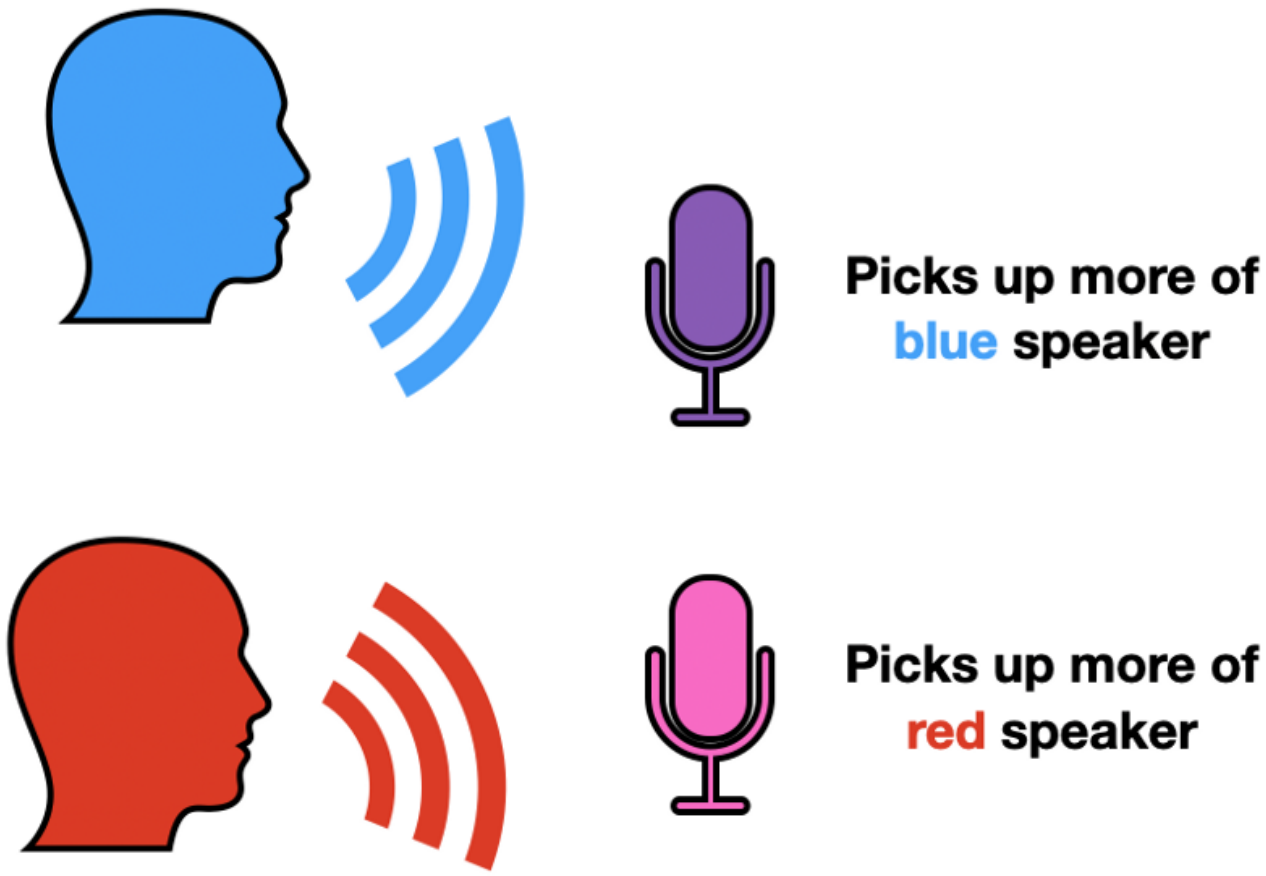
Mar 17, 2021 · 7 min read · ✨ · Listen

Independent Component Analysis (ICA)

Finding hidden factors in data

This is the final post in a two-part series on **Principal Component Analysis (PCA)** and **Independent Component Analysis (ICA)**. Although the techniques are similar, they are in fact different approaches and perform different tasks. In this post, I will provide a high level introduction to ICA, compare it to PCA, and give an example of using ICA to remove blink artifacts from EEG data.

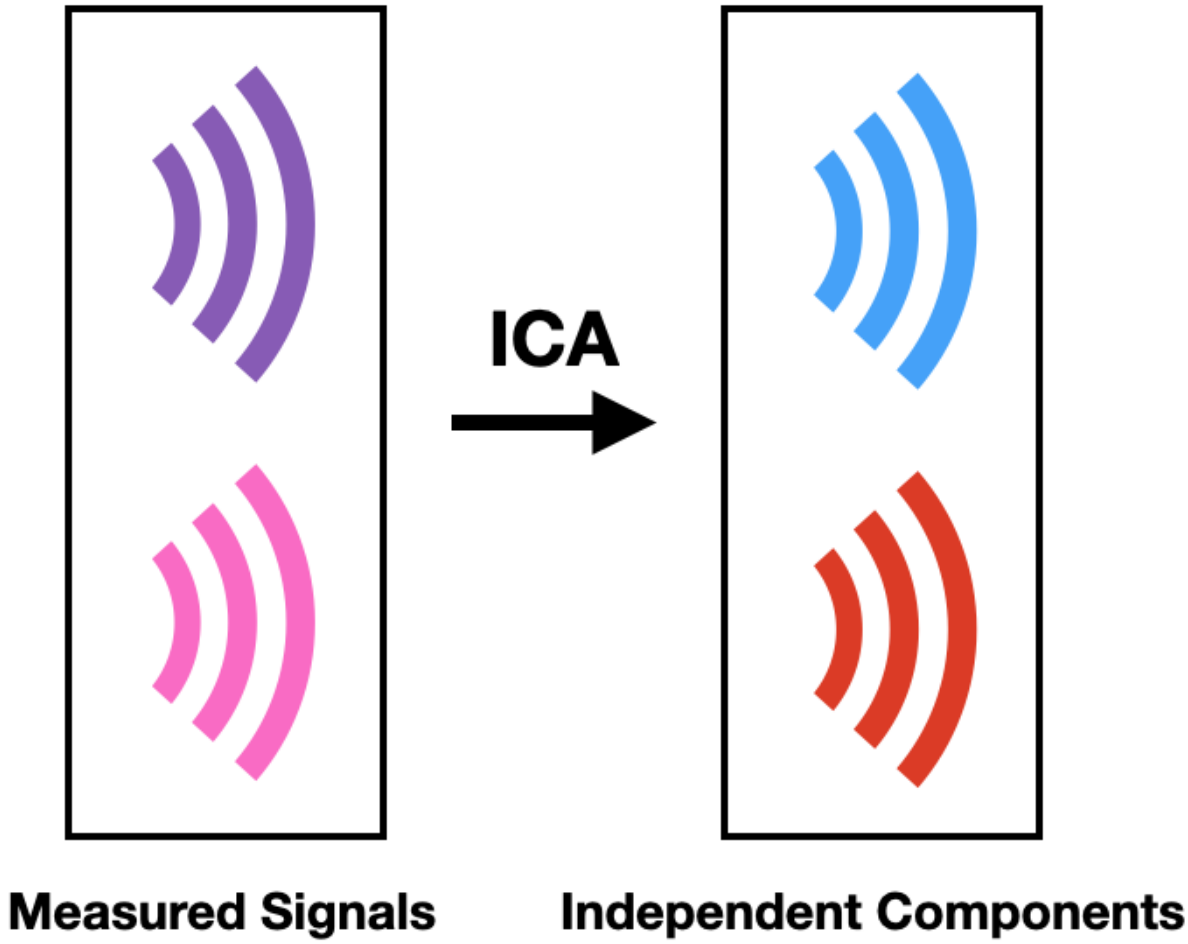




Simplest version of the “Cocktail Party Problem”. Image by author.

The standard problem used to describe ICA is the “Cocktail Party Problem”. In its simplest form, imagine two people having a conversation at a cocktail party (like the red and blue speakers above). For whatever reason, you have two microphones placed near both party-goers (like the purple and pink microphones above). Both voices are heard by *both* microphones at different volumes based on the distance between the person and microphone. In other words, we record two files that include audio from the two party-goers mixed together. The problem then is, *how can we separate the two voices in each file to obtain isolated recordings of each speaker?*

This problem is solved easily with **Independent Component Analysis (ICA)** which **transforms a set of vectors into a maximally independent set.** Returning to our “Cocktail Party Problem”, ICA will convert the two mixed audio recordings (represented by purple and pink waveforms below) into two unmixed recordings of each individual speaker (represented by blue and red waveforms below). Notice,



Converting mixed signals to independent components using ICA. Image by author.

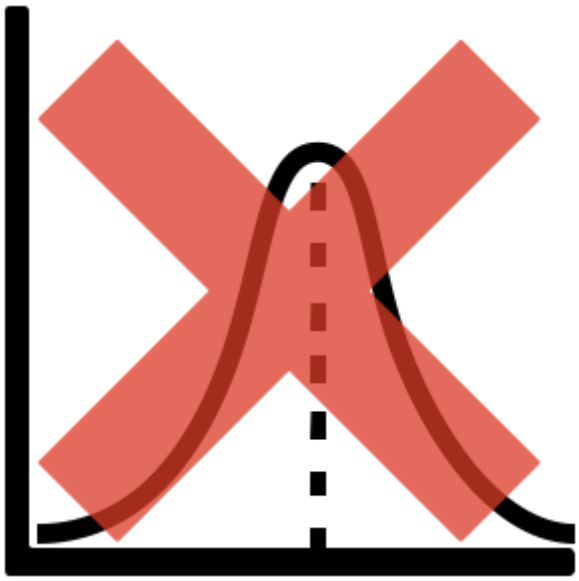
How it works

There are **two key assumptions** made in ICA. The hidden independent components we are trying to uncover must be one, **statistically independent**, and two, **non-Gaussian**. Semantically, by independent I mean information about **x** does not give you information about **y** and vice versa. Mathematically, this translates to,

$$p(x, y) = p(x)p(y)$$

Mathematical definition of statistical independence. Image by author.

Where, $p(x)$ represents the probability distribution of x. $p(x,y)$ represents the joint



Non-Gaussianity is a key assumption for ICA. Image by author.

The first assumption is the starting point of ICA. We want to disentangle information to derive a set of independent factors. If there are not multiple independent generators of information to uncover, there really isn't a need for ICA. For example, imagine using ICA for the "Cocktail Party Problem", but with only one party goer. What one could call the COVID birthday party problem. It wouldn't make much sense.

The need for the second assumption lies in the mathematics. ICA uses the idea of **non-Gaussianity** to uncover independent components. Non-Gaussianity **quantifies how *far* the distribution of a random variable is from being Gaussian.** Example measures of non-Gaussianity are kurtosis and negentropy. Why such a measure is helpful follows from the **Central Limit Theorem.** Specifically a result that states, the sum of two independent random variables has a distribution that is *closer* to Gaussian than either of the original variables. ICA combines this idea, non-Gaussianity measures, and the non-Gaussian assumption to uncover independent components hidden in data.

To illustrate this, consider a dataset with two variables **x_1** and **x_2**. These variables serve as basis that define a space i.e. we can use them to plot points in 2 dimensions. Suppose, we know the two independent components underlying the data, **s_1** and **s_2**. These two components serve as an alternative basis to describe

$$\mathbf{y} = w_1\mathbf{x}_1 + w_2\mathbf{x}_2$$

Linear combination of measured signals i.e. input variables. Image by author.

$$\mathbf{y} = a_1\mathbf{s}_1 + a_2\mathbf{s}_2$$

Linear combination of independent components. Image by author.

Going back to the Central Limit Theorem, the distribution of the sum of two random variables will be *more Gaussian* than either individual variable. Thus, when a_1 and a_2 are both non-zero, the distribution of **y** will be *more Gaussian* than either **s_1** or **s_2**. The reverse of that is, if either a_1 or a_2 is zero, then the distribution of **y** will be *less Gaussian* than the former case. And, if the non-Gaussian assumption of **s_1** and **s_2** holds, it will not be Gaussian at all since **y** will be exactly equal to one of the independent components!

In other words, the non-Gaussianity of **y** is maximized when it is directly proportional to one of the independent components. This allows us to frame ICA as an optimization problem. For example,

$$\max \quad kurt(w_1\mathbf{x}_1 + w_2\mathbf{x}_2)$$

Framing ICA as an optimization problem for a single independent component. Image by author.

Where we want to find the values of w_1 and w_2 that maximize the kurtosis of a linear combination of our known input variables. These optimal values of w_1 and w_2 will define an independent component.

$$\mathbf{s} = w_1^*\mathbf{x}_1 + w_2^*\mathbf{x}_2$$

Solutions to ICA optimization problem define independent components.

More generally, we can solve for the matrix of weights, **W**, which maximizes the

Framing ICA as an optimization problem for multiple independent components. Image by author.

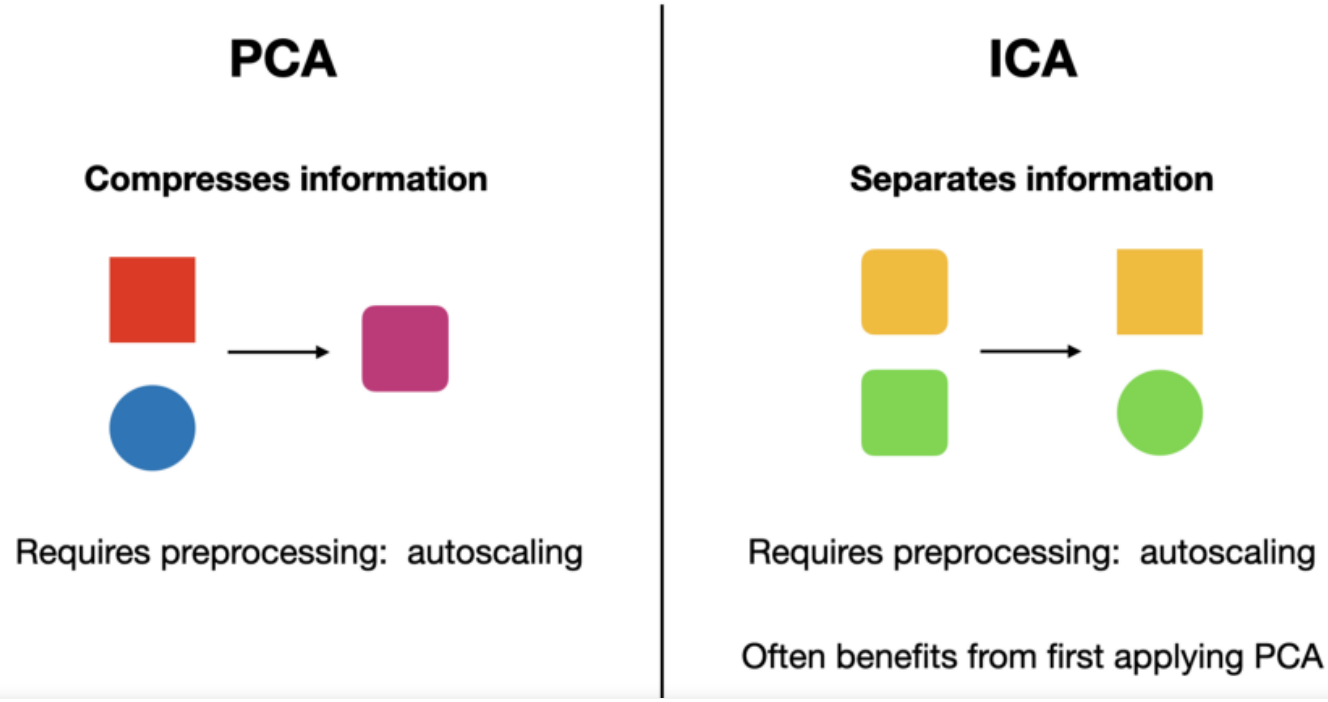
Key Points

I may have once again gone too far into the mathematical weeds. As a takeaway I will just highlight three key points of ICA:

- The number of inputs equals number of outputs
- Assumes independent components are statistically independent
- Assumes independent components are non-Gaussian

PCA vs ICA

Before moving on to an example I will briefly compare PCA and ICA. Although the two approaches may seem related, they perform different tasks. Specifically, **PCA** is often used to **compress information** i.e. dimensionality reduction. While **ICA** aims to **separate information** by transforming the input space into a maximally independent basis. A commonality is both approaches require input data to be **autoscaled** i.e. **subtract each column by its mean and divide by its standard deviation.** This is one reason why PCA is usually a good thing to do before performing ICA.

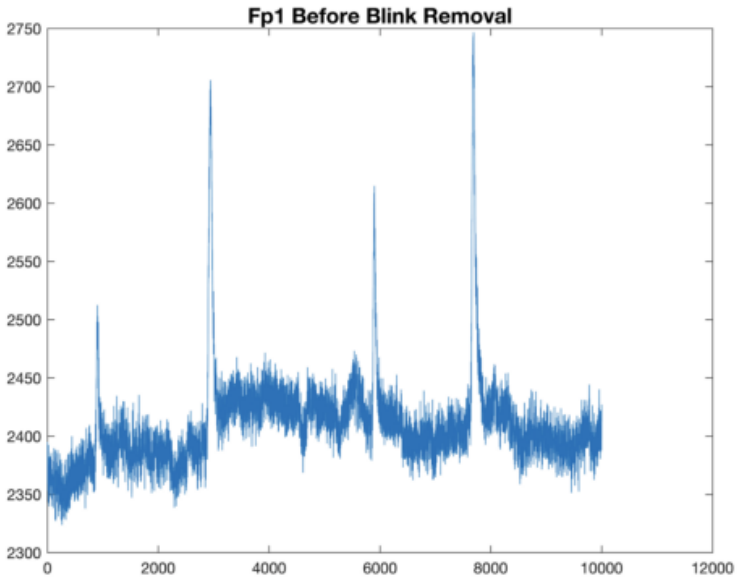


ICA to remove blink artifacts from EEG data, code is available in the **GitHub repository**.

Electroencephalography (EEG) is a technique that measures electrical activity resulting from the brain. A major disadvantage of EEG is its sensitivity to motion and other non-brain artifacts. One such artifact occurs whenever participants blink. In the below figure, blink artifacts can plainly be seen via spikes in the voltage vs time plot of the Fp1 electrode (near the front of the head).

Import Data

```
load('EEG_data')

% plot Fp1 electrode data (sits at front left part of head hear left eye)
% perturbations can be seen visually due to blinks
plot(Data(:,1))
title("Fp1 Before Blink Removal", 'FontSize', 14)
```



Importing data and plotting Fp1 voltage vs time. Image by author.

A good first step when using ICA, is first performing PCA on the dataset. Doing this in Matlab in easily done with the function *pca()*. I will note here it is critical to **autoscale** the data. This is done automatically in the *pca()* function. Also, here we start with 64 columns corresponding to 64 EEG electrode voltages measured over time. After PCA we are left with 21 columns corresponding to 21 score vectors i.e. principal components.

```
% compare their total explained variation
q=21;

% PERFORM PCA
% NOTE: it is important to normalize data! (i.e. substract mean of each
% column and divide by standard deviation. MATLAB's pca function does this
% automatically :p
[coeff,Data_PCA,latent,tsquared,explained,mu] = pca(Data, 'NumComponents', q);

% compute and display explained variation
disp(strcat("Top ", string(q), " principle components explain ", ...
    string(sum(explained(1:q))), " of variation"))
```

```
Top 21 principle components explain 99.5134 of variation
```

Code to apply PCA to dataset. Image by author.

Next, we can train an ICA model and apply it to the PCA score matrix.

ICA

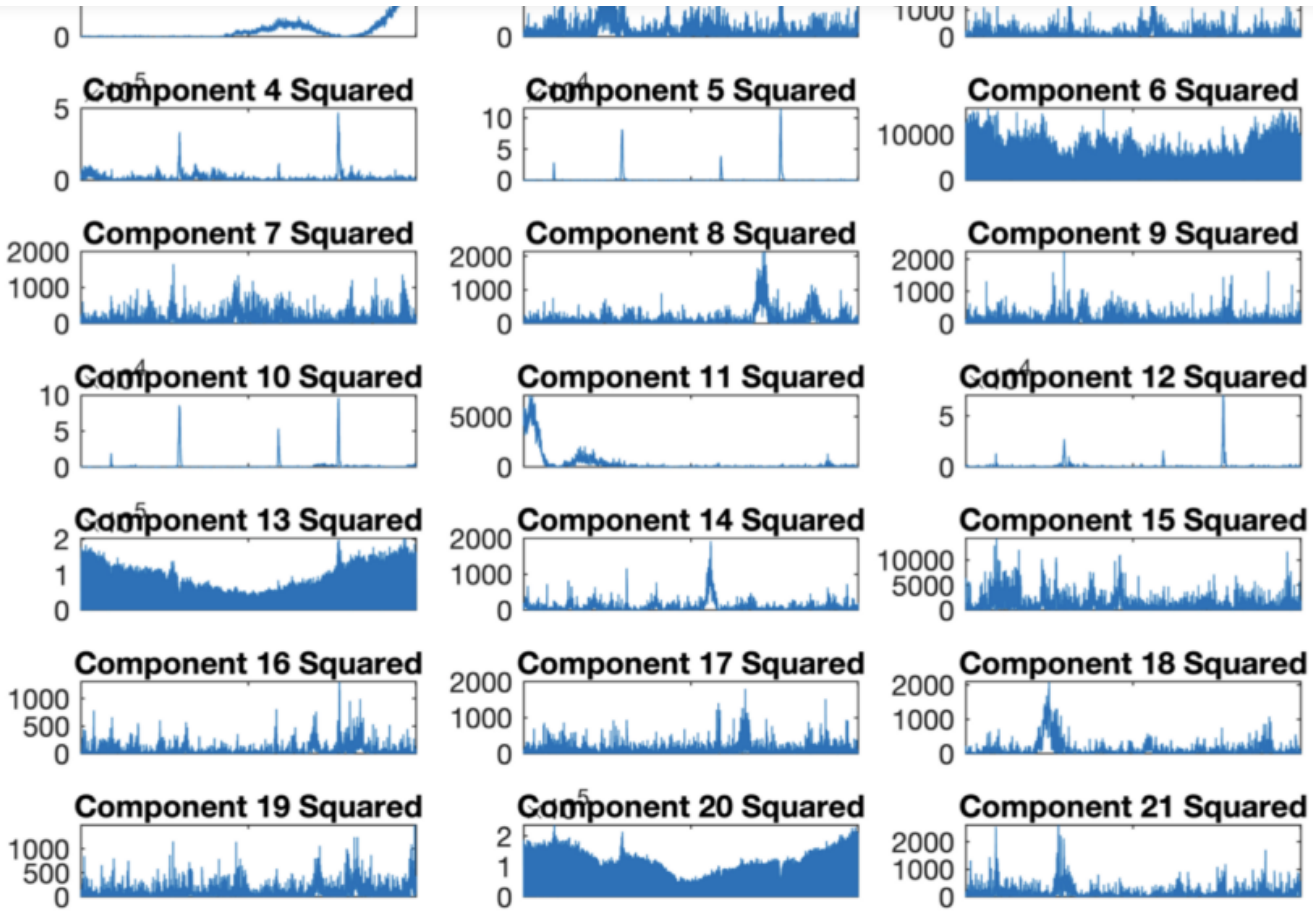
```
% compute independent components from principle components
% train ICA model
Mdl = rica(Data_PCA, q);

% apply ica
Data_ICA = transform(Mdl, Data_PCA);
```

Plot Components

```
% define number of plots per column of figure
plotsPerCol = 7;

% plot components
figure(2)
for i = 1:q

    subplot(plotsPerCol,ceil(q/plotsPerCol),i)
    plot(Data_ICA(:,i).^2)
    title(strcat("Component ", string(i), " Squared"))
    ax = gca;
    ax.XTickLabel = {};

end
```

Code to apply ICA to principal components. Image by author.

Plotting the independent components, we can inspect which ones correspond to blinking artifacts.



Plots of 21 independent components squared. Image by author.

I use a lazy heuristic to pick out independent components representing blink information. Namely, picking components who's square has 4 prominent peaks. The remaining components can be used reconstruct the original dataset without information from these blink components.

```
Components_blink = pickBlinkComponents(Data_ICA);
disp("Blink component(s):")
```

```
Blink component(s):
```

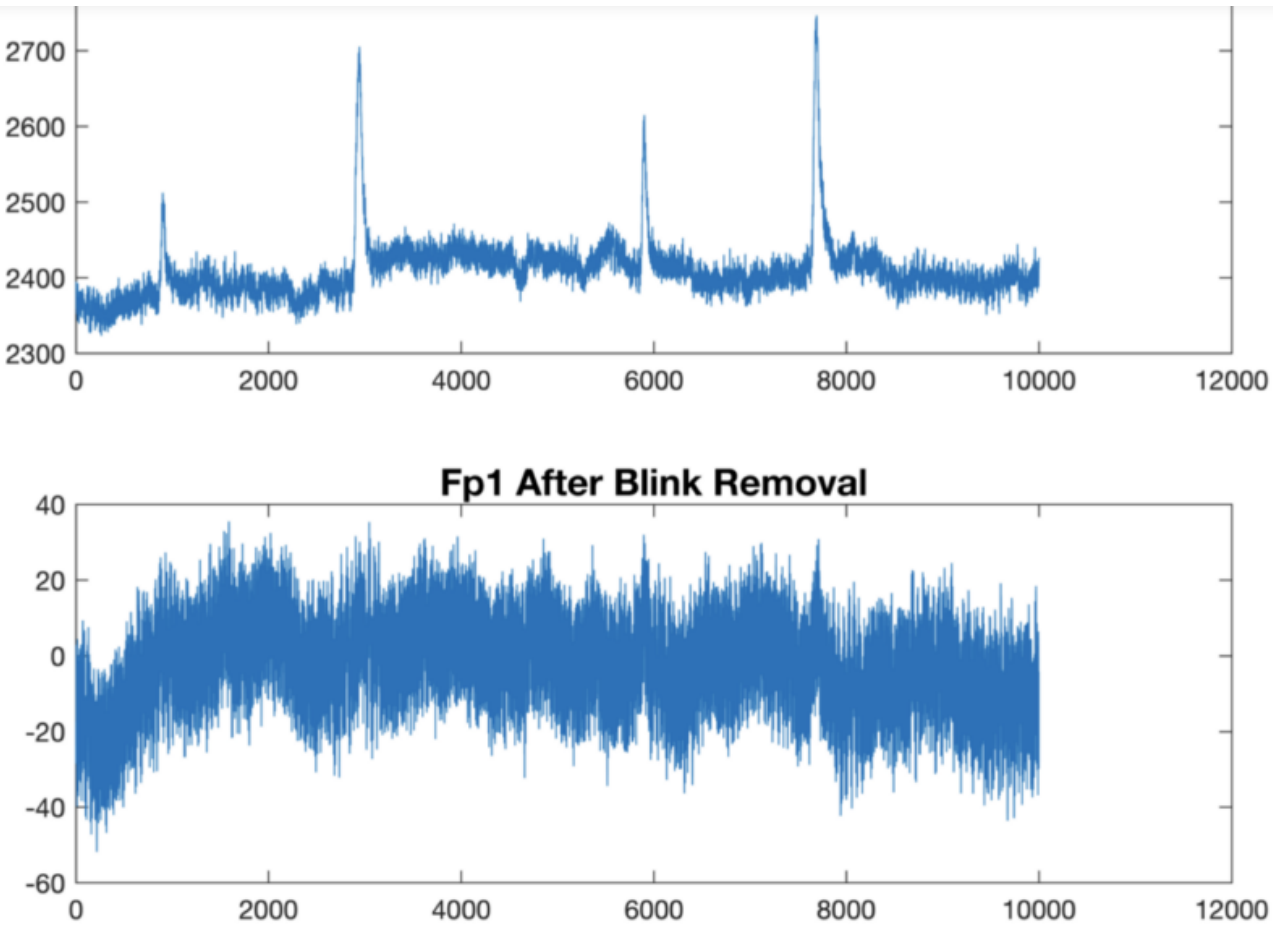
```
disp(Components_blink)
```

```
    3    20
```

```
% zero all columns corresponsing to blink components
Data_ICA_noBlinks = Data_ICA;
Data_ICA_noBlinks(:,Components_blink) = ...
    zeros(length(Data_ICA), length(Components_blink));

% perform inverse ica transform
Data_PCA_noBlinks = Data_ICA_noBlinks*Mdl.TransformWeights;

% perform inverse pca transform
Data_noBlinks = Data_PCA_noBlinks*coeff';

% plot Fp1 electrode before and after
figure(3)
subplot(2,1,1)
```

Code to pick out blink independent components and reconstruct EEG data. Image by author.

Finally, we plot the original and resulting voltage over time plot for the Fp1 electrode.



Fp1 signal before and after blink removal.

Conclusion

Independent Component Analysis (ICA) extracts hidden factors within data by transforming a set of variables to a new set that is maximally independent. ICA relies on a measure of non-Gaussianity to accomplish this task. **Principal Component Analysis (PCA)** and **ICA** aim at different goals, namely the former compresses information and the latter separates information. Despite their differences, it is often helpful to use PCA as preprocessing step for ICA. This combination of techniques has applications in areas such as: financial analysis and neuroscience.

Resources

More in this series: Principal Component Analysis | GitHub repo

Connect: My website | Book a call | Message me

[1] **Hyvärinen A, Oja E.** Independent component analysis: algorithms and applications. Neural Netw. 2000;13(4–5):411–430. doi:10.1016/s0893–6080(00)00026–5