GSLCCA Tutorial

Heather Turner, Phil Brain and Foteini Strimenopolou

For gslcca version 0.1-0, 2013-02-15
http://gslcca.r-forge.r-project.org/

1 Introduction

This tutorial demonstrates the use of the gslcca package for Generalised Semi-linear Canonical Correlation Analysis (GSLCCA) of data from EEG (Electroencephalogram) experiments.

Input in red relies on `.txt` files not included in the package, so is provided for illustration only. Input in blue can be run after the package is loaded.

2 Preparing Data for Analysis

The data required for a GSLCCA analysis are a matrix of mutlivariate responses, a time variable and possibly also treatment and subject variables. In the case of EEG experiments, the response matrix is the mean power at each time point for a number of frequencies. These power spectra are usually stored along with the other variables in a tabular format.

As an illustration we shall consider the `clonidine` data, which is included in the gslcca package, see `?clonidine`. This data was originally in a tab-delimited text file, with the first six rows and columns as follows

```
  Rat Treatment Time        0.98        1.95        2.93
1  35   Control    0 0.000073800 0.000182702 0.000153585
2  35   Control  300 0.000081100 0.000210315 0.000187891
3  35   Control  600 0.000105919 0.000269269 0.000232081
4  35   Control  900 0.000094700 0.000228982 0.000207461
5  35   Control 1200 0.000091400 0.000236652 0.000228351
6  35   Control 1500 0.000088800 0.000249708 0.000269962
```

Such data can be prepared for analysis with the gslcca package using the `readSpectra` function:

```
> clonidine <- readSpectra("Clonidine Light.txt", info = 1:3,
>     treatment = c("Control", "Low Dose", "Middle Dose", "High Dose"))
```

here the first argument is the name of the tab-delimited file, `info` gives the indices for the "information" columns, i.e. `Rat`, `Treatment` and `Time`, and `treatment` gives the levels of the treatment factor, in order of increasing dose (over-riding the default alphabetical order). The values shown are the defaults for the `info` and `treatment` arguments so can be omitted.

The function `readSpectra` has three additional arguments that allow the user to subset or aggregate the spectra that are read in. The argument `resolution` specifies the desired resolution of frequencies in Hz – data read in at a higher resolution is aggregated to obtain the

desired resolution. For example, the default setting is 1 Hz frequencies, so data read in at 0.5 Hz frequencies would be aggregated by summing over each pair of frequencies.

The arguments `end` and `nfreq` specify the end time and the number of frequencies to include in the result, respectively. The defaults for these arguments are 42900 seconds (the last 5 minute bin in a 12 hour period) and 36 frequencies (so up to 36 Hz when `resolution` is 1).

The object returned by `readSpectra` is a data frame containing the information columns with their orginal names and the matrix of spectra named `spectra`. This data frame can be passed to the data argument of the `gslcca` function (described later in Section 3).

For the `clonidine` data set, the data were then sub-sampled to keep every other time point, however this was done purely to limit the size of the data set and would not normally be done in practice.

2.1 Banded Spectra

The `bandSpectra` function can be used to aggregate the multivariate response into bands, such as the frequency bands typically used in the analysis of EEG data. For example, the `clonidine` data could be grouped into the delta (0–4 Hz), theta (4–8 Hz), alpha (8–13 Hz), beta (13–30 Hz) and gamma (>=30Hz) bands as follows:

```
> library(gslcca)
> data(clonidine)
> banded <- bandSpectra(clonidine$spectra, breaks = c(4, 8, 13, 30),
+     labels = c("delta", "theta", "alpha", "beta", "gamma"))
> head(banded)
```

```
           delta       theta       alpha        beta     gamma
[1,] 0.000521588 0.000645512 0.000294778 0.000201440 5.894e-05
[2,] 0.000791357 0.000570954 0.000284136 0.000238900 5.963e-05
[3,] 0.000736195 0.000579941 0.000254093 0.000218480 5.377e-05
[4,] 0.000751376 0.000556839 0.000235300 0.000255800 5.263e-05
[5,] 0.001749510 0.001354050 0.000959060 0.000658797 4.676e-05
[6,] 0.000626571 0.000706522 0.000266890 0.000260420 5.230e-05
```

3 GSLCCA Analysis

The aim of GSLCCA is to find a linear combination of the multivariate response that is maximally correlated with a nonlinear model. Thus we have

$$\boldsymbol{Y}\boldsymbol{a} = \rho(\boldsymbol{X}(\boldsymbol{t}, \boldsymbol{\theta})\boldsymbol{b})$$

where $\boldsymbol{Y}$ is a data matrix with rows of observations recorded at times $\boldsymbol{t}$, $\boldsymbol{a}$ is a vector of loadings, $\boldsymbol{X}(\boldsymbol{t}, \boldsymbol{\theta})$ is a matrix with columns defined by a nonlinear function with unknown parameters, $\boldsymbol{b}$ is a vector of coefficients and ρ is the correlation to be maximised.

The `gslcca` package provides the `gslcca` function to conduct GSLCCA analysis. The two key outputs from the analysis are the vector of loadings $\boldsymbol{a}$, which we describe as the *signature* and the fitted model, given by the x scores $\boldsymbol{X}(\boldsymbol{t}, \boldsymbol{\theta})\boldsymbol{b}$.

3.1 Performing GSLCCA

To demonstrate the use of `gslcca`, we shall consider a particular application of GSLCCA to the clonidine data. Section 5 gives details on how to customise the analysis, for example specifying an alternative model.

A typical call to `gslcca` may be as follows:

```
> result <- gslcca(spectra,  "Double Exponential", time=Time,
+                  subject = Rat, global = FALSE,
+                  treatment = Treatment, ref = 1, separate = FALSE,
+                  subject.smooth = TRUE, pct.explained = 0.96,
+                            data = clonidine)
```

In this example the matrix of power spectra has been specified as the multivariate response and the model is specified as `"Double Exponential"` which refers to the two compartment model:

$$\exp(-k_1 t) - \exp(-k_2 t).$$

This is equivalent to specifying the model via the symbolic formula

```
~ exp(-K1 * time) + exp(-K2 * time)
```

where `time` is the variable specified via the `time` argument and `K1` and `K2` are parameters. Specifying the model by name means that starting values for `K1` and `K2` are set by default.

The `subject` argument has been specified as `Rat` , so a separate signature will be estimated for each rat and since `global = FALSE`, the nonlinear model will also be separately estimated for each rat.

The `treatment` argument has been specified as `Treatment`, so a separate linear coefficient will be estimated for each non-reference level of the treatment factor. Since `separate = FALSE`, the nonlinear parameters will be the same. The model is fixed at zero for the reference treatment level, which is specified by `ref`.

The `subject.smooth` argument is set to `TRUE`, so the power spectra are smoothed for each rat separately prior to analysis. The `pct.explained` argument specifies that the smoothed matrix must capture at least 96% of the variance in the original data, for each rat.

Finally the `data` argument has been specified, so that the objects `spectra`, `Time`, `Rat` and `Treatment` can be found.

3.2 Examining the Result

A summary of the GSLCCA analysis can be obtained by using the `summary` function:

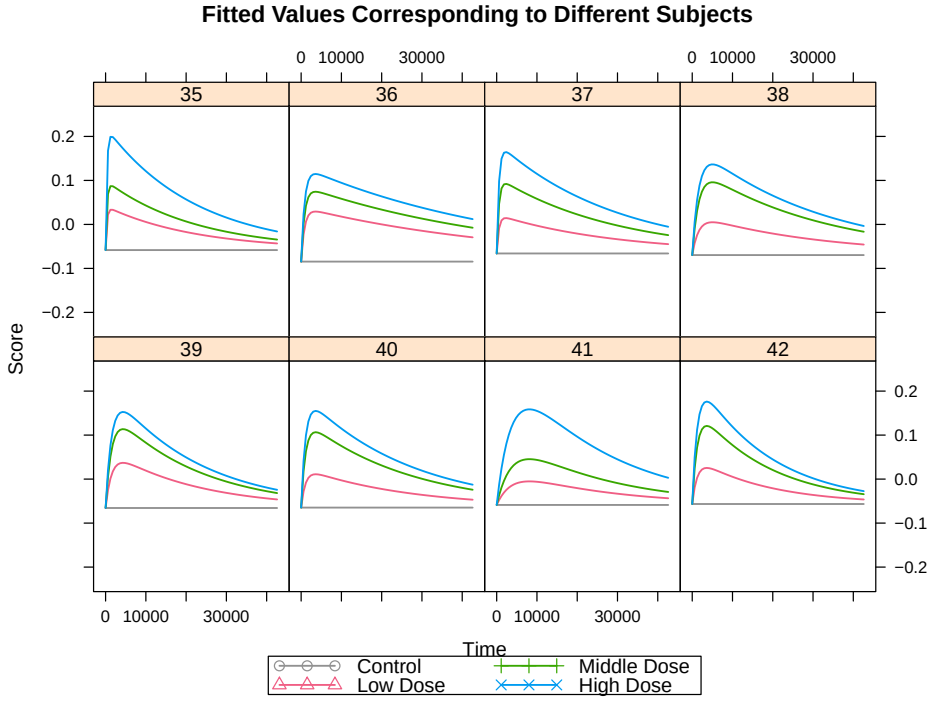
```
> summary(result)


Call:
gslcca(Y = spectra, formula = "Double Exponential", time = Time,
    subject = Rat, global = FALSE, treatment = Treatment, ref = 1,
    separate = FALSE, data = clonidine, subject.smooth = TRUE,
    pct.explained = 0.96)

Data smoothed at subject level using 5 roots

Percent variance explained by SVD approximation:
```

```
subject 35 subject 36 subject 37 subject 38 subject 39 subject 40 subject 41
    0.9929     0.9730     0.9936     0.9894     0.9828     0.9814     0.9805
subject 42
    0.9786

Nonlinear parameters:
   subject 35 subject 36 subject 37 subject 38 subject 39 subject 40 subject 41
K1 10.024     10.874     10.304     10.361     10.001     10.179     10.097
K2  5.810      6.737      6.297      7.374      7.312      6.983      8.198
   subject 42
K1  9.808
K2  7.131

Correlation at final iteration:
subject 35 subject 36 subject 37 subject 38 subject 39 subject 40 subject 41
    0.8900     0.8239     0.8876     0.8653     0.8528     0.8511     0.8006
subject 42
    0.9067
```

This summary gives some details of the smoothing performed and the model fitted for each rat. In this case the models are reasonably consistent across the rats, with the rate parameters estimated as around 10 and 7 (a model with K1 = 10 and K2 = 7 is equivalent to a model with K1 = 7 and K2 = 10 if the sign of the linear coefficient is reversed). The models fit well, giving a correlation of at least 80% for all rats – we would hope to see at least a correlation of 60%. The algorithm has converged for each rat, so an optimal model has been found for each rat, although as the model is nonlinear it is possible that this is a local optimum and a better solution could be found with different starting values.

After checking the summary, we can view the fitted model using the `plot` function:

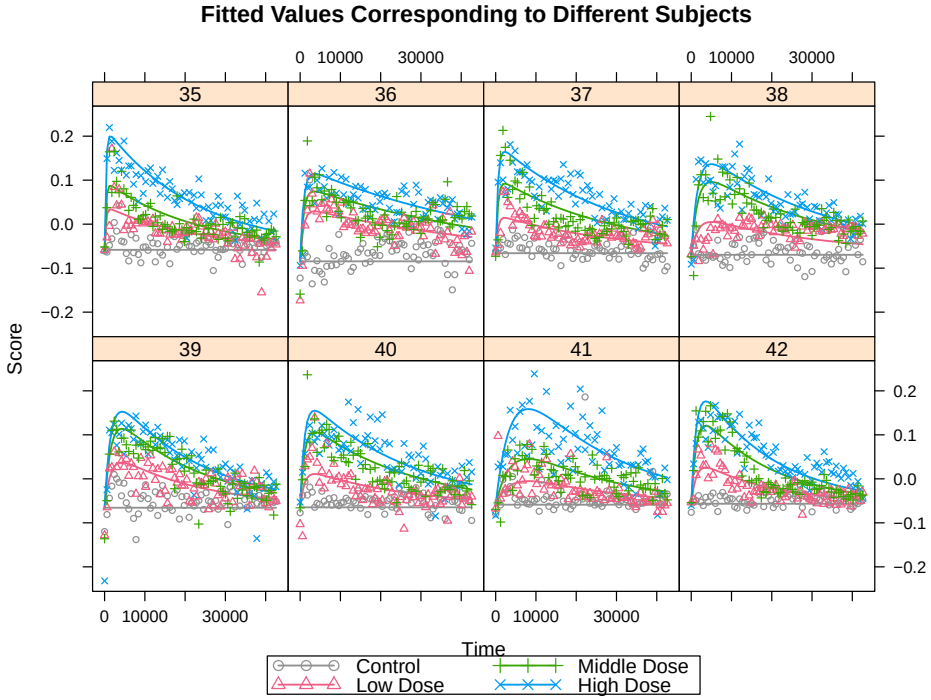
```
> plot(result, "fitted", lattice = TRUE)
```



Here the `lattice` option is set to `TRUE` to display all the models together. Again we see the models are broadly similar across rats. For each model, the amplitude varies by treatment, but the rate parameters are constant, so the curves for each treatment are proportional. The baseline for the control treatment is a non-zero constant because the $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices have been centred prior to Canonical Correlation Analysis (CCA). This adjustment is controlled by the `partial` argument.

In order to check the fit of the model, we can plot both the x scores (the fitted model) and the y scores (the projected observed values) as follows:

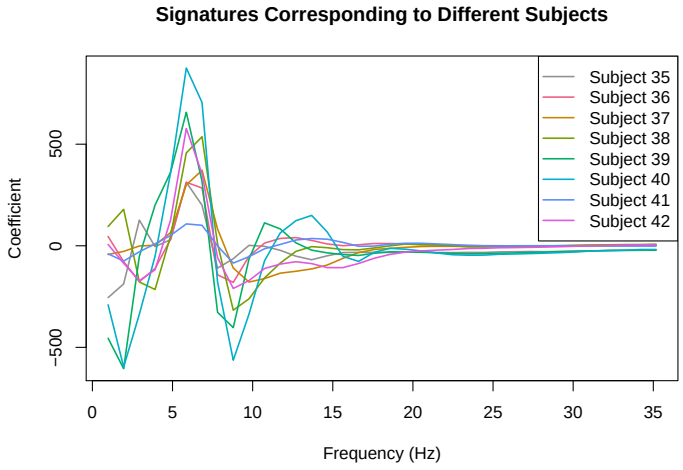
```
> plot(result, "scores", lattice = TRUE)
```



As indicated by the correlations in the summary, the sets of scores agree quite well.

Finally we can look at the model signatures, using the option `overlay = TRUE` to display them in a single plot:

```
> plot(result, "signatures", overlay = TRUE)
```



We can see that the scale varies across rats, but the shape of the signature is quite consistent, with a positive peak around 6-7 Hz and negative peaks around 2 and 9 Hz.

3.3 Saving Signatures for Further Analysis

The object returned by the `gslcca` function is a list containing several components useful for further analysis, see `?gslcca`. One of these components is `"ycoef"`, which is a data frame of the $\boldsymbol{Y}$ coefficients, i.e. the signatures, for each rat. These can be extracted directly, using `result$ycoef`or using the utility function `signatures`. For example, we can save the signatures in a tab-delimited text file as follows:

```
> write.table(signatures(result), "clonidine signatures.txt",
+             sep = "\t")
```

4 Comparing Signatures Across Experiments (Snapshot Analysis)

In the analysis of data from EEG experiments, it is hypothesised that the signature represents the relative importance of each frequency in the power spectrum in relation to the pharmacodynamic response in the brain. The signature is thus related to the drug mechanism and it is of interest to compare signatures obtained from EEG experiments using related drugs.

A simple way to do this is to use a *snapshot analysis*, that is to compare the mean signature at each frequency using a t-test. The gslcca package provides the `snapshot` function for this purpose.

As an illustration, suppose we have saved the signatures from the GSLCCA analysis of two compounds, in the files, "Compound1 signatures.txt" and "Compound2 signatures.txt". These are read in as follows:

```
> sig1 <- read.delim("Compound1 signatures.txt",
+                    check.names = FALSE)
> sig2 <- read.delim("Compound2 signatures.txt",
+                    check.names = FALSE)
```

using `check.names = FALSE` to preserve the numeric column names. We can then call the `snapshot` function to perform a snapshot analysis:

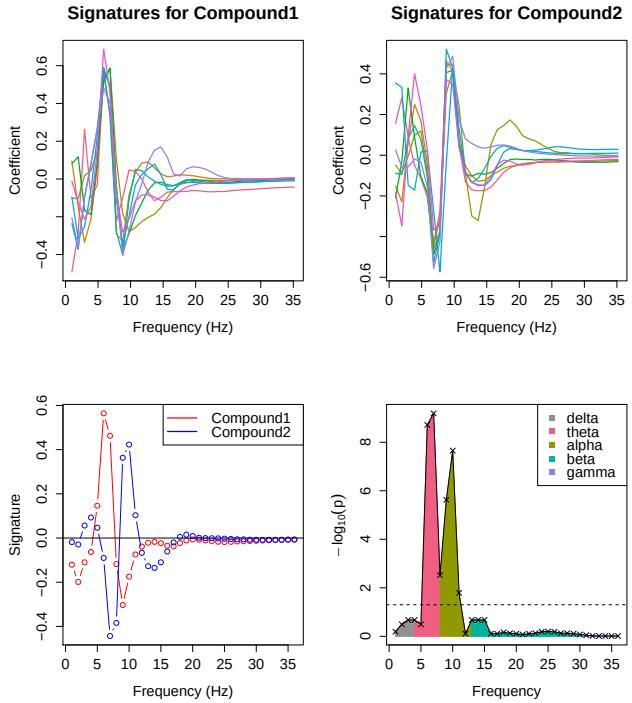
```
> result <- snapshot(sig1, sig2, normalise = TRUE)
```

Setting `nomalise = TRUE` means that the signatures will be normalised so that the sum of squared values in each signature sums to one. This normalisation removes the differences in scale between rats and makes the signatures comparable across experiments.

The `snapshot` function returns the set of (normalised) signatures for each compound, the two mean signatures and p-values from the t-tests comparing the mean signatures at each frequency. The p-values are corrected for multiple testing using Benjamini and Hochberg's false discovery rate, by default, alternative options can be specified using the `p.adjust.method` argument.

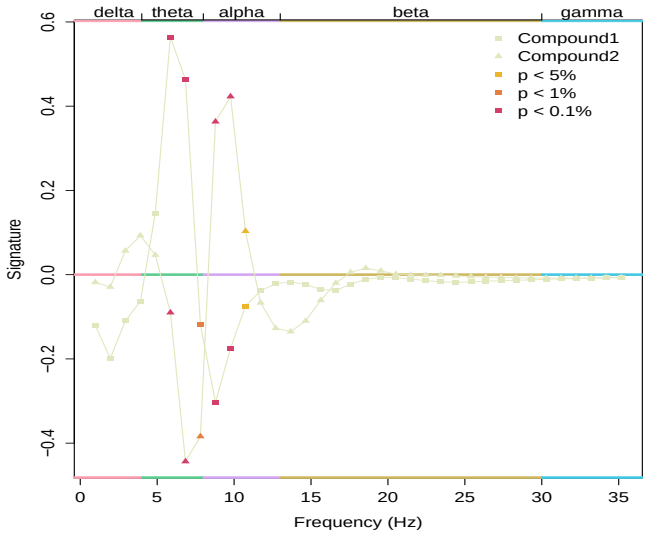
The main components of the `result` object can be displayed using `plot` with the argument `type = c("signatures", "means", "pvalue")`; the `names` argument provides names for the two compounds, which are used in the default titles/legends of the plots produced.

```
> par(mfrow = c(2,2))
> plot(result, type = c("signatures", "means", "pvalue"),
>      names = c(comp1, comp2))
```



A more compact display of the snapshot analysis can be obtained using the option `type = "compact"`:

```
> par(mfrow = c(1,1))
> plot(result, type = "compact", names = c(comp1, comp2))
```



This produces a plot of the mean signatures for each compound, color-coded by the significance level of the t-test at each frequency. As before, this shows significant differences between

the mean signatures over the conventional theta and alpha frequency bands.

5 Appendix

5.1 Model Specification

5.1.1 Using a "standard" model

The `formula` argument to the `gslcca` function specifies the "base" model, which defines the columns of $X(t, \theta)$. The gslcca package provide shortcuts for two models useful for describing the type of pharmacodynamic response that may be expected in the brain following an oral dose of a drug. Specifying the model by name means that starting values for the parameters are set automatically.

The first option is `formula = "Double Exponential"` which specifies the two compartment model:

$$\alpha(\exp(-k_1 t) + \exp(-k_2 t))$$

This is equivalent to specifying

```
formula = ~ exp(-K1 * time) - exp(-K2 * time)
```

where `time` is the variable specified via the `time` argument and `K1` and `K2` are parameters. In this case the parameters are initialised at `K1 = 9` and `K2 = 8.5`, which correspond to a time to maximum of approximately 1 hours when `time` is recorded in seconds. Alternative starting values can be specified by passing a named list to the `start` argument, e.g.

```
start = list(K1 = 8, K2 = 7.5)
```

The second model shortcut is `formula = "Critical Exponential"` which specifies the limiting case of the Double Exponential model when the rate parameters k_1 and k_2 are essentially the same:

$$\alpha t \exp(-k_1 t)$$

This corresponds to setting `formula = ~ time * exp(-K1 * time)` and `start = list(K1 = 8.5)`.