Supplementary material for "Log-Free Divergence and Covariance matrix for Compositional Data I: The Affine/Barycentric Approach"

Olivier P. Faugeras

Toulouse Schools of Economics, University Toulouse Capitole, 1, Esplanade de l'université, 31080, Toulouse cedex 06, France.

Corresponding author(s). E-mail(s): olivier.faugeras@tse-eu.fr;

This document contains supplementary material for the article titled "Log-Free Divergence and Covariance matrix for Compositional Data I: The Affine/Barycentric Approach." It provides additional simulations in Section 1, and code snippets together with explanations regarding numerical computations in Section 2.

1 Supplementary simulations

1.1 Anisotropic Generalised Barycentric Gaussian distributions with $\alpha = 1, \infty$.

For illustration and comparison purposes, we present in Figures 1 and 2 a sample of density plots of the weighted barycentric Gaussian distributions (see Definition 6), based on the W-weighted barycentric α-divergence, for $\alpha = 1, \infty$ and varying shape W and location $[\mathbf{m}]_+$ parameters.

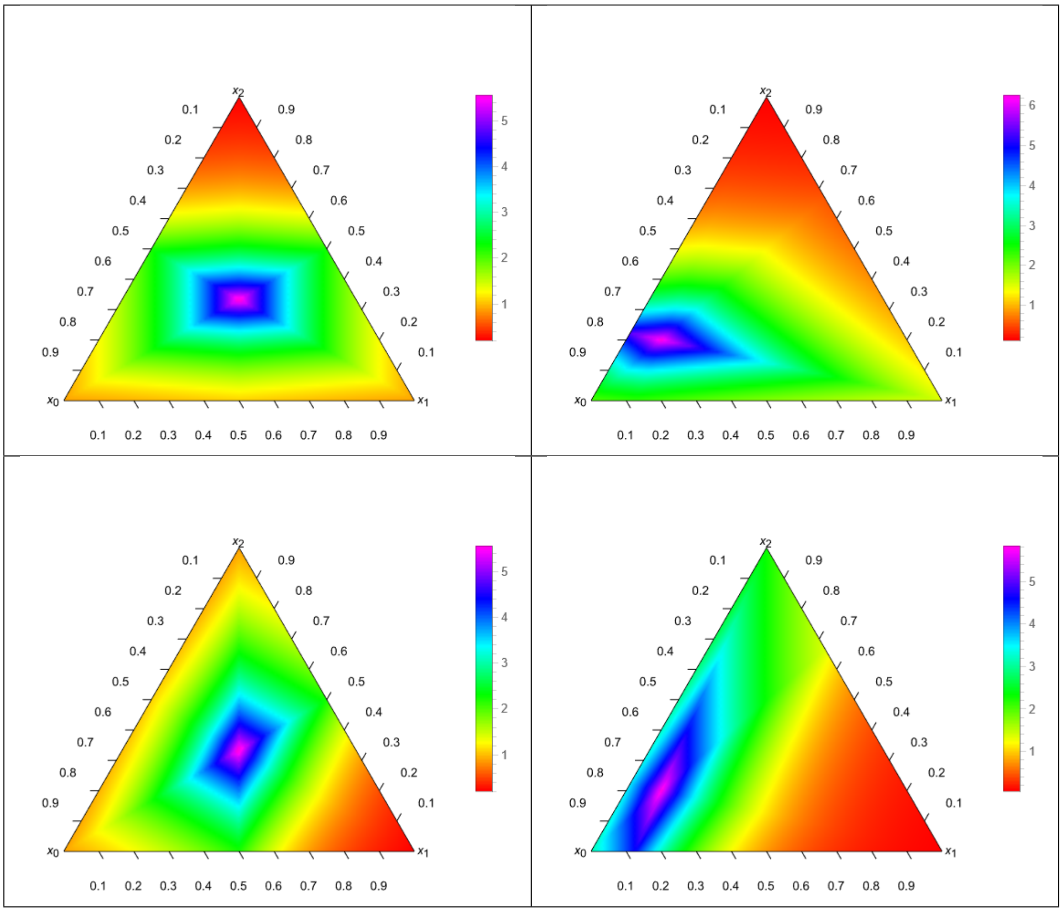

Fig. 1 Generalised Weighted Barycentric Gaussian distributions with $\alpha = 1$-divergence. Left column: centered distribution with $[\mathbf{m}]_+ = [1 : 1 : 1]_+$. Right column: a non-centered distribution with $\mathbf{m} = (0.7, 0.1, 0.2)$. $(w_{01}, w_{02}, w_{12}) = (0.8, 0.1, 0.1)$ (up), $(w_{01}, w_{02}, w_{12}) = (0.1, 0.8, 0.1)$ (down). $\sigma = 2$.

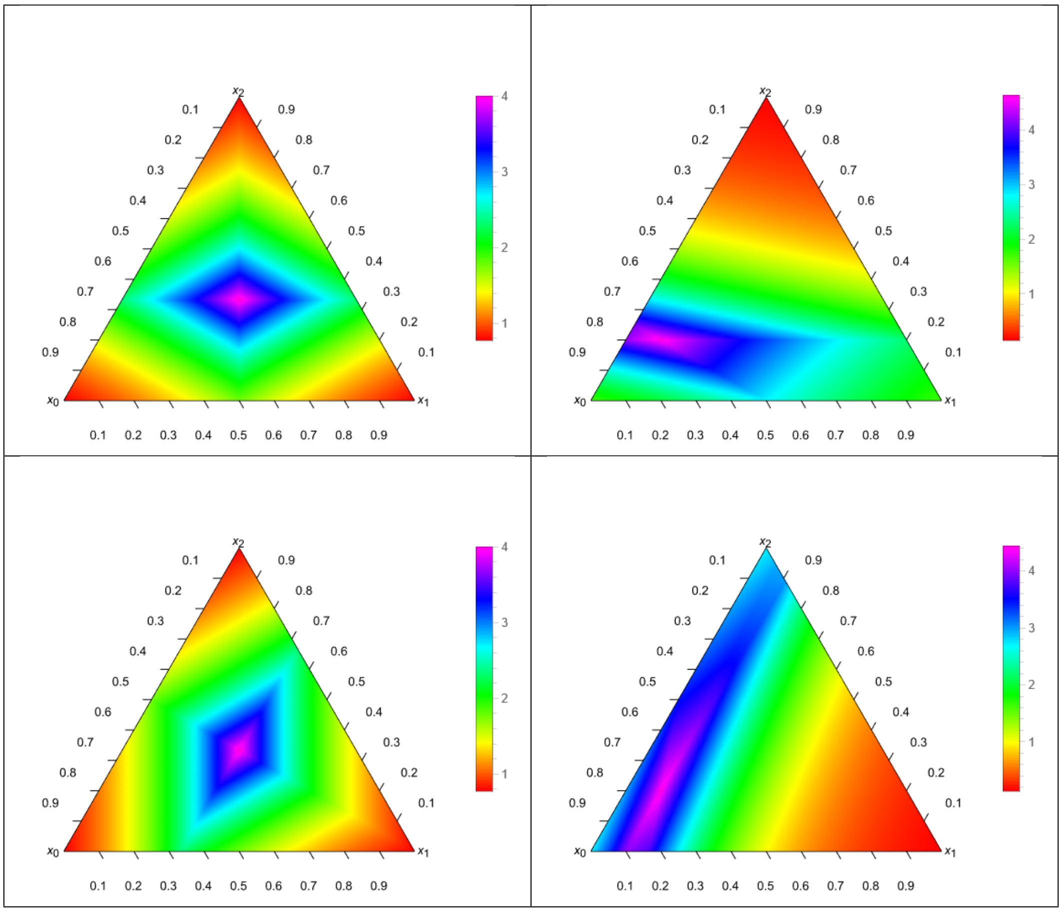

Fig. 2 Generalised Weighted Barycentric Gaussian distributions with $\alpha = \infty$-divergence. Left column: centered distribution with $[\mathbf{m}]_+ = [1 : 1 : 1]_+$. Right column: a non-centered distribution with $\mathbf{m} = (0.7, 0.1, 0.2)$. $(w_{01}, w_{02}, w_{12}) = (0.8, 0.1, 0.1)$ (up), $(w_{01}, w_{02}, w_{12}) = (0.1, 0.8, 0.1)$ (down). $\sigma = 2$.

1.2 Anisotropic Generalised Hilbert-Gaussian distributions

For illustration purposes, we present in Figure 3 density plots of the weighted Hilbert Gaussian distributions, based on the (square of) the $W-$weighted Hilbert Projective metric, defined in Remark 3.

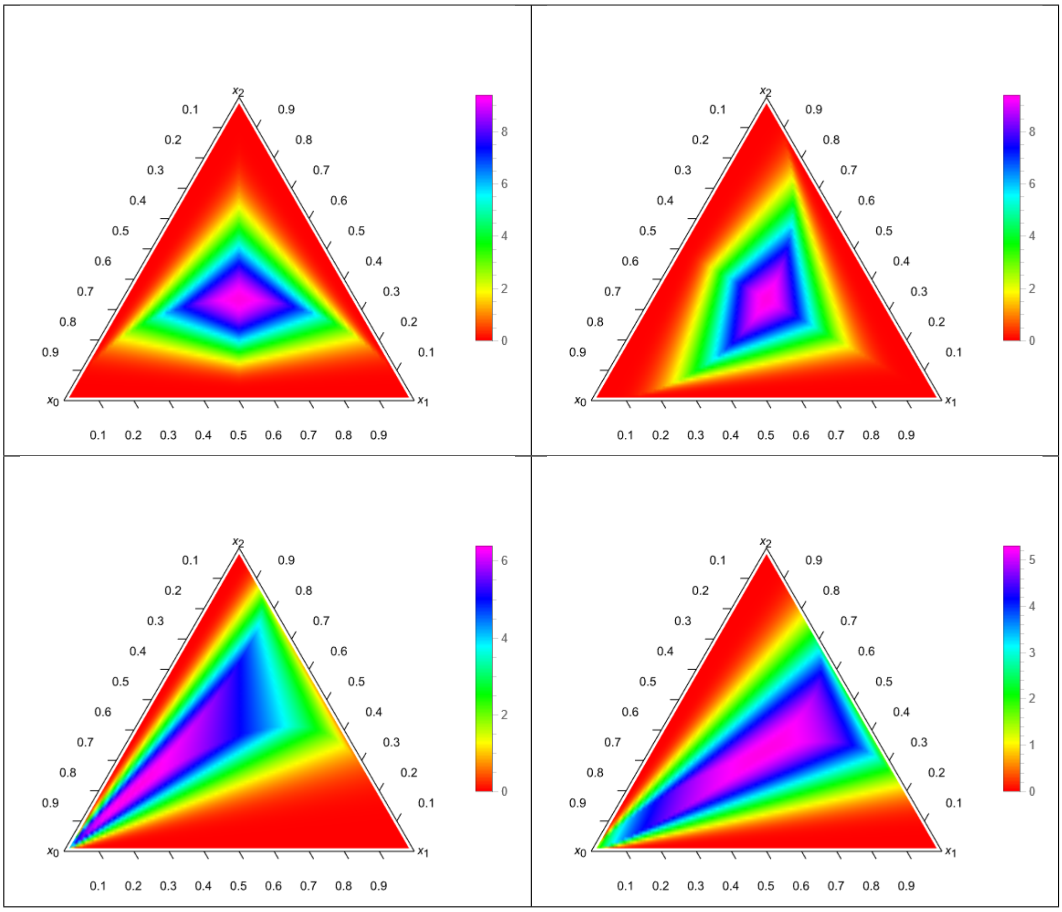

Fig. 3 Generalised Weighted Hilbert-Gaussian distributions based on the W-weighted Hilbert projective metric. $[\mathbf{m}]_+ = [1:1:1]_+$, $(w_{01}, w_{02}, w_{12}) = (0.8, 0.1, 0.1)$ (upper left), $[\mathbf{m}]_+ = [1:1:1]_+$, $(w_{01}, w_{02}, w_{12}) = (0.1, 0.8, 0.1)$ (upper right), $\mathbf{m} = (0.7, 0.1, 0.2)$, $(w_{01}, w_{02}, w_{12}) = (0.4, 0.5, 0.1)$ (lower left), $[\mathbf{m}]_+ = [1:1:1]_+$, $(w_{01}, w_{02}, w_{12}) = (0.4, 0.5, 0.1)$ (lower right). $\sigma = 100$. $\alpha = 2$.

2 Code snippets

Most of the simulations of the article were conducted in *Mathematica*(Wolfram Research (2023)), due to its ease at programming mathematical formulas. Following a suggestion of the referee, we provide below some discussion on the implementation of the main functions, together with the corresponding code fragments in *Mathematica*, and/or in *R* (R Core Team (2023)) language. These should suffice for readers looking to translate the provided code snippets into *Python* or their preferred language.

2.1 Computation of the α-barycentric divergence

The double sum $\sum_{i<j}$ in the formula (9) of the barycentric divergence $d_\alpha([\mathbf{x}]_+, [\mathbf{y}_+)$ (Definition 1) can easily (but inefficiently) be computed using two "for .. next" (nested) loops (omitted). A more efficient version, given in Listing 1 below for the R language, can be obtained by vectorization.

```
## compute alpha barycentric divergence
####### compute determinantal part using vectorized sum
compute_intermediatesum_vectorized <- function(x, y, alpha) {
  indices <- combn(length(x), 2)  # Toutes les paires (i, j) avec i < j
  terms <- abs(x[indices[1, ]] * y[indices[2, ]] - x[indices[2, ]] * y[indices[1, ]])^alpha
  sum(terms)
}
## compute alpha barycentric divergence
alphabarycentricdivergence <- function(x, y, alpha) {
  (compute_intermediatesum_vectorized(x,y,alpha)^(1/alpha))/(sum(x)* sum(y))
}
```

Listing 1 $\alpha-$Barycentric divergence using vectorized sums, in R Language

Eventually, a third, more elegant, version can be obtained by realizing that the elements in the double sums are the 2×2 minors of the matrix $[\mathbf{x}\,\mathbf{y}]$ made by binding the $\mathbf{x}$ and $\mathbf{y}$ column vectors. This gives, e.g. in a high-level language as *Mathematica*, the one line code, given in Listing 2.

```
divergence[a_, b_, alpha_] :=
  Norm[Flatten[Minors[{a, b}, 2]], alpha]/(Norm[a, 1] Norm[b, 1])
```

Listing 2 $\alpha-$Barycentric divergence, via minors, in *Mathematica* Language

These minors corresponds to the components of the exterior product $\mathbf{x} \wedge \mathbf{y}$, which itself can be described as the anti-symmetrization $\mathbf{x} \otimes \mathbf{y} - \mathbf{y} \otimes \mathbf{x}$ of the tensor product $\mathbf{x} \otimes \mathbf{y} = \mathbf{x}^T \mathbf{y}$, see Remark 2 and the details given in the follow-up/companion paper Faugeras (2024). This gives a fourth way to compute the barycentric divergence using the outer/tensor product of vectors. One then extracts the (strict upper) triangular part of the matrix representation of $\mathbf{x} \wedge y$. The code in *Mathematica*, using the command TensorWedge[.], which implements the exterior product $\wedge$ is given in Listing 3, and a slightly longer code in R, using the tensor/outer product command outer(.), is given in Listing 4.

```
closure[x_] := Normalize[x, Norm[#, 1] &]
 codadivergencewedge[a_, b_, alpha_] :=
 Norm[Flatten[UpperTriangularize[TensorWedge[closure[a], closure[b]], 1]],  alpha]
```

Listing 3 $\alpha-$Barycentric divergence, via wedge product, in *Mathematica* Language

```r
##secondversion
alphabarycentricdivergence2 <- function(x, y, alpha) {
  #normalize x and y
  xclosed<-x/sum(x)
  yclosed<-y/sum(y)

#compute |x_iy_j-x_jy_i|^\alpha et put the result as an antisymmetric matrix
  plucker<- abs(outer(xclosed,yclosed,"*")-outer(yclosed,xclosed,"*"))^alpha
  # Extract elements over the  diagonal
  upper_triangle <- plucker[upper.tri(plucker)]
  (sum(upper_triangle))^(1/alpha)
  }
}
```

Listing 4 $\alpha-$Barycentric divergence using outer products, in R Language

2.2 Barycentric covariance and variance

We give in Listing 5 an R implementation of the barycentric covariance matrix $\mathrm{Cov}([\mathbf{x}]_+,[\mathbf{y}]_+)$, Definition 8, equation (17).

```r

#barycentric covariance of 2 matrices
# compute the  matrix of pseudo scalar product of 2 pairs of vectors
pseudocov<-function(x,y,mx,my){
  (outer(x,mx,"*")-outer(mx,x,"*"))*(outer(y,my,"*")-outer(my,y,"*"))}
# barycentric covariance of two matrices
barycentric_cov<-function(X,Y){
  #Normalise each matrix (closure)
  X_normalized <- sweep(X, 1, rowSums(X), FUN="/")
  Y_normalized <- sweep(Y, 1, rowSums(X), FUN="/")

  # compute  vector of column means
  mx <- colMeans(X_normalized)
  my <- colMeans(Y_normalized)
  #note the vectors of means are already in the simplex

  n<-nrow(X);
  d<-ncol(X);
  result<-matrix(data = 0,nrow = d,ncol=d)
  for (i in 1:n) {
    result<-result+pseudocov(X_normalized[i,],Y_normalized[i,],mx,my)
  }
  return(result/n)
}
```

Listing 5 Barycentric covariance matrix using outer products, in R Language

References

Faugeras, O.P.: Log-Free Distance and Covariance matrix for Compositional Data II: The projective/exterior product approach. TSE Working Paper, n° 24-1601 (2024). https://hal.science/hal-04822302v1

R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2023). R Foundation for Statistical Computing. https://www.R-project.org/

Wolfram Research, I.: Mathematica, Version 13.3. Wolfram Research, Inc., Champaign, Illinois (2023). Wolfram Research, Inc. https://www.wolfram.com/mathematica/