DeCoVart-use-cases

Bastien Chassagnol* Gregory Nuel† Etienne Becht‡

Model

We introduce the following notations:

- $(\boldsymbol{y} = (y_{gi}) \in \mathbb{R}_+^{G\times N}$ is the global bulk transcriptomic expression, measured in N individuals.
- $\boldsymbol{X} = (x_{gj}) \in \mathcal{M}_{\mathbb{R}^{G\times J}}$ the signature matrix of the mean expression of G genes in J purified cell populations.
- $\boldsymbol{p} = (p_{ji}) \in]0,1[^{J\times N}$ the unknown relative proportions of cell populations in N samples

As in most traditional deconvolution models, we assume that the total bulk expression can be reconstructed by summing the individual contributions of each cell population weighted by its frequency, as stated explicitly in the following linear matricial relationship (Eq.(1)):

$$\boldsymbol{y} = \boldsymbol{X} \times \boldsymbol{p} \tag{1}$$

In addition, we consider the following unit simplex constraint on the cellular ratios (Eq.(2)):

$$\begin{cases} \sum_{j=1}^{J} p_j = 1 \\ \forall j \in \widetilde{J} \quad p_j \geq 0 \end{cases} \tag{2}$$

Rationale of the new generative model

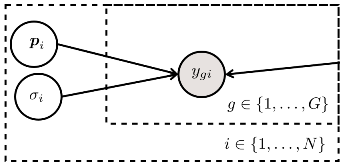
However, in real conditions with technical and environmental variability, the strict linearity of the deconvolution does not strictly hold. Thus, an additional error term is usually added, assumed to follow a *homoscedastic* zero-centred Gaussian distribution and with pairwise independent response measures while the exogenous variables (here, the purified expression profiles) are supposed determined: this set of conditions is referred to as the Gaussian-Markow assumptions. In that configuration, the MLE (maximum likelihood estimate) that bast describes this standard linear model is equal to the ordinary least squares (OLS) estimate.

In contrast to this canonical approach, in DeCovarT, we relax the *exogeneity* property by treating exogenous variables $\boldsymbol{X}$ as random variables rather than determined measures, in a process close to the approach of the DSection algorithm [1]. However, to our knowledge, we are the first to weaken the independence assumption between observations by explicitly incorporating the intrinsic covariance structure of the transcriptome of each purified cell population. To do so, we conjecture that the G-dimensional vector $\boldsymbol{x}_j$ characterising the transcriptomic expression of each cell population follows a multivariate Gaussian distribution: $\boldsymbol{x}_j \sim \mathcal{N}_G(\boldsymbol{\mu}_{.j}, \boldsymbol{\Sigma}_j)$, with $\boldsymbol{\mu}_{.j}$ the mean purified transcriptomic expression and $\boldsymbol{\Sigma}_j$ the covariance matrix, that we constrain to be positive-definite and of full rank and that is inferred using the output of the gLasso algorithm [2]. We display respectively the graphical models associated to the standard linear deconvolution model and our new innovative generative model used by the DeCovarT algorithm in subfigures a) and b), in Fig.1.

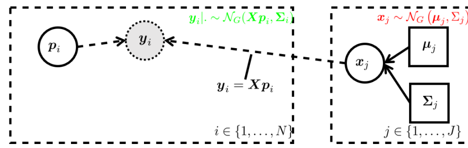
*Laboratoire de Probabilités, Statistiques et Modélisation (LPSM), UMR CNRS 8001, bastien_chassagnol@laposte.net, 0000-0002-8955-2391

†Laboratoire de Probabilités, Statistiques et Modélisation (LPSM), UMR CNRS 8001, Gregory.Nuel@math.cnrs.fr, 0000-0001-9910-2354

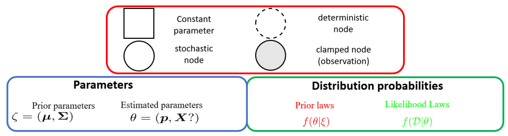
‡Les Laboratoires Servier(IRIS), etienne.becht@servier.com, 0000-0003-1859-9202



(a) Visual representation of the linear regression graphical model



(b) Visual representation of the graphical model underlying the DeCovarT generative model



(c) Legend displaying the main symbols and laws used in a graphical model.

Figure 1: We use the standard graphical convention of graphical models, as depicted in RevBayes webpage. For identifiability reasons, we conjecture that all variability arises from the stochastic nature of the covariates.

Derivation of the log-likelihood

First, we *plugged-in* the mean and covariance parameters $\zeta_j = \left(\boldsymbol{\mu}_{.j}, \boldsymbol{\Sigma}_j\right)$ inferred in the previous step. Then, by letting $\boldsymbol{\zeta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\quad \boldsymbol{\mu} = (\boldsymbol{\mu}_{.j})_{j \in \widetilde{J}} \in \mathcal{M}_{G\times J}$, $\quad \boldsymbol{\Sigma} \in \mathcal{M}_{G\times G}$ the known parameters and $\boldsymbol{p}$ the unknown cellular ratios, the conditional distribution $\boldsymbol{y}|(\boldsymbol{\zeta}, \boldsymbol{p})$ is the convolution of pairwise independent multivariate Gaussian distributions, which is also a multivariate Gaussian distribution (Eq.(3)), deduced from the *affine invariant* property of Gaussian distributions.

$$\boldsymbol{y}|(\boldsymbol{\zeta}, \boldsymbol{p}) \sim \mathcal{N}_G(\boldsymbol{\mu}\boldsymbol{p}, \boldsymbol{\Sigma}) \text{ with } \boldsymbol{\mu} = (\boldsymbol{\mu}_{.j})_{j \in \widetilde{J}}, \quad \boldsymbol{p} = (p_1, \dots, p_J) \text{ and } \boldsymbol{\Sigma} = \sum_{j=1}^{J} p_j^2 \boldsymbol{\Sigma}_j \tag{3}$$

From Eq.(3), we readily compute the associated conditional log-likelihood (Eq.(4)):

$$\ell_{\boldsymbol{y}|\boldsymbol{\zeta}}(\boldsymbol{p}) = C + \log\left(\operatorname{Det}\left(\sum_{j=1}^{J} p_j^2 \boldsymbol{\Sigma}_j\right)^{-1}\right) - \frac{1}{2}(\boldsymbol{y} - \boldsymbol{p}\boldsymbol{\mu})^\top \left(\sum_{j=1}^{J} p_j^2 \boldsymbol{\Sigma}_j\right)^{-1} (\boldsymbol{y} - \boldsymbol{p}\boldsymbol{\mu}) \tag{4}$$

First and second-order derivation of the unconstrained DeCovarT log-likelihood function

The stationary points of a function and notably maxima, are given by the roots (the values at which the function crosses the x-axis) of its gradient, in our context, the vector: $\nabla \ell : \mathbb{R}^J \to \mathbb{R}^J$ evaluated at point $\nabla \ell(\boldsymbol{p}) :]0,1[^J \to \mathbb{R}^J$. Since the computation is the same for any cell ratio p_j, we give an explicit formula for only one of them (Eq.(5)):

$$\begin{aligned}
\frac{\partial \ell_{\boldsymbol{y}|\boldsymbol{\zeta}}(\boldsymbol{p})}{\partial p_j} &= \tfrac{\partial \log(\operatorname{Det}(\boldsymbol{\Theta}))}{\partial p_j} - \tfrac{1}{2}\left[\tfrac{\partial (\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top}{\partial p_j} \boldsymbol{\Theta}(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p}) + (\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top \tfrac{\partial \boldsymbol{\Theta}}{\partial p_j}(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p}) + (\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta} \tfrac{\partial (\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})}{\partial p_j}\right] \\
&= -\operatorname{Tr}\left(\boldsymbol{\Theta} \tfrac{\partial \boldsymbol{\Sigma}}{\partial p_j}\right) - \tfrac{1}{2}\left[-\boldsymbol{\mu}_{.j}^\top \boldsymbol{\Theta}(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p}) - (\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top \Theta \tfrac{\partial \Sigma}{\partial p_j} \Theta(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p}) - (\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta}\boldsymbol{\mu}_{.j}\right] \\
&= -2p_j \operatorname{Tr}(\boldsymbol{\Theta}\boldsymbol{\Sigma}_j) + (\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta}\boldsymbol{\mu}_{.j} + p_j (\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta} \Sigma_j \boldsymbol{\Theta}(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})
\end{aligned} \tag{5}$$

Since the solution to $\nabla\left(\ell_{\boldsymbol{y}|\boldsymbol{\zeta}}(\boldsymbol{p})\right) = 0$ is not closed, we had to approximate the MLE using iterated numerical optimisation methods. Some of them, such as the Levenberg–Marquardt algorithm, require a second-order approximation of the function, which needs the computation of the Hessian matrix. Deriving once more Eq.(6) yields the Hessian matrix, $\mathbf{H} \in \mathcal{M}_{J \times J}$ is given by:

$$
\begin{aligned}
\mathbf{H}_{i,i} = \frac{\partial^2 \ell}{\partial^2 p_i} &= -2 \operatorname{Tr}\left(\boldsymbol{\Theta}\boldsymbol{\Sigma}_i\right) + 4p_i^2 \operatorname{Tr}\left(\left(\boldsymbol{\Theta}\boldsymbol{\Sigma}_i\right)^2\right) - 2p_i(\boldsymbol{y} - \boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta}\boldsymbol{\Sigma}_i\boldsymbol{\Theta}\boldsymbol{\mu}_{\boldsymbol{.i}} - \boldsymbol{\mu}_{.i}^\top \boldsymbol{\Theta}\boldsymbol{\mu}_{\boldsymbol{.i}} - \\
& 2p_i(\boldsymbol{y} - \boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta}\boldsymbol{\Sigma}_i\boldsymbol{\Theta}\boldsymbol{\mu}_{.i\,-(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta}\left(4p_i^2\boldsymbol{\Sigma}_i\boldsymbol{\Theta}\boldsymbol{\Sigma}_i - \boldsymbol{\Sigma}_i\right)\boldsymbol{\Theta}(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})}, \quad i \in \widetilde{J} \\
\mathbf{H}_{i,j} = \frac{\partial^2 \ell}{\partial p_i \partial p_j} &= 4p_j p_i \operatorname{Tr}\left(\boldsymbol{\Theta}\boldsymbol{\Sigma}_j\boldsymbol{\Theta}\boldsymbol{\Sigma}_i\right) - 2p_i(\boldsymbol{y} - \boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta}\boldsymbol{\Sigma}_i\boldsymbol{\Theta}\boldsymbol{\mu}_{\boldsymbol{.j}} - \boldsymbol{\mu}_{.i}^\top \boldsymbol{\Theta}\boldsymbol{\mu}_{\boldsymbol{.j}} - \\
& 2p_j(\boldsymbol{y} - \boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta}\boldsymbol{\Sigma}_j\boldsymbol{\Theta}\boldsymbol{\mu}_{.i\,-4p_ip_j(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})^\top \boldsymbol{\Theta}\boldsymbol{\Sigma}_i\boldsymbol{\Theta}\boldsymbol{\Sigma}_j\boldsymbol{\Theta}(\boldsymbol{y}-\boldsymbol{\mu}\boldsymbol{p})}, \quad (i,j) \in \widetilde{J}^2, i \neq j
\end{aligned}
\tag{6}
$$

in which the coloured sections pair one by one with the corresponding coloured sections of the gradient, given in Eq.(5). Matrix calculus can largely ease the derivation of complex algebraic expressions, thus we remind in Appendix Matrix calculus relevant matrix properties and derivations [1].

However, the explicit formulas for the gradient and the Hessian matrix of the log-likelihood function, given in Eq.(5) and Eq.(6) respectively, do not take into account the simplex constraint assigned to the ratios. While some optimisation methods use heuristic methods to solve this problem, we consider alternatively a reparametrised version of the problem, detailed comprehensively in Appendix Reparametrised log-likelihood.

Iterated optimisation

The MLE is traditionally retrieved from the roots of the gradient of the log-likelihood. However, in our generative framework, cancelling the gradient of Equation (4) reveals a non-closed form. Instead, iterated numerical optimisation algorithms can be used to proxy the roots, most of them considering first or second-order approximations of the function to optimise.

The *Levenberg-Marquardt algorithm* bridges the gap between between the steepest descent method (first-order) and the Newton-Raphson method (second-order) by inflating the diagonal terms of the Hessian matrix. Away from the endpoint, a second-order descent is favoured for its faster convergence pace, while the steepest approach is privileged close to the extremum, as it allows careful refinement of the step size. We use function **marqLevAlg**, since it notably introduces a stringent convergence criteria, the relative distance to the maximum (RDM), which sets apart extrema from spurious saddle points [3].

We provide additional theoretical results, such as analytical formulas for the Gradient and the Hessian in their constrained and unconstrained versions as well as simulation outputs in the vignette of the DeCovarT Github webpage.

Simulations

Simulation of a convolution of multivariate Gaussian mixtures

To assert numerically the relevance of accounting the correlation between expressed transcripts, we designed a simple toy example with two genes and two cell populations. Hence, using the simplex constraint (Eq.(2)), we only have to estimate one free unconstrained parameter, ρ_1, and then converts it back to the original ratio space using the mapping function (Eq.(7)).

We simulated "virtual" bulk mixture, $\boldsymbol{y} \in \mathcal{M}_{G \times N}$, for a set of artificial samples $N = 500$, with the following generative model:

- We tested two levels of cellular ratios, one with equi-balanced proportions ($\boldsymbol{p} = (p_1, p_2 = 1 - p_1) = (\frac{1}{2}, \frac{1}{2})$ and one with highly unbalanced cell populations: $\boldsymbol{p} = (0.95, 0.05)$.

[1] The numerical consistency of these derivatives was asserted with the **numDeriv** package, using the more stable Richardson's extrapolation

- Then, each purified transcriptomic profile of the two cell populations is drawn from a bivariate Gaussian distribution. We compared two scenarios, playing on the mean distance of centroids, respectively $\mu_{.1} = (20, 22), \mu_{.2} = (22, 20)$ and $\mu_{.2} = (20, 40), \mu_{.2} = (40, 20)$) and building the covariance matrix, $\mathbf{\Sigma} \in \mathcal{M}_{2\times 2}$ by assuming equal individual variances for each gene (the diagonal terms of the covariance matrix, $\mathrm{Diag}(\mathbf{\Sigma_1}) = \mathrm{Diag}(\mathbf{\Sigma_1}) = \boldsymbol{I}_2$) but varying the pairwise correlation between gene 1 and gene 2, $\mathbb{C}\mathrm{ov}\left[x_{1,2}\right]$, on the following set of values: $\{-0.8, -0.6, \ldots, 0.8\}$ for each of the cell population.
- As stated in Eq.(1), we assume that bulk mixture, $\mathbf{y}_{.i}$ could be directly reconstructed by summing up the individual cellular contributions weighted by their abundance, without additional noise.

Precisely, we tested the following general 8 parameters configurations listed in Table below (1) in this bivariate benchmark:

Table 1: The 8 general scenarios tested to compare the performance of DeCovarT vs standard linear deconvolution model

ID	Entropy	OVL	Proportions	Means	Variance
B1-Ho	1.000	0.1379955	0.5 / 0.5	(20,22);(22,20)	1 / 1
B1-He	1.000	0.1878551	0.5 / 0.5	(20,22);(22,20)	1 / 2
B2-Ho	0.286	0.3193959	0.95 / 0.05	(20,22);(22,20)	1 / 1
B2-He	0.286	0.4310286	0.95 / 0.05	(20,22);(22,20)	1 / 2
B3-Ho	1.000	0.0000000	0.5 / 0.5	(20,40);(40,20)	1 / 1
B3-He	1.000	0.0000000	0.5 / 0.5	(20,40);(40,20)	1 / 2
B4-Ho	0.286	0.0000000	0.95 / 0.05	(20,40);(40,20)	1 / 1
B4-He	0.286	0.0000000	0.95 / 0.05	(20,40);(40,20)	1 / 2

Practical implementation

In practice, to generate a sample with the probabilistic framework described in subfigure b) (Fig.1), you can take benefit from function you may use the natively provided DeCovarT function `simulate_bulk_mixture()` (click-on link, to access automatically to its documentation).

To compare automatically the quality of the output and the performance of several deconvolution algorithms, you may directly use function `deconvolute_ratios()`, which, in addition to infer in a parallel fashion the individual cellular ratios of n independent samples, performs the following normalisation and pre-processing tasks[2]:

1. Remove genes from both reference signature matrices and bulk profiles, presenting at least a missing value for respectively a cell population, or an individual biological sample. In the mean time, we control that the provided transcriptomic expressions are in numeric format.
2. Ensure that at least 50% of the provided genes are common between the purified reference profiles and bulk mixture[3]. We also check if valid `colnames` are provided for the purified reference matrices

[2]Note that Window users can not unfortunately perform parallel computation with a R interface, slowing considerably the operations compared to a Linux user. Additionally, with that function, we assume that the reference profile used is the same for all individuals.

[3]For the moment, you may use any `rownames` argument, however, in a near future, we plan to add automated control whether they are known and valid HGNC symbols

(in a near time, we would control that they match ontology terms, as the ones returned by R function **ontoProc**).

3. Ensure that both reference and bulk profiles are provided as raw counts (or possibly non-logged TPMs), since the DeCovarT model and its variants assume a convolution of parametric distributions on the original parameter space.

Alternatively, you may directly use any of the implemented deconvolution functions, providing at least the two required parameters `y` (individual bulk profile) and `X` (individual or global purified reference profiles), but we do not perform again the regular pre-control and processing steps described above. The main characteristics of some of the most relevant deconvolution algorithms implemented are reported in Table 2 below:

Table 2: Main characteristics of the benchmarked packages in our toy example. *DeCovarT function* refers to the R function of our package, used to implement the corresponding algorithm labelled in column 1. When different from the default values of the hyper-parameters, we detail their reviewed values in column *Hyper-parameters*

Algorithm	Key feature and inspiration	DeCovarT function	Hyper-parameters
lm	Standard OLS (ordinary least squares) using function stats::lsfit, then renormalising the ratios to enforce the unit simplex constraint	`deconvolute_ratios_abbas`	intercept =FALSE
nnls	Non-negative linear squared optimisation method, using the Lawson-Hanson algorithm (in R, package nnls::nnls())	`deconvolute_ratios_nnls`	"-"
lsei	Quadratic programming (QP) to account for the unit simplex constraint, used for isntance by deconvolution algorithm DeconRNASeq	`deconvolute_ratios_deconRNASeq`	"-"
RLR	Variants of robust linear regression (RLR)have been used by the ABIS algorithm (Monaco et al., 2019), and by the FARDEEP allgorithm (Hao et al, 2018). We used function MASS::rlm() to that purpose.	`deconvolute_ratios_monaco`	method = M
CIBERSORT	ν-SVR with linear kernel, enabling further feature selection, and removing some noise. In R, use of function e1071::best.svm	`deconvolute_ratios_CIBERSORT`	range.nu = (0.2, 0.5, 0.8) // fix=0.75
optim	First-order iterated descent optimisation algorithm, on the unconstrained log-likelihood function and without explicit formula of the gradient.	`deconvolute_ratios_basic_optim`	maxit = 2000, abstol=reltol = 10^{-6}
barrier	Variant of the previously described algorithm, but with the additional possibility to provide linear restrictions on the estimated parameters. Use of method stats::constrOptim()	`deconvolute_ratios_constrOptim`	outer.iterations = 2000,outer.eps = reltol=abstol=10^{-6}
SA	Simulated annealing (useful to optimise globally non-convex functions, especially presenting multiple local extrema). Use of function stats::optim() and method *SANN*	`deconvolute_ratios_simulated_annealing`	maxit = 2000 (not the number of iterations, but rather the number of random function evaluations, under a designed cooling temperature)
gradient	First-order iterated descent optimisation algorithm, also named gradient descent. On the reparametrised log-likelihood function, with the explicit formula of the gradient. Use of function stats::optim() and method *BFGS*	`deconvolute_ratios_first_order`	maxit = 2000, abstol=reltol = 10^{-6}
hessian	Second-order iterated descent optimisation algorithm, equivalent to a newton Raphson algorithm to retrieve the roots of the gradient. Use of function stats::nlminb()	`deconvolute_ratios_second_order`	iter.max = 2000, abs.tol=rel.tol=x.tol=xf.tol=10^{-6}, eval.max = 1
DeCoVarT	Levenberg–Marquardt algorithm, as implemented with marqLevAlg::marqLevAlg(). Penalised second-order iterated descent optimisation algorithm	`deconvolute_ratios_DeCoVarT`	epsa=epsb=epsd=10^{-6} // maxiter = 2000 // multipleTry = 1

[a] Most optimisation methods implemented in R, by default, aim at minimising a given quantity/function, since, most of them were designed at first to minimise the squared error residuals. Howver, maximising a function is equivalent to minimise its negative counterpart, which can easily be done in optimisation algorithms, by assigning parameter ‘fnscale‘ to 1.

Finally, to reproduce the results of the benchmark, you can use the highly specific function `benchmark_deconvolution_algorithms_two_genes()`: a highly specific wrapper, used as a toy example to automatically benchmark for a given number of observations n the performance of several deconvolution algorithms, as a function of the level of entropy (play on the parameter `proportions`) and overlap (play on parameter `signature_matrices` to control the average proximity of *centroids*= the averaged expression profiles, and parameters `diagonal_terms,corr_sequence` to control the topological configuration of the bivariate Gaussian distribution, playing on the parametrisation of the covariance matrix.) and `plot_correlation_Heatmap()` to visualise in a row the performance of several deconvolution algorithms (each Heatmap should be associated to one scenario of homoscedascity, entropy and average distance to centroids, then used to compare the performance, measured for a given metric, of a deconvolution algorithm). For instance, to reproduce the results provided in document Heatmaps associated to bivariate scenario, we execute the following code snippet:

```
library(DeCovarT)
library(dplyr)

#################################################################
##                execute the bivariate simulation             ##
#################################################################

RNGkind("L'Ecuyer-CMRG")
set.seed(3) # set an unique core (only for Linux users, to enable reproducible results)
deconvolution_functions <- list(
  "lm" = list(FUN = deconvolute_ratios_abbas),
  "nnls" = list(FUN = deconvolute_ratios_nnls),
  "lsei" = list(FUN = deconvolute_ratios_deconRNASeq),
  # with the new log-likelihood function and explicit gradient or Hessian
  "gradient" = list(
    FUN = deconvolute_ratios_first_order,
    additional_parameters = list(epsilon = 10^-3, itmax = 200)
  ),
  "hessian" = list(
    FUN = deconvolute_ratios_second_order,
    additional_parameters = list(epsilon = 10^-3, itmax = 200)
  ),
  "DeCoVarT" = list(
    FUN = deconvolute_ratios_DeCoVarT,
    additional_parameters = list(epsilon = 10^-3, itmax = 200)
  ),
  # with the new log-likelihood function, but stochastic estimation of the gradient
  "optim" = list(
    FUN = deconvolute_ratios_basic_optim,
    additional_parameters = list(epsilon = 10^-3, itmax = 200)
  ),
  "barrier" = list(
    FUN = deconvolute_ratios_constrOptim,
    additional_parameters = list(epsilon = 10^-3, itmax = 200)
  ),
  "SA" = list(
    FUN = deconvolute_ratios_simulated_annealing,
    additional_parameters = list(epsilon = 10^-3, itmax = 200)
  )
)
# retrieve the estimated, benchmarked parameters
```

```r
bivariate_simulation <- benchmark_deconvolution_algorithms_two_genes(
  proportions = list(
    "balanced" = c(0.50, 0.50),
    "highly_unbalanced" = c(0.95, 0.05)
  ),
  signature_matrices = list(
    "small ICD" = matrix(c(20, 22, 22, 20), nrow = 2),
    "high ICD" = matrix(c(20, 40, 40, 20), nrow = 2)
  ),
  corr_sequence = seq(-0.75, 0.75, 0.25),
  diagonal_terms = list("homoscedastic" = c(1, 1), "heteroscedastic" = c(1, 2)),
  deconvolution_functions = deconvolution_functions, n = 2000, scaled = FALSE
) %>% magrittr::extract2("simulations")


#################################################################
##                 plot the associated Heatmaps                ##
#################################################################
splitted_parameters <- split(x = bivariate_simulation, f = bivariate_simulation$ID)
bivariate_simulation_heatmap <- purrr::imap(
  splitted_parameters,
  function(.data, .name_scenario) {
    # actual call to DeCovarT function
    heatmap_per_scenario <- plot_correlation_Heatmap(.data)
    heatmap_page <- purrr::imap(heatmap_per_scenario, ~ ComplexHeatmap::draw(.x,
      padding = unit(c(0, 0, 0, 0), "cm"),
      column_title = .y, column_title_gp = grid::gpar(fontsize = 12, fontface = "bold")
    ) %>%
      grid::grid.grabExpr())
    # general organisation: 3 deconvolution algorithms per column
    heatmap_page <- gridExtra::arrangeGrob(
      grobs = heatmap_page, ncol = 3, padding = unit(0.1, "line"),
      top = ggpubr::text_grob(.name_scenario, size = 18, face = "bold")
    )
    return(heatmap_page)
  }
)

# save the actual output
ggsave("./figs/bivariate_Heatmaps.pdf",
  gridExtra::marrangeGrob(grobs = bivariate_simulation_heatmap, top = "", ncol = 1, nrow = 1),
  width = 12, height = 12, dpi = 300
)
```

Results on toy example

We compared the performance of DeCovarT algorithm with the outcome of a quadratic algorithm that specifically addresses the unit simplex constraint: the negative least squares algorithm (NNLS, [4]).

Even with a limited toy example including two cell populations characterised only by two genes, we observe that the overlap was a good proxy of the quality of the estimation: the less the two cell distributions overlap, the better the quality of the estimation is (Fig. 2):

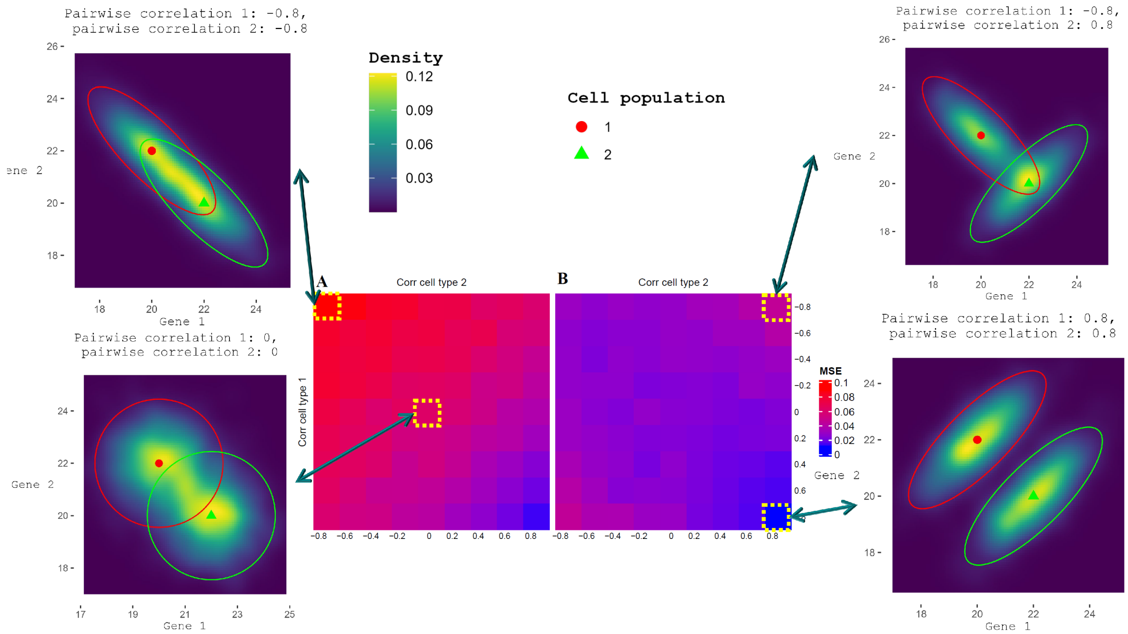


Figure 2: We used package ComplexHeatmap to display the mean square error (MSE) of the estimated cell ratios, comparing the output of the deconRNASEQ algorithm [5], in Panel **A**, with our newly implemented DeCovarT algorithm, in Panel **B**. The lower the MSE, the least noisy and biased the estimates. In addition, we added two-dimensional density plots of the central scenario, parametrised by a diagonal covariance matrix, and most extreme scenarios, with the highest gene pairwise correlation. The ellipsoids represent for each cell population the 95% confidence region and the red spherical and green triangular shapes represent respectively the centroids of cell population 1 and cell population 2.

The package used to generate the simulations and infer ratios from virtual or real biological mixtures with the DeCovarT algorithm is implemented on personal Github account DeCovarT.

Results on real datasets

The whole list of benchmarked packages is described, with their main parameters and a summary of the method used, in Table 3 below:

Table 3: Features of some gold-standard deconvolution algorithms, benchmarked in this package. *Family* returns the general type of each deconvolution algorithm, among these three classes: *supervised* algorithms, which use either a reference profile or assume a marker-based approach, in which genes are considered expressed only in one cell population. When avalaible, direct website link is indexed in *Online method*, and when implemented in our package, we give the name associated to the closely avalaible implemented function, in column *DeCovarT function*

Algorithm	Family	Key feature	Authors	DeCovarT function
TIMER	reference-based	Optimization of the condition number (CN) of expression matrix to select markers then standard OLS (ordinary least squares) using function stats::lsfit	Abbas et al., 2009 / Li et al., 2020	`deconvolute_ratios_abbas`
DECONVOLUTE	reference-based	Deconvolution to inferfor cell cycle phase-specific profiles, obtained by Simulated Annealing.	Lu et al., 2003	☒
NNLS	reference-based	Non-negative linear squared optimisation method, using the Lawson-Hanson algorithm (in R, package nnls::nnls())	Lawson et al., 1981	`deconvolute_ratios_nnls`
EPIC	reference-based	QP to account for the unit simplex constraint, weighted by the variance of the genes	Racle et al., 2017	☒
DeconRNASeq	reference-based	Quadratic programming (QP) to account for the unit simplex constraint	Gong and Szustakowski, 2013	`deconvolute_ratios_deconRNASeq`
quanTIseq	reference-based	Quadratic programming (QP) with intercept to account for residual noise	Finotello et al. 2017	☒
ABIS	reference-based	robust linear regression (RLR)	Monaco et al., 2019	`deconvolute_ratios_monaco`
FARDEEP	reference-based	robust linear regression (RLR) coupled with NNLS optimisation	Hao et al, 2018	☒
CIBERSORT	reference-based	ν-SVR with linear kernel. The most relevant genes are referred to as "support vector"	Newman et al., 2015	`deconvolute_ratios_CIBERSORT`
CIBERSORTx	unsupervised with priors	Natural extension of CIBERSORT + possibility to infer the purified gene expression profiles + batch correction, performed with COMBAT, to accout for distinct sequencing strategies	Newman et al., 2019	☒
DSection	unsupervised with priors	Bayesian framework where a prior Dirichlet, possibly noisy, is assumed on the distribution of the cellular ratios and an univariate normal prior with parameters extracted from LS solution assuming known cell-type proportions. Unknown parameters estimated using Metropolis-Hastings sampling.	Erkkila et al., 2010	☒
DeMix / DeMixt	unsupervised with priors	Designed to infer the tumoral proportion of a sample. Neither reference profiles nor mixing proportions are required but the tool can take advantage of it as strong priors. Extension to a three-component, with an additional unknown cell content has been supplied with natural extension DeMixt. Mixture is modelled as a convolution of indepent, univariate log-Normal distributions.	Ahn et al., 2013 / Wang et al., 2019	☒
MCP-counter	marker-based	MCP scores: a) were computed using log2 geometric mean of set of markers; b) correlated to known RNA proportions in the mixture.	Becht et al., 2016	☒
xCell	marker-based	xCell is a recently published method based on ssGSEA.The xCell abundance scores are computed through the following steps: ssGSEA is performed independently and averaged for each of the 489 gene sets implemented in the GSVA R package then averaged over a cell type; platform-specific corrections as well as "spillover" contaminations to account for closely related cell types are then corried out. Abundance scores are returned for 64 immune cell types.	Aran et al., 2017	☒

References

[1] T. Erkkilä, S. Lehmusvaara, P. Ruusuvuori, T. Visakorpi, I. Shmulevich, and H. Lähdesmäki, "Probabilistic analysis of gene expression measurements from heterogeneous tissues," *Bioinformatics*, 2010, doi: 10.1093/bioinformatics/btq406.

[2] R. Mazumder and T. Hastie, "The Graphical Lasso: New Insights and Alternatives," *Electronic Journal of Statistics*, Nov. 2011, doi: 10.1214/12-EJS740.

[3] M. Prague, D. Commenges, J. Guedj, J. Drylewicz, and R. Thiébaut, "NIMROD: A program for inference via a normal approximation of the posterior in models with random effects based on ordinary differential equations," *Computer Methods and Programs in Biomedicine*, Aug. 2013, doi: 10.1016/j.cmpb.2013.04.014.

[4] K. H. Haskell and R. J. Hanson, "An algorithm for linear least squares problems with equality and nonnegativity constraints," *Mathematical Programming*, Dec. 1981, doi: 10.1007/BF01584232.

[5] T. Gong and J. D. Szustakowski, "DeconRNASeq: A statistical framework for deconvolution of heterogeneous tissue samples based on mRNA-Seq data," *Bioinformatics (Oxford, England)*, 2013, doi: 10.1093/bioinformatics/btt090.

[6] J. Ahn *et al.*, "DeMix: Deconvolution for mixed cancer transcriptomes using raw measured data," *Bioinformatics (Oxford, England)*, 2013, doi: 10.1093/bioinformatics/btt301.

[7] Z. Wang *et al.*, "Transcriptome Deconvolution of Heterogeneous Tumor Samples with Immune Infiltration," *iScience*, 2018, doi: 10.1016/j.isci.2018.10.028.

[8] M. Chion, "Développement de nouvelles méthodologies statistiques pour l'analyse de données de protéomique quantitative," These en préparation, Strasbourg, 2018. Available: http://www.theses.fr/s269600

[9] M. Taboga, "Metropolis-Hastings algorithm," 2021. Available: https://www.statlect.com/fundamentals-of-statistics/Metropolis-Hastings-algorithm

Appendix A: Theoretical details

First and second-order derivation of the constrained DeCovarT log-likelihood function

To reparametrise the log-likelihood function (Eq.(4)) in order to explicitly handling the unit simplex constraint (Eq.(2)), we consider the following mapping function: $\boldsymbol{\psi} : \boldsymbol{\rho} \to \boldsymbol{p} \mid \quad \boldsymbol{\rho} \in \mathbb{R}^{J-1}, \boldsymbol{p} \in]0,1[^J$ (Eq.(7)):

$$\boldsymbol{p} = \boldsymbol{\psi}(\boldsymbol{\rho}) = \begin{cases} p_j = \frac{e^{\rho_j}}{\sum_{k<J} e^{\rho_k} + 1}, \, j < J \\ p_J = \frac{1}{\sum_{k<J} e^{\rho_j} + 1} \end{cases} \qquad \boldsymbol{\rho} = \boldsymbol{\psi}^{-1}(\boldsymbol{p}) = \left(\ln \left(\frac{p_j}{p_J} \right) \right)_{j \in \{1,\dots,J-1\}} \tag{7}$$

that is a C^2-diffeomorphism, since $\boldsymbol{\psi}$ is a bijection between $\boldsymbol{p}$ and $\boldsymbol{\rho}$ twice differentiable.

Its Jacobian, $\mathbf{J}_{\boldsymbol{\psi}} \in \mathcal{M}_{J\times(J-1)}$ is given by Eq.(8):

$$\mathbf{J}_{i,j} = \frac{\partial p_i}{\partial \rho_j} = \begin{cases} \frac{e^{\rho_i} B_i}{A^2}, & i = j, \, i < J \\ \frac{-e^{\rho_j} e^{\rho_i}}{A^2}, & i \neq j, \, i < J \\ \frac{-e^{\rho_j}}{A^2}, & i = J \end{cases} \tag{8}$$

with i indexing vector-valued $\boldsymbol{p}$ and j indexing the first-order order partial derivatives of the mapping function, $A = \sum_{j'<J} e^{\rho_{j'}} + 1$ the sum over exponential (denominator of the mapping function) and $B = A - e^{\rho_i}$ the sum over ratios minus the exponential indexed with the currently considered index i.

The Hessian (which fortunately is symmetric for each component j, as expected according to the Schwarz's theorem) of the vectorial mapping function $\boldsymbol{\psi(\rho)}$ is a third-order tensor of rank $(J-1)(J-1)J$, given by Eq.(9):

$$\frac{\partial^2 p_i}{\partial k \partial j} = \begin{cases} \frac{e^{\rho_i} e^{\rho_l}(-B_i + e^{\rho_i})}{A^3}, \ (i < J) \wedge ((i \neq j) \oplus (i \neq k)) \quad (a) \\ \frac{2e^{\rho_i} e^{\rho_j} e^{\rho_k}}{A^3}, \ (i < J) \wedge (i \neq j \neq k) \quad (b) \\ \frac{e^{\rho_i} e^{\rho_j}(-A + 2e^{\rho_j})}{A^3}, \ (i < J) \wedge (j = k \neq i) \quad (c) \\ \frac{B_i e^{\rho_i}(B_i - e^{\rho_i})}{A^3}, \ (i < J) \wedge (j = k = i) \quad (d) \\ \frac{e^{\rho_j}(-A + 2e^{\rho_j})}{A^3}, \ (i = J) \wedge (j = k) \quad (e) \\ \frac{2e^{\rho_j} e^{\rho_k}}{A^3}, \ (i = J) \wedge (j \neq k) \quad (f) \end{cases} \tag{9}$$

with i indexing $\boldsymbol{p}$, j and k respectively indexing the first-order and second-order partial derivatives of the mapping function with respect to $\boldsymbol{\rho}$. In line (a), $\oplus$ refers to the Boolean XOR operator, $\wedge$ to the AND operator and $l = \{j, k\} \setminus i$.

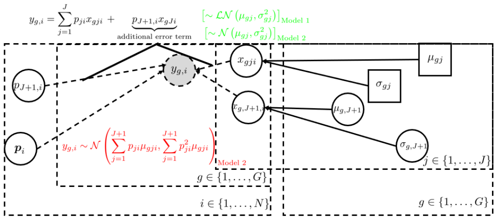
To derive the log-likelihood function in Eq.(5), we reparametrise $\boldsymbol{p}$ to $\boldsymbol{\rho}$, using a standard *chain rule formula*). Considering the original log-likelihood function, Eq.(4), and the mapping function, Eq.(7), the differential at the first order and at the second order is given by Eq.(10) and Eq.(11), respectively defined in $\mathbb{R}^{J-1}$ and $\mathcal{M}_{(J-1)\times(J-1)}$:

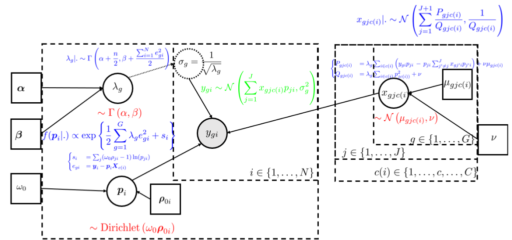
$$\left[\frac{\partial \ell_{\boldsymbol{y}|\boldsymbol{\zeta}}}{\partial \rho_j}\right]_{j<J} = \sum_{i=1}^{J} \frac{\partial \ell_{\boldsymbol{y}|\boldsymbol{\zeta}}}{\partial p_i} \frac{\partial p_i}{\partial \rho_j} \tag{10}$$

$$\left[\frac{\partial \ell^2_{\boldsymbol{y}|\boldsymbol{\zeta}}}{\partial \rho_k \rho_j}\right]_{j<J,\, k<J} = \sum_{i=1}^{J} \sum_{l=1}^{J} \left(\frac{\partial p_i}{\partial \rho_j} \frac{\partial^2 \ell_{\boldsymbol{y}|\boldsymbol{\zeta}}}{\partial p_i \partial p_l} \frac{\partial p_l}{\partial \rho_k}\right) + \sum_{i=1}^{J} \left(\frac{\partial \ell_{\boldsymbol{y}|\boldsymbol{\zeta}}}{\partial p_i} \frac{\partial^2 p_i}{\partial \rho_k \rho_j}\right) \quad (d) \tag{11}$$

Appendix B: Natural extension towards a truly Bayesian framework

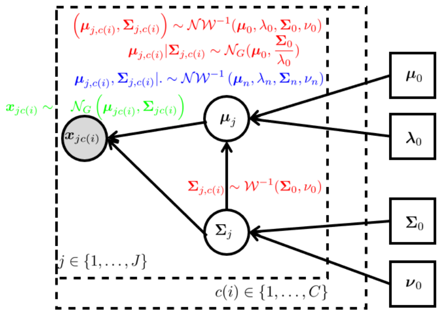
We review below in more details some papers that inspire our work as well as natural extensions of the already implemented framework in Fig.1.

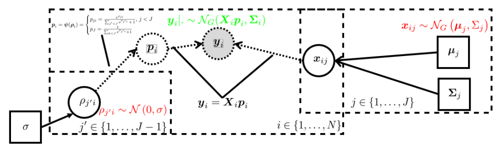




(a) Graphical represention of the DemixT generative model [6], [7]

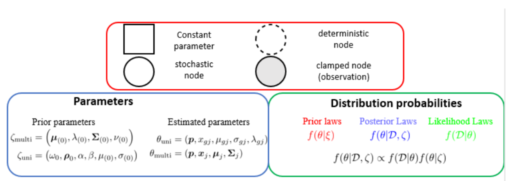
(b) Graphical represention of the DSection generative model [1]





(c) Graphical represention of [8] generative model

(d) Graphical representation of the DeCovarT generative model, in an extended Bayesian framework



(e) Legend used in the graphical model

Figure 3: To compare the generative framework across several deconvolution algorithms, we homogenise indexes and parameters, and represent dependency links using the RevBayes convention. The density functions describing the distribution of the observations, conditioned on the estimated parameters, are written in green, the prior distributions of the parameters to estimate are written in red (themselves possibly conditioned on other unknown parameters, or constant, fixed-value priors), and finally, the resulting posterior distribution parameters, conditioned on both prior and likelihood functions, following the Bayes rules, are written in blue.

We characterise in further details these models below:

a. Model a) in Fig.1 was first theorised in paper [6] to infer the proportion and characteristics of the unknown tumoral content in biological samples with mixed tumoral and normal stromal cells. It was then slightly generalised to a model with three cell proportions, still looking for the same purpose of inferring the tumoral content in sampled individuals. Briefly, in the Demix(T) model, the distribution of the bulk expression for each gene is modelled independently as the sum of univariate, cell-specific $\log_2$ Normal distributions, weighted by the cellular ratios. Additionally, they suppose that the mean and individual gene variance of each of the healthy cell populations (then excluding actually the tumoral cells, since generally tumoral expressions are highly specific to an individual, disease condition and tumoral cell line) have been characterised on physically separated, purified samples. However, in contrast with the DeCovarT model (Eq.(3)), the log-likelihood (or probability distribution) characterising a convolution of $\log_2$ Normal distributions is not closed. The estimation of the parameters is a two-step protocol:

- The unknown, general parameters of interest, namely the cellular proportions subjected to the unit simplex constraint, as well as the individual mean and variance of the tumoral gene expression, are the ones maximising the log-likelihood of the observed, bulk transcriptomic expressions, conditioned on the previously inferred, known individual mean and variance of purified, healthy cell populations. When the log-likelihood form is not available, it is approximated by numerical integration and a corresponding maximum is obtained using a Nelder-Mead procedure. Overall, this procedure is similar to the iterated conditional modes (ICM) approach of (Besag, 1986), since they do not maximise simultaneously all parameters, but rather proceeds by iteratively maximising a set of variables conditional on the rest.

- In a second time, individual references profiles are reconstituted by plugging-in the parameters estimated in the previous step, including the transcriptomic expression of normal cell populations in each sample. With a two-component model, by injecting the general linear constraint of deconvolution models (Eq.(1)), it turns out that only one parameter, either the tumoral or the healthy content, must be inferred (see Eq. 2 in [6]). Globally same framework as the one I developped, but instead considering an univariate framework. Notably, the way they account for the unknown expression of tumoral content by writing it as a function of the other variables is comparable to the proposal distribution we were looking at. Unlike a convolution of $\log_2$ Normal distributions, a convolution (applying beforehand a $\log_2$ normalisation) of Normal distributions can be related to a Normal distribution, thus obtaining a close form of the corresponding log-likelihood is straightforward. Such a model with two components only is also described in the original paper of [6][4].

b. Model b) in Fig.1 is a full Bayesian framework, in which the bulk expression of each gene in each sample, y_{gi}, follows a Normal distribution whose parameters are stochastic variables themselves rather than fixed, point-estimates. Precisely, the distribution of the inverse of the variance, what we referred to *precision*, is assumed following a Gamma distribution[5], the cellular ratios, prior to observations, follows a Dirichlet distribution[6] and the individual purified cell expressions, whose sum weighted by the cellular proportions, is assumed to equal the averaged bulk expression of the genes measured, are themselves described independently for each gene and each experimental condition by an univariate Gaussian distribution. However, while the posterior distribution of the individual, cell-specific expressions and individual bulk gene variance is *conjugated* with their respective prior distribution, the posterior distribution, since computing the normalising constant is infeasible, of the cellular ratios, can not be related to a known density distribution. To retrieve the posterior distributions, a combination of Gibbs iterations, when at least the posterior distribution of a subset of the parameters, conditioned on the currently remaining estimates, is known, and Metropolis-Hasting (MH), when the posterior distribution is only known up to a normalising constant, are performed on the parameter space Compared to the previous model, variability of the cell-specific genes and the total variability measured in bulk are considered indepedent, and so the resulting bulk observed is not described by a convolution..

c. Model c), developed by [8], details a generative model to quantify the uncertainty for differential analysis, and more precisely evaluate whether the proteomic expression, accounting for the underlying network structure, *meaningfully* diverges between two experimental conditions. The main purpose is then to derive analytically an explicit form of the posterior distribution for the difference of mean peptide intensity, from which computing a p-value and *confidence intervals* is straightforward. To do so, they propose both an univariate model (where each gene is supposed to express independently) and a multivariate model (where as in DeCovarT, the interactions between the genes are described by the covariance matrix of a multivariate Gaussian distribution.) In the multivariate setting, one major difficulty is that the variance and the mean estimates are not anymore independent, thus, when computing their corresponding posterior distributions, we have to marginalise out the parameter of interest, over the joint posterior probability of the mean and variance, conditioned on the observed values.

[4] It is not a Bayesian framework, but so close to my original representation, that I feel the need to quote it here.

[5] Alternatively, other probabilistic configurations directly parametrise the variance itself, by applying to it the so-called inverse Gamma distribution

[6] Its main advantage is that the sampled space explored respects the unit-simplex constraint (Eq.(2)), and integrating strong prior hypothetical knowledge, such as equi-balanced cell proportions, is rather straightforward. Indeed, cellular ratios that had been estimated using standard FACS methods prior to the deconvolution, can be directly plugged in the model, as vector $\boldsymbol{\rho}_{0i}$, while uncertainty on the quality of the estimate, can be directly described by unique multiplicative weight ω_{0i}.

Fortunately, [8] shows that the posterior distribution of the mean, namely $f(\boldsymbol{\mu}|.)$, could be directly related to a multivariate t-distribution (see Proposition 5.1 to get the parameters). Adapater ce travail à une hypothèse marker-based, où on sait que chaque gène n'est exprimé que par une population cellulaire, et il faudrait juste rajouter une constante mutliplicative, semble relativement faisable. L'adapter au cas plus général où les gènes sont potentiellement exprimés par toutes les populations cellulaires, semble en revanche beau plus dure. A discuter!.

d. Finally, in our extended, Bayesian framework of the DecoVarT algorithm, we still assume that prior parameters for both the mean and covariance parameters are available, while an uninformative prior is assumed on the cellular ratios, considering equi-balanced cell populations. Additionally, to handle easily the unit simplex constraint, we directly sample on the space of parameters reparametrised using the *mapping function* described in Eq.(7). Hence, assuming no correlation between the cell proportions, a simple Gaussian distribution with a diagonal covariance matrix and null mean value (with our function, considering equi-balanced proportions, namely equal to $\frac{1}{J}$, on the original space $\boldsymbol{p}$ is equivalent to initiate all cell proportions to a null vector $\mathbf{0}^{J-1}$), while the prior variance set on the cell proportions σ_0 can be directly related to the multiplicative noise constant ω_0 of the DSection framework [1].

Addtional details on extended DeCovarT Bayesian framework

Integrating over the sampling space to compute the normalising constant of the posterior distribution of the ratios, $f(\boldsymbol{p}|.)$, and the individual cellular expressions, $f(\boldsymbol{X_j}|.)$, $j \in \{1, \dots, J\}$, is intractable assertion to be checked!, thus we combine the possibilities enabled by the Metropolis-Hasting (MH) algorithm (to get rid of the normalising constant) and the Gibbs sampling algorithm (instead of sampling simultaneously the complete parameter space, we subset only a relevant part of it, conditioned on the current update of the remaining ones) to compute a MCMC estimation of the posterior distribution of these parameters. We describe below a possible pseudo-code for doing so (Algo. \ref{algo:Decovart-pseudo-code}):

We detail in Box below the general process of the Metropolis-Hasting (MH) algorithm.

Metropolis-Hasting algorithm

Consider the following definitions:

- Function $f(\theta|.)$ is called the *target distribution*, in our specific context, the posterior distribution of both the ratios and the individual cellular expression profiles[a].
- The distribution $q(\theta|\theta^{(q-1)})$ is called alternatively the **proposal distribution** or the **transition kernel**, while the transient value $\theta^{(*)}$ is called the **proposal**.
- The value returned by function K is called the **acceptance probability**.

Then, equipped with such functions and usual MCMC vocabulary, each iteration (q) of the MH algorithm can be decomposed as the following steps:

1. Draw a proposal parameter $\theta^{(*)}$ from conditional distribution $q(\theta|\theta^{(q-1)})$.
2. Determine the probability of accepting this new value, computed as such:

$$K(\theta) = \min\left(\frac{f(\theta^{(*)}|.)}{f(\theta^{(q-1)}|.)}\frac{q(\theta|\theta^{(q-1)})}{q(\theta^{(q-1)}|\theta)}, 1\right)$$

3. Draw $u \sim \text{Unif}[0,1]$.
4. Apply the following updating rule, as a function of a draw of the uniform distribution with success probability parameter $K(\theta)$:

$$\theta^{(q)} = \begin{cases} \theta^{(*)}, & u \leq K(\theta) \\ \theta^{(q-1)}, & u > K(\theta) \end{cases}$$

In the first case, we say that the proposal is *accepted*, while in the second case, it is *rejected*. However, the specific way we design the *kernel distribution*, by updating previous parameter estimates adding them a multivariate Gaussian noise (namely $\theta^{(*)} = \theta^{(q-1)} + \epsilon$), simplifies even further the acceptance probability function. Indeed, using first,

$$q(\theta^{(*)}|\theta^{(q-1)}) = q(\epsilon)$$
$$q(\theta^{(q-1)}|\theta^{*}) = q(-\epsilon)$$

and then deducing from the symmetric property of the multivariate Gaussian distribution that $q(\epsilon) = q(-\epsilon)$, the acceptance probability simplifies to:

$$K(\theta) = \min\left(\frac{f(\theta^{(q-1)} + \epsilon|.)}{f(\theta^{(q-1)}|.)}, 1\right)$$

This specific design is referred to as Random walk Metropolis-Hastings and described in further details in [9].

[a] Actually, since the acceptance probability function is the ratio of two density functions, any multiplicative constant, that reveals often intractable to compute, cancel out

We detail in Box below the general process of the Gibbs sampling algorithm, applied to the specific DeCovarT framework:

Gibbs sampling algorithm

Consider the *target distribution* $f(\boldsymbol{p}, \boldsymbol{X}|\boldsymbol{\zeta}, \mathcal{D})$, the joint density of the posterior parameters of interest, conditioned on constant, user-defined parameters $\boldsymbol{\zeta}$ and observed distribution $\mathcal{D}$, here the measured bulk estimates $\boldsymbol{Y}$. Suppose now that while we do know the joint distribution $f(\boldsymbol{p}, \boldsymbol{X}|\boldsymbol{\zeta}, \mathcal{D})$, we are however able to compute the posterior distribution, at least up to a multiplicative constant, of respectively the posterior distributions of the cellular ratios, $f(\boldsymbol{p}|\boldsymbol{X}, \boldsymbol{\zeta}, \mathcal{D})$ and the purified individual cell expression profiles, $f(\boldsymbol{X}|\boldsymbol{p}, \boldsymbol{\zeta}, \mathcal{D})$. Whenever you are able to decompose your parameters into meaningful sets, you may apply Gibbs sampling, and such distributions are referred to as *full conditional.*
In that case, the Gibbs algorithm will perform the following updated iterations:

1. Draw a proposal from $f(\boldsymbol{p}^{(q)}|\boldsymbol{X}^{(q-1)}, \boldsymbol{\zeta}, \mathcal{D})$ for the cellular ratios.
2. Use the updated cellular proportions to generate a new proposal for the purified cellular expressions: $f(\boldsymbol{X}^{(q)}|\boldsymbol{p}^{(q)}, \boldsymbol{\zeta}, \mathcal{D})$.

Generally, the idea of the Gibbs sampling is to decompose the set of parameters to sample into independent blocks, and update each of them in the same iteration, conditioned on the most updated version of the remaining parameter subsets.

Finally, when designing the proposal distribution, we must ensure it could generate any value belonging to the *support* of the target distribution, noted R_f, in other words, that $q(\theta^{(*)}|\theta^{(q-1)}) > 0$. This technical condition is indeed required for the validity of the *balance condition*, required for the *ergodicity* and the convergence to a stationary distribution of the MCMC sample. In addition, the deconvolution framework requires two additional linear constraints over the parameters:

1. First, the estimated cellular ratios are required to respect the unit simplex constraint, (Eq.(2)), handled by reparametrising the parameters using the mapping function (step (A) of Algo. \ref{algo:Decovart-pseudo-code} and Eq.(7)).
2. Secondly, we must verify that the weighted sum of the purified cell expression profiles indeed reconstitute the bulk mixture, in other words, that Eq.(1) holds. We can show by *induction* that our algorithm preserves indeed these conditions:
   i. First, consider that at step $q-1$, linear constraint $\boldsymbol{y}_i = \boldsymbol{X}_i^{(q-1)} \times \boldsymbol{p}_i^{(q-1)}$ holds.
   ii. When updating the cellular ratios in Algo \ref{algo:Decovart-pseudo-code}, conditioned on the observations and the other set of parameters, Equation (i) sets for any sample i and any cellular profile j that $\boldsymbol{x}_{ji}^{(*)} p_{ji}^{(q)} = p_{ji}^{(q-1)} \boldsymbol{x}_{ji}^{(q-1)}$, it is straightforward to show that summing over J these weighted cellular profiles is still equal to $\boldsymbol{y}_i$.
   iii. Updating the purified cellular expression profiles, conditioned by previously updated cell ratios, underscores that one of the cellular expression profiles can be rewritten as a combination of the others (in other words, that it is not a free parameter), considering $\boldsymbol{y}_i$ as a constant variable. Namely, this is equivalent to solve the following equality (Eq. (12)):

$$
\begin{gathered}
\sum_{j<J} p_{ji}^{(q)} \boldsymbol{x}_{ji}^{(q)} + p_{Ji}^{(q)} \boldsymbol{x}_{Ji}^{(q)} = \sum_{j<J} p_{ji}^{(q-1)} \boldsymbol{x}_{ji}^{(q-1)} + p_{Ji}^{(q-1)} \boldsymbol{x}_{Ji}^{(q-1)} = \boldsymbol{y}_i \\
\Longleftrightarrow \\
\sum_{j<J} \left(\cancel{p_{ji}^{(q)} \boldsymbol{x}_{ji}^{(*)}} + p_{ji}^{(q)} \boldsymbol{\varepsilon}_{ji}^{(q)} \right) + p_{Ji}^{(q)} \boldsymbol{x}_{Ji}^{(q)} = \cancel{\sum_{j<J} p_{ji}^{(q-1)} \boldsymbol{x}_{ji}^{(q-1)}} + p_{Ji}^{(q-1)} \boldsymbol{x}_{Ji}^{(q-1)} \quad (1) \\
\Longleftrightarrow \\
\boldsymbol{x}_{Ji}^{(q)} = \underbrace{\frac{p_{Ji}^{(q-1)} \boldsymbol{x}_{Ji}^{(q-1)}}{p_{Ji}^{(q)}}}_{\boldsymbol{x}_{Ji}^{(*)}} - \frac{\sum_{j<J} p_{ji}^{(q)} \boldsymbol{\varepsilon}_{ji}^{(q-1)}}{p_{Ji}^{(q)}}
\end{gathered}
\tag{12}
$$

Ou alors, sans passer par un update temporaire des profils cellulaires, et en partant directement des valeurs de $\boldsymbol{x}_{Ji}^{(q-1)}$ stockées dans la chaine MCMC??, puisque l'étape A de l'ago consiste juste à mettre à jour les ratios cellulaires, et que modifier les expressions purifiées ne changent pas temporaiement la valeur des ratios!! Si tu n'es pas d'accord, que doit-on faire avec ces "valeurs temporaires" de $\boldsymbol{X}'_{Ji}$, les stocker dans la chaine MCMC, mais dans ce cas, on aurait deux fois plus d'échantillons indépendants que pour les ratios cellulaires?

$$\boldsymbol{x}_{Ji}^{(q)} = \boldsymbol{x}'_{Ji} + \frac{\sum_{j<J} x_{ji}^{(q-1)} \left(p_{ji}^{(q-1)} - p_{ji}^{(q)}\right) - p_{ji}^{(q)} \boldsymbol{\varepsilon}_{ji}^{(q)}}{p_{Ji}^{(q)}}$$

"Si on est d'accord sur les équations précédentes, ne resterait plus qu'à déterminer les lois a posteriori de ρ et $\boldsymbol{X}$, à la constante de normalisation près"??

Algorithm 1: Pseudo-code of the generative process for our extended DeCovarT algorithm. Variable q is the sampling running index, with B the number of *burn-in* iterations to be discarded after sampling, and Q the actual length of the resulting Markov chain.

Result: MAP (Maximum a posteriori) estimates of the *purified cellular expression profiles*, $\boldsymbol{X}_{c(i)} \in \mathcal{M}_{\mathbb{R}^+_{G \times J}}$, $c(i) \in \{1, \dots, C\}$ and the *cellular ratios*, $\boldsymbol{p}_i \in]0, 1[^J$, $i \in \{1, \dots, N\}$

Input :

- Prior estimates on the mean expression of the ratios ($[\frac{1}{J}]^J$ on the original, unit simplex space $\boldsymbol{p}$, or $[0]^{J-1}$ on the reparametrised space $\boldsymbol{\rho}$, and unique variance describing the uncertainty on the estimated cellular ratios (σ_0 as the constant diagonal term of the covariance of corresponding prior Normal distribution)
- Prior estimates, possibly phenotype-dependent, of the mean, $\left[\boldsymbol{\mu}_{j(c(i))} \in \mathbb{R}^G\right]$, $j \in \{1, \dots, J\}$ and covariance estimates, $\left[\boldsymbol{\Sigma}_{j(c(i))} \in \mathcal{M}_{\mathbb{R}^{G \times G}}\right]$, $j \in \{1, \dots, J\}$ of each purified cell population.
- In addition, initial estimates of cellular ratios, $\boldsymbol{p}_{0i}$, and purified cell expression profiles, $\boldsymbol{X}_{0i}$, for each individual, such that both the general linear deconvolution constraint, namely $\boldsymbol{y}_i = \boldsymbol{X}_i \times \boldsymbol{p}_i$, and the unit simplex constraint over the ratios are respected. For instance, the estimate returned by R implemented function: lsei::limSolve().

Output: MCMC distributions of the estimated ratios (and possibly of the purified expression profiles, if not too big in terms of memory storage)

for $q \leftarrow 1$ **to** $(B + Q)$ **do**
    for $i = 1 : N$ **do**
        $\boldsymbol{\rho}_i^{(*)} = \boldsymbol{\rho}_i^{(q-1)} + \epsilon$, $\epsilon \sim \mathcal{N}_{J-1}(0, \sigma_0^2 \, \mathrm{I}_{J-1})$
        for $j = 1 : (J - 1)$ **do**
            $u \sim \mathbb{U}(0, 1)$
            if $u < \min\left(1, K_\rho\left(\rho_j^{(q-1)} \rightarrow \rho_j^{(*)}\right)\right)$ **then**
                $\rho_{ji}^{(q)} \leftarrow \rho_{ji}^{(*)}$
            else
                $\rho_{ji}^{(q)} \leftarrow \rho_{ji}^{(q-1)}$
            end
            $\boldsymbol{p}_i^{(q)} = \boldsymbol{\psi}(\boldsymbol{\rho}_i^{(q)})$
                (A) MCMC.$\boldsymbol{p_i} \leftarrow \boldsymbol{p}_i^{(q)}$
            $\boldsymbol{X}_i^{(q-1)} \leftarrow \mathrm{Diag}_J\left(\frac{\boldsymbol{p}_i^{(q-1)}}{\boldsymbol{p}_i^{(q)}}\right) \boldsymbol{X}_i^{(q-1)} \qquad (i)$
        end
        for $j = 1 : (J - 1)$ **do**
            $\boldsymbol{x}_{ji}^{(*)} = \boldsymbol{x}_{ji}^{(q-1)} + \varepsilon_{ji}, \quad \varepsilon_{ji} \sim \mathcal{N}\left(0, \mathrm{Diag}\left(\boldsymbol{\Sigma}_{ji}\right)\right)$
            $u \sim \mathbb{U}(0, 1)$
            if $u < \min\left(1, K_{\boldsymbol{x}}\left(\boldsymbol{x}_{ji}^{(q-1)} \rightarrow \boldsymbol{x}_{ji}^{(*)}\right)\right)$ **then**
                $\boldsymbol{x}_{ji}^{(q)} \leftarrow \boldsymbol{x}_{ji}^{(*)}$
            else
                $\boldsymbol{x}_{ji}^{(q)} \leftarrow \boldsymbol{x}_{ji}^{(q-1)}$
            end
        end
        $\boldsymbol{x}_{iJ}^{(q)} \leftarrow \boldsymbol{x}_{iJ}^{(q-1)} - \frac{\sum_{j<J} p_{ji}^{(q)} \varepsilon_{ji}}{p_{ji}^{(q)}} \qquad (ii)$
        MCMC.$\boldsymbol{X}_i \leftarrow \boldsymbol{X}_i^{(q)}$
    end
end
