MAMBA

The user manual

G. Palma*, L. Cella, S. Monti

* Correspondence to giuseppe.palma@cnr.it.

Contents

Introduction

This manual describes the MAMBA toolbox, designed for the flexible application of statistical voxel-based (VB) analysis in different scenarios in medical imaging and radiation oncology. The MAMBA toolbox is implemented in Matlab. It provides open-source functions to compute VB statistical models of the input data, according to a great variety of regression schemes, and to derive VB maps of the observed significance level, performing a non-parametric permutation inference. The toolbox allows for including VB and global outcomes, as well as an arbitrary amount of VB and global explanatory variables. In addition, the Matlab Parallel Computing Toolbox is exploited to take advantage of the perfect parallelizability of most workloads.

MAMBA is an open-source toolbox, freely available for academic and non-commercial purposes. It is designed to make state-of-the-art VB analysis accessible to research scientists without the programming resources needed to build from scratch their own software solutions. At the same time, the source code is handed out for more experienced users to complement their own tools, also customizing user-defined models.

Users are encouraged to freely adapt MAMBA according to their needs, and assume all responsibility and risk with respect to their use of the toolbox, which is provided "AS IS". In addition, users are welcome to cite the following references, anywhere they use MAMBA:

- G. Palma, S. Monti, and L. Cella. Voxel-based analysis in radiation oncology: A methodological cookbook. *Physica Medica*, 69:192–204, 2020. ISSN 1120-1797. doi: https://doi.org/10.1016/j.ejmp.2019.12.013. URL `https://www.sciencedirect.com/science/article/pii/S1120179719305344`

- G. Palma, L. Cella, and S. Monti. Mamba – multi-paradigm voxel-based analysis: a computational cookbot. *Submitted*, 2022

In chapter 1, we provide an extensive reference manual for the main and only functions that need to be directly called by the users of the toolbox. In chapter 2, we illustrate the use of MAMBA by means of several examples that the user can fully work out on a synthetic dataset, as well as by showing the toolbox configurations that led to clinical results previously published in the literature.

Getting started

This section describes how to install MAMBA and run a quick example.

1. Download the latest version of MAMBA from here;

2. Extract the content of `MAMBA-main.zip` to a `folder/` of your choice;

3. Open a Matlab session;
4. Add the following folders to the path of Matlab:
   - `folder/`;
   - `folder/Engine/`;
   - `folder/External/` and its subfolders;
5. Change the Matlab current folder to `folder/WorkedExamples/`;
6. Execute the following commands:

```
synthetic_cohort('N_pat', 100, 'image_size', [64 64 32], 'time_images', false)
test2
```

Chapter 1

Reference manual

In the following, we provide a detailed description of the interfaces of the four main MAMBA (Multi-pAradigM voxel-Based Analysis) functions. We will take advantage of the specifications on the mandatory arguments and options to outline the toolbox functionality.

Throughout the toolbox, the additional options of the MAMBA functions are accepted as long as they are compliant to the schemes handled by the parser `opt_pars` provided in the subfolder `Engine/`. The `opt_pars` function accepts, in arbitrary order:

- pairs of name-value arguments;
- scalar structures, from whose fields name-value pairs are extracted;
- paths to MAT-files, from whose variables name-value pairs are extracted.

The `opt_pars` function parses sequentially the input, and, if an option with the same name was already defined, its value is overwritten. The output is a list of options stored as scalar structure.

1.1 `image_read`

The function `image_read` has the following prototype:

```
➢ [img_sn, CCS, img_pre] = image_read(options)
```

The first output, **`img_sn`**, is a table of the spatially-normalized VB variables for the patients in the cohort, while **`CCS`** is a structure containing the info related to the template, as well as a field `.config` with the processed options (*i.e.*, the default options overwritten by the possible parameters explicitly assigned by the user). If the function is called with the optional third output argument, `image_read` reads and stores in **`img_pre`** the cohort VB variables in their native spaces. Both `img_sn` and `img_pre` store each VB variable of each patient not as a numeric or logical array, but as a cell containing the array itself: this guarantees higher flexibility in order to handle native images of potentially different sizes and the possible presence of a timetable of VB variables in each patient (see below the option `time_images`).

No mandatory input arguments are required by `image_read`, whose behavior is defined by a list of optional parameters.

The main additional option is **`path_proc`**, which specifies the folder containing both the template anatomy (it is expected to be found in the file `template.mat`) and a subfolder

cohort/, where the spatially normalized VB variables of each patient are stored in separate MAT-files. The template file is expected to contain a non-empty subset of the variables listed in template_images, masks and images – which define the expected common size of VB variables throughout the cohort – as well as a variable vox specifying the voxel size in mm. By default, path_proc is set to the current folder.

The options **ID_include** and **ID_exclude** specify the patient identifiers (*i.e.*, the names of the MAT-files in cohort/) to be included or excluded, respectively; the identifiers will constitute the row names of the output tables. Both options can be defined as:

- a character vector specifying the file names, possibly including the * wildcard (*e.g.*, 'Pat_*');
- a numeric vector (*e.g.*, [3 11 21:37]), which are converted to character arrays;
- a cell array containing arbitrary combinations of the above vectors (*e.g.*, {1:10 'Pat_*' 'HC_*'}).

By default, all MAT-files in cohort/ are first included (*i.e.*, ID_include is set to '*') and no one is later excluded (*i.e.*, ID_exclude is set to '').

At this point, the user may be interested in extracting just the list of patient identifiers corresponding to the above options, without actually reading the spatially normalized arrays in the cohort. If it is the case, the option **join_images** should be set to false, and:

- img_sn and img_pre are two empty n-by-0 tables with the row names given by the selected patient identifiers;
- the structure CCS only contains the field .config with the list of parsed options.

If join_images is set to true (default), the readout of info referred to template and patients is carried out.

The previously mentioned option **template_images** specifies the names of the images that have to be read in the file template.mat and saved as fields of CCS (by default, it is set to 'CT').

Similarly, the option **masks** specifies the names of the anatomical structures (defined only once in the file template.mat) that will be read and saved as fields of CCS (*e.g.*, 'brain' or {'heart' 'lungs'}; default: {}). Their union will define the global region of interest for the subsequent VB analysis (see later on in subsection on image_clin_merge the field CCS.mfix); if no mask is found within template.mat, the region of interest will be extended to the entire field of view.

The option **images** specifies the names of the images that have to be searched in the template and the cohort files (*e.g.*, 'CT' or {'MPRAGE' 'QSM' 'PET'}; the default is 'dose' to account for most VB analyses in radiation oncology). They will be saved as fields of CCS (as for the template anatomy) and as columns of the output tables img_sn and img_pre (as for the spatially normalized and native images in the structures sn and pre, respectively, in each MAT-file in cohort/). Missing images in a given patient will be filled with an array of all Not a Number (NaN) values of the same size as the template VB variables.

The option **mobile_masks** identifies the names of patient-specific masks that have to be searched in the cohort files (*e.g.*, 'lesion' or 'GTV'). Typically, such masks indicate the possible presence of pathological lesions or image artifacts, and the user may want to exclude the voxels in the mask for the specific patient in the VB analysis. Accordingly, the value of this option will lead to a patient-specific mask (see later on in subsection on image_clin_merge

the variable mpat of the output table tab) corresponding to the relative complement of the union of the patient-specific masks with respect to CCS.mfix. Missing masks in a given patient will be filled with an array of logical zeros of the same size as the template VB variables. By default, mobile_masks is set to {}.

In our experience, the image datasets collected in some clinical scenarios could include for each patient a time series of VB variables (*e.g.*, diagnostic images acquired at several time-points or radiotherapy (RT) dose maps delivered in sequential treatments or re-irradiations), in which the time variable could play a relevant role. In such cases, we encourage the use of the timetable data type, so that each VB variable, along with possible header attributes (such as the imaging modality, treatment type, etc.), is easily associated with the time. To properly handle the presence of a timetable within the structures sn and pre in each MAT-file in cohort/, the option **time_images** allows for specifying the name of the imaging timetable to be read (*e.g.*, 'tt_imaging'; by default, it is false, corresponding to no need to read any timetable). If a non-empty character vector is assigned to time_images and the timetable is actually found in a given patient file, image_read expects to find a variable Images in the timetable (see below the behaviors determined by options sigma and subsample_grain); if the timetable specified by a non-empty character vector time_images is missing in in a given patient, the corresponding cells in img_sn and img_pre will be filled with an empty timetable.

The option **sigma** defines the standard deviation of a Gaussian kernel used to filter the images (specified by the options images and time_images) read from the patients' files. It is expressed in mm as a scalar (for isotropic smoothing) or vector (for anisotropic smoothing along the coordinate axes) length (*e.g.*, 0 – corresponding to no smoothing – or [2.5 2.5 5]; by default, it is set to 5). Optionally, it is possible to also smooth the patient-specific masks (specified by the option mobile_masks) through the logical option **smooth_mobile_masks** (default: true); if smooth_mobile_masks is true, the patient-specific masks are converted to double precision.

The option **subsample_grain** can be used to perform a subsampling of the template and patient VB variables (specified by the options template_images, images, masks, mobile_masks and time_images) by averaging blocks of voxels. It can be a scalar (for cubes of voxels) or a vector (for parallelepipeds of voxels) with positive integer values (*e.g.*, 3 or [5 5 2]; by default, it is set to 1 – corresponding to no subsampling). Please note that logical VB variables (typically those specified by masks and non-smoothed mobile_masks) are first cast to double (with the customary associations false-0 and true-1), then averaged and finally converted back to logical by rounding to the nearest integer.

Finally, the option **progress_bar** specifies whether the progress of the workload should be monitored or not by progress bars (default: true).

1.2 image_clin_merge

The function image_clin_merge has two alternative prototypes.

The first prototype is:

```
➢ [tab, CCS] = image_clin_merge(img, CCS, options)
```

The first output, **tab**, is a table joining VB variables in a table **img** obtained by image_read with demographic and clinical variables contained in a separate table. The output **CCS** extends the input **CCS** by creating a new field .mfix that defines the region of interest for the VB analysis, and by adding in the existing field .config the parsed options that are specific to image_clin_merge. Besides the mandatory input arguments img and CCS, a set of

optional parameters specify, as in image_read, the way the tables are joined to produce the output tab. Please note that, if image_clin_merge is called through this prototype, possible optional parameters related to image_read will not have any pre-processing effect on the input img and CCS, with the exceptions of the options ID_include, ID_exclude, masks, mobile_masks and progress_bar. The user is, indeed, allowed to further refine the list of patients already stored in the table img (through ID_include and ID_exclude). Similarly, the user may let image_clin_merge work on the subsets of the CCS.config.masks and CCS.config.mobile_masks defined through masks and mobile_masks, respectively.

The second prototype of the function is:

```
➢ [tab, CCS] = image_clin_merge(options)
```

This prototype of the function has no mandatory input arguments, and it takes care of calling the function image_read to supply the missing img and CCS:

```
[img, CCS] = image_read(options)
```

In this case, the options of image_clin_merge should include the parameters needed to configure how image_read is expected to return its first two outputs.

In the following, we will describe the optional parameters of image_clin_merge that are not shared with image_read.

The main option is **clin_table**, which can be used to provide the table T of demographic and clinical data or the path to a MAT-file containing it (default: 'clin.mat'). The row names of the table provided by clin_table are expected to be the patient identifiers, as the join between T and img uses the vectors of row names of the two tables as key variables. Clearly, this means that a patient present in both T and img must share the same identifier.

The option **include_all_patients** can be a logical flag specifying whether the function image_clin_merge should process – within the table tab – all the patients found either in T or img (outer join; true), or just the patients present both in T and img (inner join; false – default). Setting include_all_patients to true can be useful for quickly processing derived variables (see below the option der_vars) and imputation (option clin_vars) on table T with join_images set to false.

Although in most cases the user may not need to be aware of this technicality, it could be noteworthy that the output tab is compliant with the format of img_sn and img_pre. Namely, once the join is performed, the resulting table is preliminarily parsed to ensure that each variable of each patient is a scalar cell (otherwise, the content of the variable in each row is encapsulated in a cell), and to replace variables in patients that are NaN or Not-a-Time (NaT) with empty double or datetime arrays, respectively; they will be interpreted as missing values in the subsequent processing.

In order to allow for the definition of additional variables that can be derived from the variables of the joined table, image_clin_merge accepts the option **der_vars**. It is expected to be a cell array in which each cell defines a derived variable in the form of {var_name operation}. var_name is a character vector specifying the name of the derived variable being defined, while operation is a cell array of arbitrarily nested operations on existing table variables expressed in Polish notation (*i.e.*, the first cell of operation indicates the operator, while the following cells contain the operands). The value of a variable within the table can be accessed by providing as operand a character vector variablename containing the variable name; the value of an arbitrary field .fieldname within CCS can also be accessed with the character vector ['#' fieldname]. The operators are specified as character vectors in one of the following forms:

- element-wise logical operators:

  - 'not' or '~' for NOT (complement; *e.g.*, in the der_vars cell {'non_smoker' {'~' 'smoker'}});
  - 'and' or '&' for AND (intersection; *e.g.*, {'lucky_guy' {'and' 'smoker' 'healthy'}});
  - 'or' or '|' for OR (union; *e.g.*, {'lungs' {'or' 'left_lung' 'right_lung'}});
  - 'xor' or 'symmdiff' for exclusive-OR (set symmetric difference; *e.g.*, {'registration_mismatch' {'xor' 'heart' '#heart'}});
  - 'difference': for set difference (*e.g.*, {'brain_parenchyma' {'difference' 'ICV' 'CSF'}});

- element-wise comparison:

  - 'eq', '==' or '=' for equality (*e.g.*, {'registration_match' {'=' 'heart' '#heart'}});
  - 'ne', '~=' or '<>' for inequality;
  - 'lt' or '<' ('le' or '<=') for less-than (or equal to) (*e.g.*, {'low_act' {'<' 'SUV' 5}});
  - 'gt' or '>' ('ge' or '>=') for greater-than (or equal to) (*e.g.*, {'fat' {'&' {'>=' 'CT' -120} {'<=' 'CT' -90}}});

- element-wise arithmetic operators: 'plus' or '+' (addition or unary plus); 'minus' or '-' (subtraction or unary minus); 'times' or '.*' (multiplication); 'rdivide' or './' (right division); 'ldivide' or '.\' (left division); 'power' or '.^' (power);

- matrix operators: 'mtimes' or '*' (multiplication); 'mrdivide' or '/' (right division); 'mldivide' or '\' (left division); 'mpower' or '^' (power);

- 'first' returns the first non-empty value among the operands; useful for handling variables that might have different names in patients from different sub-cohorts (*e.g.*, {'lesion' {'first' 'tumor' {'difference' 'brain' 'NAWM'} 'GTV'}});

- name of an arbitrary function in the path (*e.g.*, {'R2star' {'./' {'log' {'./' {'denoiseImage' 'Echo_1' net} {'denoiseImage' 'Echo_2' net}}} {'-' 'TE_2' 'TE_1'}}}; or {'Dice_index' {'/' {'sum' {'&' 'lungs' '#lungs'} 'all'} .5 {'+' {'sum' 'lungs' 'all'} {'sum' '#lungs' 'all'}}}});

As exploited in the above configuration of Dice_index with the '/' operator, binary element-wise logical and arithmetic operators, as well as matrix operators, allow for a simplified syntax in expressions containing two or more occurrences of the same operator in a row: for instance, {'ICV' {'|' {'|' 'CSF' 'GM'} 'WM'}} can be written equivalently as {'ICV' {'|' 'CSF' 'GM' 'WM'}}.

As shown in the above examples, if an operand is the result of a nested operation, this must be encapsulated in its own cell array, because of the widespread diffusion in Matlab of functions of indefinite arity. Please note that it is possible to define further derived variables according to operations on derived variables defined in previous cells of der_vars. By default, der_vars is set to {}.

For some applications, the final setting of der_vars can be quite complex and involve several levels of nested operations; accordingly, the saved value of this option might be hardly inspected in the standard Matlab Workspace Browser. In this context, the function Engine/arr2text.m,

which converts back any array to its Matlab expression, can enable an easier browsing of the `der_vars` definition.

The option **`select_patients`** can be a character vector specifying the name of a logical variable, either original or derived, indicating what rows (*i.e.*, patients' records) should be kept in the subsequent processing (default: `''`, corresponding to all rows).

The option **`outcome`** specifies the names of dependent variables involved in the VB statistical model (*e.g.*, `'pneumonitis'` or `{'time-to-event' 'censoring'}`; default: `{}`) that should be used for a preliminary selection among the independent variables listed in option **`clin_vars`** (*e.g.*, `{'performance_status' 'smoker' 'tumor_volume'}`; default: `{}`). The selected variables will be promoted to the output cell array `CCS.vars`. If `outcome` is a single global variable, the VB variables included in `clin_vars` are automatically selected, while the global variables in `clin_vars` will be selected according to an elastic net regularization for the associated GLM of the `outcome`. If `outcome` is not a single global variable, no variable among `clin_vars` will be automatically promoted to `CCS.vars`.

Importantly, each element of `clin_vars` can be either a character vector (as in the example above) or a cell array (*e.g.*, the slightly different `{{'performance_status' 'smoker' 'tumor_volume'} 'smoker' {'age'}}`). For those elements of `clin_vars` that are character vectors (in the above example, the variable `smoker`) no imputation is performed; otherwise, cell array elements (`performance_status` and `age`) are used to define the imputation of the first (global) variable according to a GLM including the remaining variables. This means that missing values of `performance_status` will be imputed according to a GLM including `smoker` and `tumor_volume`. Missing elements of variables in single element cell arrays (`age`) will be imputed according to the mean of the variables themselves.

Noteworthy, each variable name in `outcome` and `clin_vars` can refer to both original or derived variables.

The option **`MVA_opt`** can be a list of options that allow for a flexible configuration of the variable selection (*e.g.*, `{'lasso_distr' 'poisson' 'lasso_opt' {'Link' 'probit' 'Alpha' 0.9}}`; default: `{}`). As shown in the above example, they include the parameter `lasso_distr` (by default, it is estimated from the values assumed by the `outcome` variable as `'binomial'` – if all the value are logical or double `0` or `1` – or `'normal'`, otherwise) and the nested list of further options `lasso_opt`, which can only include options of the built-in Matlab function `lassoglm`.

The option **`vars`** can be a character vector or a cell array of character vectors specifying the table variables that are forced within the output cell array `CCS.vars`. By default, it contains the variable names of the table `img`. Variables in `vars` will therefore enter the output cell array `CCS.vars` along with the variables selected among `clin_vars`. Please note that the possible `outcome` is automatically removed from `vars`. To perform the imputation on a variable that needs to be forced within the final selection, please define its imputation model within `clin_vars` and, then, add its name within `vars`.

Unless `include_all_patients` is `true`, the output table `tab` will be trimmed to include only rows without empty `CCS.vars` or `outcome` columns.

The option **`refine_mask`** can contain a function handle or a name of a function `f(tab[, CCS])` with one or two input arguments, which returns a logical mask `m` for refining `CCS.mfix` as `CCS.mfix = CCS.mfix & m`. It might prove useful, for instance, to exclude from `CCS.mfix` regions with low image signal (MRI, CT, etc.) or dose values (RT – *e.g.*, `@(tab) mean(cat(4, tab.dose{:}), 4) > 5`, which excludes voxels with mean dose values lower or equal to 5 Gy). By default, it is set to `''`, *i.e.*, no refinement is performed.

Once the final `mfix` mask is obtained, a variable `mmob` is computed for each patient as the union (if `smooth_mobile_masks` is `false`) or the sum (if `smooth_mobile_masks` is `true`)

of the patient-specific masks, and a further variable `mpat` is obtained patient-wise as the (set or arithmetic) difference between `mfix` and `mmob`.

Finally, the option **`keep_all_columns`** contains a flag for keeping all columns in the output `tab`. If it is set to `false` (default), only `mpat`, processed `CCS.vars` and `outcome` are exported.

1.3 VBmodel

The function `VBmodel` has two alternative prototypes.

The first prototype is:

➢ `[stats_array, codec] = VBmodel(data, outcome_names, options)`

The first output, **`stats_array`**, is a #stat-by-#voxel numeric array, which contains, along the first dimension of `stats_array`, a linearized list of the voxel-wise model statistics of interest. Here, #voxel refers to the number of voxels actually flagged in the option `mfix` (see the description below). The second output, **`codec`**, is a cell array in the form of `{{stat_name_1 stat_size_1}; ...; {stat_name_n stat_size_n}}` with the info needed to reconstruct from `stats_array` the structure of the model statistics; it might be typically useful in case of multiple output statistics of interest (refer to the option `outs` described below).

The first mandatory input, **`data`**, contains all the data required to build the model, namely both independent and dependent variables and the optional patient-specific masks (meant to be defined as `mpat`). The variable `data` may be provided in one of the following alternative forms:

- a table compliant with the standard defined for the output `tab` of the function `image_clin_merge`;
- a $(2+N)$D array of VB variables in the shape of #patients-by-#dim1-by-...-by-#dim$N$-by-#variables; in this case, possible global variables must be replicated for each voxel in the central N array dimensions, while one cannot supply patient-specific masks; in addition, to build a model based on global variables only, the user has to keep the singleton in the second dimension (it is not the main reason that MAMBA has been developed for, but this specific also holds in the more relevant case of the second prototype);
- a structure with a field `.variables`, in the form of the previously described $(2+N)$D array, and an optional field `.msk`, in the form of a $(1+N)$D array (#patients-by-#dim1-by-...-by-#dim$N$) of patient-specific masks.

Each of the VB variables (variables in table; variables and masks in array/structure) can appear independently linearized in the spatial dimensions (*i.e.*, #patients-by-(#dim1$\cdot\ldots\cdot$#dim$N$)).

The second mandatory input, **`outcome_names`**, specifies what are the dependent variables in `data`. It can be:

- a character vector specifying a single variable name (*e.g.*, `'Disease'`; in this case, `data` must be a table);
- a cell array containing positive integers (*e.g.*, `{3 5}` or, equivalently, `{[3 5]}`), which specifies the variable indices, or arbitrary combinations of indices and variable names (*e.g.*, `{'Time2Event' 5}`; in this case, `data` must be a table).

The second prototype of `VBmodel` is:

➢ `[stats_array, codec] = VBmodel(EVs, outcomes, options)`

The outputs maintain the same structure described for the first prototype; on the other side, **EVs** and **outcomes** are both supplied in the forms accepted for the input `data` as previously described for the first prototype, provided that:

1. The optional patient-specific masks should be passed through `EVs` (possible masks within `outcomes` will be neglected);
2. If both `EVs` and `outcomes` are tables with row names, they will be preliminarily inner-joined using the row names as the key variable; otherwise, it is expected that the order of corresponding rows is the same in both the inputs;
3. If `EVs` or `outcomes` are provided in the form of arrays or structures, and at least one of them contains only global variables, it is not necessary to replicate the values to match the spatial dimensions of the input containing VB variables (*e.g.*, for a VB Cox regression with global outcome, `EVs` might be a 3D #patients-by-#voxels-by-#variables array, and `outcomes` might be a #patients-by-1-by-2 array given by `cat(3, Time2Event, censoring)`).

Both prototypes accept roughly the same options, with an exception that will be described below.

The main option is **`model`**, which specifies the statistical model to be trained voxelwise. It is usually a case insensitive character vector (namely, `'glm'`, `'cox'` or `'anova'`); by default, it is set to `'glm'` if there is only one outcome variable, or `'cox'` otherwise (the second dependent variable is interpreted as censoring flag). As an alternative, `model` can be a function handle (*e.g.*, `@tstat2`) that points a generic function with the following prototype:

```
➢ stats = model_prototype(single_EVs, single_outcomes, single_masks, options)
```

This function builds a single-voxel model; the output `stats` is a structure containing the model statistics; `single_EVs`, `single_outcomes` (both in the form of #patients-by-#variables arrays) and `single_masks` (in the form of #patients vector) expose the EVs, outcomes and patient-specific masks in the given voxel; and `options` configure the details of the model. A didactic example of this type of function is provided by the function `tstat2` in the subfolder `Engine/`: it develops a two-sample normal model and returns a structure with the fields `.delta` (containing the difference of the sample means) and `.t` (containing the t statistic); the option `tail` (either `'left'` or `'right'`) specifies the sign of the differences (not to be confused with the way more important option `tails` of the function `perm_test`).

Please note that all the models handled by `VBmodel` natively support the cast to logical of patient-specific masks: for each voxel, the model is fitted only on data from patients whose local `mpat` is not `0` (either logical or double). In addition, both GLM and Cox regression properly handle continuous patient-specific masks (see the comments on the option `smooth_mobile_masks` in the subsections on `image_read` and `image_clin_merge`), by modulating the weights of the associated observations.

Also, it is noteworthy that all the models natively provided with MAMBA accept a single outcome variable, with the only exception of the Cox regression, which might accept up to two outcome variables. Namely, if a single outcome variable is provided, it is interpreted as the time-to-event of uncensored observations; otherwise, the first variable represents the event time of right-censored time-to-event data, while the second (binary) variable indicates whether an observation was right-censored.

If `model` is set to `'glm'`, the option **`distr`** specifies the distribution of the outcome; its behavior, including the choice of the default, follows what already described for `lasso_distr` in the subsection on `image_clin_merge`.

In general, however, the option **fit_pars** can be set to a cell array to specify additional options of the model function (*e.g.*, `{'tail' 'left'}` for `model` set to `@tstat2`). In this context, it is worthwhile to specify that the built-in functions underlying `'glm'`, `'cox'` and `'anova'` are `glmfit`, `coxphfit` and `anovan`, respectively; this means that the elements of `fit_pars` should be valid name-value pair arguments of the above functions (*e.g.*, `{'Baseline' 0 'Ties' 'efron'}` is a valid `fit_pars` value if `model` is set to `'cox'`, but not if it is set to `'glm'`). By default, `fit_pars` is set to `{}`.

The option **outs** is a character vector or a cell array of character vectors that is used to select the statistics of interest (fields of the model structure). The user should refer to the description of the specific model function for retrieving the details of the field names of the model structure (*e.g.*, if `model` is set to `'glm'`, `outs` could be set to `{'beta' 's' 'se' 'coeffcorr' 'estdisp' 'dfe'}`). By default, `outs` is set to `'z'` if `model` is set to `'cox'`, `'f'` if `model` is set to `'anova'`, and `'t'` otherwise. Please note that, as the built-in `anovan` function does not natively return a structure of the model statistics that includes F-statistic, a structure with the single field `.f` is purposely created within `VBmodel`.

The option **vars** specifies what variables in `data` or in `EVs` are included as EVs in the model. It can be provided in the same form of `outcome_names`, or, in addition, as a numeric array of positive integers (*e.g.*, `[4:3:13 2]`). By default, it is set to `':'` (meaning all the variables in `data` or `EVs`, except `mpat` and, in the case of the first prototype, `outcome_names`). Ideally, it would be meant to coincide with `CCS.vars` returned by `image_clin_merge`. It could be useful if `keep_all_columns` in `image_clin_merge` is set to `true`, or if the user wants to perform different analyses with subsets of EVs without having to rerun `image_clin_merge`.

The option **EVOIs** specifies the EVs for which the statistics of interest should be returned. It behaves following the form of `vars` (*e.g.*, it could be set to `'age'`, `{'dose' 4 'CT'}`, `3` or `':'`). By default, it is set to `{}`: it corresponds to selecting all the terms in the design matrix, including all the EVs and a possible constant term. Please note that, if `model` is set to `'anova'`, a single EV must be selected by `EVOIs`.

The option **mfix** specifies the region of interest for the VB analysis in the form of a logical mask, as described in the subsections on `image_read` and `image_clin_merge` (indeed, ideally it would be meant to coincide with `CCS.mfix`). By default, it is set to the scalar `true` (corresponding to the selection of the entire parallelepiped spanned by the VB variables of the model, including possible patient-specific masks). If it is not a scalar, it can be a vector with the same number of elements of the VB variables; otherwise, its size must coincide with the size of possible non-linearized VB variables, or, if all the VB variables have been linearized, it is required that it contains the same number of elements of the VB variables.

If either `data` or `EVs` is not a table, the option **dim_vars** can be used to specify the dimension (a scalar integer) of `data`/`EVs` in which different variables are indexed. MAMBA identifies the default of this option exploiting all the info regarding the sizes of `mfix`, VB variables and possible patient-specific masks in order to guess how the user meant to supply the variables. In general, it is necessary to specify a value for `dim_vars` if `mfix` is a scalar and there is a single spatial variable, which is not linearized; otherwise, the spatial variables might be interpreted as a list of #dim$N$ $(N-1)$D variables (*e.g.*, if `data` is a #patients-by-#dim1-by-...-by-#dim$N$ $(N+1)$D array, please set `dim_vars` to $N+2$).

If `VBmodel` is called with the second prototype, the option **dim_vars_outcomes** allows for controlling in a similar way the interpretation of the input `outcomes`.

The option **VBmodel_workers** allows for configuring the parallelization strategy, and can be set to a nonnegative integer that specifies the maximum number of workers from the parallel pool to use in computing the VB models. By default, it is set to `inf`. Please note that, to run `VBmodel` without parallelization, `VBmodel_workers` should be set to logical or double `0`, since

logical or double `1` are interpreted as `inf`.

Finally, the option **`progress_bar`** specifies whether the progress of the workload should be monitored or not by progress bars (default: `true`).

1.4 `perm_test`

`perm_test` shares the same input signatures of `VBmodel`. Accordingly, it has the following two prototypes:

➢ `p = perm_test(data, outcome_names, options)`

➢ `p = perm_test(EVs, outcomes, options)`

The output **`p`** is a #tails-by-#voxels numeric array containing along each row the linearized *p*-map for a tailed test (see the option `tails` below) of the VB model associated with a specific EV of interest, specified by the `VBmodel` option `EVOIs`. To derive the *p*-map, `perm_test` repeatedly calls `VBmodel` on a configurable shuffling of the model variables (see the options `perm_EVs` and `perm_outcomes` below), and rely on the setting of the `VBmodel` option `outs` to estimate the VB map of the actual statistic, as well as the distribution of maxT under the null hypothesis.

As the options of `perm_test` include all the options of `VBmodel`, in the following we will describe the options of `perm_test` that are not shared with `VBmodel`.

The option **`N`** defines the number of random permutations (*i.e.*, shufflings) that will be computed to estimate the null distribution of maxT (default: `1e4`).

The option **`rng`** specifies the seed for the Mersenne Twister pseudorandom number generator (please refer to the documentation of the built-in `RandStream` function). By default, it is set to `'shuffle'`, which ensures that starting a new Matlab session will produce a different stream of shufflings.

The pair of options **`perm_EVs`** and **`perm_outcomes`** are used to configure the shuffling strategy of the permutation test. They are provided with the same form of the `VBmodel` options `vars` and `EVOIs` (*e.g.*, in the "Shuffle Y" scheme [Manly, 1986, Kennedy, 1995, Winkler et al., 2014], `perm_EVs` and `perm_outcomes` could be set to `[]` and `':'`, respectively, in order to shuffle the entire block of the outcome variables, while leaving fixed the EVs). By default, `perm_EVs` and `perm_outcomes` are set to the value of `EVOIs` and `[]`, respectively, which correspond to the "Shuffle Z" scheme [Draper and Stoneman, 1966, Kennedy, 1995, Winkler et al., 2014]: only the regressor of interest is shuffled, while both the outcomes and the nuisance variables are left fixed. Please note that possible patient-specific masks are not shuffled. Therefore, if `mpat` does not coincide with `mfix` for some patient, the *p*-map obtained with a given setting of `perm_EVs` and `perm_outcomes` (*e.g.*, `[]` and `':'`, respectively) will be different from the *p*-map obtained by complementing the value of each option (in the previous example, `':'` and `[]`, respectively), even in the limit of a large `N`. It is also worth remembering that, in general, the permutations are not involutions; hence, even if `mpat` coincides with `mfix` for all the patients and `rng` is set to ensure repeatable streams of permutations among different runs of `perm_test`, the two *p*-maps obtained by the two complementary settings of `perm_EVs` and `perm_outcomes` will be likely different, unless the `perm_test` runs out *all* the possible permutations on the set of patients.

The option **`tails`** can be a numeric array with elements $\in \{1, -1, 2\}$ (*e.g.*, `[2 -1]`; default: `1`). It specifies what test types are required (right-, left- and two-sided, respectively); two-sided tests are performed focusing on the absolute value of the considered statistic. The number of elements in `tails` determines the number or rows of the output `p`.

The option **tfce** is a logical flag that can enable the TFCE filtering of the statistic maps [Smith and Nichols, 2009]; by default, it is set to `true`. To be applied, it requires the `bwconncomp` function of the Image Processing Toolbox. The TFCE filter can be configured through the options **E** (the TFCE support exponent; default: `0.5`), **H** (the TFCE level exponent; default: `2`), **C** (the pixel/voxel connectivity; default: $3^N - 1$) and **res** (number of linearly spaced integration points; default: `100`).

The option **subsampling** can be used to enable a scheme for speeding up the execution of `perm_test` at a usually negligible cost in terms of approximation error. In particular, while the statistic map on the unpermuted data is computed on the voxels selected by `mfix` on the full grid, the maxT for the `N` shufflings is computed on a subset of `mfix`, whose points are subsampled at regular intervals along the coordinate axes, as specified by the `subsampling` value. It can be a scalar (for cubes of voxels) or a vector (for parallelepipeds of voxels) with positive integer values (*e.g.*, `3` or `[5 5 2]`; by default, it is set to `1` – corresponding to no subsampling). To compensate the bias associated with the extraction of the maximum of a statistic on a subset of its domain, the estimated values of maxT on the shufflings are multiplied by the ratio between the unshuffled maxT on the full grid and the unshuffled maxT on the subsampled grid. For large `N` (namely, `N` $\gg$ `prod(subsampling)`), the speedup tends to `prod(subsampling)`. As a rule of thumb, the p-maps obtained with a given `subsampling` can be considered accurate as long as the supports of the Fourier transforms of the model VB variables have coordinate sizes lower than `1./subsampling` (setting `sigma` > 0 in `image_read` helps to meet this condition).

The option **perm_test_workers** is similar to `VBmodel_workers`, and allows for configuring the parallelization strategy in the maxT computation on the shufflings. Setting `perm_test_workers` to a positive value allows for splitting the workload of the permutations on different workers, each of which will take care of the computation of the statistic on the entire set of voxels selected by `mfix` (regardless of the setting of `VBmodel_workers`); otherwise, the workload of each permutation on the voxels will be splitted among the workers according to the setting of `VBmodel_workers`. While it is usually advisable to run the permutation (*i.e.*, the outermost) loop in parallel to reduce the parallel overhead, for very large datasets and modest subsamplings this might lead to a prohibitive memory load (all the model variables, evaluated on the subsampled `mfix` for the entire cohort, need to be replicated for each worker) and to a burdensome memory swap: in those cases, setting `perm_test_workers` to `false` and `VBmodel_workers` to `inf` sensibly reduces the memory load. Please note that `perm_test_workers` does not affect the computation of the unpermuted statistic maps. By default, `perm_test_workers` is set to `inf`.

Finally, the option **dump2file** can be used to dynamically dump on the disk the maxT of each permutation as soon as it is computed. This can be useful if there is a non-negligible probability of unexpected stop of the run; common causes are unexpected server shutdown during a power cut or system reboot due to overheated processor (which might be not unlikely during an execution of `perm_test` with a `perm_test_workers`-to-cores ratio close to or higher than 1). By setting a non-empty character vector, `perm_test` checks if a folder named as the value of `dump2file` exists. If it does not exist, it is created, and each worker dynamically stores the freshly computed maxT in a worker-specific MAT-file therein. If the execution of `perm_test` comes to a regular end, the folder is automatically removed; otherwise, the folder and the MAT-files it contains are ready to be searched in as a starting point when the execution of `perm_test` is resumed (in this case, only the missing permutations are worked out). It is worth mentioning that the strategy implemented for the option `dump2file` is compliant with both shuffled (`'shuffle'`) and reproducible (*e.g.*, positive integers) settings of the option `rng` for the stream of permutations. Therefore, if the user is going to resume an aborted run, `perm_test` handles only the originally planned permutations that were not worked out, even for fixed random seeds.

If `dump2file` is set to true or to `''` (default), a folder with a random name (shown in the

command window) will be created in the current folder. To suppress the storage on the disk, set `dump2file` to `false`.

1.5 Function prototypes

1.5.1 image_read

➢ `[img_sn, CCS, img_pre] = image_read(options)`

Option list in alphabetical order:

- `ID_exclude`
- `ID_include`
- `images`
- `join_images`
- `masks`
- `mobile_masks`
- `path_proc`
- `progress_bar`
- `sigma`
- `smooth_mobile_masks`
- `subsample_grain`
- `template_images`
- `time_images`

1.5.2 image_clin_merge

➢ `[tab, CCS] = image_clin_merge(img, CCS, options)`

➢ `[tab, CCS] = image_clin_merge(options)`

Option list in alphabetical order:

- `clin_table`
- `clin_vars`
- `der_vars`
- `ID_exclude`
- `ID_include`
- `images` (no effect on 1st prototype)

- include_all_patients
- join_images (no effect on 1st prototype)
- keep_all_columns
- masks
- mobile_masks
- MVA_opt
- outcome
- path_proc (no effect on 1st prototype)
- progress_bar
- refine_mask
- select_patients
- sigma (no effect on 1st prototype)
- smooth_mobile_masks (no effect on 1st prototype)
- subsample_grain (no effect on 1st prototype)
- template_images (no effect on 1st prototype)
- time_images (no effect on 1st prototype)
- vars

1.5.3 VBmodel

➢ `[stats_array, codec] = VBmodel(data, outcome_names, options)`

➢ `[stats_array, codec] = VBmodel(EVs, outcomes, options)`

Option list in alphabetical order:

- dim_vars
- dim_vars_outcomes (no effect on 1st prototype)
- distr
- EVOIs
- fit_pars
- mfix
- model
- outs
- progress_bar
- vars
- VBmodel_workers

1.5.4 perm_test

➢ `p = perm_test(data, outcome_names, options)`

➢ `p = perm_test(EVs, outcomes, options)`

Option list in alphabetical order:

- `C`
- `dim_vars`
- `dim_vars_outcomes` (no effect on 1st prototype)
- `distr`
- `dump2file`
- `E`
- `EVOIs`
- `fit_pars`
- `H`
- `mfix`
- `model`
- `N`
- `outs`
- `perm_EVs`
- `perm_outcomes`
- `perm_test_workers`
- `progress_bar`
- `res`
- `rng`
- `subsampling`
- `tails`
- `tfce`
- `vars`
- `VBmodel_workers`

Chapter 2

Worked examples and clinical results

In the following, we will illustrate the use of MAMBA by means of several examples that the user can fully work out on a synthetic dataset, as well as by showing the toolbox configurations that led to clinical results previously published in the literature.

While MAMBA is intended for general-purpose VB analysis, the examples presented in the following are drawn from scenarios related to radiation oncology, since the application of VB methods in this field is relatively recent compared to the experience of medical imaging and, specifically, neuroimaging. Therefore, researchers interested in exploring radiobiology patterns through VB analysis are likely to take particular advantage of dedicated examples.

2.1 Standalone examples on a synthetic dataset

2.1.1 Generation of the synthetic dataset

The function `synthetic_cohort` in `WorkedExamples/` can be run to generate a synthetic dataset of RT patients that mimics a typical radiation oncology scenario. It can be configured through the following options:

- `clin_table` specifies the path to the output MAT-file with the clinical table (default: `'clin.mat'`);
- `path_proc` specifies the output folder containing the MAT-files with the spatially normalized VB variables of each patient (default: current folder);
- `N_pat` specifies the total number of patients to be simulated (default: `500`);
- `image_size` specifies the size of the VB variables (default: `[128 128 64]`);
- `time_images` specifies whether the follow-up CT images should be generated (default: `true`);
- `rng` specifies the seed for the MATLAB random number generator (default: `1`).

According to a somehow realistic *in silico* radiobiology, `synthetic_cohort` produces a table of demographic and clinical variables (birth date, treatment date, weight, sex, chemotherapy,

recurrence, end of follow-up, toxicity occurrence, toxicity date), a template file (containing the reference CT, the voxel size and the masks of heart, left lung and right lung) and the patient-specific files with the spatially normalized VB variables (tumor mask, dose distribution, post-RT cardiac SPECT, and follow-up CT images).

The dataset refers to a hypothetical cohort of synthetic patients treated for cubic lung cancers in variable positions (freely moving within the field-of-view considered as a flat 3-torus, so that the expected spatial distribution of the tumors is truly uniform). The dose distributions include a roughly uniform low dose bath in the entire field-of-view, with a high dose region covering the dilated tumor contour. 50% of the patients are treated with a boost of dose in a central lymph node box. Patients might also receive chemotherapy with an independent 1/2 chance.

The columns of recurrence (binary variable) and onset of treatment toxicity (continuous date) are generated as a realization of two sequences of random variables. Both chemotherapy and boost of dose represent protective factors for the recurrence, while shortening the toxicity onset; the toxicity onset also shows a minor dependence on the sex. The binary variable of toxicity occurrence selects patients in which the toxicity onset occurs during the follow-up. Of note, the dose response for the generated outcomes is not derived from a radiobiological mechanism explicitly based on patient-specific radiosensitivity maps. Indeed, the distinction between the significant VB analysis region and an underlying radiosensitivity is a complex issue (Cella et al., RadOnc 2021; Palma et al., Cancers 2021; Monti et al., Cancers 2022), which depends on the non-uniformity and spatial correlation of dose maps in the analyzed dataset, and goes beyond a more immediate test of the proposed computational tool (Palma et al., FrontOncol 2019; Wilson et al., PhysMed 2022).

A pictorial SPECT image of the heart is generated to show the cardiac perfusion following the RT treatment, and it is assumed that the dose contributes to reducing the local measured activity. Finally, a time series of follow-up CT images is generated for each patient at random time intervals, and a pulmonary fibrosis appears as a change in parenchyma density, linearly modulated from left to right, with a VB characteristic time inversely proportional to the local dose release.

Figure 2.1 reports an overview of the imaging generated for some sample synthetic patients.

In the examples described in the next subsections we assume that `synthetic_cohort` has been run with the default settings. We encourage the reader to inspect `synthetic_cohort.m` as well as the generated files in order to get a better understanding of the following statistical designs and results.

For each example, we generated a MAT-file (*e.g.*, `test1.mat`), containing the option configuration for the dataset import and the statistical analysis, as well as a function (*e.g.*, `test1.m`) for the analysis run. Each function has three optional inputs that specify the `clin_table` path (default: `'clin.mat'`), the `path_proc` folder (default: current folder) and the output folder where the MAT-file containing the results (*e.g.*, `out_test1.mat`) will be saved (default: current folder). Please note that by using customizable configuration files, the functions to run different analyses result to be quite identical, with a few exceptions for some options strictly dependent on the model chosen in the test (*e.g.*, the option `outs` in test 3).

Basically, each function consists of three main steps: the data reading by the function `image_clin_merge`; the calculation of the voxel-wise effect size (*e.g.*, the regression coefficient or the mean difference) of interest by the function `VBmodel`; and the computation of the associated p-map by the function `perm_test`.

As an example, the core of the function `test2.m` consists of the following commands:

```
conf = 'test2.mat';
[T, CCS] = image_clin_merge('clin_table', clin_table, 'path_proc', path_proc, conf);
p = ones(size(CCS.mfix));
```

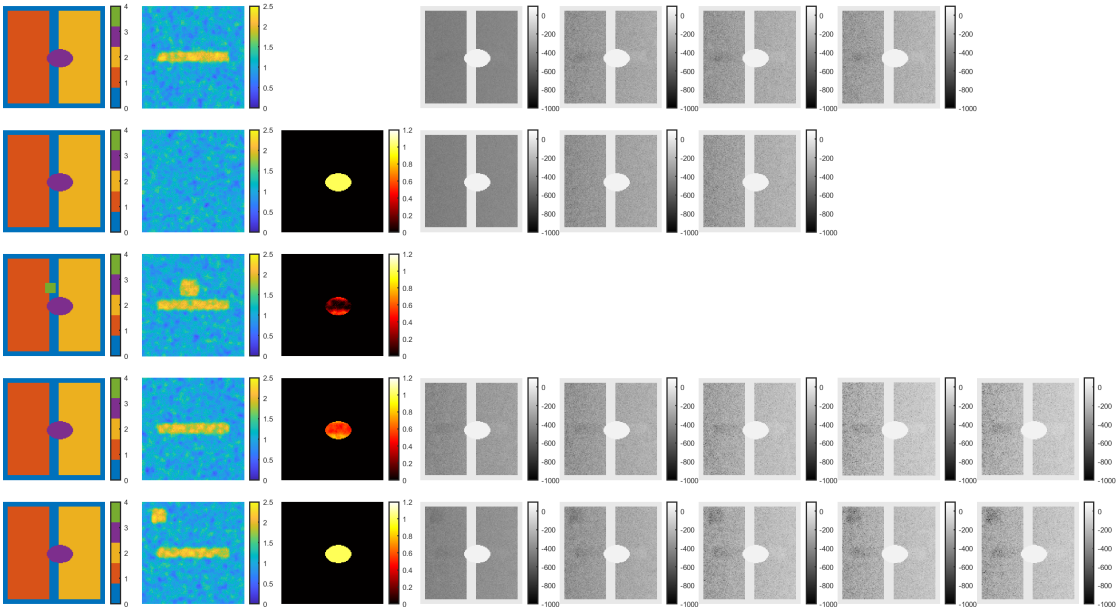

Figure 2.1: Set of coronal images generated for some synthetic patients (out in rows). From left to right: the structures (orange: right lung; yellow: left lung; purple: heart; green: tumor), the dose distribution, the cardiac SPECT and the follow-up CTs.

```
beta = zeros(size(CCS.mfix));
beta(CCS.mfix) = VBmodel(T, CCS.config.outcome, conf, 'mfix', CCS.mfix, ...
    'vars', CCS.config.vars, 'outs', 'beta');
p(CCS.mfix) = perm_test(T, CCS.config.outcome, conf, 'mfix', CCS.mfix, ...
    'vars', CCS.config.vars, 'subsampling', [4 4 2], 'N', 1000);
```

2.1.2 Test 1: minimal GLM on global binary outcome

A VB GLM is developed to evaluate statistical differences in dose distributions between patients who developed toxicity and those who did not. Here, the analysis is performed on the whole field-of-view, since no `CCS.mfix` or patient-specific masks have been set. In this minimal configuration, no nuisance variables are taken into account.

The configuration file `test1.mat` contains the following options:

```
template_images = 'CT';
images = 'dose';
outcomes = 'toxicity';
EVOIs = 'dose';
```

In this test, the function `test1.m`, beside computing the regression coefficients, executes the non-parametric permutation inference with two configurations differing from each other in the `subsampling` option: in one case it is left at its default, in the other it is set to `[4 4 2]`.

In Figure 2.2(a-b), the maps of GLM model coefficients and their significance are shown: the VB analysis identified a significant association between the toxicity endpoint and the dose values in correspondence of the central box of the simulated dose boost. In Figure 2.2(c), it can be appreciated the effectiveness of the subsampling approach in producing accurate significance maps, while providing a speedup ≈ 32.

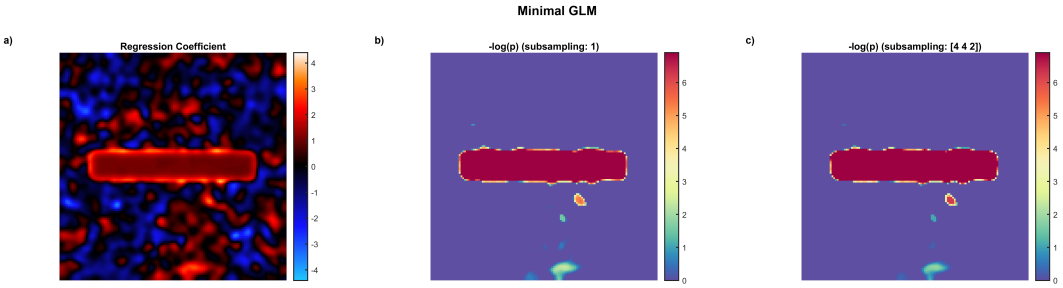

Figure 2.2: Test 1: Coronal views of the regression coefficient associated with the local dose (a) and its right-tailed significance (maxT computed on the full grid (b) and on a subsampled grid (c)) expressed as $-\log p$.

2.1.3 Test 2: GLM on global binary outcome

The relationship between lower VB dose value and recurrence is assessed via a VB GLM. The analysis is restricted to the regions defined by the option `masks`, and also patient-specific `mobile_masks` have been considered. The GLM is designed to include nuisance variables selected by an elastic net among `clin_vars`, as well as the derived variable `age_at_start`, which is forced into the model by the option `vars`. Please note that, once the default for `vars` has been overwritten, `images` need to explicitly enter the new definition.

The configuration file `test2.mat` contains the following options:

```
masks = {'heart' 'left_lung' 'right_lung'};
images = 'dose';
mobile_masks = 'tumor';
der_vars = {{'age_at_start' {'years' {'-' 'treatment_date' 'birth_date'}}}};
outcome = 'recurrence';
clin_vars = {'CHT' 'weight' 'sex'};
vars = {'age_at_start' images};
EVOIs = 'dose';
tails = -1;
```

After masking for the individual tumor volume and taking into account the patient's age at the start of treatment, the test resulted in a significant negative correlation between the tumor recurrence and the dose values in the central high dose region (Figure 2.3).

2.1.4 Test 3: *t*-test with patients' selection

This example is very similar to test 2, except that the relationship between dose distributions and recurrence is assessed via a simple VB t-test (function `tstat2`). In addition, a selection of the patients to be analyzed (age > 18 years) is performed by setting the option `select_patients` to the derived variable `adult`.

The configuration file `test3.mat` contains the following options:

```
masks = {'heart' 'left_lung' 'right_lung'};
images = 'dose';
der_vars = {{'adult' {'>' {'years' {'-' 'treatment_date' 'birth_date'}} 18}}};
outcome = 'recurrence';
select_patients = 'adult';
EVOIs = 'dose';
model = @tstat2;
fit_pars = {'tail' 'left'};
```

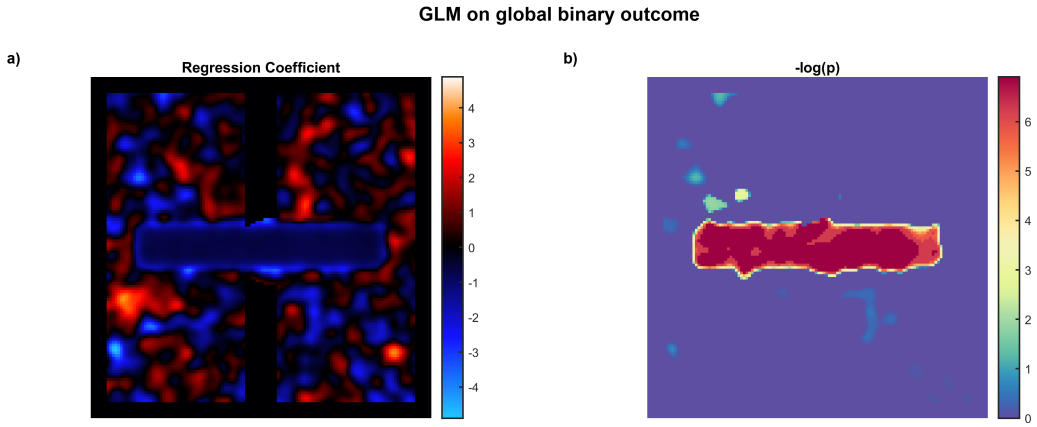

Figure 2.3: Test 2: Coronal views of the regression coefficient associated with the local dose (a) and its left-tailed significance (b) expressed as $-\log p$.

Please note that the same behavior granted by the value assigned to `fit_pars` could have been obtained by the option `tails` set to `-1`.

The obtained significance map is comparable to the one of the previous Test 2, as illustrated in Figure 2.4, where the coronal views of the mean dose difference between recurring and non-recurring patients is also shown.

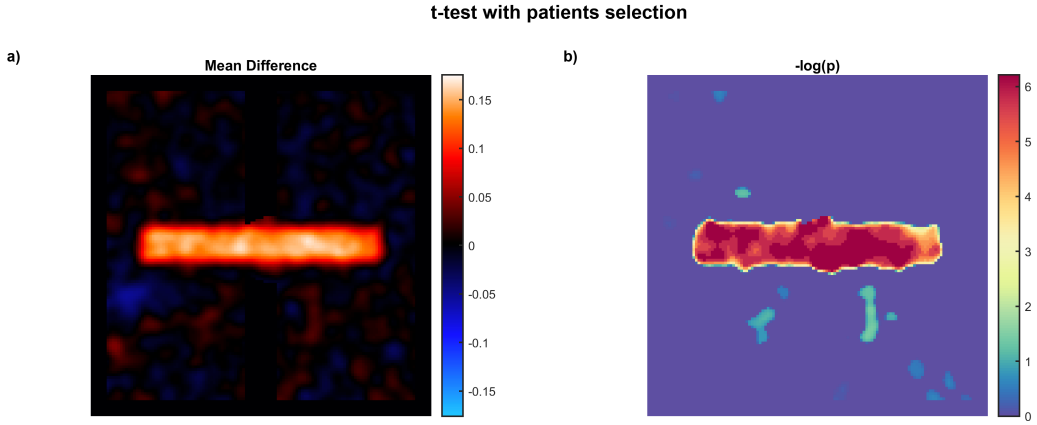

Figure 2.4: Test 3: Coronal views of the mean dose difference (a) and its right-tailed significance (b) expressed as $-\log p$.

2.1.5 Test 4: GLM on global continuous outcome

A VB GLM is designed to evaluate dose differences related to the global continuous outcome `time_to_event`, defined in the derived variables as the number of days between the treatment and the first event between toxicity occurrence and end of follow-up. The elastic net selects

nuisance variables among `weight`, `sex` and `CHT`, after having imputated missing values of `weight` according to a GLM including `sex` and `age_at_start`. In addition, the outcome is assumed to follow a Poisson distribution, as specified by the option `distr`.

The configuration file `test4.mat` contains the following options:

```
masks = {'heart' 'left_lung' 'right_lung'};
images = 'dose';
mobile_masks = 'tumor';
der_vars = { ...
    {'age_at_start' {'years' {'-' 'treatment_date' 'birth_date'}}} ...
    {'time_to_event' {'days' {'-' ...
        {'first' 'toxicity_date' 'end_follow_up'} 'treatment_date'}}}};
outcome = 'time_to_event';
clin_vars = {{'weight' 'sex' 'age_at_start'} 'sex' 'CHT'};
EVOIs = 'dose';
distr = 'poisson';
tails = -1;
```

The obtained results show a region of significant negative correlation between higher local doses and the time to toxicity occurrence (Figure 2.5).

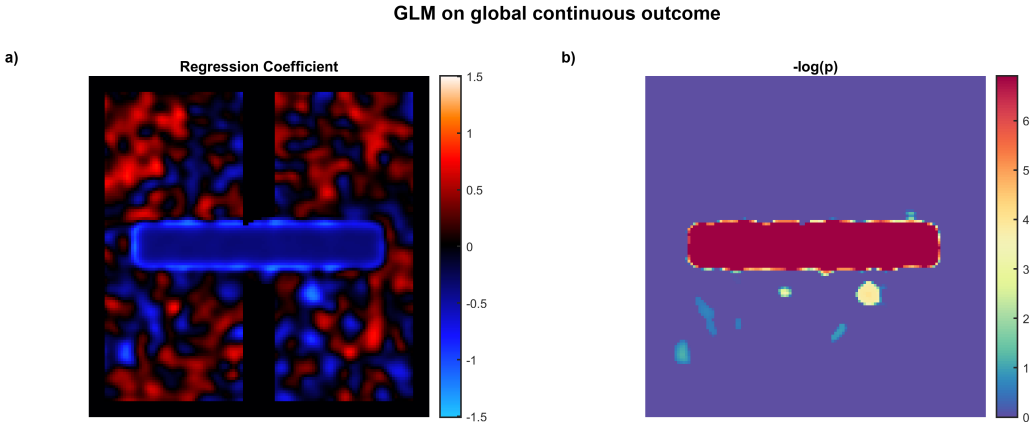

Figure 2.5: Test 4: Coronal views of the regression coefficient associated with the local dose (a) and its left-tailed significance (b) expressed as $-\log p$.

2.1.6 Test 5: Cox regression with global time-to-event and censoring

A VB Cox regression is implemented to evaluate the impact of dose distributions on toxicity development. The outcome variables `time_to_event` and `censoring` are defined in the `der_vars` option. The model is forced by the option `vars` to include `weight` as a nuisance variable, after having performed an imputation according to the mean of the variable itself, as specified by the option `clin_vars`.

The configuration file `test5.mat` contains the following options:

```
masks = {'heart' 'left_lung' 'right_lung'};
images = 'dose';
mobile_masks = 'tumor';
der_vars= { ...
    {'time_to_event' {'days' {'-' ...
```

```
        {'first' 'toxicity_date' 'end_follow_up'} 'treatment_date'}}} ...
    {'censoring' {'~' 'toxicity'}}};
outcome = {'time_to_event' 'censoring'};
clin_vars = {{'weight'}};
vars = {'weight' images};
EVOIs = 'dose';
```

After masking for tumor volume and taking into account the patient's weight, the performed analysis identified a significant relationship between the toxicity onset and the local dose in the central lymph node box (Figure 2.6).

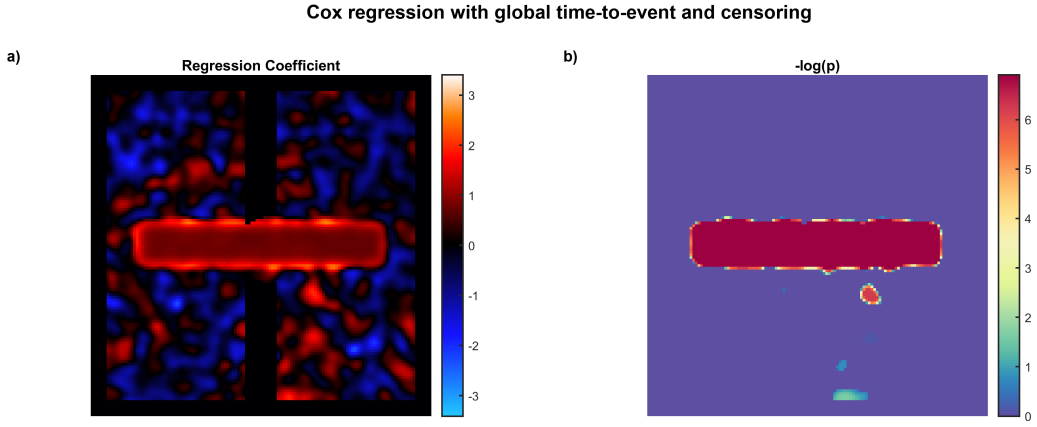

Figure 2.6: Test 5: Coronal views of the regression coefficient associated with the local dose (a) and its right-tailed significance (b) expressed as $-\log p$.

2.1.7 Test 6: GLM on VB continuous outcome and mixed VB and global EVs

The relationship between lower VB dose values and the cardiac SPECT images (Figure 2.1) is assessed via a GLM with a VB outcome. The sex EV represents a global nuisance variable. Here the template structure `heart` is eroded by a spherical structuring element of radius 1 voxel, as specified by the option `refine_mask`, to define the region to be included into the analysis. Furthermore, in this test, the TFCE is turned off.

The configuration file `test6.mat` contains the following options:

```
masks = 'heart';
images = {'dose' 'spect};
mobile_masks = 'tumor';
outcome = 'spect';
vars = ['sex' images];
refine_mask = @(x, y) morph(y.heart, sph(1), 'e');
EVOIs = 'dose';
tails = -1;
tfce = false;
```

After masking for tumor volume and including the sex as a nuisance variable, the performed VB analysis showed a region (roughly corresponding to the intersection of the central box and the heart) of significant negative correlation between higher local dose values and the local activity measured after the RT treatment (Figure 2.7).

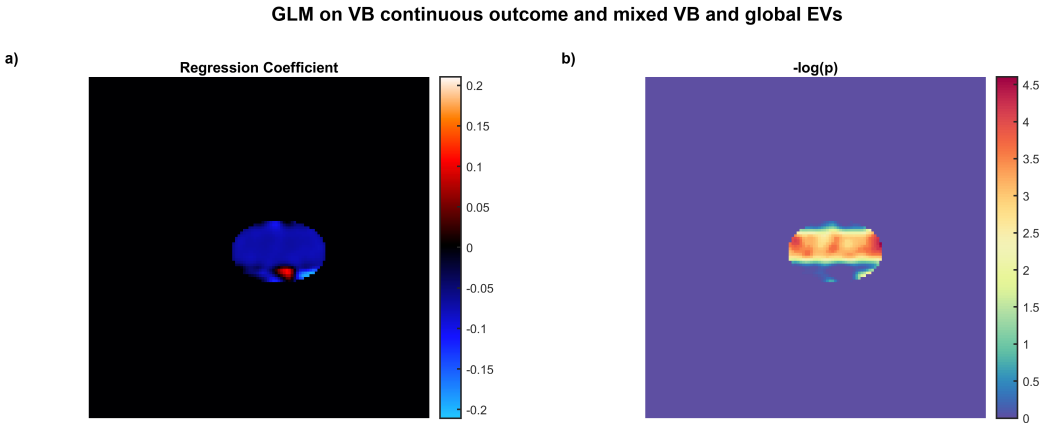

Figure 2.7: Test 6: Coronal views of the regression coefficient associated with the local dose (a) and its left-tailed significance (b) expressed as $-\log p$.

2.1.8 Test 7: GLM on VB continuous outcome with one global EV

A VB GLM is designed to evaluate the relationship between a global EV, the `mean_heart_dose`, defined as a derived variable in the option `der_vars`, and a VB outcome given by the SPECT images. As in the previous example, the TFCE is turned off, while patient-specific masks are not accounted for.

The configuration file `test7.mat` contains the following options:

```
masks = 'heart';
images = {'spect' 'dose'};
der_vars = {{'mean_heart_dose' ...
    {'/' {'sum' {'.*' 'dose' '#heart'} 'all'} {'sum' '#heart' 'all'}}}};
outcome = 'spect';
vars = 'mean_heart_dose';
EVOIs = 'mean_heart_dose';
tails = -1;
tfce = false;
```

The results of this test showed a negative correlation between the mean heart dose and the local activity as measured by the simulated cardiac SPECT (Figure 2.8).

2.1.9 Test 8: Cox regression with VB time-to-event, censoring and EV

A VB Cox regression is implemented to assess the impact of dose distributions on the development of lung fibrosis evaluated on follow-up CT images (Figure 2.1). The time-to-event and censoring are VB variables derived from the follow-up CT images contained in the patients' timetables `tt_imaging`. In particular, `VB_time2event` and `VB_censoring` are defined in the option `der_vars`, and are obtained as fields of the structure `VBcox_struct` computed by the in-house written function `censoring_event`. In this function, for each voxel, the variation of the Hounsfield Units (HU) in the follow-up CTs is fitted to an exponential curve. Assuming that the initial density of the healthy lung corresponds to –500 HU, if the exponential reaches the threshold of –400 HU within the follow-up, the VB censoring (stored in the field `x` of the output structure) is set to `false`; otherwise, it is set to `true`. Instead, the VB time-to-event (stored

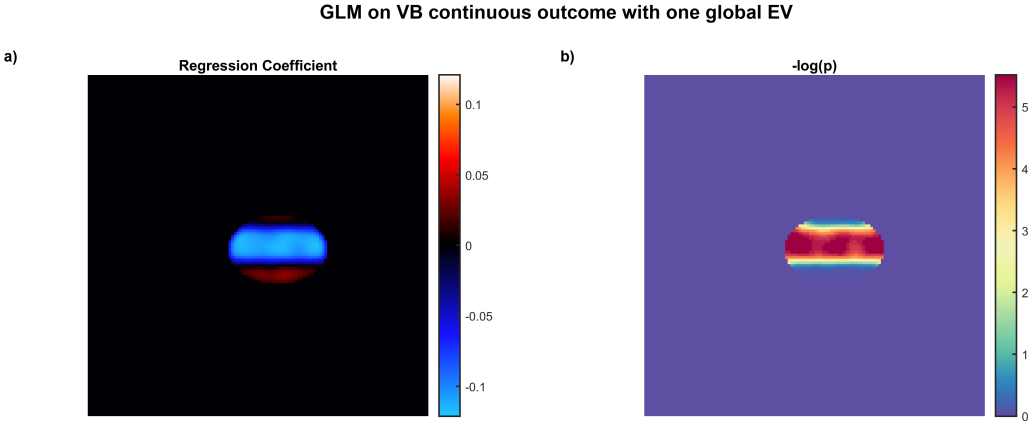

Figure 2.8: Test 7: Coronal views of the regression coefficient associated with the mean heart dose (a) and its left-tailed significance (b) expressed as $-\log p$.

in the field d) is computed as the number of days between the treatment and the first event between local fibrosis occurrence and end of follow-up. In addition, in order to guarantee that the model will be computed on reliable estimates of `VB_time2event` and `VB_censoring`, the analysis is restricted only to patients with at least 3 follow-up CTs in the `tt_imaging` timetable, as defined in the option `select_patients`, which is set to be equal to the derived variable `n_time_points`.

The configuration file `test8.mat` contains the following options:

```
masks = {'left_lung' 'right_lung'};
images = 'dose';
mobile_masks = 'tumor';
time_images = 'tt_imaging';
der_vars = {...
    {'VBcox_struct' {'censoring_event' 'tt_imaging' ...
        'Images' 'end_follow_up' 'treatment_date'}} ...
    {'VB_censoring' {'getfield' 'VBcox_struct' 'x'}} ...
    {'VB_time2event' {'getfield' 'VBcox_struct' 'd'}} ...
    {'n_time_points' {'>' {'height' 'tt_imaging'} 2}}};
outcome = {'VB_time2event' 'VB_censoring'};
select_patients = 'n_time_points';
EVOIs = 'dose';
```

The coefficient map of the Cox regression correctly identifies the left-right modulation of the synthetic dose-response relationship (Figure 2.9(a)), which appears to be largely significant at a right-tailed test in the left lung (Figure 2.9(b)).

2.2 MAMBA configuration for published studies on clinical datasets

Here, we will describe some examples of MAMBA applications in a radiation oncology setting that led to clinical results reported in the existing literature.

The first experiment is related to the application of MAMBA to a cohort of head and

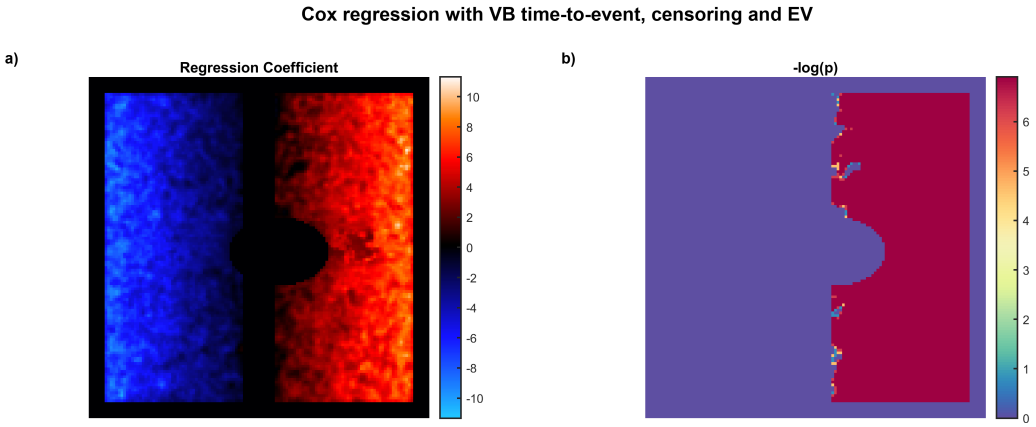

Figure 2.9: Test 8: Coronal views of the regression coefficient associated with the local dose (a) and its right-tailed significance (b) expressed as $-\log p$.

neck cancer patients undergoing RT [Monti et al., 2017]. A VB GLM was designed to assess statistical differences in dose distributions between patients who developed radiation-induced acute dysphagia and those who did not. By using the option `masks`, the analysis was restricted to the region given by the union of a set of swallowing-related structures:

```
[T, CCS] = image_clin_merge( ...
    'images', 'dose', ...
    'masks', user_defined_list_of_structures, ...
    'clin_vars', {'gender' 'age' 'tumor_stage' 'RT_technique' 'chemotherapy'}, ...
    'outcome', 'dysphagia');
```

The permutation test

```
p = perm_test(tab, 'dysphagia', ...
    'EVOIs', 'dose', ...
    'mfix', CCS.mfix);
```

highlighted significantly higher doses delivered to patients developing dysphagia in two voxel clusters in correspondence of the cricopharyngeus muscle and cervical esophagus.

In Palma et al. [2019], a cohort of non-small cell lung cancer patients treated with two different RT modalities (photons *vs.* protons) was analyzed. A VB analysis was designed to inspect dose differences between treatment modality, which was set as the only (global) EV in a GLM of the VB outcome given by Biologically Effective Dose (BED). BED maps were derived assuming an alpha-beta ratio of 3 Gy via the in-house written function `BED`. Since the BED is a nonlinear function of the dose, it was computed before any smoothing was applied by the function `local_moments` via the Gaussian kernel returned by `sph`. Therefore, it was first run

```
[img, CCS] = image_read( ...
    'images', 'dose', ...
    'masks', {'heart' 'lungs'}, ...
    'mobile_masks', 'tumor', ...
    'sigma', false);
```

and, then

```
[tab, CCS] = image_clin_merge(img, CCS, ...
    'der_vars', {...
        {'kernel' {'sph' {'./' 5 '#vox'} 'gauss'}} ...
        {'BED_maps' ...
            {'local_moments' {'BED' 'dose' 'number_of_fraction' 3} 'kernel'}} ...
        {'smoothed_tumor' {'local_moments' 'tumor' 'kernel'}} ...
        {'photons' {'=' 'RT_modality' 'IMRT'}}}, ...
    'mobile_masks', 'smoothed_tumor', ...
    'outcome', 'BED_maps', ...
    'vars', 'photons');
```

(please refer to the documentation of the functions `sph`, `local_moments` and `BED` in `Engine/` for further details on the definition of the derived variables).

This setting restricted the analysis to the union of lungs and heart, and excluded patient-wise the tumor volumes from the VB GLM.

The difference of mean BEDs in patients treated with photons and with protons was obtained as

```
delta_BED = VBmodel(tab, 'BED_maps', ...
    'EVOIs', 'photons', ...
    'outs', 'beta', ...
    'mfix', CCS.mfix);
```

and its significance was assessed separately in the two tails:

```
p = perm_test(tab, 'BED_maps', ...
    'EVOIs', 'photons', ...
    'mfix', CCS.mfix, ...
    'tails', [-1 1]);
```

The analysis identified only positive significant dose differences between photons and protons, corresponding to anatomic regions significantly spared by protons.

For the same cohort of lung cancer patients, a VB Cox regression was implemented to evaluate the impact of dose distributions on pericardial effusion (PCE) development [Cella et al., 2021]. As in the previous application, the `image_read` and `image_clin_merge` functions were called separately to appropriately configure the calculation and smoothing of BED maps and patient-specific masks. However, here the option `outcomes` was set to `{'time_to_event' 'censoring'}`, two global variables that were defined in `der_vars` (along with `'BED_maps'` and `'smoothed_tumor'`) as `{'time_to_event' {'days' {'-' {'first' 'PCE_occurrence_date' 'end_follow_up_date'} 'treatment_start_date'}}}` and `{'censoring' {'=' 'PCE_grade' 0}}`, respectively. The model was forced to include age and adjuvant chemotherapy as nuisance variables by the option `vars`, which was accordingly set to `{'BED_maps' 'age' 'adjuvant_chemotherapy'}`.

The map of the Cox regression coefficient was obtained as:

```
beta_BED = VBmodel(tab, {'time_to_event' 'censoring'}, ...
    'EVOIs', 'BED_maps', ...
    'outs', 'beta', ...
    'mfix', CCS.mfix);
```

The permutation test, which was analogously configured, highlighted regions with significant association between local dose and heart toxicity in correspondence of several cardiac structures and pulmonary segments.

Finally, in Monti et al. [2022] it is possible to observe an application of the `refine_mask` option. Here, several VB analyses were designed to evaluate the dose differences associated with

continuous global measures of radiation-induced lymphopenia. The analyses were meant to be extended to the convex hull of a dilated union of the template ribs, vertebrae, lungs and heart; however, some peripheral regions in the convex hull typically received very low dose levels in the analyzed cohort, hence the mask was refined by a VB condition on the values of mean dose among the dose maps. This was obtained by setting `masks` to `{'body' 'ribs' 'vertebrae' 'lungs' 'heart'}` and `refine_mask` to `@(tab, CCS) convexhull(morph(CCS.ribs | CCS.vertebrae | CCS.lungs | CCS.heart, sph(10./CCS.vox))) & mean(cat(4, tab.BED_maps{:}), 4) > 2` (`convexhull` refers to an in-house written function).

Bibliography

L. Cella, S. Monti, T. Xu, R. Liuzzi, A. Stanzione, M. Durante, R. Mohan, Z. Liao, and G. Palma. Probing thoracic dose patterns associated to pericardial effusion and mortality in patients treated with photons and protons for locally advanced non-small-cell lung cancer. *Radiotherapy and Oncology*, 160:148–158, 2021.

N. R. Draper and D. M. Stoneman. Testing for the inclusion of variables in linear regression by a randomisation technique. *Technometrics*, 8(4):695–699, 1966. ISSN 00401706. URL http://www.jstor.org/stable/1266641.

P. E. Kennedy. Randomization tests in econometrics. *Journal of Business & Economic Statistics*, 13(1):85–94, 1995. ISSN 07350015. URL http://www.jstor.org/stable/1392523.

B. F. Manly. Randomization and regression methods for testing for associations with geographical, environmental and biological distances between populations. *Researches on Population Ecology*, 28(2):201–218, 1986.

S. Monti, G. Palma, V. D'Avino, M. Gerardi, G. Marvaso, D. Ciardo, R. Pacelli, B. A. Jereczek-Fossa, D. Alterio, and L. Cella. Voxel-based analysis unveils regional dose differences associated with radiation-induced morbidity in head and neck cancer patients. *Scientific reports*, 7(1):1–8, 2017.

S. Monti, T. Xu, Z. Liao, R. Mohan, L. Cella, and G. Palma. On the interplay between dosiomics and genomics in radiation-induced lymphopenia of lung cancer patients. *Radiotherapy and Oncology*, 167:219–225, 2022. ISSN 0167-8140. doi: https://doi.org/10.1016/j.radonc.2021.12.038. URL https://www.sciencedirect.com/science/article/pii/S0167814021090861.

G. Palma, S. Monti, T. Xu, E. Scifoni, P. Yang, S. M. Hahn, M. Durante, R. Mohan, Z. Liao, and L. Cella. Spatial dose patterns associated with radiation pneumonitis in a randomized trial comparing intensity-modulated photon therapy with passive scattering proton therapy for locally advanced non-small cell lung cancer. *International Journal of Radiation Oncology*Biology*Physics*, 104(5):1124–1132, 2019. ISSN 0360-3016. doi: https://doi.org/10.1016/j.ijrobp.2019.02.039. URL https://www.sciencedirect.com/science/article/pii/S0360301619302743.

G. Palma, S. Monti, and L. Cella. Voxel-based analysis in radiation oncology: A methodological cookbook. *Physica Medica*, 69:192–204, 2020. ISSN 1120-1797. doi: https://doi.org/10.1016/j.ejmp.2019.12.013. URL https://www.sciencedirect.com/science/article/pii/S1120179719305344.

G. Palma, L. Cella, and S. Monti. Mamba – multi-paradigm voxel-based analysis: a computational cookbot. *Submitted*, 2022.

S. M. Smith and T. E. Nichols. Threshold-free cluster enhancement: Addressing problems of smoothing, threshold dependence and localisation in cluster inference. *NeuroImage*, 44(1): 83–98, 2009. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2008.03.061. URL https://www.sciencedirect.com/science/article/pii/S1053811908002978.

A. M. Winkler, G. R. Ridgway, M. A. Webster, S. M. Smith, and T. E. Nichols. Permutation inference for the general linear model. *NeuroImage*, 92:381–397, 2014. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2014.01.060. URL https://www.sciencedirect.com/science/article/pii/S1053811914000913.