Package ‘spm12r’

January 9, 2019

Type Package

Title Wrapper Functions for ‘SPM’ (Statistical Parametric Mapping)
Version 12 from the ‘Wellcome’ Trust Centre for ‘Neuroimaging’

Version 2.8.1

Date 2019-01-01

Description Installs ‘SPM12’ to the R library directory and has
associated functions for ‘fMRI’ and general imaging utilities,
called through ‘MATLAB’.

License GPL-2 | file LICENCE

Imports R.utils, neurobase, utils, matlabr (>= 1.5.2), oro.nifti,
git2r

Suggests knitr, rmarkdown, kirby21.t1 (>= 1.7), kirby21.fmri (>= 1.7),
scales, httr

SystemRequirements MATLAB

Encoding UTF-8

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author John Muschelli [cre, aut]

Maintainer John Muschelli <muschelliJ2@gmail.com>

Repository CRAN

Date/Publication 2019-01-09 12:50:03 UTC

R topics documented:

   acpc_reorient ........................................... 2
   add_spm_dir ........................................... 3
   build_spm12_coregister .................................. 4
   build_spm12_first_level .................................. 6
   build_spm12_first_level_spec ................................ 7
Function that AC/PC re-orientsthe images for SPM spatial normalization routine. Uses nii_setOrigin from http://www.mccauslandcenter.sc.edu/CRNL/sw/spm8/spm.zip

Usage
acpc_reorient(infiles, modality = c("T1", "T2", "CT", "fMRI"),
spmdir = spm_dir(), verbose = TRUE, install_dir = NULL, ...)
add_spm_dir

Arguments

infiles (character) Files to reorient. First file will be used to estimate AC/PC, then rest will be transformed
modality T1, T2, CT, fMRI (T2*)
spmdir (character) path for SPM12. If NULL, assumes SPM12 is in matlabpath and so is spm8/toolbox Must have nii_setOrigin installed. In system.file("", package="spm12") from http://www.mccauslandcenter.sc.edu/CRNL/sw/spm8/spm.zip
verbose (logical) Print diagnostic output
install_dir directory to download SPM12
... Arguments to pass to run_matlab_code

Value

Exit code from MATLAB. If not zero, there was an error

Description

Add SPM Directory

Usage

add_spm_dir(x, spmdir = spm_dir(VERBOSE = verbose), verbose = TRUE)
add_spm12_dir(x, spmdir = spm_dir(VERBOSE = verbose), verbose = TRUE)

Arguments

x Character vector of commands
spmdir SPM Directory
verbose Print diagnostic messages

Value

A character vector
build_spm12_coregister

**Batch SPM12 Coregister (Estimate and Reslice)**

**Description**

Performs SPM12 coregistration estimation and reslicing on an Image

**Usage**

```python
build_spm12_coregister(fixed, moving, other_files = NULL, 
cost_fun = c("nmi", "ecc", "ncc"), separation = c(4, 2), 
tol = c(0.02, 0.02, 0.02, 0.001, 0.001, 0.001, 0.01, 0.01, 0.01, 0.01, 0.001, 
0.001, 0.001), fwhm = c(7, 7), interp = c("bspline4", 
"nearestneighbor", "trilinear", paste0("bspline", 2:3), paste0("bspline", 5:7)), wrap_x = FALSE, wrap_y = FALSE, wrap_z = FALSE, 
mask = FALSE, prefix = "r", verbose = TRUE, ...)

build_spm12_coregister_estimate(fixed, moving, other_files = NULL, 
cost_fun = c("nmi", "ecc", "ncc"), separation = c(4, 2), 
tol = c(0.02, 0.02, 0.02, 0.001, 0.001, 0.001, 0.01, 0.01, 0.01, 0.01, 0.001, 0.001, 0.001), fwhm = c(7, 7), verbose = TRUE, ...)

build_spm12_coregister_reslice(fixed, moving, interp = c("bspline4", 
"nearestneighbor", "trilinear", paste0("bspline", 2:3), paste0("bspline", 5:7)), wrap_x = FALSE, wrap_y = FALSE, wrap_z = FALSE, 
mask = FALSE, prefix = "r", verbose = TRUE, ...)

spm12_coregister_wrapper(..., func = c("build_spm12_coregister", 
"build_spm12_coregister_reslice", "build_spm12_coregister_estimate"), 
add_spm_dir = TRUE, spmdir = spm_dir(verbos = verbose, install_dir = 
install_dir), clean = TRUE, verbose = TRUE, outdir = NULL, 
install_dir = NULL)

spm12_coregister(..., add_spm_dir = TRUE, spmdir = spm_dir(verbos = verbose, 
install_dir = install_dir), clean = TRUE, verbose = TRUE, 
outdir = NULL, install_dir = NULL)

spm12_coregister_estimate(..., add_spm_dir = TRUE, 
spmdir = spm_dir(verbos = verbose, install_dir = install_dir), 

clean = TRUE, verbose = TRUE, outdir = NULL, install_dir = NULL)

spm12_coregister_reslice(..., add_spm_dir = TRUE, 
spmdir = spm_dir(verbos = verbose, install_dir = install_dir), 

clean = TRUE, verbose = TRUE, outdir = NULL, install_dir = NULL)
```
Arguments

- **fixed**: File that is assumed fixed
- **moving**: Moving file to be registered to fixed space
- **other_files**: Other files to register to fixed, in same space as moving file
- **cost_fun**: Cost function
- **separation**: The average distance between sampled points (in mm). Can be a vector to allow a coarse registration followed by increasingly fine
- **tol**: The accuracy for each parameter. Iterations stop when differences between successive estimates are less than the required
- **fwhm**: Gaussian smoothing to apply to the 256x256 joint histogram. Other information theoretic coregistration methods use fewer bins,
- **interp**: Interpolator for sampling in fixed space
- **wrap_x**: Wrap in x-direction
- **wrap_y**: Wrap in y-direction
- **wrap_z**: Wrap in z-direction
- **mask**: Mask the data. With masking enabled, the program searches through the whole time series looking for voxels which need to be sampled from outside the original images. Where this occurs, that voxel is set to zero for the whole set of images
- **prefix**: Prefix to append to front of image filename
- **verbose**: Print diagnostic messages
- **...**: Additional arguments to pass to run_matlabbatch
- **func**: not used
- **add_spm_dir**: Add SPM12 directory from this package
- **spmdir**: SPM dir to add, will use package default directory
- **clean**: Remove scripts from temporary directory after running
- **outdir**: Directory to copy results. If full filename given, then results will be in dirname(filename)
- **install_dir**: Directory to download SPM12

Value

List of output files, the matlabbatch object, and the script

Examples

```bash
## Not run:
fname = paste0("~/Desktop/D2/scratch/",
              "100-318_20070723_0957_CT_3_CT_Head--SS_0.01_SyN_ROI.nii.gz")
spm = spm12_coregister(
fixed = fname,
moving = fname,
other_files = fname,
execute = FALSE)
```
build_spm12_first_level

SPM12 fMRI First Level Model

Description

SPM12 fMRI First Level Model

Usage

build_spm12_first_level(..., outdir = NULL,
   est_args = list(write_residuals = FALSE, method = "Classical", bayesian = NULL), verbose = TRUE)

spm12_first_level(..., outdir = NULL, est_args = list(write_residuals = FALSE, method = "Classical", bayesian = NULL), add_spm_dir = TRUE, spmdir = spm_dir(verbos = verbose, install_dir = install_dir),
   clean = TRUE, verbose = TRUE, install_dir = NULL)

Arguments

... Arguments passed to build_spm12_first_level_spec
outdir output directory for results
est_args Arguments passed to build_spm12_fmri_est
verbose Print diagnostic messages
add_spm_dir Add SPM12 directory from this package
spmdir SPM dir to add, will use package default directory
clean Remove scripts from temporary directory after running
install_dir directory to download SPM12

Value

A list of objects, including an spm object and output files.
Description

SPM12 fMRI First Level Specification

Usage

build_spm12_first_level_spec(scans = NULL, outdir = NULL,
units = c("scans", "secs"), slice_timed = TRUE, nslices = NULL,
ref_slice = NULL, tr, condition_mat = NULL, condition_list = NULL,
regressor_mat = NULL, regressor_list = NULL, hpf = 128,
time_deriv = FALSE, disp_deriv = FALSE, interactions = FALSE,
global_norm = c("none", "Scaling"), mthresh = 0.8, mask = NULL,
correlation = c("AR(1)", "none", "FAST"), n_time_points = NULL,
verbose = TRUE, overwrite = TRUE, ...

spm12_first_level_spec(..., outdir = NULL, add_spm_dir = TRUE,
spmdir = spm_dir(verbos = verbose, install_dir = install_dir),
clean = TRUE, verbose = TRUE, overwrite = TRUE,
install_dir = NULL)

Arguments

scans images to run
outdir output directory for results
units The onsets of events or blocks can be specified in either scans or seconds.
slice_timed Were the image slice-time corrected
nslices If the data were slice-time corrected, the number of slices of the image
ref_slice If the data were slice-time corrected, the reference slice
tr The repetition time, in seconds
condition_mat multiple condition mat/txt file
condition_list List of conditions: see spm12_condition. This should be a list (or a list of lists) which have the items: name, onset, duration, time_mod_order, param_mod, orth. name does not need to be specified if it is a named list of lists.
regressor_mat multiple regressor mat/txt file
regressor_list List of regressors: see spm12_regressor. This should be a list (or a list of lists) which have the items: name, value, and n_time_points. name does not need to be specified if it is a named list of lists.
hpf High pass filter, in seconds.
time_deriv  time derivative. The time derivative allows the peak response to vary by plus or minus a second.
disp_deriv  dispersion derivative, allows the width of the response to vary.
interactions  Model interactions, Generalized convolution of inputs with basis set.
global_norm  Global intensity normalisation
mthresh  Masking threshold, defined as proportion of globals.
mask  Specify an image for explicitly masking the analysis.
correlation  Serial correlations in fMRI time series
n_time_points  Number of time points
verbose  Print diagnostic messages
overwrite  If a SPM.mat file exists in the outdir, should the file be removed?
...  Arguments passed to run_spm12_script
add_spm_dir  Add SPM12 directory from this package
spmdir  SPM dir to add, will use package default directory
clean  Remove scripts from temporary directory after running
install_dir  directory to download SPM12

Value
A list of objects, including an spm object and output files.

---

build_spm12_realign  SPM12 Realign (Estimate and Reslice)

Description
Performs SPM12 realignment estimation and reslicing on an Image

Usage

build_spm12_realign(filename, time_points = NULL, fwhm = 5,
  quality = 0.9, separation = 4, register_to = c("first", "mean"),
  est_interp = c("bspline2", "trilinear", paste0("bspline", 3:7)),
  wrap_x = FALSE, wrap_y = FALSE, wrap_z = FALSE,
  weight_image = NULL, reslice = c("all+mean", "all", "2:n", "mean"),
  reslice_interp = c("bspline4", "nearestneighbor", "trilinear",
    paste0("bspline", 2:3), paste0("bspline", 5:7), "fourier"),
  mask = FALSE, prefix = "r", verbose = TRUE, ...

spml2_realign(..., add_spm_dir = TRUE, spmdir = spm_dir(verbos =
  verbose, install_dir = install_dir), clean = TRUE, retimg = FALSE,
  reorient = FALSE, verbose = TRUE, outdir = NULL,
  install_dir = NULL)
Arguments

filename  Files to be realigned and resliced

time_points  A vector of time points to run realignment. If filename is a 4D file, then will do all the time points. Otherwise, filename must be a character vector of 3D files or a list of 3D nifti objects.

fwhm  Full-Width Half Max to smooth. Gaussian smoothing to apply to the 256x256 joint histogram.

quality  Quality versus speed trade-off. Highest quality (1) gives most precise results, whereas lower qualities gives faster realignment.

separation  The average distance between sampled points (in mm). Can be a vector to allow a coarse registration followed by increasingly fine

register_to  Should the files be registered to the first or the mean

est_interp  Interpolator for estimation

wrap_x  wrap in x-direction

wrap_y  wrap in y-direction

wrap_z  wrap in z-direction

weight_image  weighting image to weight each voxel of the reference image during estimation. The weights are proportional to the inverses of the standard deviations. May be used when there is a lot of motion.

reslice  Options for reslicing all - all images in filename, 2:n - all images in filename 2:length(filename), all+mean - all images and the mean, mean - mean only

reslice_interp  Interpolator for reslicing

mask  Mask the data. With masking enabled, the program searches through the whole time series looking for voxels which need to be sampled from outside the original images. Where this occurs, that voxel is set to zero for the whole set of images

prefix  Prefix to append to front of image filename

verbose  Print diagnostic messages

...  Arguments passed to run_spm12_script

add_spm_dir  Add SPM12 directory from this package

spmdir  SPM dir to add, will use package default directory

clean  Remove scripts from temporary directory after running

reting  (logical) return image of class nifti

reorient  (logical) If reting, should file be reoriented when read in?

outdir  Directory to copy results. If full filename given, then results will be in dirname(filename)

install_dir  directory to download SPM12

Value

List of output files, the matlabbatch object, and the script
Examples

dims = rep(10, 4)
temp_nii = array(rnorm(prod(dims)), dim = dims)
temp_nii = oro.nifti::nifti(temp_nii)
res = build_spm12_realign(temp_nii)

bwconncomp

Label Connected Clusters of Certain Size

Description

Get Cluster of certain size from spm_bwlabel

Usage

bwconncomp(infile, outfile = NULL, retimg = TRUE, conn = 26,
            reorient = FALSE, spmdir = spm_dir(), verbose = TRUE,
            install_dir = NULL)

Arguments

  infile      input filename
  outfile     output filename
  retimg      Return the image instead of matlab output
  conn        Connections to be passed to MATLAB’s bwconncomp
  reorient    If retimg, then this argument is passed to readNIfTI
  spmdir      (character) path for SPM12. If NULL, assumes SPM12 is in matlabpath.
  verbose     Print Diagnostics
  install_dir directory to download SPM12

Value

Name of output file or nifti object, depending on retimg

Examples

library(neurobase)
set.seed(1234)
dims = c(30, 30, 10)
arr = array(rnorm(prod(dims)), dim = dims)
nim = nifti(arr)
mask = datatyper(nim > 1)
## Not run:
c = bwconncomp(mask)
tab = table(c(cc))
## End(Not run)
filename_check

Perform filename checks for SPM

Description
Checks a filename to see if nifti and expands paths to absolute paths

Usage
filename_check(filename, ...)

Arguments
filename filename of an image or nifti object
... arguments passed to checknii

Value
Character of filename

See Also
cHECKnii

group_xyz

Center of Gravity for Multiple Areas

Description
Find Center of Gravity of Each Area of Image, after thresholding

Usage
group_xyz(img, k = 1, ...)

Arguments
img Object of class nifti
k Minimum number of voxels for a cluster. See spm_bwlabel
... Arguments passed to spm_bwlabel

Value
Matrix of 3 columns
install_spm12  \hspace{1cm} Install SPM12 file into directory

**Description**

Install spm12 scripts to spm12r for script capabilities

**Usage**

```
install_spm12(lib.loc = NULL, verbose = TRUE, install_dir = NULL)
```

**Arguments**

- `lib.loc` a character vector with path names of R libraries. Passed to `system.file`
- `verbose` print diagnostic messages
- `install_dir` Alternative directory to download SPM12

**Examples**

```r
tdir = tempfile()
dir.create(tdir)
in_ci <- function() {
  nzchar(Sys.getenv("CI"))
}
if (!Platform$OS.type == "unix" | in_ci()) { # windows problem
  res = install_spm12(install_dir = tdir)
  res = install_spm12(install_dir = tdir)
}
```

ntime_points  \hspace{1cm} Get number of time points from a file

**Description**

Extracts the number of time points from a nifti object, list or character

**Usage**

```
ntime_points(filename)
```

**Arguments**

- `filename` List of nifti objects, a vector of character filenames, or a single 4D nifti

**Value**

Vector of time points
run_matlabbatch

Run Matlab Batch from List

Description

Run Matlab Batch from List

Usage

run_matlabbatch(spm, add_spm_dir = TRUE, clean = TRUE,
verbose = TRUE, gui = FALSE, ...)

matlabbatch_job(spm, add_spm_dir = TRUE, gui = FALSE, ...)

matlabbatch_to_script(spm, batch_prefix = "matlabbatch1.", ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spm</td>
<td>List of the class matlabbatch</td>
</tr>
<tr>
<td>add_spm_dir</td>
<td>should SPM12 directory be added to the script</td>
</tr>
<tr>
<td>clean</td>
<td>Remove scripts from temporary directory after running</td>
</tr>
<tr>
<td>verbose</td>
<td>Print diagnostic messages</td>
</tr>
<tr>
<td>gui</td>
<td>is the GUI necessary?</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments to pass to matlabbatch_job</td>
</tr>
<tr>
<td>batch_prefix</td>
<td>prefix to add to the names of spm</td>
</tr>
</tbody>
</table>

Value

Result of run_matlab_script

run_spm12_script

Wrapper for running spm12_script

Description

Runs spm12_script with wrapper for spm12r functions

Usage

run_spm12_script(script_name, jobvec = NULL, mvec = NULL,
add_spm_dir = TRUE, spmdir = spm_dir(verbos = verbose),
clean = TRUE, verbose = TRUE, single_thread = FALSE, ...)

build_spm12_script(script_name, jobvec = NULL, mvec = NULL,
add_spm_dir = TRUE, spmdir = spm_dir(verbos = verbose),
verbose = TRUE, install_dir = NULL, ...)
**spm12_condition**

**Arguments**

- **script_name**: Name of the script filename without .m ext, passed to `spm12_script`
- **jobvec**: Vector of characters to be substituted in _job.m file
- **mvec**: Vector of characters to be substituted in .m file
- **add_spm_dir**: Add SPM12 directory from this package
- **spmdir**: SPM dir to add, will use package default directory
- **clean**: Remove scripts from temporary directory after running
- **verbose**: Print diagnostic messages
- **single_thread**: Should the flag `-singleCompThread` be executed to limit MATLAB to a single computational thread?
- **...**: Arguments to pass to `spm12_script`
- **install_dir**: Directory to download SPM12

**Value**

Result of `run_matlab_script`

---

**spm12_condition**  
*Build Conditions for SPM12 first level model*

**Description**

Build Conditions for SPM12 first level model

**Usage**

```matlab
spm12_condition(name, onset, duration, time_mod_order = 0, param_mod = NULL, orth = TRUE)

spm12_condition_list(cond)
```

**Arguments**

- **name**: Name of the condition
- **onset**: vector of onset of the condition
- **duration**: vector of duration of the condition, must be the same length as onset
- **time_mod_order**: time modulation order. This option allows for the characterization of linear or nonlinear time effects. Zero means no modulation
- **param_mod**: parametric modulation. Not currently supported in `spm12r`
- **orth**: Orthogonalize the regressors within trial types.
- **cond**: List of conditions
Value

A list of objects, each with a name, onset, duration, and other condition values.

Examples

```r
res = spm12_condition(  
nname = "condition1",  
onset = c(0, 2, 4, 6, 8),  
duration = rep(1, 5) )  
print(res)  
L = list(  
cond1 = list(onset = c(0, 2, 4, 6, 8), duration = rep(1, 5)),  
cond2 = list(onset = c(0, 2, 4, 6, 8) + 1, duration = rep(1, 5))  
)  
res = spm12_condition_list(L)  
print(res)
```

---

**spm12_contrast**

*Build contrasts for SPM12 first level model*

**Description**

Build contrasts for SPM12 first level model

**Usage**

```r
spm12_contrast(name, weights, replicate = c("none", "repl", "replsc",  
"sess", "both", "bothsc"))  
spm12_contrast_list(cons, type = "T")
```

**Arguments**

- **name**: Name of the contrast
- **weights**: Weights of the contrast, must be the same length as the number of regressors
- **replicate**: If there are multiple sessions with identical conditions, one might want to specify contrasts which are identical over sessions. Options are no replication (none), replicate (repl), replicate + scale (replsc), create per session (sess), Both: Replicate + Create per session (both), Both: Replicate + Scale + Create per session (bothsc)
- **cons**: List of contrasts
- **type**: type of contrast, T-statistic or F-statistic

**Value**

A list of objects, each with a name and value
Examples

```r
res = spm12_contrast(name = "condition1", weights = c(1, rep(0, 8)))
print(res)
contrasts = list(
  name = "LeftHand",
  weights = c(1, rep(0, 7)),
  replicate = "none",
  type = "T" ),
  name = "RightHand",
  weights = c(0, 1, rep(0, 6)),
  replicate = "none",
  type = "T" ),
  name = "AllEffects",
  weights = rbind(
    c(1, rep(0, 7)),
    c(0, 1, rep(0, 6))
  ),
  replicate = "none",
  type = "F"
)
res = spm12_contrast_list(contrasts)
print(res)
```

Description

SPM12 fMRI Contrasts

Usage

```r
spm12_contrast_manager(..., add_spm_dir = TRUE,
  spmdir = spm_dir(verbose = verbose), clean = TRUE, verbose = TRUE,
  install_dir = NULL)

build_spm12_contrast_manager(spm, delete_existing = TRUE,
  contrast_list = NULL, ...)
```

Arguments

- ...: Arguments passed to `matlabbatch_to_script`
- add_spm_dir: Add SPM12 directory from this package
- spmdir: SPM dir to add, will use package default directory
- clean: Remove scripts from temporary directory after running
**spm12_contrast_query**

**verbose**
Print diagnostic messages

**install_dir**
directory to download SPM12

**spm**
Path to SPM.mat file

**delete_existing**
Delete existing contrasts

**contrast_list**
List of contrasts to pass to `spm12_contrast_list` for conversion

---

**Value**

A list of output and results

---

**spm12_contrast_query**

Build contrasts query for SPM12 results

---

**Description**

Build contrasts query for SPM12 results

**Usage**

```r
spm12_contrast_query(weights = Inf, name = "", threshold_type = c("FWE", "none", "FDR"), threshold = 0.05, number_extent_voxels = 0, mask_type = c("None", "Contrast", "Image"), mask_image = NULL, mask_contrast = NULL, mask_threshold = 0.05, mask_inclusive = TRUE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>weights</code></td>
<td>Weights of the contrast, must be the same length as the number of regressors</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Name of the contrast</td>
</tr>
<tr>
<td><code>threshold_type</code></td>
<td>Type of thresholding done, if any</td>
</tr>
<tr>
<td><code>threshold</code></td>
<td>Threshold value</td>
</tr>
<tr>
<td><code>number_extent_voxels</code></td>
<td>Number of voxel extent to call an area a cluster</td>
</tr>
<tr>
<td><code>mask_type</code></td>
<td>Type of mask, if any</td>
</tr>
<tr>
<td><code>mask_image</code></td>
<td>If <code>mask_type</code> = &quot;Image&quot;, then the filename of the mask</td>
</tr>
<tr>
<td><code>mask_contrast</code></td>
<td>Vector of weights for the contrast that will be used as the mask if <code>mask_type</code> = &quot;Contrast&quot;</td>
</tr>
<tr>
<td><code>mask_threshold</code></td>
<td>If <code>mask_type</code> = &quot;Contrast&quot;, the threshold for the mask</td>
</tr>
<tr>
<td><code>mask_inclusive</code></td>
<td>Is the mask inclusive? If FALSE, then values in the mask which are zero will be included in the output</td>
</tr>
<tr>
<td><code>cons</code></td>
<td>List of contrasts</td>
</tr>
</tbody>
</table>
Value

A list of objects, each with a titlestr (title string), contrast, threshold description, threshold value, extent of voxels, and a mask (if appropriate)

Examples

```r
res = spm12_contrast_query(name = "condition1", weights = 1)
print(res)
contrasts = list(
  list(name = "All Contrasts",
       weights = Inf
  ),
  list(name = "RightHand",
       weights = 2)
)
res = spm12_contrast_query_list(contrasts)
print(res)
```

**spm12_coregister.deprecated**

*SPM12 Coregister (Estimate and Reslice)*

Description

Performs SPM12 coregistration estimation and reslicing on an Image

Usage

```r
spm12_coregister.deprecated(fixed, moving, other.files = NULL,
    prefix = "r", add_spm_dir = TRUE, spmdir = spm_dir(),
    clean = TRUE, verbose = TRUE, outdir = NULL, install_dir = NULL,
    ...)```

Arguments

- **fixed**: File that is assumed fixed
- **moving**: moving file to be registered to fixed space
- **other.files**: Other files to register to fixed, in same space as moving file
- **prefix**: Prefix to append to front of image filename
- **add_spm_dir**: Add SPM12 directory from this package
- **spmdir**: SPM dir to add, will use package default directory
- **clean**: Remove scripts from temporary directory after running
- **verbose**: Print diagnostic messages
- **outdir**: Directory to copy results. If full filename given, then results will be in dirname(filename)
- **install_dir**: directory to download SPM12
- **...**: Arguments passed to run_spm12_script
Value

Result from run_matlab_script

---

**spm12_fmri_est**  
**SPM12 FMRI Estimation**

Description

SPM12 FMRI Estimation

Usage

```r
spm12_fmri_est(..., add_spm_dir = TRUE, spmdir = spm_dir(verbos = 
verbose), clean = TRUE, verbose = TRUE, install_dir = NULL)

build_spm12_fmri_est(spm, write_residuals = FALSE, 
method = c("Classical", "Bayesian", "Bayesian2"),
bayesian = list(space.volume.block_type = "Slices", signal = "UGL", ARP 
= 3, noise.UGL = 1, LogEv = "No", anova.first = "No", anova.second = 
"Yes", gcon = list(name = "", convec = numeric(0))), ...)
```

Arguments

- `...`: Arguments passed to `matlabbatch_to_script`
- `add_spm_dir`: Add SPM12 directory from this package
- `spmdir`: SPM dir to add, will use package default directory
- `clean`: Remove scripts from temporary directory after running
- `verbose`: Print diagnostic messages
- `install_dir`: Directory to download SPM12
- `spm`: Path to SPM.mat file
- `write_residuals`: Should residuals be written?
- `method`: Method for model estimation
- `bayesian`: If method = "Bayesian", this is for a 1st level model Bayesian estimation and this list specifies the parameters

Value

A list of output and results
Description

Performs SPM12 (Spatial) Normalization on an Image

Usage

```r
spm12_normalize(filename, other.files = NULL,
  bounding_box = matrix(c(-90, -126, -72, 90, 90, 108), nrow = 2, byrow =
  TRUE), biasreg = 0.001, biasfwhm = 60, regularization = c(0, 0.001,
  0.5, 0.05, 0.2), affine = c("mni", "eastern", "subj", "none", ""),
  smoothness = 0, sampling_distance = 3, voxel_size = c(2, 2, 2),
  interp = c("bspline4", "nearestneighbor", "trilinear",
  paste0("bspline", 2:3), paste0("bspline", 5:7)), add_spm_dir = TRUE,
  spmdir = spm_dir(verbose = verbose, install_dir = install_dir),
  clean = TRUE, verbose = TRUE, install_dir = NULL, ...)
```

Arguments

- **filename**: File to be normalized to the template
- **other.files**: Files to be written using the estimated normalization
- **bounding_box**: matrix (2x3) of the bounding box to use. Default is for MNI 2mm template size
- **biasreg**: Amount of bias regularization
- **biasfwhm**: FWHM of Gaussian smoothness of bias.
- **regularization**: parameters for warping regularization
- **affine**: Space to register the image to, using an affine registration
- **smoothness**: FWHM of smoothing done
- **sampling_distance**: Smoothness of the warping field. This is used to derive a fudge factor to account for correlations between neighboring voxels. Smoother data have more
- **voxel_size**: The voxel sizes (x, y & z, in mm) of the written normalised images.
- **interp**: Interpolator for sampling in fixed space
- **add_spm_dir**: Add SPM12 directory from this package
- **spmdir**: SPM dir to add, will use package default directory
- **clean**: Remove scripts from temporary directory after running
- **verbose**: Print diagnostic messages
- **install_dir**: directory to download SPM12
- **...**: Arguments passed to `run_spm12_script`

Value

List of output filenames
spm12_normalize_est  SPM12 Normalize (Estimate)

Description

Estimate SPM12 (Spatial) Normalization from image

Usage

```
spm12_normalize_est(filename, biasreg = 0.001, biasfwhm = 60,
     regularization = c(0, 0.001, 0.5, 0.05, 0.2), affine = c("mni",
     "eastern", "subj", "none", ""), smoothness = 0,
     sampling_distance = 3, add_spmdir = TRUE, spmdir = spmdir(verbose
     = verbose, install_dir = install_dir), clean = TRUE, verbose = TRUE,
     install_dir = NULL, ...)
```

Arguments

- **filename**: File to be normalized to the template
- **biasreg**: Amount of bias regularization
- **biasfwhm**: FWHM of Gaussian smoothness of bias.
- **regularization**: parameters for warping regularization
- **affine**: Space to register the image to, using an affine registration
- **smoothness**: FWHM of smoothing done
- **sampling_distance**: amount of smoothing of the warping field. This is used to derive a fudge factor to account for correlations between neighbouring voxels. Smoother data have more
- **add_spmdir**: Add SPM12 directory from this package
- **spmdir**: SPM dir to add, will use package default directory
- **clean**: Remove scripts from temporary directory after running
- **verbose**: Print diagnostic messages
- **install_dir**: directory to download SPM12
- **...**: Arguments passed to run_spm12_script

Value

Result from run_matlab_script
Description

Applies SPM12 (Spatial) Normalization to images

Usage

```r
spm12_normalize_write(deformation, other.files = NULL,
  bounding_box = matrix(c(-90, -126, -72, 90, 90, 108), nrow = 2, byrow =
  TRUE), voxel_size = c(2, 2, 2), interp = c("bspline4",
  "nearestneighbor", "trilinear", paste0("bspline", 2:3), paste0("bspline",
  5:7)), retimg = FALSE, reorient = FALSE, add_spm_dir = TRUE,
  spmdir = spm_dir(verbos = verbose, install_dir = install_dir),
  clean = TRUE, verbose = true, install_dir = NULL, ...)

build_spm12_normalize_write(deformation, other.files = NULL,
  bounding_box = matrix(c(-90, -126, -72, 90, 90, 108), nrow = 2, byrow =
  TRUE), voxel_size = c(2, 2, 2), interp = c("bspline4",
  "nearestneighbor", "trilinear", paste0("bspline", 2:3), paste0("bspline",
  5:7)), verbose = true, ...)
```

Arguments

- **deformation**: Filename of deformation (nifti)
- **other.files**: Files to be written using the estimated normalization
- **bounding_box**: matrix (2x3) of the bounding box to use. Default is for MNI 2mm template size
- **voxel_size**: The voxel sizes (x, y & z, in mm) of the written normalised images.
- **interp**: Interpolator for sampling in fixed space
- **retimg**: Logical indicating if image should be returned or result from `run_matlab_script`
- **reorient**: if retimg=TRUE pass to `readNIfTI`
- **add_spm_dir**: Add SPM12 directory from this package
- **spmdir**: SPM dir to add, will use package default directory
- **clean**: Remove scripts from temporary directory after running
- **verbose**: Print diagnostic messages
- **install_dir**: directory to download SPM12

Value

List of SPM object, results, and output filenames
Examples

dims = rep(10, 3)
temp_nii = array(rnorm(prod(dims)), dim = dims)
temp_nii = oro.nifti::nifti(temp_nii)
res = build_spm12_normalize_write(temp_nii,
other.files = temp_nii,
install.dir = tempdir())

Description

Performs SPM12 realignment estimation and reslicing on an Image

Usage

spm12_realign Deprecated(filename, fwhm = 5, register.to = c("first",
"mean"), reslice = c("all", "2:n", "all+mean", "mean"), prefix = "r",
add_spm_dir = TRUE, spmdir = spm_dir(verb = verb, install_dir =
install_dir), clean = TRUE, verb = TRUE, outdir = NULL,
install_dir = NULL, ...)

Arguments

filename Files to be realigned and resliced
fwhm Full-Width Half Max to smooth
register.to Should the files be registered to the first or the mean
reslice Options for reslicing all - all images in filename, 2:n - all images in filename
prefix:all+mean - all images and the mean, mean - mean only
prefix Prefix to append to front of image filename
add_spm_dir Add SPM12 directory from this package
spmdir SPM dir to add, will use package default directory
clean Remove scripts from temporary directory after running
verb Print diagnostic messages
outdir Directory to copy results. If full filename given, then results will be in dirname(filename)
install_dir directory to download SPM12
... Arguments passed to run_spm12_script

Value

Result from run_matlab_script
**spm12_realign_est**  
*SPM12 Realign (Estimate)*

**Description**
Performs SPM12 Realignment estimation on an Image

**Usage**
```
spm12_realign_est(filename, fwhm = 5, register_to = c("first", "mean"),
                  add_spm_dir = TRUE, spmdir = spm_dir(verbospe = verbose, install_dir =
                  install_dir), clean = TRUE, verbose = TRUE, outdir = NULL,
                  install_dir = NULL, ...)
```

**Arguments**
- `filename`: Files to be realigned
- `fwhm`: Full-Width Half Max to smooth
- `register_to`: Should the files be registered to the first or the mean
- `add_spm_dir`: Add SPM12 directory from this package
- `spmdir`: SPM dir to add, will use package default directory
- `clean`: Remove scripts from temporary directory after running
- `verbose`: Print diagnostic messages
- `outdir`: Directory to copy results. If full filename given, then results will be in `dirname(filename)`
- `install_dir`: Directory to download SPM12
- `...`: Arguments passed to `run_spm12_script`

**Value**
Character list of filenames from output

---

**spm12Regressor**  
*Build Regressors for SPM12 first level model*

**Description**
Build Regressors for SPM12 first level model

**Usage**
```
spm12Regressor(name, value, n_time_points)

spm12RegressorList(reg, n_time_points)
```
Arguments

- **name**: Name of the regressor
- **value**: Value of the regressor, must be the same length as `n_time_points`
- **n_time_points**: Number of time points for the analysis
- **reg**: List of regressors

Value

A list of objects, each with a name and value

Examples

```r
res = spm12_regressor(name = "condition1", value = c(rep(1, 10), rep(0, 10)), n_time_points = 20)
print(res)
L = list(
  cond1 = list(value = c(rep(1, 10), rep(0, 10)), n_time_points = 20),
  cond2 = list(value = c(rep(0, 10), rep(1, 10)), n_time_points = 20)
)
res = spm12_regressor_list(L, n_time_points = 20)
print(res)
```

Description

SPM12 Results Report

Usage

```r
spm12_results(..., add_spm_dir = TRUE, spmdir = spm_dir(verbose =
  verbose, install_dir = install_dir), clean = TRUE, verbose = TRUE,
  display = FALSE, desktop = FALSE, install_dir = NULL)

build_spm12_results(spm, units = c("Volumetric", "Scalp-Time",
  "Scalp-Frequency", "Time-Frequency", "Frequency-Frequency"),
  result_format = c("none", "ps", "eps", "png", "pdf", "jpg", "tif",
  "fig", "csv", "nidi"), write_images = c("none", "threshold_spm",
  "binary_clusters", "nary_clusters"), contrast_list = NULL,
  image_basename = NULL, ...)
```
Arguments

... Arguments passed to `matlabbatch_to_script`
add_spm_dir Add SPM12 directory from this package
spmdir SPM dir to add, will use package default directory
clean Remove scripts from temporary directory after running
verbose Print diagnostic messages
display Run `run_matlab_script` with the display option on, which is required in some cases of output. May fail if no displays are available.
desktop Run `run_matlab_script` with the desktop option on. May fail if no displays are available.
install_dir directory to download SPM12
spm Path to SPM.mat file
units Units of the data
result_format Output format to save, if any
write_images Type of images to write out
contrast_list List of contrasts (or just one), to pass to `spm12_contrast_query_list`
image_basename Base stub of filenames, if any are to be written out

Value

A list of output and results

---

`spm12_script` *Find SPM12 Script*

---

Description

Copies the SPM12 script from the scripts directory to a temporary file

Usage

```
spm12_script(script_name, outdir = tempdir())
```

Arguments

- `script_name` Name of the script filename without ".m" ext
- `outdir` Path to copy scripts and run

Value

Character vector of script paths

Examples

```
spm12_script(script_name = "Segment")
```
**Description**

Performs SPM12 Segmentation on an Image

**Usage**

```plaintext
spm12_segment(filename, set_origin = TRUE, biasreg = 0.001, biasfwhm = 60, native = TRUE, dartel = FALSE, modulated = FALSE, unmodulated = FALSE, bias_field = FALSE, bias_corrected = FALSE, n_gaus = c(1, 1, 2, 3, 4, 2), smoothness = 0, sampling_distance = 3, regularization = c(0, 0.001, 0.5, 0.05, 0.2), affine = c("mni", "eastern", "subj", "none"), mrf = 1, def_inverse = TRUE, def_forward = TRUE, warp_cleanup = c("light", "none", "thorough"), retimg = TRUE, add_spm_dir = TRUE, spmdir = spm_dir, verbose = verbose, install_dir = install_dir, clean = TRUE, verbose = TRUE, reorient = FALSE, install_dir = NULL, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>File to be segmented</td>
</tr>
<tr>
<td>set_origin</td>
<td>Run <code>acpc_reorient</code> on image first. Warning, this will set the orientation differently</td>
</tr>
<tr>
<td>biasreg</td>
<td>Amount of bias regularization</td>
</tr>
<tr>
<td>biasfwhm</td>
<td>FWHM of Gaussian smoothness of bias.</td>
</tr>
<tr>
<td>native</td>
<td>Keep tissue class image (c*) in alignment with the original.</td>
</tr>
<tr>
<td>dartel</td>
<td>Keep tissue class image (rc*) that can be used with the Dartel toolbox.</td>
</tr>
<tr>
<td>modulated</td>
<td>Keep modulated images. Modulation is to compensate for the effect of spatial normalisation.</td>
</tr>
<tr>
<td>unmodulated</td>
<td>Keep unmodulated data</td>
</tr>
<tr>
<td>bias_field</td>
<td>save a bias corrected version of your images</td>
</tr>
<tr>
<td>bias_corrected</td>
<td>save an estimated bias field from your images</td>
</tr>
<tr>
<td>n_gaus</td>
<td>The number of Gaussians used to represent the intensity distribution for each tissue class. Can be 1:8 or infinity</td>
</tr>
<tr>
<td>smoothness</td>
<td>FWHM of smoothing done</td>
</tr>
<tr>
<td>sampling_distance</td>
<td>smoothingess of the warping field. This is used to derive a fudge factor to account for correlations between neighbouring voxels. Smoother data have more</td>
</tr>
<tr>
<td>regularization</td>
<td>parameters for warping regularization</td>
</tr>
<tr>
<td>affine</td>
<td>Space to register the image to, using an affine registration</td>
</tr>
</tbody>
</table>
**Description**

Performs SPM12 slice timing correction on images

**Usage**

```python
spm12_slice_timing(filename, time_points = NULL, nslices, tr, ta = tr - tr/nslices, slice_order = 1:nslices, ref_slice, prefix = "a", add_spm_dir = TRUE, spmdir = spm_dir(observe = verbose, install_dir = install_dir), clean = TRUE, verbose = TRUE, outdir = tempdir(), retimg = TRUE, reorient = FALSE, install_dir = NULL, ...)

build_spm12_slice_timing(filename, time_points = NULL, nslices, tr, ta = tr - tr/nslices, slice_order = 1:nslices, ref_slice, prefix = "a", add_spm_dir = TRUE, spmdir = spm_dir(observe = verbose, install_dir = install_dir), clean = TRUE, verbose = TRUE, outdir = tempdir(), install_dir = NULL, ...)
```
Arguments

filename: Files to be slice-time corrected

time_points: A vector of time points to run slice-time correction. If filename is a 4D file, then will do all the time points. Otherwise, filename must be a character vector of 3D files or a list of 3D nifti objects.

nslices: Number of slices in the images

tr: Repetition time (in seconds)

ta: Time between the first and the last slice within one scan

slice_order: Order slices were taken (if not specified, assumed ascending), bottom slice = 1

ref_slice: Reference slice

prefix: Prefix to append to front of image filename

add_spm_dir: Add SPM12 directory from this package

spmdir: SPM dir to add, will use package default directory

clean: Remove scripts from temporary directory after running

verbose: Print diagnostic messages

outdir: Directory to copy results

reorient: (logical) If retime, should file be reoriented when read in? Passed to `readNIfTI`.

install_dir: Directory to download SPM12

...: Arguments passed to `run_spm12_script`

Value

List of results, the SPM job, the script and the outfile

Description

Performs SPM12 slice timing correction on images

Usage

```r
spm12_slice_timing.deprecated(filename, time_points = NULL, nslices, tr, ta = tr - tr/nslices, slice_order = 1:nslices, ref_slice, prefix = "a", add_spm_dir = TRUE, spmdir = spm_dir(verbosenum = verbose = TRUE, install_dir = NULL), clean = TRUE, verbosenum = TRUE, outdir = tempdir(), retimg = FALSE, reorient = FALSE, install_dir = NULL, ...)
```
**Arguments**

- **filename**: Files to be slice-time corrected
- **time_points**: Time points to run slice-time correction. If `filename` is a 4D file, then will do all the time points. Otherwise, `filename` must be a character vector of 3D files or a list of 3D nifti objects.
- **nslices**: Number of slices in the images
- **tr**: Repetition time (in seconds)
- **ta**: Time between the first and the last slice within one scan
- **slice_order**: Order slices were taken (if not specified, assumed ascending), bottom slice = 1
- **ref_slice**: Reference slice
- **prefix**: Prefix to append to front of image filename
- **add_spm_dir**: Add SPM12 directory from this package
- **spmdir**: SPM dir to add, will use package default directory
- **clean**: Remove scripts from temporary directory after running
- **verbose**: Print diagnostic messages
- **outdir**: Directory to copy results
- **reorient**: (logical) return image of class nifti
- **reinit**: (logical) If reinit, should file be reoriented when read in? Passed to `readNIFTI`.
- **install_dir**: Directory to download SPM12
- **...**: Arguments passed to `run_spm12_script`

**Value**

Output filenames

**Description**

Performs SPM12 Smoothing on an Image

**Usage**

```r
spm12_smooth(filename, reinit = FALSE, fwhm = 8, dtype = c("SAME", "UINT8", "INT16", "INT32", "FLOAT32", "FLOAT64"), implicit_mask = FALSE, prefix = "s", add_spm_dir = TRUE, spmdir = spm_dir(verbosse = verbose, install_dir = install_dir), clean = TRUE, verbose = TRUE, reorient = FALSE, install_dir = NULL, ...)
```
$spm_{bwlabel}$

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>File to be smoothed</td>
</tr>
<tr>
<td>retimg</td>
<td>Logical indicating if image should be returned or result from $run_{matlab_script}$</td>
</tr>
<tr>
<td>fwhm</td>
<td>Full-Width Half Max to smooth</td>
</tr>
<tr>
<td>dtype</td>
<td>data type for the output format</td>
</tr>
<tr>
<td>implicit_mask</td>
<td>Should an implicit mask be used. An &quot;implicit mask&quot; is a mask implied by a particular voxel value (0 for images with integer type, NaN for float images).</td>
</tr>
<tr>
<td>prefix</td>
<td>Prefix to append to front of image filename</td>
</tr>
<tr>
<td>add_spm_dir</td>
<td>Add SPM12 directory from this package</td>
</tr>
<tr>
<td>spmdir</td>
<td>SPM dir to add, will use package default directory</td>
</tr>
<tr>
<td>clean</td>
<td>Remove scripts from temporary directory after running</td>
</tr>
<tr>
<td>verbose</td>
<td>Print diagnostic messages</td>
</tr>
<tr>
<td>reorient</td>
<td>if retimg=TRUE pass to $readNIfTI$</td>
</tr>
<tr>
<td>install_dir</td>
<td>directory to download SPM12</td>
</tr>
<tr>
<td>...</td>
<td>Arguments passed to $run_{spm12_script} readNIfTI</td>
</tr>
</tbody>
</table>

**Value**

Result from $run_{matlab\_script}$ or nifti file, depending on retimg

---

$spm_{bwlabel}$ $SPM \text{ BWLabel Clusters of Certain Size}$

**Description**

Get Cluster of certain size from $spm_{bwlabel}$

**Usage**

```plaintext
spm_{bwlabel}(infile, outfile = NULL, retimg = TRUE, k = NULL, 
topN = NULL, margin = NULL, binary = TRUE, 
spmdir = spm_dir(verboste = verbose, install_dir = install_dir), 
reorient = FALSE, verbose = TRUE, install_dir = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>infile</td>
<td>input filename</td>
</tr>
<tr>
<td>outfile</td>
<td>output filename</td>
</tr>
<tr>
<td>retimg</td>
<td>Return the image instead of matlab output</td>
</tr>
<tr>
<td>k</td>
<td>Minimum cluster size needed</td>
</tr>
<tr>
<td>topN</td>
<td>Top number of clusters kept (used if k is NULL)</td>
</tr>
<tr>
<td>margin</td>
<td>Margin to loop over if wanted in 2D</td>
</tr>
</tbody>
</table>
Binary (logical) Should the result be binary or numbered with clusters.

spmdir SPM directory (for MATLAB)

reorient If retimg, then this argument is passed to readNIFTI

verbose Print Diagnostics

install_dir directory to download SPM12

**Value**

Output from `run_matlab_script` or `nifti` object, depending on retimg

Result from `run_matlab_script`

**Note**

Taken from http://en.wikibooks.org/wiki/SPM/How-to#How_to_remove_clusters_under_a_certain_size_in_a_binary_mask.

---

**Description**

Returns the SPM12 directory

**Usage**

```
spm_dir(verbosE = FALSE, install_dir = NULL)
```

**Arguments**

- `verbose` print diagnostic messages, passed to `install_spm12`
- `install_dir` directory to download SPM12

**Value**

Character vector of spm12 paths
**spm_directory**

<table>
<thead>
<tr>
<th>Description</th>
<th>SPM Directory</th>
</tr>
</thead>
</table>

**Usage**

```
spm_directory(spm)
```

**Arguments**

```
spm                  Path to SPM.mat file
```

**Value**

Path to working directory

---

**spm_probs_to_seg**  
**SPM Probability to Hard Segmentation**

<table>
<thead>
<tr>
<th>Description</th>
<th>Converts probability images from SPM segmentation to a hard, choose-one segmentation</th>
</tr>
</thead>
</table>

**Usage**

```
spm_probs_to_seg(img, ties.method = c("first", "last", "random"))
```

**Arguments**

```
img                        list of images for probabilities for each class
ties.method               a character string specifying how ties are handled. See max.col. Note, order of ties is different than max.col.
```

**Value**

Object of class nifti

**Examples**

```
## Not run:
spm_seg = spm12_segment(image)$outfiles
seg = spm_probs_to_seg(spm_seg)

## End(Not run)
```
### spm_version

**SPM Version**

**Description**

SPM Version

**Usage**

```r
spm_version()
```

```r
spm12_version()
```

**Value**

Character vector of length 1

**Examples**

```r
if (matlab::have_matlab()) {
  spm_version()
}
```

---

### spm_xmat

**SPM X Matrix**

**Description**

SPM X Matrix

**Usage**

```r
spm_xmat(spm)
```

**Arguments**

```r
spm Path to SPM.mat file
```

**Value**

Matrix of values
Index

acpc_reorient, 2, 27
add_spm12_dir (add_spm_dir), 3
add_spm_dir, 3

build_spm12_contrast_manager
  (spm12_contrast_manager), 16
build_spm12_coregister, 4
build_spm12_coregister_estimate
  (build_spm12_coregister), 4
build_spm12_coregister_reslice
  (build_spm12_coregister), 4
build_spm12_first_level, 6
build_spm12_first_level_spec, 6, 7
build_spm12_fmri_est, 6
build_spm12_fmri_est (spm12_fmri_est), 19
build_spm12_normalize_write
  (spm12_normalize_write), 22
build_spm12_realign, 8
build_spm12_results (spm12_results), 25
build_spm12_script (run_spm12_script), 13
build_spm12_slice_timing
  (spm12_slice_timing), 28
bwconncomp, 10

cHECKnii, 11

cHECKnii_check, 11

group_xyz, 11

install_spm12, 12, 32

matplotlib_batch_job, 13
matplotlib_batch_job (run_matlabbatch), 13
matplotlib_batch_to_script, 16, 19, 26
matplotlib_batch_to_script
  (run_matlabbatch), 13
max.col, 33

ntime_points, 12

readNIfTI, 22, 28–31
run_matlab_code, 3
run_matlab_script, 13, 14, 22, 26, 28, 31, 32
run_matlabbatch, 5, 13
run_spm12_script, 8, 9, 13, 18, 20–24, 28–31

spm12_condition, 7, 14
spm12_condition_list (spm12_condition), 14

spm12_contrast, 15
spm12_contrast_list, 17
spm12_contrast_list (spm12_contrast), 15
spm12_contrast_manager, 16
spm12_contrast_query, 17
spm12_contrast_query_list, 26
spm12_contrast_query_list
  (spm12_contrast_query), 17
spm12_coregister
  (build_spm12_coregister), 4
spm12_coregister deprecated, 18
spm12_coregister_estimate
  (build_spm12_coregister), 4
spm12_coregister_reslice
  (build_spm12_coregister), 4
spm12_coregister_wrapper
  (build_spm12_coregister), 4
spm12_first_level
  (build_spm12_first_level), 6
spm12_first_level Spec
  (build_spm12_first_level Spec), 7
spm12_fmri_est, 19
spm12_normalize, 20
spm12_normalize_est, 21
spm12_normalize_write, 22
spm12_realign (build_spm12_realign), 8
spm12_realign deprecated, 23
spm12_realign_est, 24
spm12_regressor, 7, 24
spm12_regressor_list(spm12_regressor), 24
spm12_results, 25
spm12_script, 13, 14, 26
spm12_segment, 27
spm12_slice_timing, 28
spm12_slice_timing.deprecated, 29
spm12_smooth, 30
spm12_version(spm_version), 34
spm_bwlabel, II, 31
spm_dir, 32
spm_directory, 33
spm_probs_to_seg, 33
spm_version, 34
spm_xmat, 34
system.file, 12