MS Lesion Segmentation

Kristin Linn

2017-03-30

Background

- Multiple sclerosis is a chronic disease of the central nervous system (brain, spinal chord, optic nerves)
- Symptoms can include blurred vision, loss of balance, poor coordination, slurred speech, tremors, numbness, extreme fatigue, problems with memory and concentration, paralysis, and blindness.
- Affects more than 2.3 million worldwide.
- Affects more women than men (approximately 2:1)

Background

- Multiple sclerosis lesions in the brain are areas of active inflamation, demylenation, or permanent tissue damage.
- MRI is well-suited for assessing lesion burden (volume and patterns) because lesions appear as hyperintensities on FLAIR, T2-w, and PD images and as hypointensities on T1-w images.
- Obtaining manual lesion segmentations is often resource intensive, so accurate and efficient methods for automatic segmentation are necessary for scalability and research progress.

Goals of this tutorial

- Apply OASIS (Sweeney et al. 2013), an automatic lesion segmentation model, to obtain predicted lesion probability maps.
- Compare the results using the default OASIS settings to those obtained after re-training the model using our data.

Loading Data

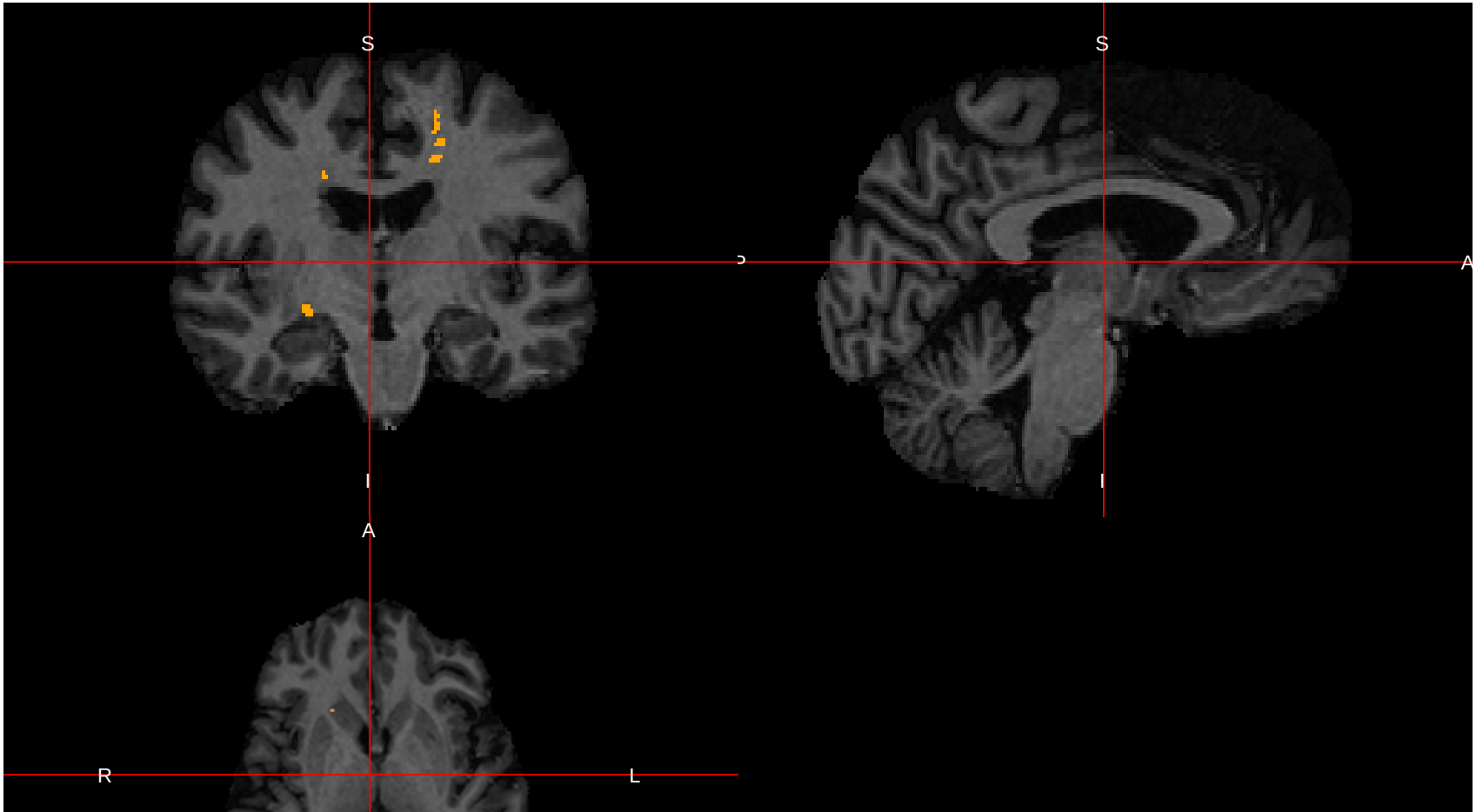
- Let's read in the images, brain masks, and manual lesion segmentations.

```
library(ms.lesion)
library(neurobase)
library(fslr)
library(scales)
library(oasis)
tr_files = get_image_filenames_list_by_subject(group = "tra
ts_files = get_image_filenames_list_by_subject(group = "tes
tr_t1s = lapply(tr_files, function(x) readnii(x["MPRAGE"]))
tr_t2s = lapply(tr_files, function(x) readnii(x["T2"]))
tr_flairs = lapply(tr_files, function(x) readnii(x["FLAIR"]
tr_pds = lapply(tr_files, function(x) readnii(x["PD"]))
tr_masks = lapply(tr_files, function(x) readnii(x["Brain_Ma
tr_golds = lapply(tr_files, function(x) readnii(x["mask2"])
ts_t1s = lapply(ts_files, function(x) readnii(x["MPRAGE"]))
ts_t2s = lapply(ts_files, function(x) readnii(x["T2"]))
```

Visualization

- Here's the T1 volume for training subject 05 with the 'gold standard' manual lesion segmentation overlayed.

```
les_mask = tr_golds$training05
ortho2(tr_t1s$training05, les_mask, col.y = "orange")
```



MS Lesion Segmentation with OASIS

- OASIS is Automated Statistical Inference for Segmentation (Sweeney et al. 2013)
- The OASIS algorithm takes FLAIR, T1, T2, and PD images from patients with multiple sclerosis (MS) and produces OASIS probability maps of MS lesion presence, which can be thresholded into a binary lesion segmentation.
- OASIS uses logistic regression of the labels on the images, smoothed versions of the images, and some interaction terms

Default OASIS Model

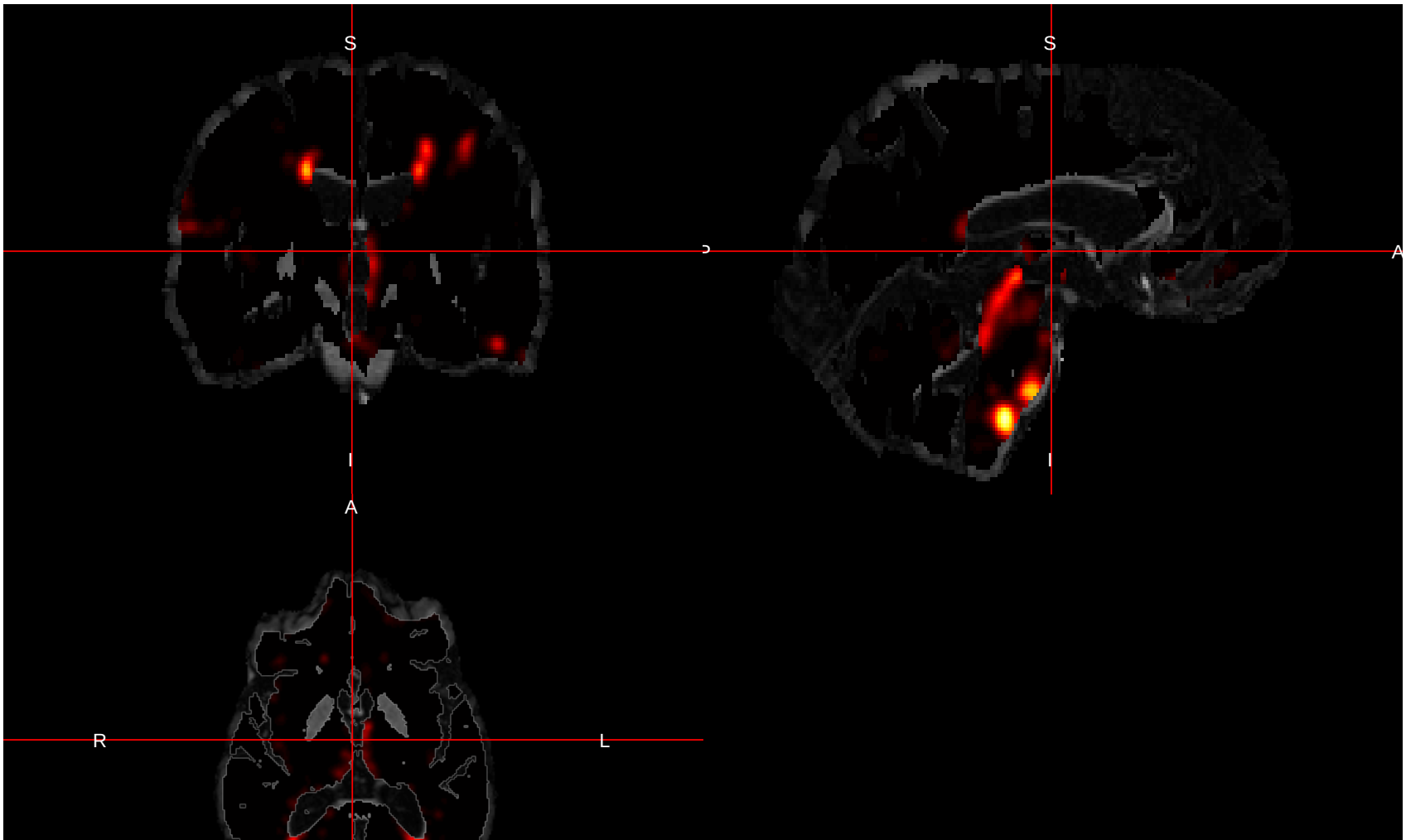
- The OASIS library comes with default parameters that can be used to generate probability maps for new test subjects
- Here we apply the default model to obtain OASIS probability maps for the test subjects.

```
default_predict_ts = function(x){
  res = oasis_predict(
      flair=ts_flairs[[x]], t1=ts_t1s[[x]],
      t2=ts_t2s[[x]], pd=ts_pds[[x]],
      brain_mask=ts_masks[[x]],
      preproc=FALSE, normalize=TRUE,
      model=oasis::oasis_model)
  return(res)
}
default_probs_ts = lapply(1:3, default_predict_ts)
```

Vizualization

- Let's look at the probability map for test subject 01:

```
les_mask = default_ts[[1]]
ortho2(ts_t1s$test01, les_mask)
```



Thresholding

- To get a final estimated segmentation, we must choose a cutoff to binarize the OASIS probability maps.
- The `binary` argument in the `oasis_predict` function is FALSE by default, resulting in the output being the probability map
- Setting binary=TRUE will return the thresholded version, using the input to the `threshold` argument (default = 0.16).
- In practice, we might want to use a grid search over thresholds and cross validation to choose the cutoff.

Default OASIS Model

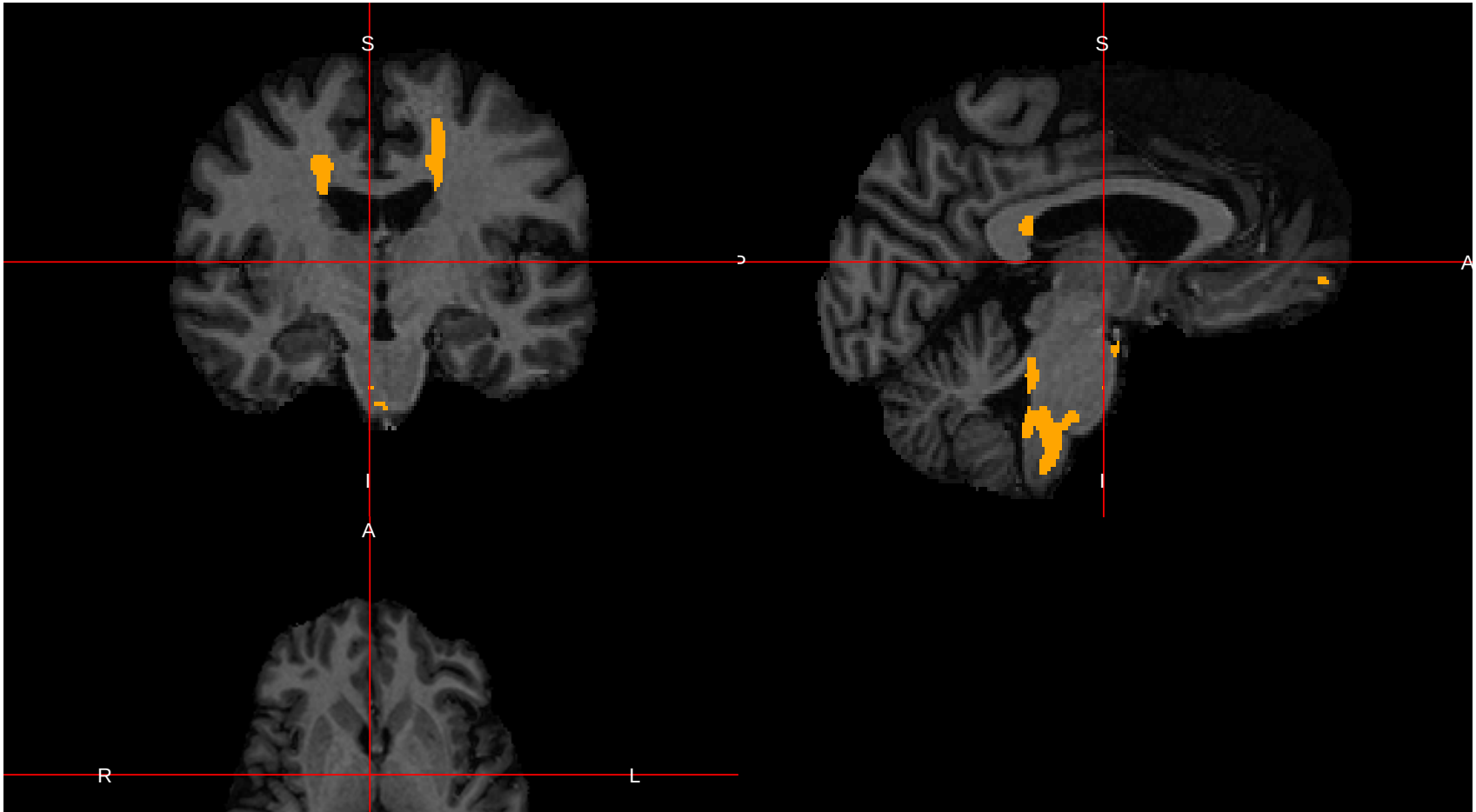
- To evaluate how the default model performs, we need to compare the predictions to a gold standard.
- Let's therefore obtain OASIS probability maps for our training subjects.
- We will use the default threshold to binarize.

```
default_predict_tr = function(x){
  res = oasis_predict(
      flair=tr_flairs[[x]], t1=tr_t1s[[x]],
      t2=tr_t2s[[x]], pd=tr_pds[[x]],
      brain_mask=tr_masks[[x]],
      preproc=FALSE, normalize=TRUE,
      model=oasis::oasis_model, binary=TRUE)
  return(res)
}
default_probs_tr = lapply(1:5, default_predict_tr)
```

Default OASIS Model Results

- Here's the T1 volume for training subject 05 with the OASIS segmentation overlayed.

```
les_mask = default_tr[[5]]
ortho2(tr_t1s$training05, les_mask, col.y = "orange")
```



Default OASIS Model Results

- Let's see how well OASIS corresponds with the gold standard manual segmentations for the training subjects.

```
tbls = lapply(1:5, function(x) table(c(tr_golds[[x]]), c(de
lapply(tbls, function(x) (2*x[2,2])/(2*x[2,2] + x[1,2] + x
```

```
[[1]]
[1] 0.4938917

[[2]]
[1] 0.6860189

[[3]]
[1] 0.4316734

[[4]]
[1] 0.2891665
```

Improving Results

- We might improve the results by re-training the OASIS model using our five training subjects.
- To retrain the model using new data, binary masks of 'gold standard' lesion segmentations are needed and should be in T1 space.

Making OASIS data frames

- OASIS requires a particular data frame format
- OASIS has an option to preprocess your data for you (preproc)
- OASIS has an option to normalize the intensities of your data for you using a whole-brain normalization (normalize)
- `make_df()` below is a helper function

```
library(oasis)
make_df = function(x){
  res = oasis_train_dataframe(
      flair=tr_flairs[[x]], t1=tr_t1s[[x]],
      t2=tr_t2s[[x]], pd=tr_pds[[x]],
      gold_standard=tr_golds[[x]],
      brain_mask=tr_masks[[x]],
      preproc=FALSE, normalize=TRUE,
      return_preproc=FALSE)
  return(res$oasis_dataframe)
}
oasis_dfs = lapply(1:5, make_df)
```

Training OASIS

- The function `oasis_training` takes the data frames we made and fits a logistic regression, where the outcome vector consists of all subjects' voxel-level data (top 85% in intensity)
- The function `do.call` is a useful R function that applies the function named in the first argument to all elements of the list specified in the second argument.

```
model = do.call("oasis_training", oasis_dfs)
```

References

Sweeney, Elizabeth M, Russell T Shinohara, Navid Shiee, Farrah J Mateen, Avni A Chudgar, Jennifer L Cuzzocreo, Peter A Calabresi, Dzung L Pham, Daniel S Reich, and Ciprian M Crainiceanu. 2013. "OASIS Is Automated Statistical Inference for Segmentation, with Applications to Multiple Sclerosis Lesion Segmentation in MRI." *NeuroImage: Clinical* 2. Elsevier: 402–13.