Supplement: Skull Stripping CT data

Goal

In this tutorial, we will discuss skull-stripping (or brain-extracting) x_ray computed tomography (CT) scans. We will use data from TCIA (http://www.cancerimagingarchive.net/) as there is a great package called `TCIApathfinder` to interface with TCIA.

Using TCIApathfinder

In order to use `TCIApathfinder`, please see the vignette to obtain API keys. Here we will look at the collections:

```
library(TCIApathfinder)
library(dplyr)
series_instance_uid = "1.3.6.1.4.1.14519.5.2.1.2857.3707.893926543922125108620513439908"
download_unzip_series = function(series_instance_uid,
                                 verbose = TRUE) {
  tdir = tempfile()
  dir.create(tdir, recursive = TRUE)
  tfile = tempfile(fileext = ".zip")
  tfile = basename(tfile)
  if (verbose) {
    message("Downloading Series")
  }
  res = save_image_series(
    series_instance_uid = series_instance_uid,
    out_dir = tdir,
    out_file_name = tfile)
  if (verbose) {
    message("Unzipping Series")
  }
  stopifnot(file.exists(res$out_file))
  tdir = tempfile()
  dir.create(tdir, recursive = TRUE)
  res = unzip(zipfile = res$out_file, exdir = tdir)
  L = list(files = res,
           dirs = unique(dirname(normalizePath(res))))
  return(L)
}
# Download and unzip the image series

file_list = download_unzip_series(
  series_instance_uid = series_instance_uid)
```

```
Downloading Series
```

```
Unzipping Series
```

Converting DICOM to NIfTI

We will use `dcm2niix` to convert from DICOM to NIfTI. The function `dcm2niix` is wrapped in `dcm2niir`. We will use `dcm2niir::dcm2nii` to convert the file. We use `check_dcm2nii` to grab the relevant output files:

```
library(dcm2niir)
dcm_result = dcm2nii(file_list$dirs)
```

```
#Copying Files

# Converting to nii

'/Library/Frameworks/R.framework/Versions/3.5/Resources/library/dcm2niir/dcm2niix' -9 -z y -f %p_%t_%s
```

```
dcm_result$nii_after
```

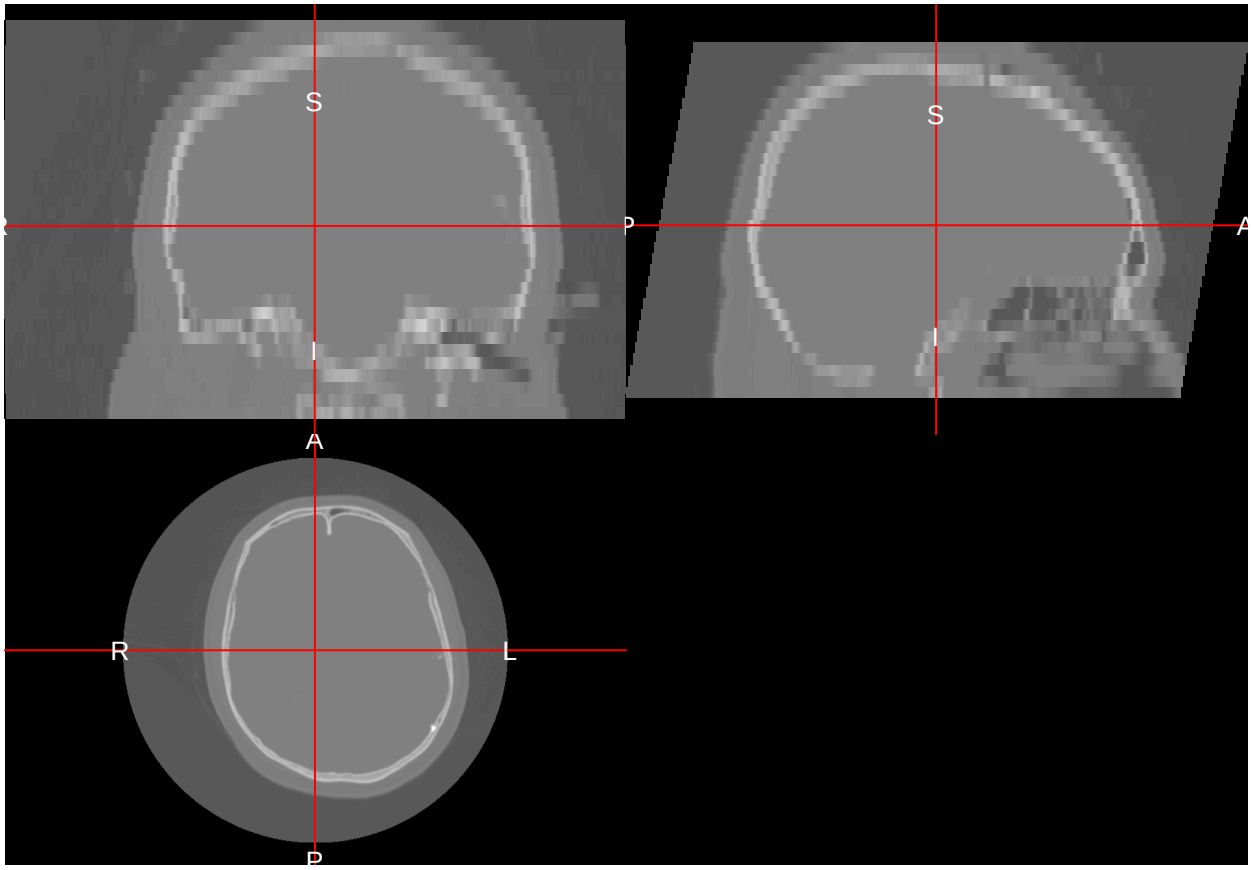
```
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b67343fb7a/HEAD_STD_20010124161
[2] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b67343fb7a/HEAD_STD_20010124161
```

```
result = check_dcm2nii(dcm_result)
result
```

```
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b67343fb7a/HEAD_STD_20010124161
attr(,"json_file")
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b67343fb7a/HEAD_STD_20010124161
```

Here we read the data into `R` into a `nifti` object:

```
library(neurobase)
img = readnii(result)
ortho2(img)
```

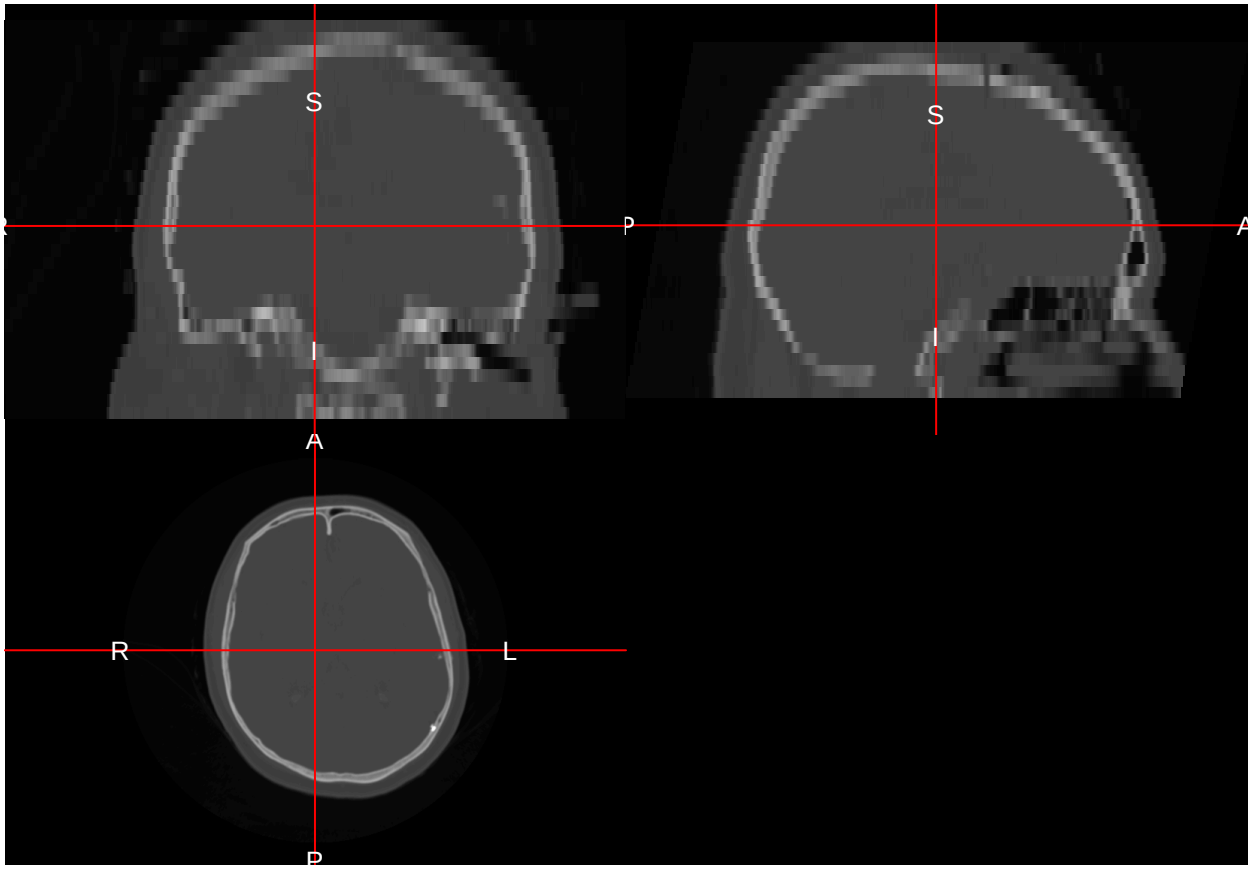



```
range(img)
```

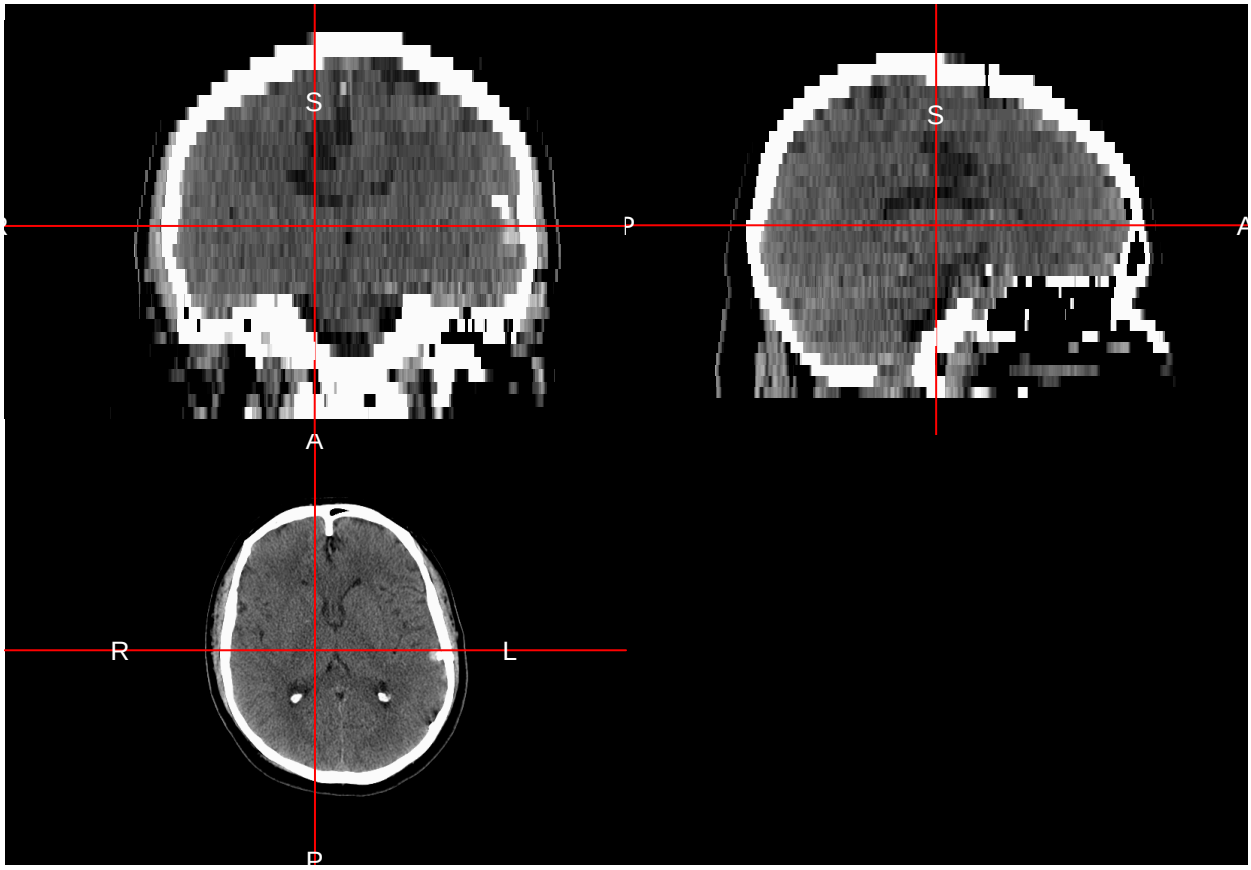
```
[1] -3024  3071
```

Here we will use `neurobase::rescale_img` to make sure the minimum is −1024 and the maximum is 3071. The minimum can be lower for areas outside the field of view (FOV). Here we plot the image and the Winsorized version to see the brain tissue:

```
img = rescale_img(img, min.val = -1024, max.val = 3071)
ortho2(img)
```




```
ortho2(img, window = c(0, 100))
```



Skull Strip

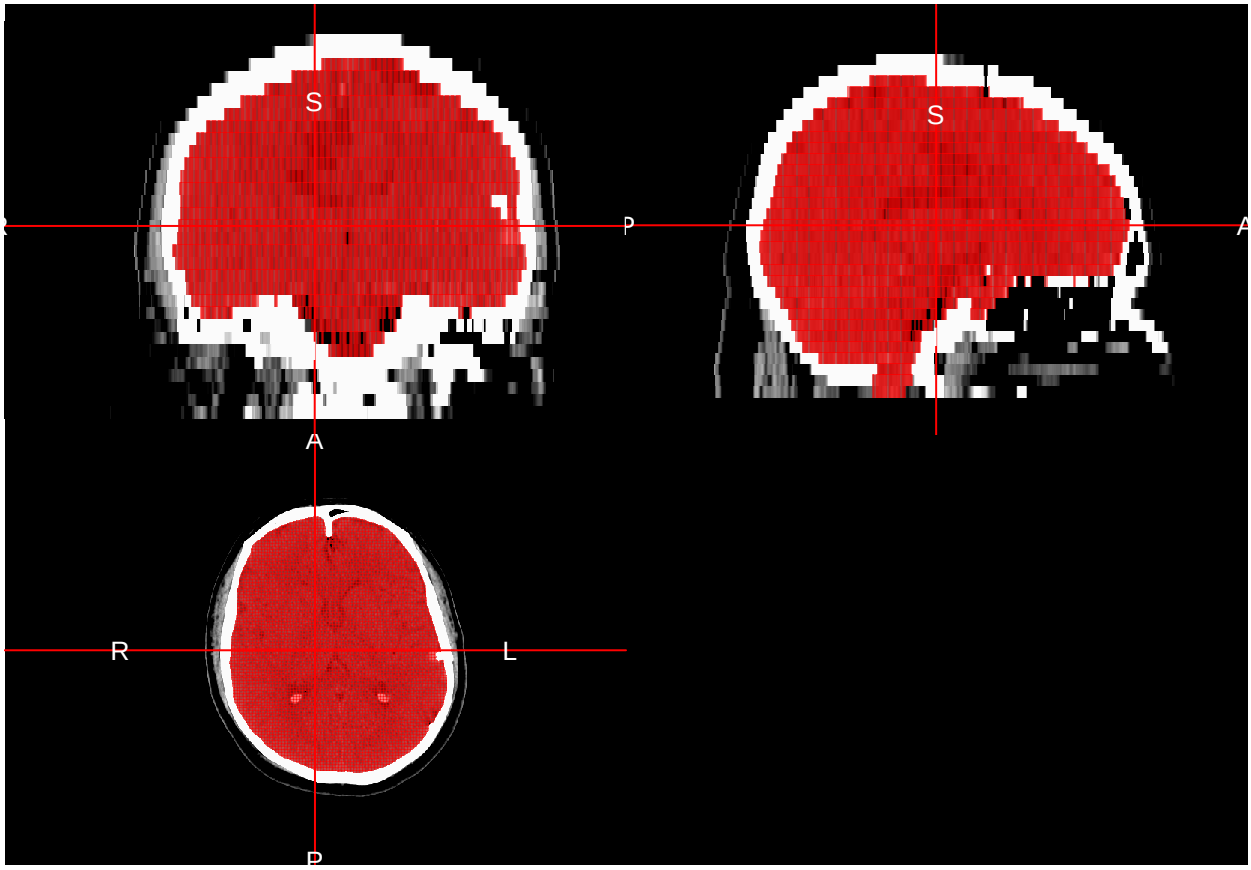
We can skull strip the image using `CT_Skull_Strip` or `CT_Skull_Stripper`. The `CT_Skull_Stripper` has a simple switch to use `CT_Skull_Strip` or `CT_Skull_Strip_robust`.

```
library(ichseg)
ss = CT_Skull_Strip(img, verbose = FALSE)
```

```
Warning in get.fsl(): Setting fsl.path to /usr/local/fsl
```

```
Warning in get.fsloutput(): Can't find FSLOUTPUTTYPE, setting to NIFTI_GZ
```

```
ortho2(img, ss > 0,
       window = c(0, 100),
       col.y = scales::alpha("red", 0.5))
```

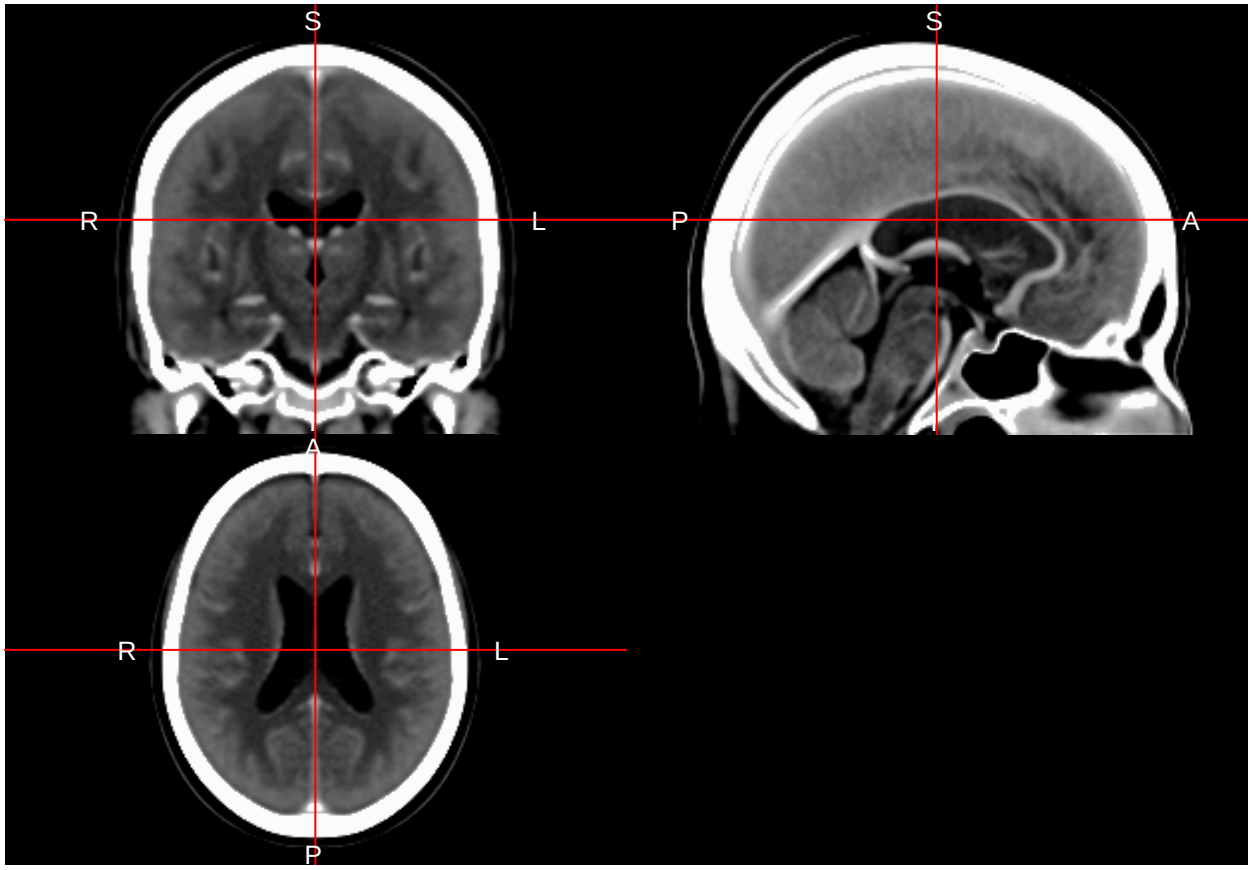


The `CT_Skull_Strip_robust` function does 2 neck removals using `remove_neck` from `extrantsr` and then find the center of gravity (COG) twice to make sure the segmentation focuses on the head. In some instances, the whole neck is included in the scan, such as some of the head-neck studies in TCIA.

Registration

Here we register the image to the template image from Rorden (2012). We will use the `registration` function from the `extrantsr` package. The `extrantsr` package uses the `ANTsR` package to perform the registration, and simply wraps multiple commands together. We will use a Symmetric Normalization (SyN) type of registration, which first uses an affine registration, then combines it with a symmetric non-linear diffeomorphism. The output file `reg$outfile` is the registered image.

```
template_image = ichseg::ct_template(type = "image")
ortho2(template_image, window = c(0, 100))
```




```
reg = extrantsr::registration(
  img, template.file = template_image,
  typeofTransform = "SyN",
  interpolator = "Linear")
```

```
# Running Registration of file to template

# Applying Registration output is

$fwdtransforms
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b6340605961Warp.nii.gz"
[2] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b6340605960GenericAffine.mat"

$invtransforms
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b6340605960GenericAffine.mat"
[2] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b6340605961InverseWarp.nii.gz"

$prev_transforms
character(0)

# Applying Transformations to file

 [1] "-d"
 [2] "3"
 [3] "-i"
 [4] "<pointer: 0x7f945e53cda0>"
 [5] "-o"
 [6] "<pointer: 0x7f945e4b1520>"
```

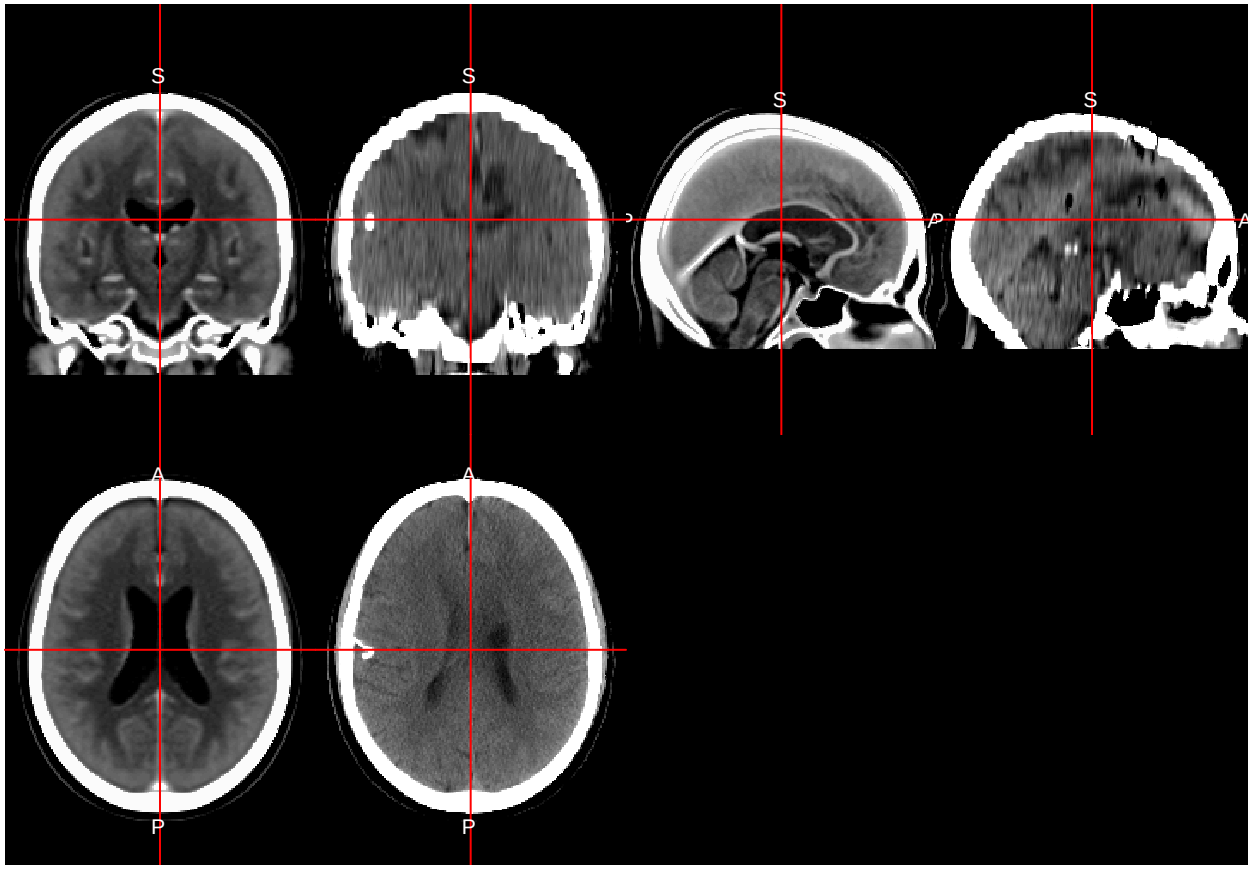
```
 [7] "-r"
 [8] "<pointer: 0x7f947a3a0fe0>"
 [9] "-n"
[10] "linear"
[11] "-t"
[12] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b6340605961Warp.nii.gz"
[13] "-t"
[14] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b6340605960GenericAffine.mat"
```

```
# Writing out file
```

```
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b61ea25720.nii.gz"
```

```
# Reading data back into R
```

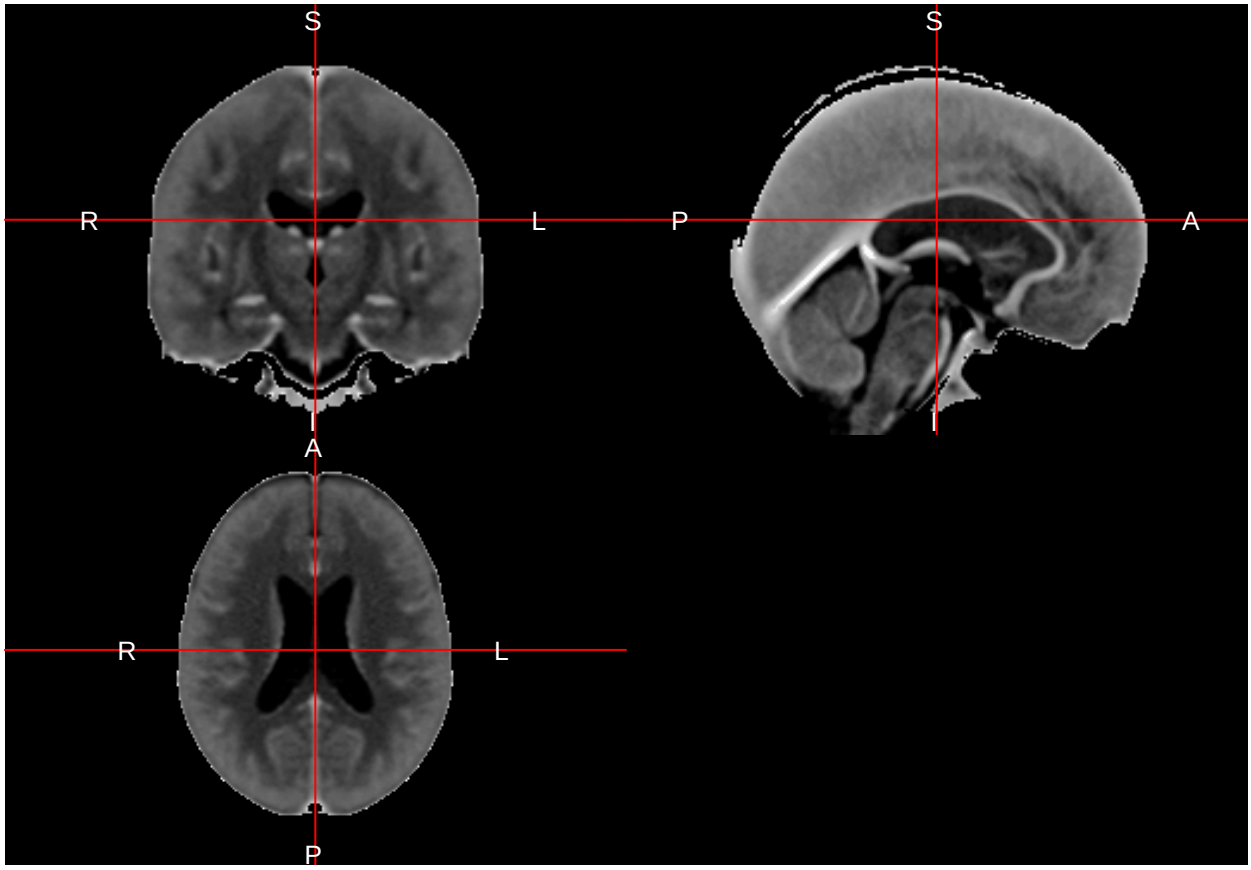
```
wimg = window_img(reg$outfile, window = c(0, 100))
double_ortho(template_image, wimg, window = c(0, 100))
```



We see relatively good alignment between the template image (left) and the registered image (right)

Here we will use the skull-stripped template and perform the same registration with the skull-stripped image.

```
template_brain = ichseg::ct_template(type = "brain")
ortho2(template_brain, window = c(0, 100))
```




```
brain_reg = extrantsr::registration(
  ss, template.file = template_brain,
  typeofTransform = "SyN",
  interpolator = "Linear")
```

```
# Running Registration of file to template

# Applying Registration output is

$fwdtransforms
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b65aaaf36a1Warp.nii.gz"
[2] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b65aaaf36a0GenericAffine.mat"

$invtransforms
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b65aaaf36a0GenericAffine.mat"
[2] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b65aaaf36a1InverseWarp.nii.gz"

$prev_transforms
character(0)

# Applying Transformations to file

 [1] "-d"
 [2] "3"
 [3] "-i"
 [4] "<pointer: 0x7f945e558cf0>"
 [5] "-o"
 [6] "<pointer: 0x7f947a3bdbb0>"
```

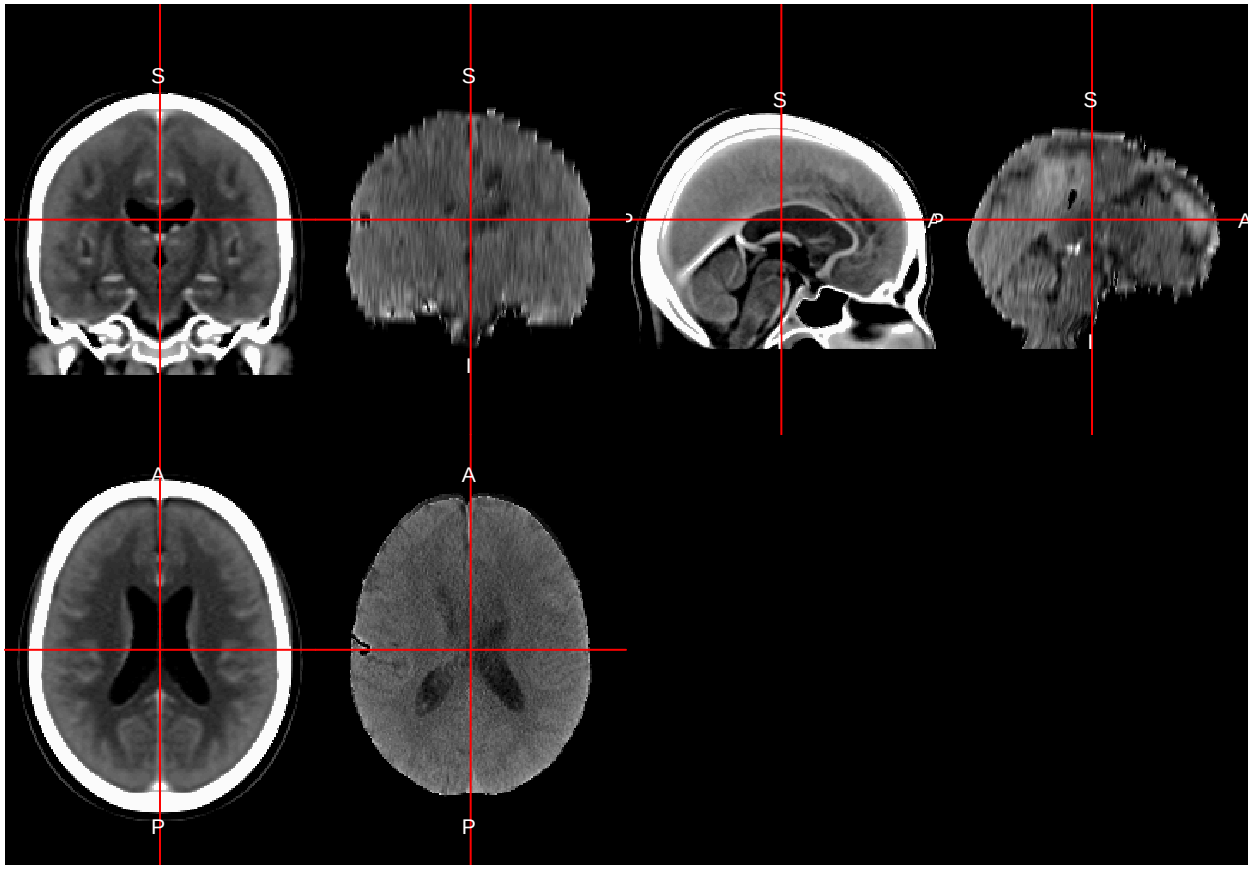
```
 [7] "-r"
 [8] "<pointer: 0x7f945e49ef20>"
 [9] "-n"
[10] "linear"
[11] "-t"
[12] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b65aaaf36a1Warp.nii.gz"
[13] "-t"
[14] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b65aaaf36a0GenericAffine.mat"
```

```
# Writing out file
```

```
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//RtmpogYndC/file58b65e8e751d.nii.gz"
```

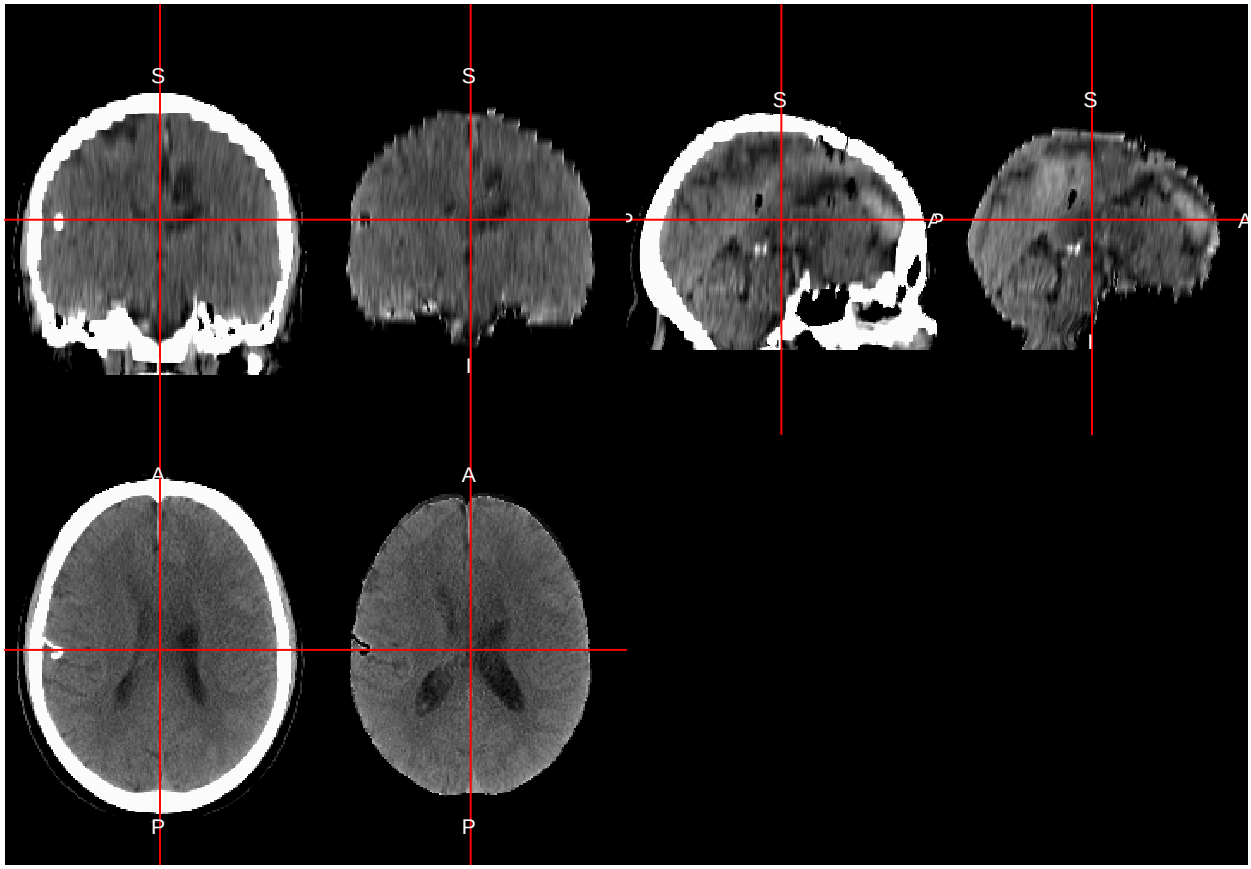
```
# Reading data back into R
```

```
wbrain = window_img(brain_reg$outfile, window = c(0, 100))
double_ortho(template_image, wbrain, window = c(0, 100))
```



We see again good alignment, but we see that there are some stark differences in these registrations when we compare them:

```
double_ortho(wimg, wbrain)
```


S
S
S
S
A
P
I
I
A
A
P
P