Connecting Google Storage to R Studio

Graham Chickering

11/7/2020

```
library(tensorflow)
library(tfdatasets)
library(keras)
library(cloudml)
library(readr)
library(tidyverse)
library(googleCloudStorageR)
library(googleAuthR)
library(sparklyr)
library(magick)
library(ggplot2)
library(cowplot)
library(reprex)
library(imager)
```

This is where we set up a connection between Google Cloud Storage and R

```
gcs_auth(json_file="account_credentials.json")

gcs_get_bucket("medical_images")
```

```
## ==Google Cloud Storage Bucket==
## Bucket:          medical_images
## Project Number:  1048020973776
## Location:        US
## Class:           STANDARD
## Created:         2020-11-07 18:56:59
## Updated:         2020-11-07 18:56:59
## Meta-generation: 1
## eTag:            CAE=
```

```
gcs_global_bucket("medical_images")
```

```
## Set default bucket name to 'medical_images'
```

```
gcs_get_global_bucket()
```

```
## [1] "medical_images"
```

```
objects <- gcs_list_objects()

all_images<-as.data.frame(objects[-c(1,2),-3])

image<-gcs_get_object(objects$name[[9]], saveToDisk="patient0.png", overwrite = TRUE)
```

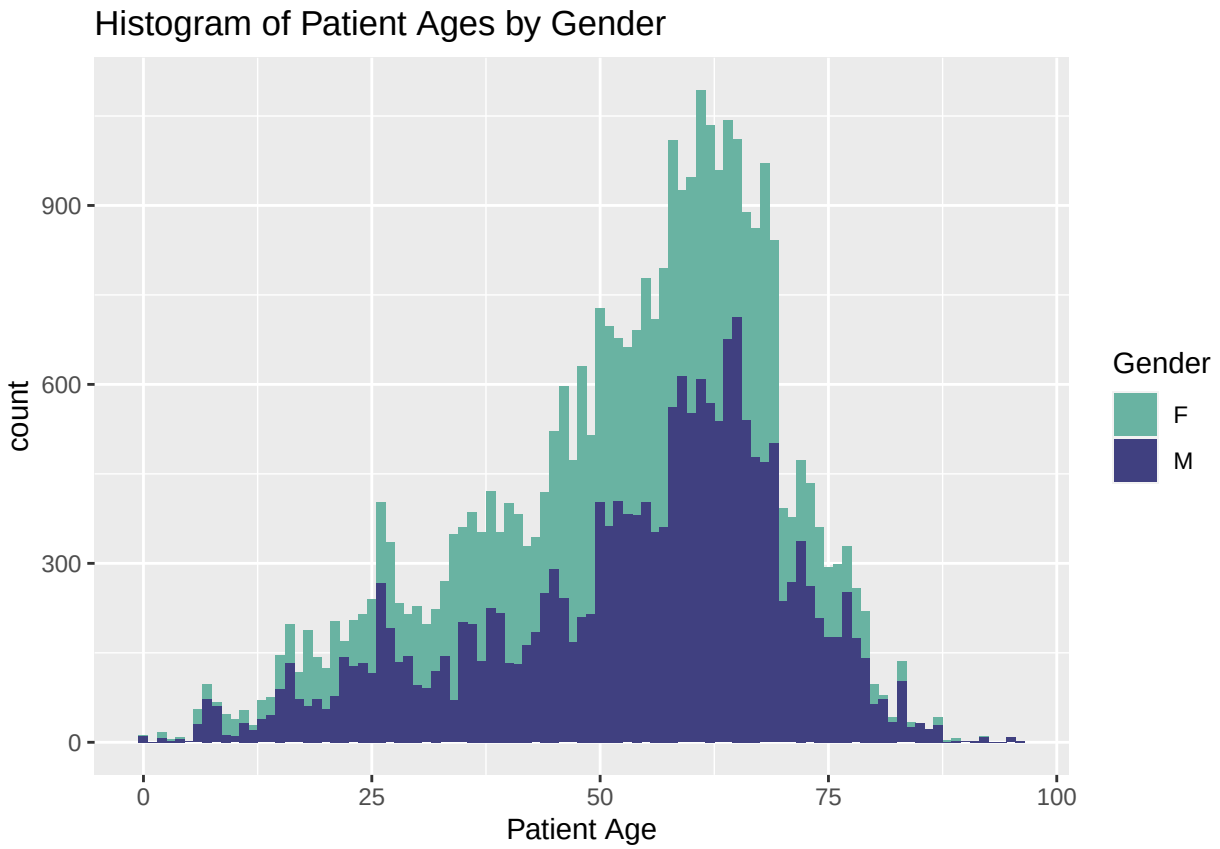
```
## 2020-11-12 00:08:15 -- Saved archive/minideeplesion/000001_01_01/109.png to patient0.png (200.9 Kb)
```

```
medical_data<-gcs_get_object(objects$name[[2]])

#data_dir <- gs_data_dir("gs://medical_images/archive")
#medical_data<-read_csv(file.path("gs://medical_images/archiveDL_info.csv"))
#medical_data <- read_csv(file.path(data_dir, "DL_info.csv"))

path<-"gs://medical_images/archive/minideeplesion/"
path2<-"archive/minideeplesion/"

medical_data<-medical_data %>%
   janitor::clean_names() %>%
  mutate(first_part=substr(file_name,1,12), second_part=substr(file_name,14,20),
         file_path=paste0(path,first_part,paste("/", second_part, sep="")),
         object_path= paste0(path2,first_part,paste("/", second_part, sep="")),
         radius=substr(lesion_diameters_pixel,1,6),
         lesion_type=case_when(
           coarse_lesion_type == -1 ~"Unknown",
           coarse_lesion_type == 1 ~ "Bone",
           coarse_lesion_type == 2 ~ "Abdomen",
           coarse_lesion_type == 3 ~ "Mediastinum",
           coarse_lesion_type == 4 ~ "Liver",
           coarse_lesion_type == 5 ~"Lung",
           coarse_lesion_type == 6 ~ "Kidney",
           coarse_lesion_type == 7 ~ "Soft tissue",
           coarse_lesion_type == 8 ~ "Pelvis"
         ) ) %>%
  select(-first_part,-second_part) %>% arrange(file_name)
```

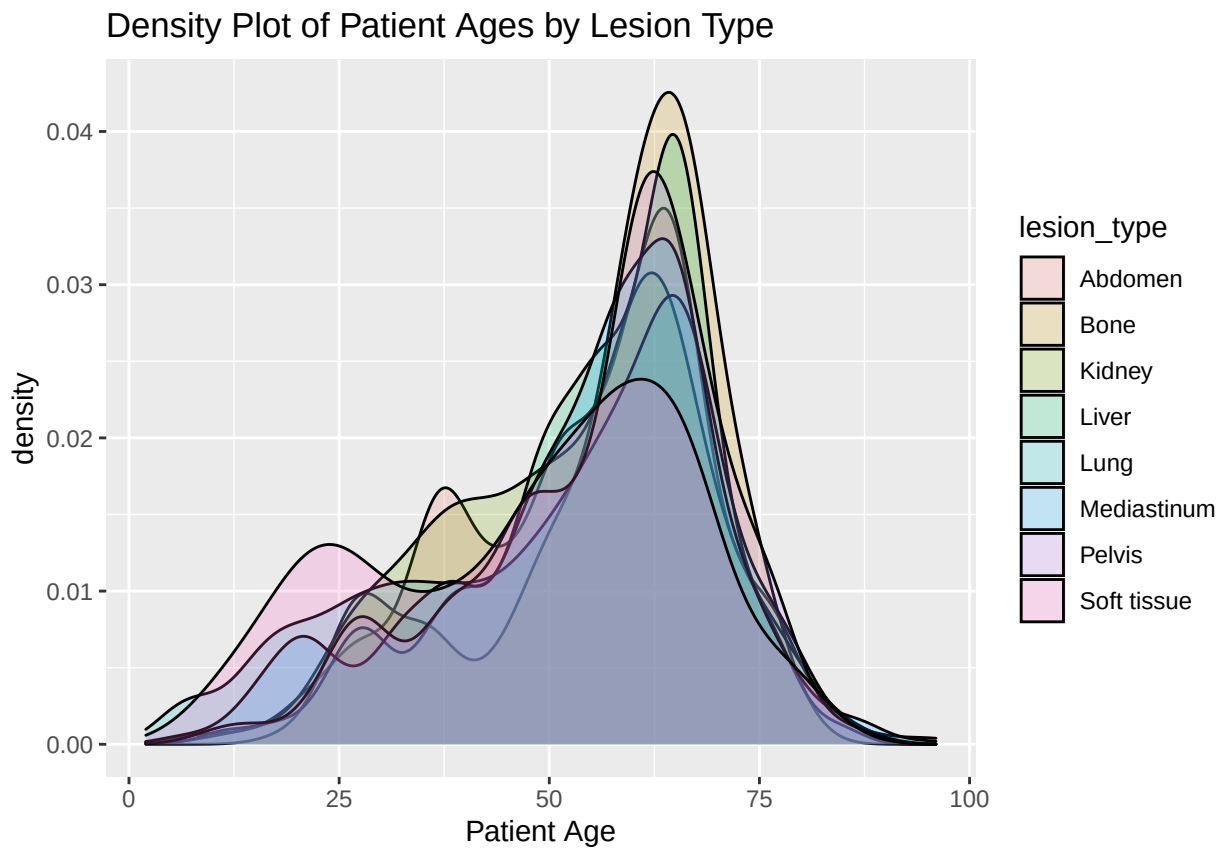
Exploratory Data Analysis

-These graphs will show the characteristics of the patients and types of legions of the patients in the study

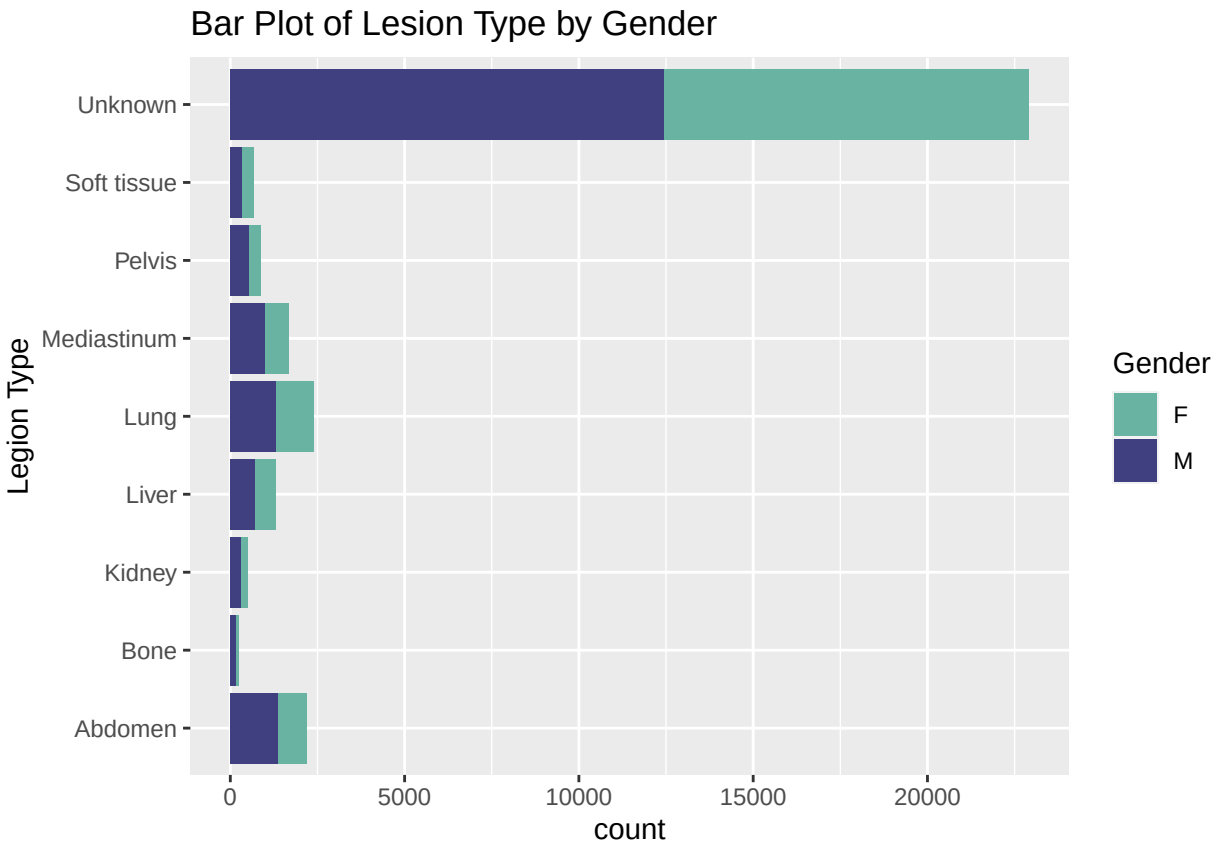
```
patients <- medical_data %>%
  filter(patient_age < 120) %>%
  mutate(radius=round(as.numeric(radius),0)) %>%
  rename(Gender = patient_gender)

ggplot(data = patients, aes(x = patient_age, fill = Gender)) +
  geom_histogram(binwidth = 1) +
  scale_fill_manual(values = c("#69b3a2", "#404080")) +
  labs(x = "Patient Age", title = "Histogram of Patient Ages by Gender")
```



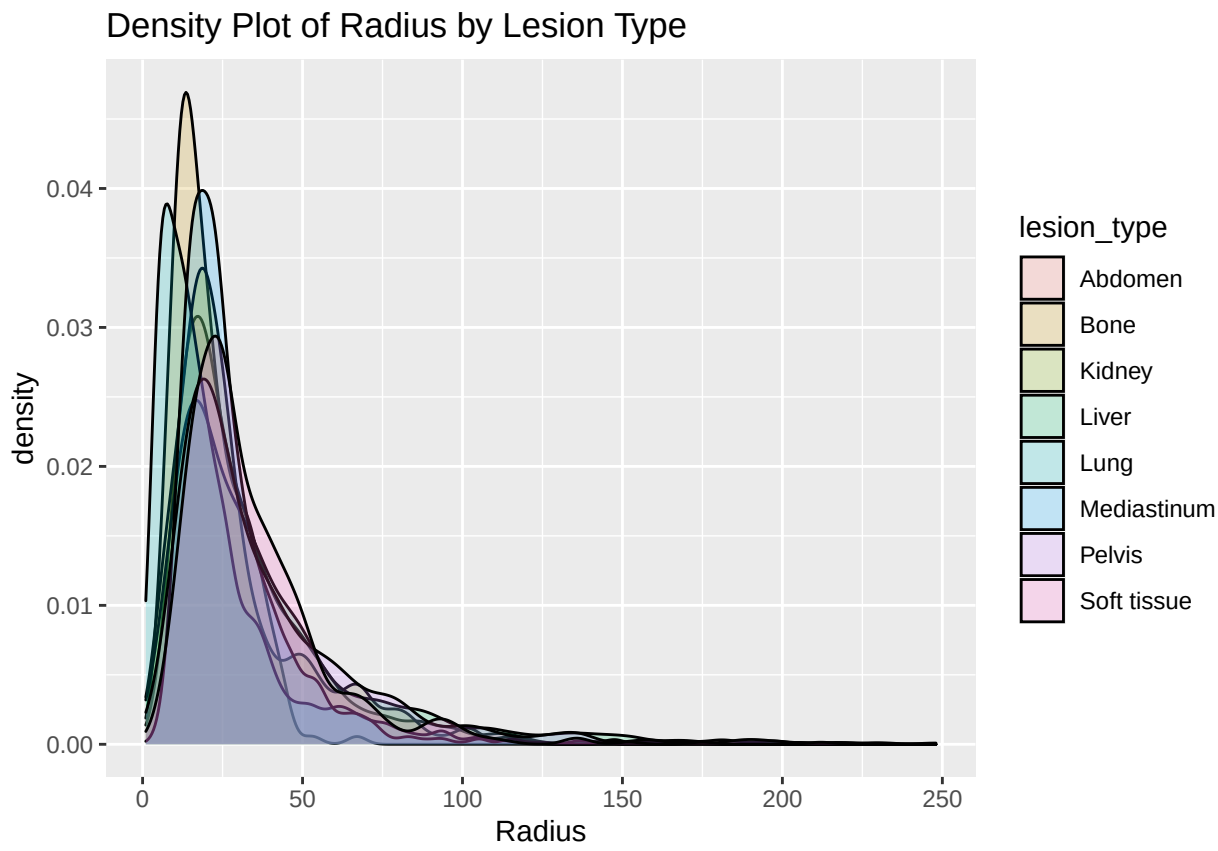

```
patients2<-patients %>% filter(lesion_type != "Unknown")

ggplot(patients2, aes(x=patient_age, fill = lesion_type)) +
  geom_density(alpha = 0.2) +
  labs(x = "Patient Age", title = "Density Plot of Patient Ages by Lesion Type")
```




```
ggplot(data = patients, aes(x = lesion_type, fill=Gender)) +
  geom_bar() +
  coord_flip()+
  scale_fill_manual(values = c("#69b3a2", "#404080")) +
  labs(x = "Legion Type", title = "Bar Plot of Lesion Type by Gender")
```




```
patients2<-patients %>% filter (radius<250 & lesion_type != "Unknown")

ggplot(data = patients2, aes(x = radius,fill=lesion_type)) +
  geom_density(alpha=0.2) +
  labs(x = "Radius", title = "Density Plot of Radius by Lesion Type")
```



This to to print out some of the images of the actual lesions

-This needs to be changed, meant to have a box around where the legion is

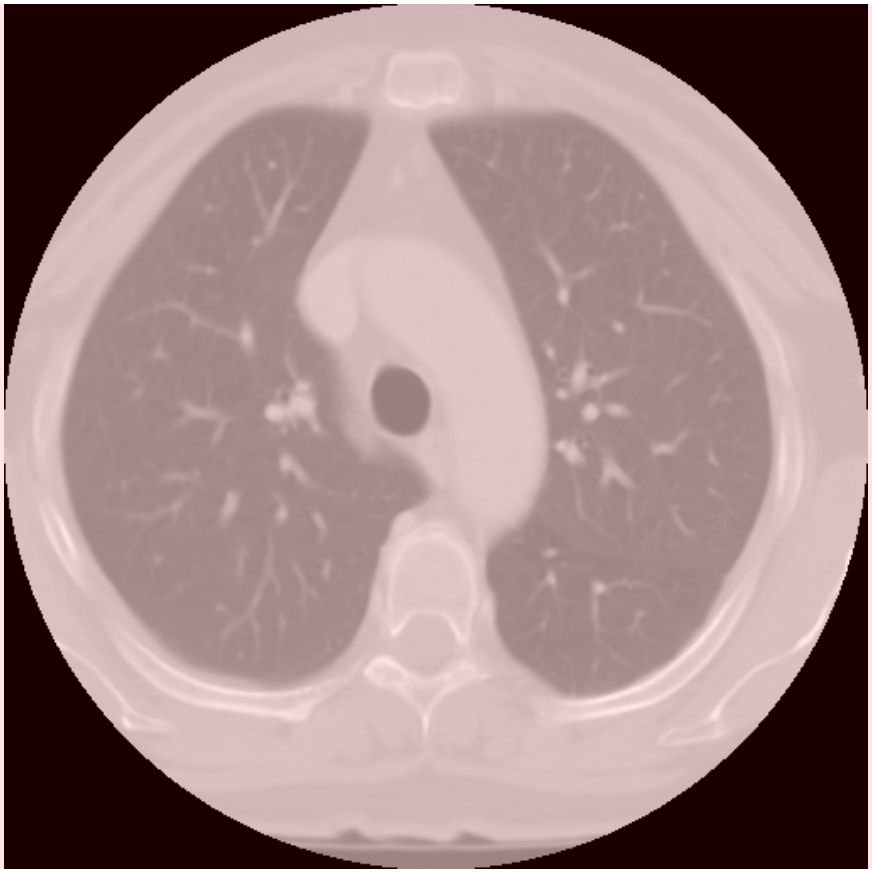
```
image<-gcs_get_object(objects$name[[9]], saveToDisk="patient0.jpeg", overwrite = TRUE)
```

```
## 2020-11-12 00:08:20 -- Saved archive/minideeplesion/000001_01_01/109.png to patient0.jpeg (200.9 Kb)
```

```
img<-image_read("patient0.jpeg")
img<-image_normalize(img)

image_info(img)
```

```
## # A tibble: 1 x 7
##   format width height colorspace matte filesize density
##   <chr>  <int>  <int> <chr>      <lgl>    <int> <chr>
## 1 PNG      512    512 Gray       FALSE        0 72x72
```

```
image<-ggdraw() +
  draw_image(
    img, scale = 1
  ) + geom_rect(aes(xmin = 0, xmax = 20, ymin = 0, ymax = 20), alpha = 1/10,fill = "red")
image
```



##This is where I will begin my work with Spark now that all the data is available from Google Cloud Storage -This is not complete

```
#spark_install()

sc <- spark_connect(master = "local", version = "2.3")


medical <- copy_to(sc, medical_data)
images<-copy_to(sc,objects )
medical
```

```
## # Source: spark<medical_data> [?? x 22]
##    file_name patient_index study_index series_id key_slice_index
##    <chr>             <dbl>       <dbl>     <dbl>           <dbl>
##  1 000001_0~             1           1         1             109
##  2 000001_0~             1           2         1              14
##  3 000001_0~             1           2         1              17
##  4 000001_0~             1           3         1              88
##  5 000001_0~             1           4         1              17
##  6 000002_0~             2           1         1             162
##  7 000002_0~             2           1         1             176
##  8 000002_0~             2           2         1              50
##  9 000002_0~             2           2         1              52
## 10 000002_0~             2           2         1              65
## # ... with more rows, and 17 more variables: measurement_coordinates <chr>,
```

```
## #   bounding_boxes <chr>, lesion_diameters_pixel <chr>,
## #   normalized_lesion_location <chr>, coarse_lesion_type <dbl>,
## #   possibly_noisy <dbl>, slice_range <chr>, spacing_mm_px <chr>,
## #   image_size <chr>, dicom_windows <chr>, patient_gender <chr>,
## #   patient_age <dbl>, train_val_test <dbl>, file_path <chr>,
## #   object_path <chr>, radius <chr>, lesion_type <chr>
```

```
spark_web(sc)

spark_disconnect(sc)
```

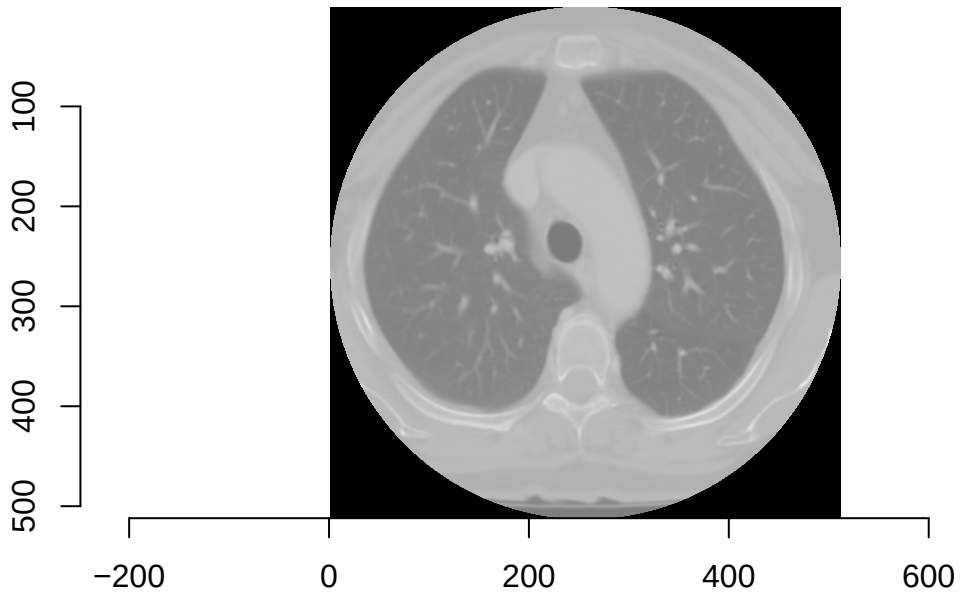
Data Preprocessing To Set Up Testing and Training Sets of the medical images

-This is not complete

```
im <- load.image("patient0.jpeg") %>% grayscale()
```

```
## Warning in grayscale(.): Image appears to already be in grayscale mode
```

```
plot(im)
```




```
dim(im)
```

```
## [1] 512 512   1   1
```

```
x<-as.matrix(im)
```

```
set.seed(123)
random <- sample(1:nrow(all_images), 0.8 * nrow(all_images)) # 80%: training data, 20%: test data
train <- all_images[random, ]
train2<- train %>% left_join(medical_data, by= c("name"="object_path"))

test <- all_images[-random, ]
test2<- test %>% left_join(medical_data, by= c("name"="object_path"))
#image2<-gcs_get_object(train$name[[1]], saveToDisk="patient1.jpeg", overwrite = TRUE)
```

This is where I will do more of the machine learning and convolution neural netowrks within Spark

```
# data_dir <- gs_data_dir("gs://medical_images/archive/minideeplesion")
# images <- list.files(data_dir, pattern = ".png", recursive = TRUE)
# length(images)
#
# classes <- list.dirs(data_dir, full.names = FALSE, recursive = FALSE)
# classes
#
# list_ds <- file_list_dataset(file_pattern = paste0(data_dir, "/*/*"))
# list_ds %>% reticulate::as_iterator() %>% reticulate::iter_next()
# list_ds
#
# get_label <- function(file_path) {
#   parts <- tf$strings$split(file_path, "/")
#   parts[-2] %>%
#     tf$equal(classes) %>%
#     tf$cast(dtype = tf$float32)
# }
#
# decode_img <- function(file_path, height = 224, width = 224) {
#
#   size <- as.integer(c(height, width))
#
#   file_path %>%
#     tf$io$read_file() %>%
#     tf$image$decode_jpeg(channels = 3) %>%
#     tf$image$convert_image_dtype(dtype = tf$float32) %>%
#     tf$image$resize(size = size)
# }
#
# preprocess_path <- function(file_path) {
#   list(
#     decode_img(file_path),
#     get_label(file_path)
#   )
# }
```

```
#
# labeled_ds <- list_ds %>%
#   dataset_map(preprocess_path, num_parallel_calls = tf$data$experimental$AUTOTUNE)
#
# labeled_ds %>%
#   reticulate::as_iterator() %>%
#   reticulate::iter_next()
```

```
# prepare <- function(ds, batch_size, shuffle_buffer_size) {
#
#   if (shuffle_buffer_size > 0)
#     ds <- ds %>% dataset_shuffle(shuffle_buffer_size)
#
#   ds %>%
#     dataset_batch(batch_size) %>%
#     # `prefetch` lets the dataset fetch batches in the background while the model
#     # is training.
#     dataset_prefetch(buffer_size = tf$data$experimental$AUTOTUNE)
# }
#
# model <- keras_model_sequential() %>%
#   layer_flatten() %>%
#   layer_dense(units = 128, activation = "relu") %>%
#   layer_dense(units = 128, activation = "relu") %>%
#   layer_dense(units = 5, activation = "softmax")
#
# model %>%
#   compile(
#     loss = "categorical_crossentropy",
#     optimizer = "adam",
#     metrics = "accuracy"
#   )
#
# model %>%
#   fit(
#     prepare(labeled_ds, batch_size = 32, shuffle_buffer_size = 1000),
#     epochs = 5,
#     verbose = 2
#   )
```

```
# trial<-flow_images_from_directory(directory=data_dir,
#                             generator = image_data_generator(rescale=1/255),
#                             target_size=c(256,256), color_mode = "grayscale")
# trial
# data_dir <- gs_data_dir("gs://medical_images/archive/minideeplesion/000002_02_01/044.png")
# trial<-image_load(path="gs://medical_images/archive/minideeplesion/000002_02_01", grayscale=TRUE)
```