Tutorial 3 Worksheet AY 22/23 Sem 1

DSA2101

Introduction to the dataset

The data for this week comes from the UCI machine learning repository.

Unzip the file `grammatical_facial_expression.zip` and place it in your `data` folder. On my local computer, I now see the following structure:

```
list.files("../data/grammatical_facial_expression/")[1:3]
```

```
## [1] "a_affirmative_datapoints.txt" "a_affirmative_targets.txt"
## [3] "a_conditional_datapoints.txt"
```

The data comes from video images of facial expressions when using Brazilian sign language. Facial expressions are an essential component of sign language, since they modify the meaning of the sign. These facial expressions are known as Grammatical Facial Expressions (GFEs). This dataset was collected to see if it is possible to train a classifier to identify the presence of a GFE from the facial image of the signer. There are 9 GFEs considered:

- `wh_questions`: who/what/when/where/how questions
- `yes/no questions`: yes/no questions
- `doubt questions`: doubt questions
- `topics`
- `assertions`
- `negatives`
- `conditional clauses`
- `focus`
- `relative clauses`

The videos were captured by a Microsoft Kinect sensor. The videos come from two users (`a` and `b`). In each video, a user performs (five times), in front of the sensor, five sentences in Libras (Brazilian Sign Language) that require the use of a grammatical facial expression. Using Microsoft Kinect, they have obtained:

- a text file (`*_datapoints.txt`) containing one hundred coordinates (x, y, z) of points from eyes, nose, eyebrows, face contour and iris; each line in the file corresponds to points extracted from one frame. The z-coordinate is depth from the sensor. The first value in each line is a timestamp for that frame in the video.
- a corresponding text (`*_targets.txt`) file with a manually labelled classification of whether there is a GFE or not. Each line in this file corresponds to a frame in the `datapoints.txt` file.

The dataset is organized in 36 files: 18 datapoint files and 18 target files, one pair for each video which compose the dataset.The name of the file refers to each video: the letter corresponding to the user (a and b), name of grammatical facial expression and a specification (target or datapoints).

Questions

1. Use `scan` to read in one of the datapoints files. What does it return?

```
tmp_frames<- scan("../data/grammatical_facial_expression/a_negative_datapoints.txt",
                  skip=1)
```

2. We would like to represent the coordinates in each frame as a matrix with dimension 100x3. We would then like to represent the information in each datapoints file as a multidimensional array (100 x 3 x n) where n is the number of frames. Here is an *example*:

```
tmp_classes <- scan("../data/grammatical_facial_expression/a_negative_targets.txt")
l_class_vec <- length(tmp_classes) # no. of frames
output_array <- array(tmp_frames[-seq(1, length(tmp_frames), by=301)],
                      dim=c(3, 100, length(tmp_classes)))
output_array <- aperm(output_array, c(2,1,3))
```

`output_array` is now of dimension 100x3x1124. We can access the coordinates in the first frame with `output_array[ , , 1]`, and so on.

3. Write a function `read.fe_data` that works in the following way, and returns an object of class `gfe`. You can find an example of such an object in `example_gfe.rds`.

```
args(read.fe_data)
```

```
## function (path_to_dir, user, facial_expression)
## NULL
```

```
obj1 <- read.fe_data("../data/grammatical_facial_expression", user="b",
                     "emphasis")
str(obj1)
```

```
## List of 2
##  $ frames: num [1:100, 1:3, 1:1344] 329 327 323 320 317 ...
##  $ info  :'data.frame':  1344 obs. of  4 variables:
##   ..$ timestamp        : num [1:1344] 1.39e+09 1.39e+09 1.39e+09 1.39e+09 1.39e+09 ...
##   ..$ user             : chr [1:1344] "b" "b" "b" "b" ...
##   ..$ facial_expression: chr [1:1344] "emphasis" "emphasis" "emphasis" "emphasis" ...
##   ..$ fe_present       : num [1:1344] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "class")= chr "gfe"
```

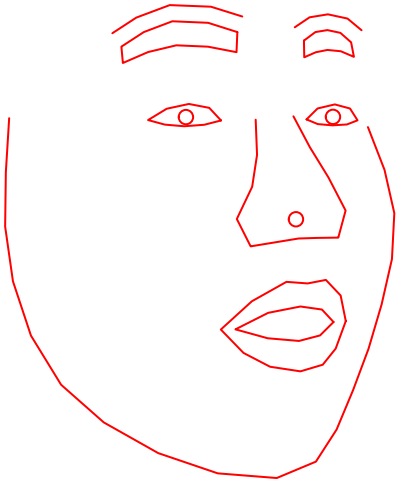
2. The 100 coordinates for each frame correspond to the following facial features:

- 0 - 7 (x,y,z) - left eye
- 8 - 15 (x,y,z) - right eye
- 16 - 25 (x,y,z) - left eyebrow
- 26 - 35 (x,y,z) - right eyebrow
- 36 - 47 (x,y,z) - nose
- 48 - 67 (x,y,z) - mouth
- 68 - 86 (x,y,z) - face contour
- 87 (x,y,z) - left iris
- 88 (x,y,z) - right iris
- 89 (x,y,z) - nose tip
- 90 - 94 (x,y,z) - line above left eyebrow

- 95 - 99 (x,y,z) - line above right eyebrow

Write a plot method for objects of class `gfe`, that works in the following way. The `id` argument selects and plots the id-th frame in the `gfe` object.

```
plot(obj1, id=1000)
```



Hints

1. If you need more information about the data, do refer to the file `data_descriptions.txt` and `view_points.png`.
2. The `gfe` object contains a multi-dimensional array. It can be created with `array`. Remember that R is column-major, so store your coordinates in the correct way.
3. Code like this week is not for presenting to others, but for us to work with our data efficiently. It is worth taking the time to write it well, because we might use it frequently, and so might our colleagues. What other functions might be useful when we are exploring our data?
4. When writing up the function, practice on one of the frames first. Understand the data completely before packaging it into a function.