Term Project Report

DA5020, Spring 2018

Dylan Rose

Contents

Introduction

There is growing scientific and practical interest in using large sets of human eye movement behavior data for biometric purposes (Itti, 2015). Eye movement data meets a number of the criteria for "big" data analytics and storage. First, they have significant "volume" in that raw eye movement data records typically contains thousands of records describing horizontal and vertical coordinates of gaze (fovea placement in screen coordinates). Second, they have significant "variety" in terms of experimental meta-data and stimuli. The latter ranges from procedurally generated images of simple geometric shapes to minutes or hours long video sequences displayed at 4k.

Third, they have unique challenges in terms of data "veracity" and "validity". Eye movement data is inherently noisy and non-stationary, frequently contains missing or impossible values that require special error-handling. Raw gaze data is not typically useful on its own, but requires processing into a set of "macro" eye movement type events, such as saccades (large ballistic movements that reposition the center of vision over new inspection targets) and fixations (periods of relative oculomotor stability). This process depends on algorithms with many user selected parameter values. This means that it has at least four of the six "V" descriptors used to define data as "big" (Jain, Schedlbauer, & Durant, 2015). Despite this, it is still most commonly stored and distributed in text or binary data files (though see Wilming et al. (2017) for an excellent recent effort away from this).

Project goals and justification

For my term project, I've collected several eye movement data sets that I will ultimately use to test eye biometric classifiers for *user task* in my dissertation. My goal for the project is to integrate them, link them tightly to their associated stimuli, and to evaluate the merits of a storing and retrieving them from a true database instead of from the raw data sources.

There are two particular challenges to working with them that have informed my decision about choice of database. The first is their hierarchical structure: individual data points are nested within trials, within subjects, and these finally within an experiment. This means the data is poorly structured for traditional relational databases, as getting it to third normal form would require the use of many adjacency lists, nested set representations, or a cumbersome number of "intermediary" data tables to appropriately express the numerous "many to many" relationships formed within the data, See this and this site for some discussion of the issues involved.

The second is the experimental stimuli, and the importance of maintaining strong relationships between gaze and stimulus data. Eye movement positions are not typically used as features in related biometric

classifiers by themselves. Rather, they used as indexes from an image aligned map or model of scene content believed to drive different pathways of eye movement control activity in the brain. To be useful, the database should therefore permit fast, accurate sampling directly from these maps. Though this constraint can be addressed through supplementary text-files/meta-data that provide file paths to the appropriate stimuli, this approach is fragile and cumbersome at scale. SQL does have a "blob" data type that makes it possible to store (small) images in a table, and many NoSQL database systems allow very significant data type flexibility, but the consensus appears to be that images should not be stored in databases but in the host file-system or a specialized file-system/container within it (see this site for some discussion of this point also).

It is therefore clear that a NoSQL DBMS is the correct choice for this project. The freedom from normalization constraints relaxes the difficulty of building a schema that accommodates the nested structure of the data. Though I will not be storing the stimuli as would be ideal, but only the paths to them alongside the data for the project, it is likely to be significantly easier to do so if I eventually wanted to test this. Finally, I will likely expand the database to include other data sets in the future. As there's no risk of an insufficient relational schema breaking relationships or causing anomalies at update, a NoSQL DBMS will make updating it in this way much safer. The project will therefore aggregate:

1. eye movement data – only fixation records at this time
2. experimental meta-data (equipment parameters, sampling times, etc)
3. paths to experimental stimuli, if they are available
4. subject demographic data, if it is available

into a single MongoDB document.

Format of Each Subset of the Raw Data

We're working with four different kinds of data that need to get merged for this project. Each type can be thought of as a table, though not one that necessarily corresponds to the forms specified by relational DBs.

Fixation data

Fixation data are points that describe where a particular subject was looking in a particular image during a specific trial at a particular time, as well as for how long. The fixation data table has eight variables: horizontal position ("x_position"), vertical position ("y_position"), fixation duration ("duration"), the experimental subject from whose data it was taken ("subject"), the number of the fixation within a particular subject/trial record ("fixation"), the image the subject was inspecting during the trial from which the data

	x_position	y_position	subject	image	fixation	task_type	duration	experiment
1	834	532	1082	7484.jpg	1	free_viewing	750	vip
2	719	549	1082	7484.jpg	2	free_viewing	308	vip
3	568	523	1082	7484.jpg	3	free_viewing	258	vip
4	845	484	1082	7484.jpg	4	free_viewing	233	vip
5	985	457	1082	7484.jpg	5	free_viewing	183	vip
6	1061	639	1082	7484.jpg	6	free_viewing	100	vip

Figure 1: Example of fixation data

	experiment	source	screen_width_pix	screen_height_pix	viewing_distance_in	eye_tracker	sampling_rate_hz	trial_duration_seconds	list_of_tasks
1	nu	http://custom	1920	1080	19.68	eyelink_1000	60	5	free_viewing,scene_description,object_counting
2	vip	http://mmas.comp.nus.edu.sg/VIP_files/VIP_Dataset.zip	1680	1050	19.68	smi_red_250	120	5	free_viewing,anomaly_detection
3	viu	https://labs.psych.ucsb.edu/eckstein/miguel/research_pages/saliencydata.html	800	600	19.68	eyelink_1000	250	2	free_viewing,saliency_search,cued_object_search
4	wilmings	datadryad:monitor_width_pix	NA	NA	NA	500	6	NA	<NA>

Figure 2: Example of fixation data

was taken (“image”), the task type a subject was performing for a particular trial (“task_type”), and the experiment the data is associated with (“experiment”) (Figure 1).

Experimental meta-data

Experimental meta-data describes things like the hardware/displays used, stimulus display times, etc. The meta-data for the experiments has nine variables: the name of the experiment (“experiment”), a link to the source of the data (“source”), the width of the monitor used in the experiment (“monitor_width_pix”),the height of the monitor used in the experiment (“monitor_height_pix”), the distance between the subject and the monitor (“viewing_distance”), the eye tracking hardware used (“eye_tracker_used”), the sampling rate for the data (“eye_tracking_sampling_rate”), the length of the trial (“trial_duration”), and the tasks in the experiment (“tasks”) (Figure 2).

Stimuli data

Stimuli information for these sets are essentially path information/labels used to align subject/trial data records with this data where necessary, and to make it easy to access the images in the file-system/consistently

	full_image_path
1	/Users/dylanrose/Sync/projects/work_projects/class_work/collecting_storing_data/term_project/input_data/viu_dataset/Images/image_r_1.jpg
2	/Users/dylanrose/Sync/projects/work_projects/class_work/collecting_storing_data/term_project/input_data/viu_dataset/Images/image_r_10.jpg
3	/Users/dylanrose/Sync/projects/work_projects/class_work/collecting_storing_data/term_project/input_data/viu_dataset/Images/image_r_100.jpg
4	/Users/dylanrose/Sync/projects/work_projects/class_work/collecting_storing_data/term_project/input_data/viu_dataset/Images/image_r_101.jpg
5	/Users/dylanrose/Sync/projects/work_projects/class_work/collecting_storing_data/term_project/input_data/viu_dataset/Images/image_r_102.jpg
6	/Users/dylanrose/Sync/projects/work_projects/class_work/collecting_storing_data/term_project/input_data/viu_dataset/Images/image_r_103.jpg

	image	image_labels	experiment
1	image_r_1.jpg	1	viu
2	image_r_10.jpg	10	viu
3	image_r_100.jpg	100	viu
4	image_r_101.jpg	101	viu
5	image_r_102.jpg	102	viu
6	image_r_103.jpg	103	viu

Figure 3: Example of fixation data

	subject	experiment	age	gender	neuropsych_status
1	7742	vip	22	F	NA
2	6470	vip	22	F	NA
3	619	vip	22	F	NA
4	2134	vip	22	F	NA
5	5094	vip	22	F	NA
6	6848	vip	22	F	NA

Figure 4: Example of fixation data

name new images generated by doing some image processing on them. There are four variables in the data frame as a result: the name of the image file, including the file extension ("image"), the name of the image without the file extension ("image_labels"), the full file path to the image on my system ("full_image_path"), and the experiment the image was inspected in ("experiment") (Figure 3).

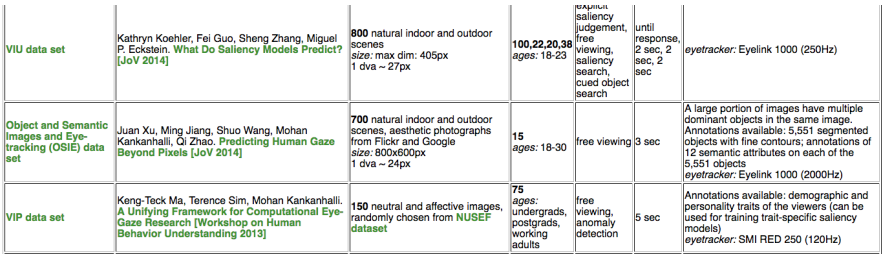
Subject demographic data

Subject meta data describes subject parameters, such as their age, gender, etc. The demographic data table contains the following following five variables for each subject: some subject-specific id-code ("subject"), their gender if it was available ("gender"), their age if it was available ("age"), their neuro-psychological status ("neuropsych_status"), and the experiment they were in ("experiment") (Figure 4).

Raw Data Collection

Collecting the raw data for this project was straightforward. The raw data are four sets of eye movement data, three of which are taken from public sources, and one from an experiment I ran here. Eye movements made by observers in each experiment were made in response to a minimum of two tasks while inspecting unedited images of natural scenes. The sets are as follows:

1. The VIP dataset taken from the MIT Computational Saliency Model Benchmark site. seventy-five



VIU data set	Kathryn Koehler, Fei Guo, Sheng Zhang, Miguel P. Eckstein. What Do Saliency Models Predict? [JoV 2014]	**800** natural indoor and outdoor scenes *size:* max dim: 405px 1 dva ~ 27px	**100,22,20,38** *ages:* 18-23	explicit saliency judgement, free viewing, saliency search, cued object search	until response, 2 sec, 2 sec, 2 sec	*eyetracker:* Eyelink 1000 (250Hz)
Object and Semantic Images and Eye-tracking (OSIE) data set	Juan Xu, Ming Jiang, Shuo Wang, Mohan Kankanhalli, Qi Zhao. Predicting Human Gaze Beyond Pixels [JoV 2014]	**700** natural indoor and outdoor scenes, aesthetic photographs from Flickr and Google *size:* 800x600px 1 dva ~ 24px	**15** *ages:* 18-30	free viewing	3 sec	A large portion of images have multiple dominant objects in the same image. Annotations available: 5,551 segmented objects with fine contours; annotations of 12 semantic attributes on each of the 5,551 objects *eyetracker:* Eyelink 1000 (2000Hz)
VIP data set	Keng-Teck Ma, Terence Sim, Mohan Kankanhalli. A Unifying Framework for Computational Eye-Gaze Research [Workshop on Human Behavior Understanding 2013]	**150** neutral and affective images, randomly chosen from NUSEF dataset	**75** *ages:* undergrads, postgrads, working adults	free viewing, anomaly detection	5 sec	Annotations available: demographic and personality traits of the viewers (can be used for training trait-specific saliency models) *eyetracker:* SMI RED 250 (120Hz)

Figure 5: Download link for the VIU and VIP datasets.

subjects inspected a set of 150 neutral and emotional affecting images while performing either a free viewing or an anomaly detection task. Each subject was shown each image for five seconds.

2. The VIU dataset taken from the same database as the VIP set. Eighty subjects performed one of three separate tasks: free viewing (twenty-two subjects), saliency search (twenty subjects), and cued object search (thirty-eight). Subjects were given two seconds to inspect each image.

3. The Wilmings-Onat "head-fixed" dataset was taken from a dedicated data repository containing a number of eye movement datasets. For this experiment, nineteen observers performed one of two tasks while viewing images of "urban scenes": free viewing and guided viewing. Subjects were given six seconds to inspect each image.

4. A custom data set of eye movement data I collected from fifteen Northeastern University undergraduate students (referred to hereafter as "NU" dataset). Subjects completed 210 trials, each of which required them to perform one of three tasks: free viewing, scene viewing in preparation for scene description, and object counting. Each subject completed seventy trials for each task on a randomly ordered list of 210 images of natural scenes.

For the first three, some effort had been made to clean and package the data in a systematic way. Both the VIP and VIU sets were available from the same website and provided access to meta-data in a straightforward tabular format (Figure 5).

The data for the Wilmings set was also straightforward to obtain. I downloaded the provided hdf5 file of gaze data, as well as the folder of images and the meta-data about subjects for the experiment directly from DataDryad.org (see Figure 6). I then wrote a simple piece of processing code in python that accessed the data from the hdf5 file (see Figure 7).

The data from the NU dataset had already been pre-processed into subject/trial specific text files containing fixation data, so I didn't need to do anything there, either.

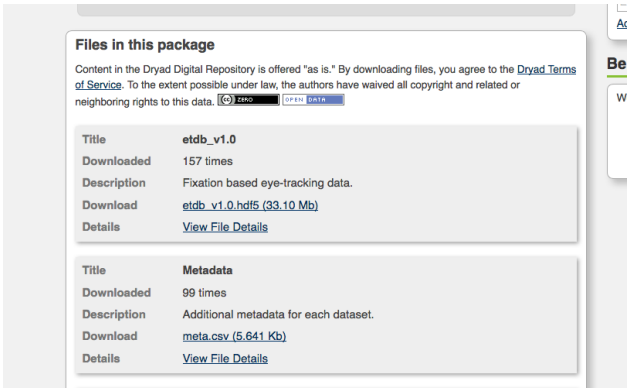


Figure 6: Download link for Wilmings dataset

```
#!/usr/bin/env

import numpy as np
import pandas as pd
import h5py
import fixmat

#Load the head fixed data
head_fixed_data, meta = fixmat.load('etdb_v1.0.hdf5', 'Head Fixed')

#Filter down so that we're only working with the "urban" stimuli
urban_stimuli_only = head_fixed_data.loc[head_fixed_data.category!=10]

#Grab the variables & rename them appropriately
x_position = urban_stimuli_only.x
y_position = urban_stimuli_only.y
subject    = urban_stimuli_only.SUBJECTINDEX
image      = urban_stimuli_only.filenumber
fixation   = urban_stimuli_only.fix
bin_tasks  = urban_stimuli_only.guided_viewing.as_matrix()
duration   = urban_stimuli_only.end - urban_stimuli_only.start

#Do a little cleanup
task_type = []
for each_task_instance,each_task_index in enumerate(bin_tasks):
    if bin_tasks[each_task_instance] == 0:
        task_type.append("free_viewing")
    else:
        task_type.append("guided_viewing")

#Setup the output frame and write it out
output_frame = pd.DataFrame()
output_frame["x_position"] = x_position
output_frame["y_position"] = y_position
output_frame["subject"]    = subject
output_frame["image"]      = image
output_frame["fixation"]   = fixation
output_frame["task_type"]  = task_type
output_frame["duration"]   = duration
output_frame.to_csv("./wilmings_fixation_data.csv",index=False)
```

Figure 7: Python script for getting data out of the Wilmings hdf5 file

```
name:
source:
monitor_width_pix:
monitor_height_pix:
viewing_distance:
eye_tracker_used:
eye_tracking_sampling_rate:
trial_duration:
tasks:
```

Figure 8: Experiment meta data text file template.

Pre-processing the Raw Data

This was by far the most labor intensive part of the project. While accessing the data from its original set of sources was easy, the differences in file naming and storage formats made getting all of the data into *R* in a consistent way quite challenging. I'll talk about the difficulties associated with each experiment separately, but the code required to do so generally followed a consistent set of steps at a certain level of abstraction:

1. Read the data in from (typically) plain text or structured text files, making any experiment-specific tweaks necessary based on the differences in the data structure.
2. Apply some experiment-specific data cleaning function to its content (set variable types, drop data for certain subjects/image types, etc.)
3. Bind the cleaned data together into a single data frame.

Experimental meta-data

The first thing I had to do was create some kind of unified representation of the meta-data associated with each experiment. This includes information about things like the size of the display the images were shown on, the length of the recording period within a trial, and so on. To do this, I made a simple template .txt file (Figure 8)

These files can then be simply read together and aggregated to provide a table of this information for all of the experiments we have data for.

```
Version:    IDF Event Detector 3.0.18
Sample Rate:    120
Subject:    1082




Table Header for Fixations:
Event Type  Trial   Number  Start   End Duration    Location X  Location Y
Dispersion X    Dispersion Y    Plane   Avg. Pupil Size X   Avg Pupil Size Y

Table Header for Saccades:
Event Type  Trial   Number  Start   End Duration    Start Loc.X Start Loc.Y
End Loc.X   End Loc.Y   Amplitude   Peak Speed  Peak Speed At   Average Speed
   Peak Accel. Peak Decel. Average Accel.

Table Header for Blinks:
Event Type  Trial   Number  Start   End Duration

Table Header for User Events:
Event Type  Trial   Number  Start   Description

UserEvent   1   1   18589046081 # Message: 7484.jpg
Fixation L  1   1   18589047189 18589797940 750751  834.38  532.22  42  56  -1
  15.67   15.67
Saccade L   1   1   18589797940 18589814566 16626   819.12  543.29  777.00
542.20  0.56    66.73   1.00    33.44   3409.48 -1620.75    2221.81
Fixation L  1   2   18589814566 18590123187 308621  719.29  549.22  63  13  -1
  14.49   14.49
Saccade L   1   2   18590123187 18590139931 16744   713.78  553.53  602.62
544.54  1.01    115.51  1.00    60.19   5285.03 -5976.05    4432.35
Fixation L  1   3   18590139931 18590398426 258495  568.81  523.19  68  28  -1
  15.28   15.28
```

Figure 9: Example VIP data file

VIP set

The VIP experiment dataset was probably the most frustrating and difficult one to work with. Each subject's data came in an "events.txt" file generated by the eye-tracking system experiment manager. Though they're plain text, they're not well-structured (see Figure 9). For example, they have twenty lines of header material, the data for different images/trials is mixed within them in chunks separated by comments containing the name of the image, and different types of data (saccades,fixations,blinks,errors) are all mixed up together in-line.

This meant that the "read" function for the VIP is the single longest piece of code for the entire project (roughly 40 lines, see Figure 10), as it requires dealing with a number of special cases/has to select specific types of data out of the mix.

Though the experimenters provide some of the image data, a great deal of it is actually taken from an image dataset that is not publicly available. The image data/paths have therefore been just treated as missing for the purposes of this project. Subject demographic data was provided in a csv file which was easy to load and align with the data on the basis of matching subject id strings between the two data frames.

```
#Extract data from gaze data file:
read_vip_text_data <- function(file_path,subject_lookup_table){
    if(file.exists(file_path)){
        raw_data <- read.csv(file=file_path, header=F, sep="\t", skip=20)

        #The subject info appears in the 4th element of the first split on Ubuntu, and in the fifth element on OSX,
        #so this switch checks this and gets the correct subject info accordingly
        if (system_type == 'Darwin'){
            subject  <- unlist(str_split(unlist(str_split(unlist(str_split(file_path,"s/"))[5]," "))[1],"_"))[2]
        } else {
            subject  <- unlist(str_split(unlist(str_split(unlist(str_split(file_path,"s/"))[4]," "))[1],"_"))[2]
        }

        task <- str_replace(subject_lookup_table$task[subject_lookup_table$subject_id==subject],"-","_")
        trial_break_indexes  <- which(raw_data$V1 =="UserEvent")
        user_events_only     <- as.character(unlist(unname(as.list(filter(raw_data,V1=="UserEvent")%>%select(V5)))))
        #To give ourselves trial number/inspected stimuli info, we gotta do some complicated looping. Hold tight:
        subjects_to_skip <- c("6393","7446","9320","9756")
        if (subject %in% subjects_to_skip){
            output_fixation_frame <- data.frame(NA,NA,NA,NA,NA,subject,task)
            colnames(output_fixation_frame) <- c("V7","V8","V3","V6","image","subject","task")
            return(output_fixation_frame)
        } else {
        output_fixation_frame <- data.frame()
        for (each_break in 1:(length(trial_break_indexes)-1)){
            data_start_point <- trial_break_indexes[each_break]
            data_end_point   <- trial_break_indexes[each_break+1]-1
            all_data_types_in_range <- slice(raw_data,data_start_point:data_end_point)
            fixation_only_data <- filter(all_data_types_in_range,V1=="Fixation R") %>% select(V7,V8,V3,V6)
            image <- rep(unlist(str_split(rep(user_events_only[each_break]),": "))[2],nrow(fixation_only_data))
            fixation_only_data$image <- as.factor(image)
            output_fixation_frame <-rbind(output_fixation_frame,fixation_only_data)
        }
        output_fixation_frame$subject <- rep(subject,nrow(output_fixation_frame))
        output_fixation_frame$task    <- rep(task,nrow(output_fixation_frame))
        return(output_fixation_frame)
    }
    } else {
        sprintf('Cant find file %s at the specified path!',file_path)
    }
}
```

Figure 10: VIP data read function

VIU set

The VIU experiment set was much less complicated. Fixation data for each subject/trial was stored in well-structured plain-text files, separated by task (e.g. Figure 11)

I therefore simply had to load the file for each task and paste them together to get the fixation data frame setup.

The image data was available for this experiment, and to read it in/process it I simply looped over each location in the image sub-directory for that dataset to get the image path/image names. The image label variable information just took a bit of string processing to separate the file from the file extension. No subject demographic data was provided for this experiment, so it was all treated as a set of NAs.

Wilmings set

To get the fixation data for this experiment, I simply ran the extraction python code I presented in Figure 7. This dumped the fixation data into a simple csv file with the appropriate column names for all of the subjects who completed the experiment. The stimuli were available for this experiment, and I followed the same procedure for getting the stimuli paths/labels/names that I used for the VIU experiment. Finally, the subject meta-data was provided by the original posters of the data in a simple csv file, so it was easy to read.

FreeViewing.txt

x	y	obs	image	fixation
216	186	1	1	1
199	194	1	1	2
212	189	1	1	3
252	155	1	1	4
344	199	1	1	5
256	70	1	1	6
344	165	1	1	7
262	86	1	1	8
334	159	1	1	9
330	137	1	1	10
335	171	1	1	11
336	191	1	1	12
240	68	1	1	13
NaN	NaN	1	1	14
NaN	NaN	1	1	15
250	211	1	2	1
152	243	1	2	2

Figure 11: VIU free-viewing task fixation raw data example

```
```{r,message=FALSE,warning=FALSE,tidy=TRUE}
vip_data <- merge(merge(vip_fixation_data_frame,vip_subject_dataframe,
    by=c("subject","experiment")),vip_image_dataframe,by=c("image","experiment"))
viu_data <- merge(merge(viu_fixation_data_frame,viu_subject_dataframe,
    by=c("subject","experiment")),viu_image_dataframe,by=c("image","experiment"))
wilmings_data <- merge(merge(wilmings_fixation_data_frame,wilmings_subject_dataframe,
    by=c("subject","experiment")),wilmings_image_dataframe,by=c("image","experiment"))
nu_data <- merge(merge(nu_fixation_data_frame,nu_subject_dataframe,
    by=c("subject","experiment")),nu_image_dataframe,by=c("image","experiment"))
all_data<-rbind(vip_data,viu_data,wilmings_data,nu_data)
```
```

Figure 12: Data merge process for each experiment prior to row-wise binding across experiments.

NU set

This data set was the one from my experiment. The raw fixation data was available in a set of csv files, and could be read simply by looping through the directory where the data was stored and reading in each file's data. The stimuli are available for this experiment, but for reasons beyond the scope of this project report I could not include their information here. They were therefore treated as missing. Subject demographic meta-data was available in a basic csv file, and it too could be read into *R* straightforwardly.

Storage

Once all of the data had been collected for each of the four experiments, getting it ready for storage was easy. The fixation, image, and demographic data types within a particular experiment were bound together using a set of nested merges: first between the fixation data and the subject demographic data on the variables "subject" and "experiment", then between the resulting table and the image meta-data table on the "image" and "experiment" variables. Each of these subject-specific merged data frames was then bound together into a final data frame by row-wise binding (Figure 12)

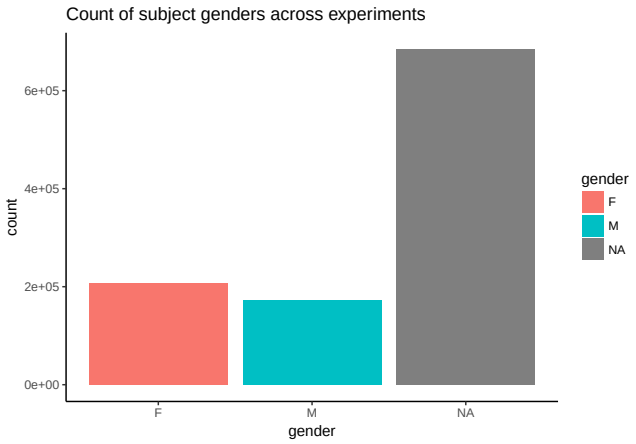
```{r,message=FALSE,warning=FALSE,tidy=TRUE}
fixation_data_connection <- mongo("fixation_data")
fixation_data_connection$drop()
fixation_data_connection$insert(all_data)

#Separate insert for the experiment meta-data
fixation_data_connection$insert(all_experiments_parameter_table)
```

Figure 13: Data merge process for each experiment prior to row-wise binding across experiments.

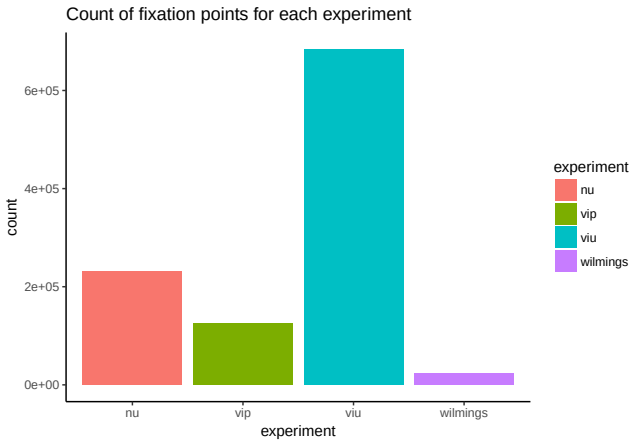
We then create a connection to a MongoDB document called "fixation_data" and insert the "all_data" frame, as well as the frame of experimental meta-data into that same document (Figure 13)

Testing

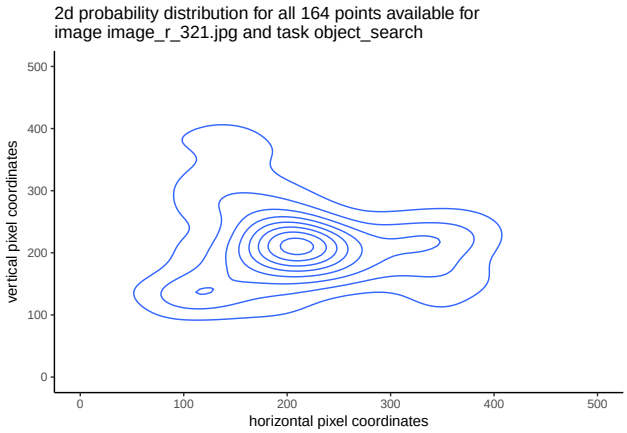
Just to test out what this lets us do, we can produce nice plots of things like the gender of subjects across experiments:



or the total number of fixation records in each of the experiments:



or even cool stuff like plotting a smooth 2d distribution for all the points available for a particular image and task:



Discussion, limitations, next steps

Discussion

I believe I've accomplished my stated objectives for my project, and am extremely pleased with the results. Although a significant amount of work was required to move from raw data to the database, the advantages of using a database rather than the raw data as it was available by default are *immediately* apparent. While the need to aggregate many kinds of information cleanly across experiments like this is less common in experimental contexts, for my purposes it enables forms of information manipulation and aggregation that would be prohibitively time and effort consuming. Being able to compare, e.g. spatial distributions of gaze points across observers and images for the *same task type across experiments*, is a completely novel and exciting experience.

While I've had to lean on records documenting the location of specific stimuli in my host file system instead of including image files in the database itself, the form of the resulting database would make it extremely easy to pull the image files in directly using scripting in other languages.

Finally, although the pre-processing pipeline and the initial data insertion operation into the Mongo document is slow, querying the database–even for data sets that are going to be large–is incredibly fast. The time elapsed on the execution of even fairly complex queries is nowhere near that required to execute the initial preprocessing pipeline for a particular experiment, which is the closest appropriately time-performance comparison.

Limitations

The most significant limitation to using *MongoDB* in this context, and particularly with *mongolite* in *R*, is the paste/sprintf manipulations required to create procedurally driven queries. Though this can be addressed by using filtering with something like *dplyr* following a broader query, *mongolite*'s functionality feels incomplete in this way.

The data aggregation pipeline as I've created it is pretty robust, but it is also slow: it takes several minutes to build the database from start to finish. Replacing loops within functions, as well as using parallelized versions of the same code, should help address this down the line.

Next steps

For my next steps, I'd like to begin exploring the use of MongoDB database creation/querying outside of *R*. For example, though it is possible to overcome the schematic complexity of hierarchical data such as this by dumping it all into a single *MongoDB* document, *MongoDB* itself naturally supports hierarchical structuring of data properties – *mongolite* commands don't seem to accommodate this function straightforwardly.

I'd also like to build out more robust exception handling/sets of modification to a core set of code that would permit R to flexibly extract all four types of data from a source folder with a consistent structure. To do this correctly requires true object oriented code to create classes and experiment specific methods, and *R*'s functionality for doing so wasn't a part of the course instruction is beyond my experience with *R* outside the context of the course.

Though it might require the use of a different language, I would like to build-in code to move stimuli into a consistent set of positions in a specialized file-system object, such as an hdf5 file. This would make it possible to build two separate but extremely tightly linked objects containing the gaze data and related stimuli via path specifications in the former to elements in the file-system object.

Finally, I'd like to explore the use of the MapReduce functionality *MongoDB* provides to enable in-database processing and data alignment between the true, raw gaze data (just lists of gaze coordinate positions), and fixations. This would make it possible to apply a consistent set of algorithms to move from raw data to the data ultimately to be analyzed. This is also likely to be significantly faster than applying functions individually to each raw data set as they are added to the database, permitting increased flexibility in terms of testing the effects of different parameter settings within those algorithms.

References

Itti, L. (2015). New Eye-Tracking Techniques May Revolutionize Mental Health Screening. *Neuron*, *88*(3), 442–444.

Jain, Y., Schedlbauer, M., & Durant, K. (2015). *Data Collection, Integration and Analysis* (Vol. 17).

Wilming, N., Onat, S., Ossandón, J. P., Açik, A., Kietzmann, T. C., Kaspar, K., Gameiro, R. R., et al. (2017). An extensive dataset of eye movements during viewing of complex images. *Scientific Data*, *4*, 160126.