Term Project Code

Dylan Rose

We'll begin with the data aggregation and collection code, which has to be custom-configured for each of the experiment data-sets. So let's do some setup, library loading, and format control first:

Setup and sanity checking

```
rm(list = ls())

setwd("~/Sync/projects/work_projects/class_work/collecting_storing_data/term_project/")

# We need to flag one piece of code later to change slightly based on the
# operating system this is run on (OSX vs Ubuntu), so get the os name here:
system_type <- unname(Sys.info()[1])
number_of_sets <- 4

# library(h5)
library(mongolite)
library(knitr)
library(tidyverse)
library(stringr)
library(magrittr)

# Wraps text for in-line code chunks, making the output document bit more
# readable
opts_chunk$set(width = 20, comment = "", warning = FALSE, message = FALSE, echo = TRUE,
    tidy = TRUE, size = "small")
```

All of the data is stored in the "input_data" sub-directory. We'll use a bit of code to loop through and get some basic information about the experiments.:

```
directory_to_check_for_input_data <- "./input_data"
input_data_sub_dirs <- list.dirs(path = directory_to_check_for_input_data, full.names = FALSE,
    recursive = FALSE)
```

Now we can proceed to set up paths for each dataset to be able to access stuff from them as we go:

```
full_set_paths <- paste(paste(getwd(), "/input_data/", sep = ""), input_data_sub_dirs,
    sep = "")
```

We also need to check that each directory has a text file – "experiment_parameters.txt" – that has some information about each experiment we're going to need down the line, so setup paths for those:

```
paths_to_parameters_files <- paste(full_set_paths, "/experiment_parameters.txt",
    sep = "")
```

Now to do some basic testing/sanity checking. To proceed confidently, we need to test that:

1. the number of data sets in the "input_data" folder matches the number described for the project.
2. R says that each folder exists.
3. R says that each folder contains a text file of experimental parameter data.

```
# Test 1: number of subdirectories matches the number of data sets described
# for the experiment:
correct_num_sub_dirs <- length(input_data_sub_dirs) == number_of_sets

# Test 2: each folder exists, so the sum of the exist queries for each
# folder is equal to the number of sets:
number_of_existing_sub_dirs <- sum(unlist(lapply(full_set_paths, dir.exists)))
correct_num_existing_sub_dirs <- number_of_existing_sub_dirs == number_of_sets

# Test 3: each folder contains a experiment parameters text file, so the
# number of exist queries for these files is equal to the number of sets
number_of_experiment_parameters_files <- sum(unlist(lapply(paths_to_parameters_files,
    file.exists)))
correct_num_experiment_param_files <- number_of_experiment_parameters_files ==
    number_of_sets

(sanity_check_df <- data.frame(correct_num_sub_dirs, correct_num_existing_sub_dirs,
    correct_num_experiment_param_files))
```

```
  correct_num_sub_dirs correct_num_existing_sub_dirs
1                 TRUE                          TRUE
  correct_num_experiment_param_files
1                               TRUE
```

If all of the above read true, then we should feel safe proceeding.

Setup the experiment parameters table

Next step is to get the experiment parameters data from each text file in each of the data set sub-directories. To do this, we'll first write a couple of functions:

```
# function to get a list of readlines output and return a correctly
# configured/cleaned up list
clean_experiment_params_list <- function(list_of_experiment_params) {
    split_data <- unlist(lapply(list_of_experiment_params, str_split, pattern = ":"))
    split_data_output <- data.frame(split_data[2], paste(split_data[4], split_data[5],
        sep = ":"), as.integer(split_data[7]), as.integer(split_data[9]), as.double(split_data[11]),
        split_data[13], as.integer(split_data[15]), as.integer(split_data[17]),
        split_data[19])
    colnames(split_data_output) <- c("experiment", "source", "screen_width_pix",
        "screen_height_pix", "viewing_distance_in", "eye_tracker", "sampling_rate_hz",
        "trial_duration_seconds", "list_of_tasks")
    return(split_data_output)
}

get_experimental_parameters_from_txt <- function(path_to_param_text_file) {
    data_from_text_file <- readLines(path_to_param_text_file, warn = FALSE)
    output_data_frame <- clean_experiment_params_list(data_from_text_file)
    return(output_data_frame)
}
```

Now apply it and bind the resulting list of data frames together into one frame:

```
all_experiments_parameter_table <- do.call("rbind", lapply(paths_to_parameters_files,
    get_experimental_parameters_from_txt))
```

VIP Dataset

The eye movement data for the VIP dataset is stored in a set of text files generated by the iView eye-tracking data preprocessing software. To start, we're gonna loop through these text files and get all of the fixation data from each of them. To do that we'll use the following functions:

```
# 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)
    }
}

# Cleanup function for fixation data output
clean_vip_text_data <- function(data_frame_to_clean) {
    colnames(data_frame_to_clean) <- c("x_position", "y_position", "fixation",
        "duration", "image", "subject", "task_type")
    data_frame_to_clean$x_position <- as.integer(data_frame_to_clean$x_position)
    data_frame_to_clean$y_position <- as.integer(data_frame_to_clean$y_position)
    data_frame_to_clean$fixation <- as.integer(data_frame_to_clean$fixation)
    data_frame_to_clean$duration <- as.integer(data_frame_to_clean$duration/1000)
    data_frame_to_clean$image <- as.character(data_frame_to_clean$image)
    data_frame_to_clean$subject <- as.character(data_frame_to_clean$subject)
    data_frame_to_clean$task_type <- as.character(data_frame_to_clean$task_type)
    return(data_frame_to_clean)
}
```

Now get the list of text data files on the path:

```
# Get the list of text files
vip_data_file_path <- full_set_paths[str_detect(full_set_paths, "vip_dataset")]
text_files_on_path <- list.files(paste(vip_data_file_path, "/Events", sep = ""),
    "*.txt", full.names = TRUE)
vip_text_data_files <- text_files_on_path[!str_detect(text_files_on_path, "experiment")]
```

To make the read_vip_text_data function work, we're gonna need a lookup table from a csv file that indicates which subject completed which task:

```
vip_lookup_table <- read.csv(paste(vip_data_file_path, "/subjects.csv", sep = ""))
```

Now run the function using an lapply and bind the list of dataframes together:

```
subjects_to_drop <- c("6393", "7446", "9320", "9756")
all_vip_fixation_data_list <- lapply(vip_text_data_files, read_vip_text_data,
    subject_lookup_table = vip_lookup_table)
all_vip_fixation_data_frame <- clean_vip_text_data(data.frame(do.call("rbind",
    all_vip_fixation_data_list)))
all_vip_fixation_data_frame <- all_vip_fixation_data_frame[, c(1, 2, 6, 5, 3,
    7, 4)]
vip_fixation_data_frame <- filter(all_vip_fixation_data_frame, !subject %in%
    subjects_to_drop)
vip_fixation_data_frame$experiment <- rep(as.character("vip"), nrow(vip_fixation_data_frame))
```

For this one we've already got the data we need for the subject table:

```
vip_subject_data <- filter(vip_lookup_table, !subject_id %in% subjects_to_drop)
subject <- as.character(vip_subject_data$subject_id)
experiment <- rep("vip", length(subject))
gender <- vip_subject_data$gender
age <- unlist(lapply(lapply(str_split(vip_subject_data$agegroup, "-"), as.integer),
    mean))
neuropsych_status <- rep(NA, length(subject))
vip_subject_dataframe <- data.frame(subject, experiment, age, gender, neuropsych_status,
    stringsAsFactors = FALSE)
vip_subject_dataframe$subject <- unlist(as.character(vip_subject_dataframe$subject))
```

```
vip_subject_dataframe$gender <- str_replace(str_replace(as.character(vip_subject_dataframe$gender),
    "Female", "F"), "Male", "M")
```

Now for the image data. We don't actually have the images for this experiment, just their names, so the path info is just NAs:

```
vip_image_raw_data <- read.csv(paste(vip_data_file_path, "/nusef_images.txt",
    sep = ""), header = FALSE)
image <- vip_image_raw_data$V1
image_labels <- str_replace(image, ".[jJ][pP][gG]", "")
full_image_path <- rep(NA, nrow(vip_image_raw_data))
experiment <- rep("vip", nrow(vip_image_raw_data))
vip_image_dataframe <- data.frame(full_image_path, image, image_labels, experiment,
    stringsAsFactors = FALSE)
vip_image_dataframe$experiment <- as.character(vip_image_dataframe$experiment)
vip_image_dataframe$image <- as.character(vip_image_dataframe$image)
```

VIU Dataset

The data for the VIU dataset are stored in a set of task-specific text files. This makes loading and aggregating the data a bit easier. The one thing that needs to be added, at least initially, is a column identifying which task the observer was performing.

```
read_viu_text_data <- function(file_path) {
    if (file.exists(file_path)) {
        task_type <- str_split(file_path, "/", simplify = TRUE)
        task_type <- str_split(task_type[length(task_type)], ".txt", simplify = TRUE)[1]
        raw_data <- read.table(file_path, header = TRUE)
        raw_data$task_type <- rep(task_type, nrow(raw_data))
        return(as.data.frame(raw_data))
    } else {
        sprintf("Cant find file %s at the specified path!", file_path)
    }
}
```

This function handles converting the variable types and renaming the columns appropriately for the fixation data frame once it's been generated:

```
clean_viu_text_data <- function(data_frame_to_clean) {
    colnames(data_frame_to_clean) <- c("x_position", "y_position", "subject",
        "image", "fixation", "task_type")
    data_frame_to_clean$x_position <- as.integer(data_frame_to_clean$x_position)
    data_frame_to_clean$y_position <- as.integer(data_frame_to_clean$y_position)
    data_frame_to_clean$duration <- rep(NA, length(data_frame_to_clean$y_position))
    data_frame_to_clean$subject <- as.character(data_frame_to_clean$subject)
    data_frame_to_clean$image <- as.character(data_frame_to_clean$image)
    data_frame_to_clean$fixation <- as.integer(data_frame_to_clean$fixation)
    data_frame_to_clean$task_type <- as.character(data_frame_to_clean$task_type)
    return(data_frame_to_clean)
}
```

First get the paths to the text data containing the fixation data:

```
# Get the list of text files
viu_data_file_path <- full_set_paths[str_detect(full_set_paths, "viu_dataset")]
```

```
text_files_on_path <- list.files(viu_data_file_path, "*.txt", full.names = TRUE)
viu_text_data_files <- text_files_on_path[!str_detect(text_files_on_path, "experiment")]
```

Now get the list of the gaze data text files and lapply this function over it, producing a concatenated data table of the fixation data for each subject/trial/task type:

```
all_viu_fixation_data_list <- lapply(viu_text_data_files, read_viu_text_data)
all_viu_fixation_data_frame <- clean_viu_text_data(data.frame(do.call("rbind",
    all_viu_fixation_data_list)))
```

Subjects 20-22 in this experiment are missing some data, so we're not going to include them here:

```
subjects_to_drop <- c(20, 21, 22)
viu_fixation_data_frame <- filter(all_viu_fixation_data_frame, !subject %in%
    subjects_to_drop)
viu_fixation_data_frame$subject <- as.character(viu_fixation_data_frame$subject)
viu_fixation_data_frame$experiment <- rep(as.character("viu"), nrow(viu_fixation_data_frame))
```

Finally, fix up the strings for the tasks so that they're consistently named with the other experiments:

```
viu_fixation_data_frame$task_type <- str_replace(viu_fixation_data_frame$task_type,
    "FreeViewing", "free_viewing")
viu_fixation_data_frame$task_type <- str_replace(viu_fixation_data_frame$task_type,
    "ObjectSearch", "object_search")
viu_fixation_data_frame$task_type <- str_replace(viu_fixation_data_frame$task_type,
    "SaliencyViewing", "saliency_viewing")
```

Now set up the subject demographic data:

```
subject <- as.character(unique(all_viu_fixation_data_frame$subject))
experiment <- rep("viu", length(subject))
age <- rep(NA, length(subject))
gender <- rep(NA, length(subject))
neuropsych_status <- rep(NA, length(subject))
viu_subject_dataframe <- data.frame(subject, experiment, age, gender, neuropsych_status,
    stringsAsFactors = FALSE)
viu_subject_dataframe$experiment <- as.character(viu_subject_dataframe$experiment)
viu_subject_dataframe$subject <- as.character(viu_subject_dataframe$subject)
```

Now for the meta-data on the images. This will give the name of the image, it's full file path, and a key aligning it with the numerical image number used for each subject/trial pair:

```
full_image_path <- list.files(paste(viu_data_file_path, "/Images", sep = ""),
    "*.jpg", full.names = TRUE)
image <- str_subset(unlist(str_split(full_image_path, "/Images/")), "image_*")
image_labels <- str_extract(unlist(str_split(str_subset(unlist(str_split(full_image_path,
    "/image_r_*")), ".jpg"), ".jpg")), "^[0-9]{1,3}$")
image_labels <- image_labels[!is.na(image_labels)]
experiment <- rep("viu", length(image_labels))
viu_image_dataframe <- data.frame(full_image_path, image, image_labels, experiment)
viu_image_dataframe$image <- as.character(viu_image_dataframe$image)
viu_image_dataframe$experiment <- as.character(viu_image_dataframe$experiment)
```

In this case we have to replace the "image_label" variable with the "image" variable to make matching work later on:

```
viu_fixation_data_frame$image <- viu_image_dataframe$image[match(viu_fixation_data_frame$image,
    viu_image_dataframe$image_label)]
```

Wilmings Dataset

The data for the Wilmings dataset are stored in an hdf file. As the repository provides clean python code to extract data from it by experiment, I've just written a bit of preprocessing code to wrap around that and spit the data out into a single csv,formatted correctly:

```
wilmings_fixation_data_frame <- read.csv(paste(full_set_paths[str_detect(full_set_paths,
    "wilmings")], "/wilmings_fixation_data.csv", sep = ""))
wilmings_fixation_data_frame$experiment <- rep(as.character("wilmings"), nrow(wilmings_fixation_data_fra
```

The subject meta-data is also stored in a decently clean csv file, so load that in and do the few necessary fixes:

```
wilmings_subject_dataframe <- read.csv(paste(full_set_paths[str_detect(full_set_paths,
    "wilmings")], "/additional_meta/meta_headfixed.csv", sep = ""))
wilmings_subject_dataframe$experiment <- rep("wilmings", nrow(wilmings_subject_dataframe))
wilmings_subject_dataframe$neuropsych_status <- rep(NA, nrow(wilmings_subject_dataframe))
colnames(wilmings_subject_dataframe) <- c("subject", "gender", "age", "experiment",
    "neuropsych_status")
wilmings_subject_dataframe <- wilmings_subject_dataframe[, c(1, 4, 3, 2, 5)]
wilmings_subject_dataframe$gender <- str_replace(str_replace(wilmings_subject_dataframe$gender,
    "f", "F"), "m", "M")
```

Finally, the images are stored in a "stimuli" sub-folder, so we'll get a list of them from the folder directly:

```
image <- list.files(paste(full_set_paths[str_detect(full_set_paths, "wilmings")],
    "/stimuli/", sep = ""), "*.png")
image_labels <- str_extract(image, "[0-9]{1,2}")
full_image_path <- list.files(paste(full_set_paths[str_detect(full_set_paths,
    "wilmings")], "/stimuli", sep = ""), "*.png", full.names = TRUE)
experiment <- rep("wilmings", length(image_labels))
wilmings_image_dataframe <- data.frame(full_image_path, image, image_labels,
    experiment)
wilmings_image_dataframe$image <- as.character(wilmings_image_dataframe$image)
```

We need to do the same matching procedure we did for the VIU dataset because the image label/image name variables have their content switched in each of those frames:

```
wilmings_fixation_data_frame$image <- wilmings_image_dataframe$image[match(wilmings_fixation_data_frame$
    wilmings_image_dataframe$image_label)]
```

NU Dataset

The fixation data for this experiment are kept in a set of subject/trial specific csv files. So first we need to get a list of those:

```
nu_data_file_path <- full_set_paths[str_detect(full_set_paths, "nu_dataset")]
nu_text_data_files <- list.files(paste(nu_data_file_path, "/data", sep = ""),
    "*.csv", full.names = TRUE)
```

We'll get task information per-trial from another csv which we'll use as a lookup table while reading the fixation data files. So load that here:

```
nu_task_lookup <- read.csv(paste(nu_data_file_path, "/in_lab_expt_subject_image_by_trial_salience_map.cs
    sep = ""))
```

Now a function to read in the data from each appropriately, and one to do some cleaning:

```
read_nu_text_data <- function(file_path, task_lookup_table) {
    if (file.exists(file_path)) {
        output_fixation_frame <- read.csv(file_path, header = TRUE) %>% select(x_position,
            y_position, fixation_number)
        subject_trial_info <- unlist(str_split(unlist(str_split(unlist(str_split(file_path,
            "data/"))[4], "-"))[1], "_"))
        output_fixation_frame$subject <- rep(subject_trial_info[1], nrow(output_fixation_frame))
        output_fixation_frame$image <- rep(filter(task_lookup_table, subject_id ==
            subject_trial_info[1] & trial_n == as.integer(subject_trial_info[3]))$image_name,
            nrow(output_fixation_frame))
        output_fixation_frame$task <- rep(as.character(filter(task_lookup_table,
            subject_id == subject_trial_info[1] & trial_n == subject_trial_info[3])$task_type),
            nrow(output_fixation_frame))
        return(as.data.frame(output_fixation_frame))
    } else {
        sprintf("Cant find file %s at the specified path!", file_path)
    }
}

clean_nu_text_data <- function(data_frame_to_clean) {
    colnames(data_frame_to_clean) <- c("x_position", "y_position", "fixation",
        "subject", "image", "task_type")
    data_frame_to_clean$duration <- rep(NA, nrow(data_frame_to_clean))
    data_frame_to_clean$x_position <- as.integer(data_frame_to_clean$x_position)
    data_frame_to_clean$y_position <- as.integer(data_frame_to_clean$y_position)
    data_frame_to_clean$fixation <- as.integer(data_frame_to_clean$fixation)
    data_frame_to_clean$subject <- as.character(data_frame_to_clean$subject)
    data_frame_to_clean$image <- as.character(data_frame_to_clean$image)
    data_frame_to_clean$task_type <- as.character(data_frame_to_clean$task)
    data_frame_to_clean <- data_frame_to_clean[, c(1, 2, 4, 5, 3, 6, 7)]
    return(data_frame_to_clean)
}
```

Here we loop over the files and return the subject/trial specific fixation data, then clean/rbind it into the output data frame for this set:

```
all_nu_fixation_data_list <- lapply(nu_text_data_files, read_nu_text_data, task_lookup_table = nu_task_
nu_fixation_data_frame <- clean_nu_text_data(data.frame(do.call("rbind", all_nu_fixation_data_list)))
nu_fixation_data_frame$experiment <- rep(as.character("nu"), nrow(nu_fixation_data_frame))
```

For the subject data, we have a csv file containing at least some of the necessary information:

```
nu_subject_dataframe <- read.csv(paste(nu_data_file_path, "/subject_number_record.csv",
    sep = ""))
nu_subject_dataframe <- filter(nu_subject_dataframe, use_dont_use == "use") %>%
    select(subject_id, gender)
colnames(nu_subject_dataframe) <- c("subject", "gender")
nu_subject_dataframe$age <- rep(NA, nrow(nu_subject_dataframe))
nu_subject_dataframe$experiment <- rep("nu", nrow(nu_subject_dataframe))
nu_subject_dataframe$neuropsych_status <- rep(NA, nrow(nu_subject_dataframe))
nu_subject_dataframe <- nu_subject_dataframe[, c(1, 4, 3, 2, 5)]
```

For the image data, we've gotta do a little more construction work, as we have the image used per trial, and so can get the label/filename, but not the full path. What data we do have is already in the task lookup csv, so just get it out of there:

```
image <- as.character(nu_task_lookup$image_name)
image_labels <- str_replace(image, ".jpg", "")
full_image_path <- rep(NA, length(image_labels))
experiment <- rep("nu", length(image_labels))
nu_image_dataframe <- data.frame(full_image_path, image, image_labels, experiment)
nu_image_dataframe$image <- as.character(nu_image_dataframe$image)
```

Bind all the experiment-specific data together, then rbind:

Now that we have all of the appropriate data from all of the experiments, formatted correctly, we can go ahead and bind it together:

```
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)
```

Database setup/data insert

To store this data in our *MongoDB* database, all we have to do is create the connection and insert the data:

```
fixation_data_connection <- mongo("fixation_data")
fixation_data_connection$drop()
fixation_data_connection$insert(all_data)
```

```
List of 5
 $ nInserted  : num 1062536
 $ nMatched   : num 0
 $ nRemoved   : num 0
 $ nUpserted  : num 0
 $ writeErrors: list()
```

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

```
List of 5
 $ nInserted  : num 4
 $ nMatched   : num 0
 $ nRemoved   : num 0
 $ nUpserted  : num 0
 $ writeErrors: list()
```

Testing the database

Now we should do some queries of differing complexity just as a test run to check the time it takes to complete these a range of query types:

```
system.time(female_count <- fixation_data_connection$count("{\"gender\":\"F\"}"))
```

```
   user  system elapsed
  0.000   0.000   0.696
```

```
system.time(male_age_27_count <- fixation_data_connection$count("{\"gender\":\"M\",
\t\"age\":27}"))
```

```
   user  system elapsed
  0.000   0.000   0.609
```

```
system.time(free_viewing_female_22_only_data <- fixation_data_connection$find("{\"task_type\":\"free_vi
\t\"gender\":\"F\",
\t\"age\":22}"))
```

```
   user  system elapsed
  0.338   0.016   0.942
```

Let's check to make sure we're getting sane results. If we filter the data in the "all_data" table and count the number of twenty-two year-old women doing free viewing, the number of rows in the data queried from database and in the table should match:

```
female_22_free_viewing_data_from_table <- filter(all_data, gender == "F", age ==
    22, task_type == "free_viewing")
nrow(female_22_free_viewing_data_from_table) == nrow(free_viewing_female_22_only_data)
```

```
[1] TRUE
```

Looks like we're getting sensible results in good times, so we're done here.

Data for testing

Submitted with my package for the assignment is a bson file you can load to validate that my queries and the like are correct. There's no way to easily share the raw data, so this will be the limit in terms of my deliverables, but you should be able to get some workable results to see that what I've done is correct.

One important difference between what I'm working on and what you'll have is that I can't share the data from my experiment for IRB related reasons. Sorry.

```
data_export_subset <- filter(all_data, experiment != "nu")
meta_data_export_subset <- filter(all_experiments_parameter_table, experiment !=
    "nu")
subset_for_export_connection <- mongo("subset_for_export")
subset_for_export_connection$insert(data_export_subset)
```

```
List of 5
 $ nInserted  : num 831844
 $ nMatched   : num 0
 $ nRemoved   : num 0
 $ nUpserted  : num 0
 $ writeErrors: list()
```

```
subset_for_export_connection$insert(meta_data_export_subset)
```

```
List of 5
 $ nInserted  : num 3
 $ nMatched   : num 0
```

```
 $ nRemoved   : num 0
 $ nUpserted  : num 0
 $ writeErrors: list()
```

```
subset_for_export_connection$export(file("./dr_project_export_subset.bson"),
    bson = TRUE)
```