```
# This script reads in an excel file exported from REDCap and writes a csv usable by the app

# Run this script each time you pull down a new excel file, then re-deploy the app
# Previous iterations used the "raw" format and matched key-value pairs,
#    but since I'm already having to rewrite/harmonize I'm using the labeled CSV
#    from here on out

library(plyr)
library(dplyr)
library(lubridate)
library(tidyverse)
library(padr)
library(ggmap)
library(leaflet)
library(htmltools)
library(shinydashboard)
library(REdaS)
library(rsconnect)
library(readxl)

# Read in most recent dataset.  First column name occasionally gets corrupted
data <- read.csv("SPOTOnCML_DATA_LABELS_2021-02-05_1448.csv", stringsAsFactors = F)
colnames(data)[1] <- "PIN"

# Redcap maps to two events: DBS and Baseline
baseline <- filter(data, Event.Name == "Baseline") # Country and site info
dbs <- filter(data, Event.Name == "DBS") # Patient info
dbs <- dbs %>% select(-c(Country, Institution)) # These live in baseline

# Read in Luke's manually-curated list of institutions
clinics <- read.csv("institution_locations.csv", header = T, stringsAsFactors = F)

# Prune extra columns that app won't need, glom together the useful bits
#    Some of these might be needed for calculating other statistics
dbs <- dbs %>% select(c(PIN, Data.Access.Group, Sample.Date, WBC,
                        Receive.Date, Test.Date, Notes, X.BCR.ABL.ABL..IS., Mutation.Testing.Results))
baseline <- baseline %>% select(c(PIN, Country, Institution,
                                  Test.Requested, Date.of.birth))
df <- join(baseline, dbs, by = "PIN")

# Fix a RedCAP typo
df$Institution <- gsub("Venzuela", "Venezuela", df$Institution)

# Create a working df for institution key, country key, institution name
#    Need this for later "join" functions
clins <- clinics[, 1:3]
colnames(clins)[3] <- "Institution"



### In late 2019, MO switched entirely to LabWare and will use that
#     for all further updates. Cecilia mentioned in early '21 that
#     (she hopes) that may change in the near future

labware <- read_xlsx(list.files(".", pattern = "Customer"))
labware_key <- read_xlsx("Labware_key.xlsx")

# Prune. Transit time was used by previous iterations / stats but doesn't do any harm to calculate
labware <- labware %>%
  select(-c(`Text Id`, Analysis, `...7`, `Date Reviewed`)) %>%
  mutate(transitTime = round(`X Arrival Date` - `Sampled Date`, 0))

# Some customer names are mis-entered, so this gets broken up
labware$Customer <- gsub("BPKoirala Memorial Cancer Hospital", "BPKOIRALA", labware$Customer)

    # There are two Bhutan hospitals, so until Zaneta can answer that, pick one
labware$Customer <- gsub("Bhutan", "JIGME_DORJI_WANGCHUC", labware$Customer)

    # Two mystery samples, oh boy!  Set those to NA so they don't get picked up by the map
labware$Customer <- gsub("nil", NA, labware$Customer)

# Create a smaller working df for labware stuff.  Helps rename things properly
sites <- select(labware_key, c(institution, country, `LabWare Customer Name`, i_key, c_key))
colnames(sites) <- c("Institution", "Country", "Customer", "i_key", "c_key")
labware <- join(labware, sites, by = "Customer")


### Zaneta doesn't know how to export very well, so Luke had to clean
#     recent data before it was importable. Most significantly, Jill's
#     previous data ran through "Receive Date" 4/23/20, so everything
#     prior to that date in Zaneta's data was removed to avoid double-counting.
#     Similarly, some samples were run for BCR-ABL *only* for MA QC

postJill <- read_xlsx("DBS 2020 for LM_Labware_Cleaned.xlsx")
```

```
# Further semi-manual cleanup, oh joy
postJill <- postJill %>% select(-c(`Ordering Physician`, `Sample Type`, Comments))

# Recoding with gsub is simpler than manually adjusting cell values
postJill$Institution <- gsub("Banco Municipal de Sangre-Venezuela",
                             "Banco Municipal de Sangre", postJill$Institution)
postJill$Institution <- gsub("Vanuatu",
                             "Vila Central Hospital", postJill$Institution)
postJill$Institution <- gsub("Colonial War Memorial Hospital-Fiji",
                             "Colonial War Memorial Hospital", postJill$Institution)

# Clean up extra dfs. Can be removed; Luke just got sick of the Environment clutter
rm(baseline, data, dbs, labware_key)


# Redcap, Labware, and post-Jill counts will overlap institutions, so counting is tough.
#    Simplest solution Luke could think of was to summarise counts individually for each data set

# For each data set,
#   Group by institution, count ("summarise(n()")),
#   append to the key (contained in "clins" or "sites),
#   prune extra columns, and name that count's column based on dataset
redcapsum <- df %>% group_by(Institution) %>% summarise(n())
redcapsum <- join(redcapsum, clins, by = "Institution")
redcapsum <- redcapsum %>% select(-Institution, c_key)
colnames(redcapsum)[1] <- "rc"

labwaresum <- labware %>% group_by(Customer) %>% summarise(n())
labwaresum <- join(labwaresum, sites, by = "Customer")
labwaresum <- labwaresum %>% select(c(`n()`, i_key))
colnames(labwaresum)[1] <- "lw"

postJillsum <- postJill %>% group_by(Institution) %>% summarise(n())
postJillsum <- join(postJillsum, sites, by = "Institution")
postJillsum <- postJillsum %>% select(-c(Institution, Country, Customer, c_key))
colnames(postJillsum)[1] <- "pj"

  # Create temporary df with each *possible* institution, not each one *present*
temp <- data.frame(c(1:max(sites$i_key)), c(0))
colnames(temp) <- c("i_key", "n")

  # Glom together the three sets of counts
temp <- join(temp, redcapsum, by = "i_key")
temp <- join(temp, labwaresum, by = "i_key")
temp <- join(temp, postJillsum, by = "i_key")

  # NAs are carried over by the joins. Set to zero to avoid screwing up the sum
temp[is.na(temp)] <- 0
  # Sum across all three data sets
temp <- temp %>% mutate(total = as.numeric(rc) + lw + pj) %>% select(-c(n, lw, pj, rc, c_key))

  # Glom this summed count onto our master list of clinics
Counts <- join(temp, clinics, by = "i_key")
  # Calculate total distance traveled for each clinic (clinic's distance * sample number)
Counts <- Counts %>% mutate(clinic.miles = as.numeric(total)*as.numeric(distance_mi))

# The app will require HTML formatting for the popup
Counts <- Counts %>%
  mutate(Popup = paste(
    paste0("<b>", institution, "</b>"),
    paste0("<i>", country, "</i>"),
    paste0("Number of Samples: ", total),
    # paste0("Median Time to Sample Intake (days): ", medianTransit),
    sep = "<br/>"))

# This df is for the three summary statistics in the top bar of the app
summTable <- Counts %>%
  summarise(`Project Duration` = paste(Sys.Date() -
                                 as.Date("2017-09-22", format = "%Y-%m-%d"), "days"),
            `Samples Received` = sum(Counts$total, na.rm = T),
          # `Samples Processed` = length(dbs$bcr_abl_ct[!is.na(dbs$bcr_abl_ct)]),
            `Total Miles Travelled by Samples` = round(sum(Counts$clinic.miles, na.rm = T), -3)
          # `Median Testing Time` = paste(median(testTime, na.rm = TRUE), "days")
          # `Last Updated` = format(as.Date(max(test_date, na.rm = TRUE)), "%d %b %Y")
)
colnames(summTable) <- c("Project Duration", "Samples Processed", "Total Miles Travelled by Samples")

# Export two minimal data tables that the app can call directly
write.csv(summTable, file = "summCSV.csv", row.names = F)
write.csv(Counts, file = "CountCSV.csv", row.names = F)
```