Datafile download

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(googlesheets4)
library(qualtRics)
library(lubridate)
library(googledrive)
```

```
##
## Attaching package: 'googledrive'
##
## The following objects are masked from 'package:googlesheets4':
##
##     request_generate, request_make
```

Note that both the Qualtrics surveys and the Google sheet containing PID roster data are not world readable. Executing this script will not work for most users. It is still provided for documentation purposes.

```r
gs4_auth(path="/home1/ralmond/ownCloud/Projects/shiny-protocol-369419-250fe24b2096.json")

#gs4_auth(email="russell.g.almond@gmail.com", scopes="https://www.googleapis.com/auth/spreadsheets")
#gs4_deauth()
PPStudents <- read_sheet("1iTlM3LrTIbOZ_Fo9ROJ4VR68CJREGFlJOlZKOQ9oC8Y")
```

```
## v Reading from "PP Fall 2022 Student Assent_November 16 2022_10pm".
```

```
## v Range 'Sheet0'.
```

```r
#PPStudents
```

Consent and Assent

Both consent and assent were done through Qualtrics survey, however, a few paper forms were used instead. These were entered into the master roster spreadsheet

Download Roster

The roster is divided into sheets by teacher name and class period. The third letter of the teachers name (which was unique for the participating teachers) was used as an identifier.

```r
nclasses <- 18
rostersheet <- gs4_get("1RyNbYrXeJCz_UuWw-OdQtHm4WGkJhnCRGvvRC-hRuUg")
classnames <- pull(rostersheet$sheets,"name")[1:nclasses]
indexes <- 1:nclasses
teachers <- stringi::stri_match_first_regex(classnames,"[[:alpha:]]*$")
tcode <- substr(teachers,3,3)
period <- substr(classnames,1,1)
classcode <- paste(tcode,period,sep="")
#data.frame(classnames,tcode,period,classcode)
```

```r
class_roster <- read_sheet(rostersheet$spreadsheet_id,indexes[1])
mutate(class_roster,class=classcode[1],teacher=tcode[1],
       period=as.numeric(period[1])) -> roster
for (cl in indexes[-1]) {
  class_roster <- read_sheet(rostersheet$spreadsheet_id,cl)
  mutate(class_roster,class=classcode[cl],teacher=tcode[cl],
         period=as.numeric(period[cl])) %>%
    rbind(roster,.) -> roster
}
revname <- function(names) {
  sapply(strsplit(names,","),function(nme)
    paste(trimws(nme[2]),trimws(nme[1])))
}

mutate(roster,Student=revname(pull(roster,"Student Name"))) -> roster
```

Student names in the official system include middle names, but students don't include middle names when signing the assent forms. Lets convert the assent names to regexps, and then try to match that way.

```r
name2regex <-function(name) {
  paste(strsplit(name," ")[[1]],collapse=".*")
}
findName<- function(name,namelist) {
  ind <- grep(name2regex(name),namelist,ignore.case=TRUE)
  if (length(ind) == 0)
    warning("No match found for ",name," in rosters.\n")
  if (length(ind)>1) {
    warning("The name ",name," matched multiple names: ",
            paste(namelist[ind],collapse=", "),".\n")
  }
  ind
}
```

The column `PPStudents$Q4` contains the names of the consenting students. Match this to the column roster$Student.

```r
nomatch <- character()
multimatch <- character()
students <- pull(roster,"Student")

# for (stud in pull(PPStudents,Q4)[-1]) {
#   ind <- findName(stud,students)
#   if (length(ind) == 0) nomatch <- c(nomatch,stud)
#   else if (length(ind) > 1) multimatch <- c(multimatch,stud)
#   else roster[ind,"Assent (0/1)"] <- TRUE
# }
```

```
names(roster) <- c("StudentName", "Nickname", "StudyID", "StudentEmail",
                   "ParentName", "ParentEmail", "Consent", "Assent",
                   "Pretest_P", "ECT_P", "POT_P", "Att_Mon_11_28",
                   "Att_Tue_11_29", "Att_Wed_11_30", "Att_Thu_12_1",
                   "Att_Fri_12_2","class","teacher","period", "Student")
```

Roster has some weird stuff in Consent column. Clean it out.

```
oldRoster <- select(roster,ParentName,ParentEmail,Consent,Student)
roster %>% select(!Consent) %>%
  mutate(ParentEmail=NA_character_,StudentEmail=NA_character_,
        Consent=NA) -> roster
# These next lines are for working with previous data.  Will work entirely from script.
# for (irow in 1:nrow(roster)) {
#   if(!is.null(roster[[irow,"ParentEmail"]]))
#     roster[[irow,"ParentEmail1"]] <- as.character(roster[[irow,"ParentEmail"]])
#   if(!is.null(roster[[irow,"Consent"]]))
#     if(is.numeric(roster[[irow,"Consent"]][[1]]) ||
#        isTRUE(roster[[irow,"Consent"]][[1]]))
#       roster[[irow,"Consent1"]] <- as.numeric(roster[[irow,"Consent"]][[1]])
# }
# select(roster,!Consent) %>%
#   mutate(Consent=Consent1,ParentEmail=ParentEmail1) %>%
#   select(!Consent1) -> roster
# roster %>% select(!consent1) %>% select(!Conset) %>% select(!ParentEmail1) ->
#   roster
# roster %>% select(!Consent1) -> roster
```

Download consent and assent forms.

```
qualtricsIDs <- c(PRE="SV_bkALP80IYWnEgkK",ECT="SV_3BKybDb3f2v4QU6",
                  POT="SV_1MotNHu1dPEAJnM",Assent="SV_07esYssbhDccXs1",
          Consent="SV_8vsI01klt7DSPMG")
```

```
consent <- fetch_survey(qualtricsIDs["Consent"])
```

```
##   |                                                                              |

##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   StartDate = col_datetime(format = ""),
##   EndDate = col_datetime(format = ""),
##   Progress = col_double(),
##   `Duration (in seconds)` = col_double(),
##   Finished = col_logical(),
##   RecordedDate = col_datetime(format = ""),
##   RecipientLastName = col_logical(),
##   RecipientFirstName = col_logical(),
##   RecipientEmail = col_logical(),
##   ExternalReference = col_logical(),
##   LocationLatitude = col_double(),
##   LocationLongitude = col_double(),
##   Q4_Size = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
## 'StartDate', 'EndDate', and 'RecordedDate' were converted without a specific timezone
## * To set a timezone, visit https://www.qualtrics.com/support/survey-platform/managing-your-account/
## * Timezone information is under 'User Settings'
## * See https://api.qualtrics.com/instructions/docs/Instructions/dates-and-times.md for more
```

```r
assent <- fetch_survey(qualtricsIDs["Assent"])
```

```
##   |                                                                              |
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   StartDate = col_datetime(format = ""),
##   EndDate = col_datetime(format = ""),
##   Progress = col_double(),
##   `Duration (in seconds)` = col_double(),
##   Finished = col_logical(),
##   RecordedDate = col_datetime(format = ""),
##   RecipientLastName = col_logical(),
##   RecipientFirstName = col_logical(),
##   RecipientEmail = col_logical(),
##   ExternalReference = col_logical(),
##   LocationLatitude = col_double(),
##   LocationLongitude = col_double(),
##   Q2_Size = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```r
mutate(assent,StartDate=force_tz(StartDate,"EST")) -> assent
all(pull(assent,"Finished"))
```

```
## [1] TRUE
```

```r
all(pull(consent,"Finished"))
```

```
## [1] TRUE
```

No incomplete surveys, so assume all are offering consent.

Match Assent Forms

Original script provided non-matching names. Hidden for archival version.

```r
assenters <- na.omit(unique(pull(assent,"Q4")))
allStudents <- pull(roster,"Student")
length(assenters)
```

```
## [1] 378
```

```r
nomatch <- character()
multimatch <- character()
for (student in assenters) {
 ind <- findName(student,allStudents)
```

```
  if (length(ind) == 0) nomatch <- c(nomatch,student)
  else if (length(ind) > 1) multimatch <- c(multimatch,student)
  else roster[ind,"Assent"] <- TRUE
}
length(nomatch)
```

```
## [1] 16
```

```
length(multimatch)
```

```
## [1] 17
```

This next bit of code deals with students who go by a different name than the official name from the class roster.

Once again, output is hidden to conceal student names.

```
snConsent <- pull(consent,"Q8")
for (nme in nomatch) {
  ind <- findName(nme,snConsent)
  if (length(ind)==1) {
   # cat("Student name ",nme," matches ",snConsent[ind],".\n")
  }
}

print("Nicknames:")
nicknames <- pull(roster,"Nickname")
for (nme in nomatch) {
  ind <- findName(nme,nicknames)
  if (length(ind)==1) {
  #  cat("Student name ",nme," matches ",allStudents[ind],".\n")
  }
}
```

Needed human intervention for the remaining cases. Thesee were added to an “Aliases” sheet in the roster.

```
aliases <- read_sheet(rostersheet$spreadsheet_id,"Aliases")
```

```
## v Reading from "Physics Playground rosters".
```

```
## v Range ''Aliases''.
```

```
mmatch <- character()
for (student in aliases$RosterName[!is.na(aliases$UsedName)]) {
 ind <- findName(student,allStudents)
  if (length(ind) == 0) nomatch <- c(nomatch,student)
  else if (length(ind) > 1) mmatch <- c(mmatch,student)
  else roster[ind,"Assent"] <- TRUE
}
mmatch
```

```
## character(0)
```

```
sum(roster[["Assent"]],na.rm=TRUE)
```

```
## [1] 331
```

```
nrow(roster)
```

```
## [1] 385
```

In a few cases we have multiple responses from the student. We checked the response time versus the bell schedule as many of the student completed the assent form during class.

```
bells <- make_datetime(2022,12,2,c(8,9,9,10,11,13,14,15),
                       c(0,0,55,50,45,15,10,00),tz="EST")
whichPeriod<- function (dt) {sum(dt>bells)}
multiMatchBells <- data.frame(name=character(),period=numeric())
for (stud in multimatch) {
  rows <- grep(stud,pull(assent,Q4))
  for (row in rows) {
    per <- whichPeriod(assent[[row,"StartDate"]])
    #cat("Studnet ",assent[[row,"Q4"]]," filled out the questionnaire in period ",per,".\n")
    multiMatchBells <- rbind(multiMatchBells,list(stud,per))
  }
}
names(multiMatchBells) <- c("Student", "Period")
#multiMatchBells
```

Parent Consent

```
nomatchConsent <- character()
multiConsent <- character()

for (irow in 1:nrow(consent)) {
  student <- consent[[irow,"Q8"]]
  ind <- findName(student,allStudents)
  if (length(ind) == 0) nomatchConsent <- c(nomatchConsent,student)
  else if (length(ind) > 1) multihConsent <- c(multiConsent,student)
  else {
    roster[ind,"Consent"] <- TRUE
    roster[ind,"ParentName"] <- consent[[irow,"Q6"]]
    roster[ind,"ParentEmail"] <- consent[[irow,"Q9"]]
  }
}
length(nomatchConsent)
```

```
## [1] 23
```

```
length(multiConsent)
```

```
## [1] 0
```

Again, some manual checking for consents which were not matched correctly.

```
reconcilliation <- read_sheet(rostersheet,"Unmatched Consents")
```

```
## v Reading from "Physics Playground rosters".

## v Range ''Unmatched Consents''.

## New names:
## * `` -> `...3`
```

```
reconcilliation %>% mutate(RosterName=if_else(RosterName=="NA",NA_character_,RosterName)) -> reconcillia
```

```
conStudent <- consent$Q8
for (i in 1:nrow(reconcilliation)) {
  student <- reconcilliation[[i,"RosterName"]]
  if (is.na(student)) next
  irow <- findName(reconcilliation[[i,"student"]],conStudent)
  if (length(irow) !=1) next
  ind <- findName(student,allStudents)
  if (length(ind)==1)  {
    roster[ind,"Consent"] <- TRUE
    roster[ind,"ParentName"] <- consent[[irow,"Q6"]]
    roster[ind,"ParentEmail"] <- consent[[irow,"Q9"]]
  }
}
```

Look for paper consent forms

```
oldRoster$Consent1 <- NA_integer_
ncon <- sapply(oldRoster$Consent,is.numeric)
oldRoster$Consent1[ncon] <- sapply(oldRoster$Consent[ncon],as.numeric)
oldRoster$Email1 <- NA_character_
nemail <- sapply(oldRoster$ParentEmail,is_character)
oldRoster$Email1[nemail] <- sapply(oldRoster$ParentEmail[nemail],as.character)
pConsent_p <- is.na(roster$Consent)&!is.na(oldRoster$Consent1)
oldRoster %>% filter(pConsent_p) %>% select(Student,ParentName,Consent1,Email1) -> paperConsent
length(paperConsent)
```

```
## [1] 4
```

```
stopifnot(!any(roster[pConsent_p,"Student"] != paperConsent$Student))
roster$ParentName[pConsent_p] <- paperConsent$ParentName
roster$ParentEmail[pConsent_p] <- paperConsent$Email1
roster$Consent[pConsent_p] <- paperConsent$Consent1
```

At this point, NAs are FALSE

```
roster$Assent[is.na(roster$Assent)] <- FALSE
roster$Consent[is.na(roster$Consent)] <- FALSE
sum(roster$Assent&roster$Consent)
```

```
## [1] 276
```

So total of 276 unique study IDs meeting both consent and assent.

Note, that because of absences, not all students completed pre and post-tests. So screen on having answered at least 1/2 of the items on these tests.

Pretest Checking

```
pretest <- fetch_survey(qualtricsIDs["PRE"], force=TRUE, label=FALSE, convert=FALSE)
```

```
##   |                                                                              |
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   StartDate = col_datetime(format = ""),
##   EndDate = col_datetime(format = ""),
##   IPAddress = col_character(),
##   RecordedDate = col_datetime(format = ""),
##   ResponseId = col_character(),
##   RecipientLastName = col_logical(),
##   RecipientFirstName = col_logical(),
##   RecipientEmail = col_logical(),
##   ExternalReference = col_logical(),
##   DistributionChannel = col_character(),
##   UserLanguage = col_character(),
##   ID = col_character(),
##   First = col_character(),
##   Last = col_character(),
##   Q80 = col_character(),
##   Age = col_character(),
##   Ethnicity_9_TEXT = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```r
pretestInfo <- select(pretest,all_of(c("Finished","ID","RecordedDate",
                                       "First","Last","Q80")))
validEmail <- function (eml) {
  valid <- !is.na(eml)
  valid[valid] <- stringi::stri_detect_fixed(eml[valid],'@')
  valid[valid] <- !stringi::stri_detect_fixed(eml[valid],"fsus")
  valid
}
mutate(pretestInfo,Email=if_else(validEmail(Q80),Q80,NA_character_)) ->
  pretestInfo
pretestInfo %>% mutate(Email=sapply(strsplit(Email," "),last)) -> pretestInfo
mutate(pretestInfo,Finished=as.logical(Finished)) -> pretestInfo
pretestInfo$NQcount <- apply(!is.na(select(pretest,starts_with("A-NQ"))),1,sum)
pretestInfo$FQcount <- apply(!is.na(select(pretest,starts_with("A-FQ"))),1,sum)
pretestInfo$FL5 <- apply(!is.na(select(pretest,starts_with("FL_5"))),1,sum)
pretestInfo <- mutate(pretestInfo,
                      ValidResponse = Finished | ((NQcount >7)+(FQcount >6) + (FL5>13)) >2) %>%
               mutate(TotalReponses = NQcount + FQcount+FL5)
```

Check for both duplicates (students attempting pretest more than once) as well as mismatches between the name, collected in the pretest, and the names from the roster/assent process.

Old criteria (P0) Qualtrics marked the assessment as finished. New criteria (P) student completed at least half of the cognitive and non-cognitive items.

```r
duplicateIDs <- list()
nomatch <- character()
ProblemNames <- list()
for (irow in 1:nrow(pretestInfo)) {
  if (!pretestInfo[[irow,"ValidResponse"]]) next
  ID <- pretestInfo[[irow,"ID"]]
  name <- paste(pretestInfo[[irow,"First"]],pretestInfo[[irow,"Last"]])
  idmatch <- grep(ID,roster$StudyID)
  if (length(idmatch) > 1) {
    duplicateIDs <- c(duplicateIDs,list(list(ID=ID,matches=idmatch)))
  }
  namematch <- findName(name,roster$Student)
  if (length(namematch) < 1) {
    nomatch <- c(nomatch,name)
    next
  }
  if (length(idmatch)==0L) ind <- namematch
  else  ind <- intersect(idmatch,namematch)
  if (length (ind) < 1) {
          ProblemNames <- c(ProblemNames,list(list(name=name,ID=ID,
                                                  Problem="Name ID mismatch")))
      next
  }
  if (length(ind) > 1) {
    ProblemNames <- c(ProblemNames,list(list(name=name,ID=ID,
                                                Problem="Multiple Matches")))
  } else {
    roster[ind,"Pretest_P"] <- TRUE
    roster[ind,"Pretest_PO"] <- pretestInfo[[irow,"Finished"]]
    roster[ind,"StudyID"] <- ID
    pre_eml <- pretestInfo[[irow,"Email"]]
    if (!is.na(pre_eml)) {
      eml <- roster[[ind,"StudentEmail"]]
      if (!is.na(eml) && nchar(eml)>0) {
        if (eml != pre_eml) {
          ProblemNames <- c(ProblemNames, list(list(name=name,ID=ID,
                                                    Problem="Email mismatch",
                                                    Pretest=pre_eml,
                                                    Roster=eml)))
        }
      } else {
        roster[[ind,"StudentEmail"]] <- pre_eml
      }
    }
  }

}
length(duplicateIDs)
```

```
## [1] 0
```

```
length(ProblemNames)
```

```
## [1] 5
```

Again fix nicknames and other name issues.

```
aliases <- read_sheet(rostersheet$spreadsheet_id,"Aliases")
```

```
## v Reading from "Physics Playground rosters".
```

```
## v Range ''Aliases''.
```

```
mutate(pretestInfo, Name=paste(First,Last)) -> pretestInfo
is.na(pretestInfo$Name) <- is.na(pretestInfo$First) & is.na(pretestInfo$Last)
for (arow in 1:nrow(aliases)) {
  if (is.na(aliases[[arow,"AltName"]])) next
  irow <- grep(aliases[[arow,"AltName"]],pretestInfo$Name)
  if (!pretestInfo[[irow,"ValidResponse"]]) next
  ID <- pretestInfo[[irow,"ID"]]
  name <- aliases[[arow,"RosterName"]]
  idmatch <- grep(ID,roster$StudyID)
  if (length(idmatch) > 1) {
    duplicateIDs <- c(duplicateIDs,list(list(ID=ID,matches=idmatch)))
  }
  namematch <- findName(name,roster$Student)
  if (length(namematch) < 1) {
    nomatch <- c(nomatch,name)
    next
  }
  if (length(idmatch)==0L) ind <- namematch
  else  ind <- intersect(idmatch,namematch)
  if (length (ind) < 1) {
        ProblemNames <- c(ProblemNames,list(list(name=name,ID=ID,
                                            Problem="Name ID mismatch")))
     next
  }
  if (length(ind) > 1) {
    ProblemNames <- c(ProblemNames,list(list(name=name,ID=ID,
                                          Problem="Multiple Matches")))
  } else {
    roster[ind,"Pretest_P"] <- TRUE
    roster[ind,"StudyID"] <- ID
    pre_eml <- pretestInfo[[irow,"Email"]]
    if (!is.na(pre_eml)) {
      eml <- roster[[ind,"StudentEmail"]]
      if (!is.na(eml) && nchar(eml)>0) {
        if (eml != pre_eml) {
          ProblemNames <- c(ProblemNames, list(list(name=name,ID=ID,
                                                    Problem="Email mismatch",
                                                    Pretest=pre_eml,
                                                    Roster=eml)))
        }
      } else {
        roster[[ind,"StudentEmail"]] <- pre_eml
      }
    }
  }

}
```

Check for duplicate Pretest IDs

In a few cases, IDs were re-used for students who joined the study after the first day. So need to check if duplicate IDs are same person retaking the test or multiple people re-using an ID.

```
usedIDs <- pretestInfo$ID[pretestInfo$ValidResponse]
duplicates <- unique(usedIDs[duplicated(usedIDs)])
duplicates
```

```
## [1] "A1663" "E1281" "E0325" "A1102" "B1247" "B1731" "B1146"
```

```
reused <- sapply(duplicates, function(id) {
  recs <- pretestInfo[pretestInfo$ID==id,]
  #cat(id,":",paste(recs$Name,collape=", "),"\n")
  all(tolower(recs$Name[1])==tolower(na.omit(recs$Name[-1])))
})
```

```
names(reused) <- duplicates
reused
```

```
## A1663 E1281 E0325 A1102 B1247 B1731 B1146
##  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

Check to see if we can find a criteria for selecting which pretest to use if there are multiple pretests.

```
if (FALSE) { ## Do not print to hide student names)
pretestInfo[pretestInfo$ID %in% duplicates[reused],]  %>% arrange(ID)
pretestInfo[pretestInfo$ID %in% duplicates[!reused],] %>% arrange(ID)
}
```

Flag duplicates as missing pretest.

```
roster[match(duplicates[!reused],roster$StudyID),c("Pretest_P","Pretest_P0")]<- FALSE
roster[match(duplicates[reused],roster$StudyID),"Pretest_P0"]<-FALSE
roster[is.na(roster$Pretest_P0)&roster$Consent&roster$Assent,c("StudyID","Consent","Assent","Pretest_P"
```

```
## # A tibble: 15 x 5
##    StudyID Consent Assent Pretest_P Pretest_P0
##    <chr>     <dbl> <lgl>  <lgl>     <lgl>
##  1 D3137         1 TRUE   TRUE      NA
##  2 C3148         1 TRUE   TRUE      NA
##  3 C0976         1 TRUE   TRUE      NA
##  4 E0224         1 TRUE   TRUE      NA
##  5 F2923         1 TRUE   TRUE      NA
##  6 D1854         1 TRUE   TRUE      NA
##  7 <NA>          1 TRUE   NA        NA
##  8 D2147         1 TRUE   TRUE      NA
##  9 D3542         1 TRUE   TRUE      NA
## 10 D1911         1 TRUE   TRUE      NA
## 11 A0741         1 TRUE   NA        NA
## 12 B3856         1 TRUE   NA        NA
## 13 F3766         1 TRUE   TRUE      NA
## 14 B1258         1 TRUE   NA        NA
## 15 E1067         1 TRUE   TRUE      NA
```

```
ptmiss <- grep("^[A-F][0-9]+$",roster[is.na(roster$Pretest_P),]$StudyID,value=TRUE)
grep("^[A-F][0-9]+$",roster[is.na(roster$Pretest_P0),]$StudyID,value=TRUE)
```

```
##  [1] "D3137" "C3148" "C0976" "E0224" "D1359" "F2923" "D1854" "D2147" "D3542"
## [10] "D1911" "A0741" "B3856" "F3766" "B1258" "D2440" "F1595" "E1067"
```

```
roster[ptmiss,]
```

```
## # A tibble: 4 x 21
##   StudentName Nickname StudyID StudentEmail ParentName ParentEmail Assent
##   <chr>       <chr>    <chr>   <chr>        <chr>      <chr>       <lgl>
## 1 <NA>        <NA>     <NA>    <NA>         <NA>       <NA>        NA
## 2 <NA>        <NA>     <NA>    <NA>         <NA>       <NA>        NA
## 3 <NA>        <NA>     <NA>    <NA>         <NA>       <NA>        NA
## 4 <NA>        <NA>     <NA>    <NA>         <NA>       <NA>        NA
## # i 14 more variables: Pretest_P <lgl>, ECT_P <lgl>, POT_P <lgl>,
## #   Att_Mon_11_28 <dbl>, Att_Tue_11_29 <dbl>, Att_Wed_11_30 <dbl>,
## #   Att_Thu_12_1 <list>, Att_Fri_12_2 <dbl>, class <chr>, teacher <chr>,
## #   period <dbl>, Student <chr>, Consent <dbl>, Pretest_P0 <lgl>
```

Safe to mark all missing pretests as false.

```
roster$Pretest_P[is.na(roster$Pretest_P)] <- FALSE
roster$Pretest_P0[is.na(roster$Pretest_P0)] <- FALSE
salvagedPretests <- roster$StudyID[roster$Pretest_P&!roster$Pretest_P0]
length(salvagedPretests)
```

```
## [1] 20
```

```
#roster[match(salvagedPretests,roster$StudyID),]
```

New criteria added to the number of duplicates. Choose the duplicate with more non-missing responses.

```
dontkeep <- function (id) {
  candidates <- grep(id,pretestInfo$ID)
  recs <- pretestInfo[candidates,]
  if (reused[id]) {
    if (sum(recs$Finished) == 1) {
      candidates[!recs$Finished]
    } else {
      candidates[-which.max(recs$TotalReponses)]
    }
  } else {
    candidates
  }
}
deleteRows <- lapply(duplicates,dontkeep)
names(deleteRows) <- duplicates
deleteRows
```

```
## $A1663
## [1] 216
##
## $E1281
## [1] 309 328
##
## $E0325
## [1] 316 343
##
## $A1102
## [1] 341 353
##
## $B1247
## [1] 360 361
##
## $B1731
## [1] 277
##
## $B1146
## [1] 363 403
```

```
pretestClean <- cbind(pretest[-unlist(deleteRows),c(6,8,18,22:91)],
                      pretestInfo[-unlist(deleteRows),c(1,8:12)])
finished <- pretestClean$ID[pretestClean$Finished]
responded <- pretestClean$ID[pretestClean$ValidResponse]
setdiff(responded,finished)
```

```
## [1] "F3801" "F1528" "D1988" "D2350"
```

Check for Mid- and Post-tests

Look at the ECT and POT tests (depending on treatment group, students recieved one of the two on Wed and the other one on Friday).

```
pottest <- fetch_survey(qualtricsIDs["POT"],force=TRUE, label=FALSE, convert=FALSE) %>%
  mutate(Finished=as.logical(Finished))
```

```
##   |                                                                              |

##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   StartDate = col_datetime(format = ""),
##   EndDate = col_datetime(format = ""),
##   IPAddress = col_character(),
##   RecordedDate = col_datetime(format = ""),
##   ResponseId = col_character(),
##   RecipientLastName = col_logical(),
##   RecipientFirstName = col_logical(),
##   RecipientEmail = col_logical(),
##   ExternalReference = col_logical(),
##   DistributionChannel = col_character(),
##   UserLanguage = col_character(),
##   UserID1 = col_character(),
##   Q90 = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
pottest$NQcount <- apply(!is.na(select(pottest,starts_with("A-NQ"))),1,sum)
pottest$FQcount <- apply(!is.na(select(pottest,starts_with("A-FQ"))),1,sum)
pottest$IMIcount <- apply(!is.na(select(pottest,starts_with("IMI"))),1,sum)
pottest$PAcount <- apply(!is.na(select(pottest,starts_with("PA"))),1,sum)
pottest$FL5count <- apply(!is.na(select(pottest,starts_with("FL_5"))),1,sum)
pottest$SupCount <- apply(!is.na(select(pottest,starts_with("Support"))),1,sum)
ecttest <- fetch_survey(qualtricsIDs["ECT"],force=TRUE, label=FALSE, convert=FALSE) %>%
  mutate(Finished=as.logical(Finished))
```

```
##   |                                                                              |

##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   StartDate = col_datetime(format = ""),
##   EndDate = col_datetime(format = ""),
##   IPAddress = col_character(),
##   RecordedDate = col_datetime(format = ""),
##   ResponseId = col_character(),
##   RecipientLastName = col_logical(),
##   RecipientFirstName = col_logical(),
##   RecipientEmail = col_logical(),
##   ExternalReference = col_logical(),
##   DistributionChannel = col_character(),
##   UserLanguage = col_character(),
##   UserID1 = col_character(),
##   Q90 = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
ecttest$NQcount <- apply(!is.na(select(ecttest,starts_with("A-NQ"))),1,sum)
ecttest$FQcount <- apply(!is.na(select(ecttest,starts_with("A-FQ"))),1,sum)
ecttest$IMIcount <- apply(!is.na(select(ecttest,starts_with("IMI"))),1,sum)
ecttest$PAcount <- apply(!is.na(select(ecttest,starts_with("PA"))),1,sum)
ecttest$FL5count <- apply(!is.na(select(ecttest,starts_with("FL_5"))),1,sum)
ecttest$SupCount <- apply(!is.na(select(ecttest,starts_with("Support"))),1,sum)
```

```
pottest %>% select(Finished,contains("count",ignore.case=TRUE)) %>%
  summary()
```

```
##   Finished          NQcount         FQcount         IMIcount
##  Mode :logical   Min.   :0.000   Min.   :0.000   Min.   : 0.00
##  FALSE:16        1st Qu.:6.000   1st Qu.:6.000   1st Qu.:36.00
##  TRUE :342       Median :6.000   Median :6.000   Median :36.00
##                  Mean   :5.852   Mean   :5.849   Mean   :35.27
##                  3rd Qu.:6.000   3rd Qu.:6.000   3rd Qu.:36.00
##                  Max.   :6.000   Max.   :6.000   Max.   :36.00
##     PAcount         FL5count        SupCount
##  Min.   :0.000   Min.   : 0.00   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:12.00   1st Qu.:0.000
##  Median :7.000   Median :12.00   Median :8.000
##  Mean   :4.821   Mean   :11.83   Mean   :5.587
##  3rd Qu.:7.000   3rd Qu.:12.00   3rd Qu.:8.000
##  Max.   :7.000   Max.   :12.00   Max.   :8.000
```

```
ecttest %>% select(Finished,contains("count",ignore.case=TRUE)) %>%
  summary()
```

```
##   Finished          NQcount         FQcount         IMIcount
##  Mode :logical   Min.   : 0.00   Min.   :0.000   Min.   : 0.00
##  FALSE:20        1st Qu.:11.00   1st Qu.:8.000   1st Qu.:36.00
##  TRUE :335       Median :11.00   Median :8.000   Median :36.00
##                  Mean   :10.65   Mean   :7.727   Mean   :35.42
##                  3rd Qu.:11.00   3rd Qu.:8.000   3rd Qu.:36.00
##                  Max.   :11.00   Max.   :8.000   Max.   :36.00
##     PAcount         FL5count        SupCount
##  Min.   :0.000   Min.   : 0.00   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:19.00   1st Qu.:0.000
##  Median :7.000   Median :19.00   Median :8.000
##  Mean   :4.876   Mean   :18.79   Mean   :5.656
##  3rd Qu.:7.000   3rd Qu.:19.00   3rd Qu.:8.000
##  Max.   :7.000   Max.   :19.00   Max.   :8.000
```

Check for duplicates

```
pottest <-
  mutate(pottest,TotalResponses=NQcount+FQcount+IMIcount+PAcount+FL5count+SupCount) %>%
  mutate(ValidResponses=(NQcount >3) & (FQcount >4) & (FL5count >9))
potID0 <- pottest$UserID1[pottest$Finished]
potIDs <- pottest$UserID1[pottest$ValidResponses]
potduplicates <- unique(potIDs[duplicated(potIDs)])
```

```
print("POT duplicates")
```

```
## [1] "POT duplicates"
```

```
potduplicates
```

```
## [1] "D1359" "D1124" "A2079" "D3654" "F1494" "F1562"
```

```
ecttest <-
  mutate(ecttest,TotalResponses=NQcount+FQcount+IMIcount+PAcount+FL5count+SupCount) %>%
  mutate(ValidResponses=(NQcount >5) & (FQcount >4) & (FL5count >9))
ectID0 <- ecttest$UserID1[ecttest$Finished]
ectIDs <- ecttest$UserID1[ecttest$ValidResponses]
ectduplicates <- unique(ectIDs[duplicated(ectIDs)])
print("ECT duplicates")
```

```
## [1] "ECT duplicates"
```

```
ectduplicates
```

```
## [1] "D1359" "D0314"
```

```
pottest[pottest$UserID1 %in% potduplicates,
        c("UserID1","Finished","ValidResponses","NQcount","FQcount","IMIcount","PAcount",
          "FL5count","SupCount","TotalResponses")] %>% arrange(UserID1)
```

```
## # A tibble: 12 x 10
##    UserID1 Finished ValidResponses NQcount FQcount IMIcount PAcount FL5count
##    <chr>   <lgl>    <lgl>            <int>   <int>    <int>   <int>    <int>
##  1 A2079   TRUE     TRUE                 6       6       36       0       12
##  2 A2079   TRUE     TRUE                 6       6       36       7       12
##  3 D1124   TRUE     TRUE                 6       6       36       0       12
##  4 D1124   TRUE     TRUE                 6       6       33       0       12
##  5 D1359   TRUE     TRUE                 6       6       36       7       12
##  6 D1359   TRUE     TRUE                 6       6       36       7       12
##  7 D3654   TRUE     TRUE                 6       6       36       7       12
##  8 D3654   TRUE     TRUE                 6       6       36       0       12
##  9 F1494   TRUE     TRUE                 6       6       36       7       12
## 10 F1494   TRUE     TRUE                 6       6       36       0       12
## 11 F1562   TRUE     TRUE                 6       6       36       0       12
## 12 F1562   TRUE     TRUE                 6       6       36       0       12
## # i 2 more variables: SupCount <int>, TotalResponses <int>
```

```
ecttest[ecttest$UserID1 %in% ectduplicates,
        c("UserID1","Finished","ValidResponses","NQcount","FQcount","IMIcount","PAcount",
          "FL5count","SupCount","TotalResponses")] %>% arrange(UserID1)
```

```
## # A tibble: 4 x 10
##   UserID1 Finished ValidResponses NQcount FQcount IMIcount PAcount FL5count
##   <chr>   <lgl>    <lgl>            <int>   <int>    <int>   <int>    <int>
## 1 D0314   TRUE     TRUE                11       8       35       7       19
## 2 D0314   FALSE    TRUE                11       8       35       0       19
## 3 D1359   TRUE     TRUE                11       8       36       7       19
## 4 D1359   TRUE     TRUE                11       8       36       7       19
## # i 2 more variables: SupCount <int>, TotalResponses <int>
```

```
potdup.ect <- sapply(potduplicates,function (id) any(grepl(id,ecttest$UserID1)))
ectdup.pot <- sapply(ectduplicates,function(id) any(grepl(id,pottest$UserID1)))
```

```
potdup.ect
```

```
## D1359 D1124 A2079 D3654 F1494 F1562
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
ectdup.pot
```

```
## D1359 D0314
##  TRUE  TRUE
```

Looks like nobody took the wrong test.

Pick the record with the most responses.

```
dontkeep1 <- function (id,test) {
  candidates <- grep(id,test$UserID1)
  recs <- test[candidates,]
  candidates[-which.max(recs$TotalResponses)]
}
deletePotRows <- lapply(potduplicates,function (id) dontkeep1(id,pottest))
names(deletePotRows) <- potduplicates
deletePotRows
```

```
## $D1359
## [1] 87
##
## $D1124
## [1] 177
##
## $A2079
## [1] 198
##
## $D3654
## [1] 231
##
## $F1494
## [1] 273
##
## $F1562
## [1] 287
```

```
deleteEctRows <- lapply(ectduplicates,function (id) dontkeep1(id,ecttest))
names(deleteEctRows) <- ectduplicates
deleteEctRows
```

```
## $D1359
## [1] 258
##
## $D0314
## [1] 354
```

```
ecttest <- ecttest[-unlist(deleteEctRows),]
pottest <- pottest[-unlist(deletePotRows),]
```

Now transfer information about completed forms to the roster.

```
studyIDs <- roster$StudyID
roster$POT_P <- FALSE
roster$POT_PO <- FALSE
```

```
roster$ECT_P <- FALSE
roster$ECT_P0 <- FALSE
roster[na.omit(match(unique(potIDs),studyIDs)),"POT_P"] <- TRUE
roster[na.omit(match(unique(potID0),studyIDs)),"POT_P0"] <- TRUE
roster[match(potduplicates,studyIDs),"POT_P0"] <- FALSE
roster[na.omit(match(unique(ectIDs),studyIDs)),"ECT_P"] <- TRUE
roster[na.omit(match(unique(ectID0),studyIDs)),"ECT_P0"] <- TRUE
roster[match(ectduplicates,studyIDs),"ECT_P0"] <- FALSE
```

Again check for the number of additional student included using the new rules.

```
salvagedEcttests <- roster$StudyID[roster$ECT_P&!roster$ECT_P0]
#roster[match(salvagedPretests,roster$StudyID),]
length(salvagedEcttests)
```

```
## [1] 7
```

```
salvagedPottests <- roster$StudyID[roster$POT_P&!roster$POT_P0]
#roster[match(salvagedPretests,roster$StudyID),]
length(salvagedPottests)
```

```
## [1] 12
```

Write back out for more manual checking:

```
mutate(roster,valid=Consent&Assent&Pretest_P&ECT_P&POT_P) %>%
  mutate(valid0=Consent&Assent&Pretest_P0&ECT_P0&POT_P0) ->
  roster
#  mutate(valid=if_else(is.na(valid),FALSE,valid)) -> roster
salvaged <- roster$valid&!roster$valid0
summary(salvaged)
```

```
##    Mode   FALSE    TRUE
## logical     359      26
```

Total of 26 student results salvaged by the more lenient inclusion rules.

```
#roster[salvaged,]
```

Write back out the roster information with PID for use in study honorarium process.

```
roster %>% sheet_write(rostersheet,sheet="Checked")
```

```
## v Writing to "Physics Playground rosters".

## v Writing to sheet 'Checked'.
```

```
# multiMatchBells %>% sheet_write(rostersheet,sheet="Multimatched Students")
#data.frame(student=nomatchConsent) %>% #sheet_write(rostersheet,sheet="Unmatched Consents")
```

```
write(roster$StudyID[salvaged],"salvaged.txt")
write(roster$StudyID[roster$valid],"validIDs.txt")
```

Write out data with invalid ID and Identifying information removed.

```
goutids <- list(roster="1DyFdhhBE8UvhyLtWbifnZphBUZV3QzoC8yjQ7oHkaZg",
                validIDs="1lBp0LhNdSf0r1OtwC2cyqInB2npLbjHOKUMh4z_nzfM",
                pretest="1wnQyfXmSABXsUmfKPh-nLPEfZtiBvcdYqLq-Tcc24vU",
                ect="1KMrpnK60720779GSKWzpjIdw1qaKy3eCbSIXy6fkTzQ",
                pot="10aTP3zVPJ7p3oYzXitqBBDth4Q3zIrA_eQVc0KS39jc",
                allIDs="1BPkBMEE5DOtZ3MBt5C1ab4djddAOhUyAIPMiR4WRhBQ")
gIESdrivePath <- "https://drive.google.com/drive/folders/1P1nQ5dHT8Wp2K5gvORyR2RG2Y-i4-JFb/"
#gIESdrive <- googledrive::as_dribble(gIESdrivePath)
```

The game used a different ID system, with a different ID for the (MTW) and second (TF) halves of the study. Merge these IDs with the valid IDs to filter the game log data for included students.

```
AllIDs <- read_sheet("1BPkBMEE5DOtZ3MBt5C1ab4djddAOhUyAIPMiR4WRhBQ")
```

```
## v Reading from "UserIDs2".

## v Range 'Sheet1'.
```

Join PP Id information with Roster fields to make ValidID data.

```
vtable <- select(roster,StudyID,valid) %>% filter(!is.na(StudyID))
vids <- pull(vtable,valid)
names(vids) <- pull(vtable,StudyID)
head(vids)
```

```
## A3430 C3104 C3306 D3418 D3137 C3115
##  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

```
validIDs <- select(roster,StudyID, class, teacher, period, valid, valid0) %>%
  filter(!is.na(StudyID)) %>% filter(valid) %>%
  left_join(AllIDs,by=join_by(StudyID))
vidList = validIDs$StudyID
validIDs
```

```
## # A tibble: 244 x 12
##    StudyID class teacher period valid valid0 ID1    ID2    Midtest Posttest
##    <chr>   <chr> <chr>    <dbl> <lgl> <lgl>  <chr>  <chr>  <chr>   <chr>
##  1 A3430   a1    a            1 TRUE  TRUE   066ad1 066ad2 ECT     POT
##  2 C3104   a1    a            1 TRUE  TRUE   176cd1 176cd2 ECT     POT
##  3 D3418   a1    a            1 TRUE  TRUE   217dd1 217dd2 POT     ECT
##  4 D3137   a1    a            1 TRUE  FALSE  248dd1 248dd2 POT     ECT
##  5 C3115   a1    a            1 TRUE  TRUE   180cd1 180cd2 ECT     POT
##  6 A3294   a1    a            1 TRUE  TRUE   002ad1 002ad2 ECT     POT
##  7 F3126   a1    a            1 TRUE  TRUE   360fd1 360fd2 ECT     POT
##  8 C3429   a1    a            1 TRUE  TRUE   181cd1 181cd2 ECT     POT
##  9 B3283   a1    a            1 TRUE  TRUE   099bd1 099bd2 POT     ECT
## 10 A3351   a1    a            1 TRUE  TRUE   042ad1 042ad2 ECT     POT
## # i 234 more rows
## # i 2 more variables: Condition <chr>, Number <dbl>
```

Write this out for log file analysis.

```
validIDs %>% sheet_write(gs4_get(goutids$validIDs),sheet="ValidIDs-Deidentified")
```

```
## v Writing to "ValidIDs-Deidentified".

## v Writing to sheet 'ValidIDs-Deidentified'.
```

Clean PII out of pretest data.

```r
pretestClean %>% filter(!is.na(ID)) %>% filter(vids[ID]) %>%
  mutate(StudyID=ID) %>%
  sheet_write(gs4_get(goutids$pretest),"Valid-Only")
```

```
## v Writing to "PretestData".

## v Writing to sheet 'Valid-Only'.
```

```r
junkCols <- c("StartDate","EndDate", "Status", "IPAddress", "Progress",
              "ResponseId", "RecipientLastName", "RecipientFirstName",
              "RecipientEmail", "ExternalReference", "LocationLatitude",
              "LocationLongitude", "DistributionChannel", "UserLanguage")
ecttest %>% select(!any_of(junkCols)) %>% mutate(StudyID=UserID1) %>%
   filter(!is.na(StudyID)) %>% filter(vids[StudyID]) %>%
  sheet_write(gs4_get(goutids$ect),"Valid-Only")
```

```
## v Writing to "ECTTest".

## v Writing to sheet 'Valid-Only'.
```

```r
pottest %>% select(!any_of(junkCols)) %>% mutate(StudyID=UserID1) %>%
   filter(!is.na(StudyID)) %>% filter(vids[StudyID]) %>%
  sheet_write(gs4_get(goutids$pot),"Valid-Only")
```

```
## v Writing to "POTtest".

## v Writing to sheet 'Valid-Only'.
```