Counts Check

Chang Y. Chung

December 30, 2014

Contents

Introduction

This document implements some checks regarding win, loss, and skip counts in the data as attached in Matt's email I have received on Dec. 30, 2014.

Read and verify raw data

First, uncompress .zip file in order to get the data files out:

```
unzip -u -qq Received/20141230/data_precleaned.zip
```

Unzipping it creates the data_precleaned directory where the uncompressed data, .csv files reside. Let's read them into data frames.

```
rm(list=ls())
setwd("~/Professional/Matt/")
library(sqldf)

## Loading required package:  gsubfn
## Loading required package:  proto
## Loading required package:  RSQLite
## Loading required package:  DBI
## Loading required package:  methods

options(width=90)

readcsv <- function(file) read.csv(file=file, header=T, sep=",", stringsAsFactors=F,
    na.strings="NA")
fs <- (function(dir) paste(dir, list.files(dir), sep="/"))("data_precleaned")
csvs <- fs[grepl("_cleaned", fs)]
```

```
fn <- function(s) {w <- strsplit(s, "_")[[1]]; paste0(w[4], w[3], substr(w[8],1,1))}
ge <- globalenv()
dimension <- function(s) {d <- dim(eval(parse(text=s), envir=ge));
    sprintf('%s (%d rows, %d columns)', s, d[1], d[2])}
df <- function(csv) {n <- fn(csv); assign(n, readcsv(csv), envir=ge); dimension(n)}
Map(df, csvs)
```

```
## $`data_precleaned/wikisurvey_504_ideas_2014-05-19T19_04_52Z_public_cleaned.csv`
## [1] "ideas504p (594 rows, 15 columns)"
##
## $`data_precleaned/wikisurvey_504_ideas_2014-05-19T19_04_52Z_restricted_cleaned.csv`
## [1] "ideas504r (594 rows, 15 columns)"
##
## $`data_precleaned/wikisurvey_504_nonvotes_2014-05-19T19_04_58Z_public_cleaned.csv`
## [1] "nonvotes504p (8420 rows, 23 columns)"
##
## $`data_precleaned/wikisurvey_504_nonvotes_2014-05-19T19_04_58Z_restricted_cleaned.csv`
## [1] "nonvotes504r (8420 rows, 23 columns)"
##
## $`data_precleaned/wikisurvey_504_votes_2014-05-19T19_04_55Z_public_cleaned.csv`
## [1] "votes504p (26724 rows, 23 columns)"
##
## $`data_precleaned/wikisurvey_504_votes_2014-05-19T19_04_55Z_restricted_cleaned.csv`
## [1] "votes504r (26724 rows, 23 columns)"
##
## $`data_precleaned/wikisurvey_608_ideas_2014-05-19T13_56_59Z_public_cleaned.csv`
## [1] "ideas608p (489 rows, 15 columns)"
##
## $`data_precleaned/wikisurvey_608_ideas_2014-05-19T13_56_59Z_restricted_cleaned.csv`
## [1] "ideas608r (489 rows, 15 columns)"
##
## $`data_precleaned/wikisurvey_608_nonvotes_2014-05-19T13_57_09Z_public_cleaned.csv`
## [1] "nonvotes608p (14657 rows, 23 columns)"
##
## $`data_precleaned/wikisurvey_608_nonvotes_2014-05-19T13_57_09Z_restricted_cleaned.csv`
## [1] "nonvotes608r (14657 rows, 23 columns)"
##
## $`data_precleaned/wikisurvey_608_votes_2014-05-19T13_57_06Z_public_cleaned.csv`
## [1] "votes608p (28715 rows, 23 columns)"
##
## $`data_precleaned/wikisurvey_608_votes_2014-05-19T13_57_06Z_restricted_cleaned.csv`
## [1] "votes608r (28715 rows, 23 columns)"
```

Notice that the number of rows has been changed for some data, assuming that "public" and "restricted" were "cleaned" and "raw" before, respectively.

- nonvotes 504 restricted has 8420 rows vs 8786, before
- votes 504 restricted has 26724 rows vs 28471, before
- nonvotes 608 restricted has 14657 rows vs 14454, before

Wins and Losses

For this, we are re-using the check1 function from reviewRaw2.pdf with a minor modification that allows public/restricted types.

```
library(sqldf)
str2df <- function(str) eval(as.name(str))
check1 <- function(svyid, type) {
    votes <- str2df(paste0('votes', svyid, type))
    validVotes <- votes[votes$Valid, ]
    wins <- sqldf('select [Winner.ID], count(*) as wins from validVotes group by 1')
    losses <- sqldf('select [Loser.ID], count(*) as losses from validVotes group by 1')
    ideas <- str2df(paste0('ideas', svyid, type))
    standing <- sqldf('select i.[Wikisurvey.ID], i.[Idea.ID],
        i.Wins as ideasWins, i.Losses as ideasLosses,
        case when w.wins is null then 0 else w.wins end as votesWins,
        case when l.losses is null then 0 else l.losses end as votesLosses
        from ideas i
            left outer join wins w on i.[Idea.ID] = w.[Winner.ID]
            left outer join losses l on i.[Idea.ID] = l.[Loser.ID]')
    nomatch <- sqldf('select * from standing
        where ideasWins != votesWins or ideasLosses != votesLosses')
    nomatch
}
check1(504, 'p')

## Loading required package:  tcltk

## [1] Wikisurvey.ID Idea.ID       ideasWins     ideasLosses   votesWins     votesLosses
## <0 rows> (or 0-length row.names)

check1(504, 'r')

## [1] Wikisurvey.ID Idea.ID       ideasWins     ideasLosses   votesWins     votesLosses
## <0 rows> (or 0-length row.names)

check1(608, 'p')

## [1] Wikisurvey.ID Idea.ID       ideasWins     ideasLosses   votesWins     votesLosses
## <0 rows> (or 0-length row.names)

check1(608, 'r')

## [1] Wikisurvey.ID Idea.ID       ideasWins     ideasLosses   votesWins     votesLosses
## <0 rows> (or 0-length row.names)
```

Skips

For checking the skip counts, we re-use the check2 function with a slight change in order to accomodate two different types of data: public and restricted.

```
library(sqldf)
check2 <- function(svyid, type) {
    nonvotes <- str2df(paste0('nonvotes', svyid, type))
```

```
    noNA <- sqldf('select * from nonvotes
        where [Left.Choice.ID] is not null and [Record.Type] = \'Skip\'')
    left <- sqldf('select [Left.Choice.ID], count(*) as nSkips from noNA group by 1')
    right <- sqldf('select [Right.Choice.ID], count(*) as nSkips from noNA group by 1')
    ideas <- str2df(paste0('ideas', svyid, type))
    names(ideas)[names(ideas) == 'Times.involved.in.Cant.Decide'] <- 'Skips'
    skips <- sqldf('select i.[Wikisurvey.ID], i.[Idea.ID], i.Skips as ideaSkips,
        case when l.nSkips is null then 0 else l.nSkips end as nonvoteLeftSkips,
        case when r.nSkips is null then 0 else r.nSkips end as nonvoteRightSkips
        from ideas as i
            left outer join left as l on i.[Idea.ID] = l.[Left.Choice.ID]
            left outer join right as r on i.[Idea.ID] = r.[Right.Choice.ID]')
    nomatch <- sqldf('select * from skips
        where ideaSkips != (nonvoteLeftSkips + nonvoteRightSkips)')
    nomatch
}
check2(504, 'p')

## [1] Wikisurvey.ID     Idea.ID           ideaSkips         nonvoteLeftSkips
## [5] nonvoteRightSkips
## <0 rows> (or 0-length row.names)

check2(504, 'r')

## [1] Wikisurvey.ID     Idea.ID           ideaSkips         nonvoteLeftSkips
## [5] nonvoteRightSkips
## <0 rows> (or 0-length row.names)

check2(608, 'p')

## [1] Wikisurvey.ID     Idea.ID           ideaSkips         nonvoteLeftSkips
## [5] nonvoteRightSkips
## <0 rows> (or 0-length row.names)

check2(608, 'r')

## [1] Wikisurvey.ID     Idea.ID           ideaSkips         nonvoteLeftSkips
## [5] nonvoteRightSkips
## <0 rows> (or 0-length row.names)
```

Done

This conclude checking counts for Matt's new data.