Measure_discrepancy_check_withgoldenstandard

Denise Abcede

2021

At the organization I worked for, data validating of a specific set of assessments [n=20] were performed by manually checking each patient and time point [n=6 time points]. Through R, I created this measure discrepancy check method to automate most discrepancy-detecting. This method reduced data validating time by 70% and reduced human error significantly. For this method to work, we had to have a golden standard dataset with correct raters and time points for me to detect if a measure was missing, if there was an incorrect rater added or if time points by visit did not match.

Before merging the golden standard file with the measure you'd like to clean, I ensured the merging variables match in both datasets. It was also good practice to delete variables that are unnecessary for the cleaning and mirror the datasets so the order of the merging variables are in the same order.

```
#view var names
variable.names(gs)
```

```
## [1] "PATIENTI"        "RATER"           "GSVISIT"         "DATE"
## [5] "GSVISITBEGINDAT" "GSVISITENDDATE"
```

```
variable.names(m1)
```

```
## [1] "PATIENTI"         "RATER"            "MEASURE1VISIT"    "DATE"
## [5] "MEASURE1BEGINDAT" "MEASURE1ENDDATE"
```

```
#drop any unwanted variables
gs <- gs[!is.na(gs$PATIENTI), ]
m1 <- m1[!is.na(m1$PATIENTI), ]

#drop unnecessary variables
gs <- gs[,c(1:6)]

# reorder variables so both datasets have similar structure
# ex.m1<-m1[,c(1,20,4,10:12,14:19)]
# ex.gs-gfss[,c(1,20,4,10:12,14:19)]

#change "m1VISIT" and "gsvisit" variables to "VISIT" for standardization of merging variables across me
names(m1)[c(3,5,6)] <- c("VISIT","BEGINDAT","ENDDATE")
names(gs)[c(3,5,6)] <- c("VISIT","BEGINDAT","ENDDATE")


#variables that will not be merged should include their measure name
names(m1)[4] <- "M1.DATE"
names(gs)[4] <- "GS.DATE"

names(m1)
```

```
## [1] "PATIENTI" "RATER"    "VISIT"    "M1.DATE"  "BEGINDAT" "ENDDATE"
```

```
names(gs)
```

```
## [1] "PATIENTI" "RATER"    "VISIT"    "GS.DATE"  "BEGINDAT" "ENDDATE"
```

When determining the best type of merge to capture discrepancies, I decided it was best to do a full merge on both sides. While I want to be sure all patients are captured and the golden standard Should indicate that, if a row in the measure side is not merged with the golden standard measure then it will not appear. Therefore, we will not know if there is ane error that this code did not address.

```
merge1<-merge(gs,m1,by=c("PATIENTI","RATER","VISIT","BEGINDAT","ENDDATE"), all.x=TRUE, all.y=TRUE)
```

I created tables and new columns to easily indicate what specific patient data to check based on common discrepancies.

```
attach(merge1)
is.factor(PATIENTI)
```

```
## [1] FALSE
```

```
VISIT <- factor(VISIT, levels=c("33","0","3","6","9","12"))
merge1 <- merge1[order(PATIENTI,VISIT), ]
(dis_rater <- table(PATIENTI,RATER)) #view patient IDs where 2 raters are used for 1 patient = discrepa
```

```
##          RATER
## PATIENTI  69 123 128 236 241 690
##   3012001  0   3   0   0   0   0
##   3012002  0   7   1   0   0   0
##   3012003  0   1   0   0   0   0
##   3012004  3   0   0   0   0   0
##   3012005  3   1   0   0   0   0
##   3012006  6   0   0   1   0   0
##   3012007  6   0   0   0   0   0
##   3012008  0   7   0   0   0   0
##   3012009  1   3   0   0   0   0
##   3012010  5   0   0   0   0   0
##   3012011  0   4   0   0   1   0
##   3012012  0   1   0   0   0   0
##   3012013  0   4   0   0   0   0
##   3012014  2   0   0   0   0   0
##   3012015  1   0   0   0   0   0
##   3012016  3   0   0   0   0   1
##   3012017  0   2   0   0   0   0
##   3012018  0   1   0   0   0   0
##   3012019  1   0   0   0   0   0
##   3012020  0   1   0   0   0   0
##   3012021  0   3   0   0   0   0
##   3012022  0   1   0   0   0   0
##   3012023  1   0   0   0   0   0
##   3012024  5   1   0   0   0   0
```

```
(dis_visit <- table(PATIENTI, VISIT)) #view patient Ids where any timepoint >1 as there should only be
```

```
##          VISIT
## PATIENTI  33 0 3 6 9 12
##   3012001  1 1 1 0 0  0
##   3012002  1 1 2 1 1  1
##   3012003  1 0 0 0 0  0
```

```
##   3012004  1 1 1 0 0  0
##   3012005  1 1 2 0 0  0
##   3012006  1 1 1 2 1  1
##   3012007  1 1 1 1 1  1
##   3012008  1 1 1 1 1  2
##   3012009  2 1 1 0 0  0
##   3012010  0 2 1 2 1  0
##   3012011  1 1 1 2 0  0
##   3012012  0 0 0 0 1  0
##   3012013  2 1 1 0 0  0
##   3012014  1 1 0 0 0  0
##   3012015  1 0 0 0 0  0
##   3012016  1 2 1 0 0  0
##   3012017  1 1 0 0 0  0
##   3012018  1 0 0 0 0  0
##   3012019  1 0 0 0 0  0
##   3012020  1 0 0 0 0  0
##   3012021  1 2 0 0 0  0
##   3012022  0 0 1 0 0  0
##   3012023  1 0 0 0 0  0
##   3012024  1 1 2 2 0  0
```

```r
#mark missing values
merge1$missing_m1 <- ifelse(is.na(M1.DATE),1,0) #create new binary variable to mark for missing values
merge1$missing_gs <- ifelse(is.na(GS.DATE),1,0)
merge1$missing_visit <- ifelse(is.na(VISIT),1,0)
merge1$missing_rater <- ifelse(is.na(RATER),1,0)

#export R into csv files for further review and data cleaning
write.csv(merge1, "m1_discrep_check_finalproduct.csv")
write.csv(dis_rater, "discrepancy_rater_finalproduct.csv")
write.csv(dis_visit, "discrepancy_visit_finalproduct.csv")
```

With the output, the team is able identify missing data and incorrectly added data.