QBS181 Midterm

Mark Taylor

Setup

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(odbc)
library(sqldf)
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite
```

```
library(stringr)
library(SASxport)
```

1

```
DIQ <- read.xport("DIQ_I.XPT")

library(DBI)

#dbWriteTable(con, SQL("mtaylor.DIQ"), DIQ)
```

a) List the data-related issues you see in this data set

1. Many columns are coded, when really they should be using a descriptor value instead
2. Some columns containing meaningful numerical values also have codes intermixed into them, which could lead to confusion when analyzing if codes were to be interpreted as numerical values
3. Some columns that are coded would be much better suited to contain boolean values instead. Boolean values are easier to work with and more readable, and in these instances edge cases that cannot be described with a Boolean are neglible or can be expressed as NULL
4. Some columns contain their units in another column, most notably lengths of time that need corresponding units to make any sense. In one column there might be the value 1, but you need the corresponding column to know if that means 1 second, 1 minute, 1 hour, etc.
5. The column names are all encoded and not possible for a human to intuitively know what information they contain
6. There are many rows which are missing a lot of information, seeming to be filled mostly with NULL values

b) How will you address each data-related issue?

1. Because we know the descriptor associated with each code, in instances where we should replace a code with a descriptor we check each code and replace with the descriptor it should contain. For the most part, these are categories that should be readily understood and have semantic meaning
2. In an instance where codes occur in a column that should just contain values, we can check for the code and replace it with a meaningful value (or a NULL in many cases, as these often represent a lack of information). These codes are never in the range of possible values, so they can either be set to meaningful values (666 representing an infant being replaced with zero) or NULL (such as instances where a response is unknown or unanswered)
3. These instances are easy to interpret, usually we just need to simply replace 1s with TRUE and 2s with FALSE. We treat all other possibilities as NULLs in these instances, which will be justified in the next part.
4. For these instances, we need to first parse out the unit from the table containing the unit (as it typically is encoded). Then we can look through the table with the actual values and apply the appropriate units
5. I simply made new columns for everything as I went along, using names containing semantic information
6. For the most part I avoided imputing data for the sake of filling vacant values. In some instances there is a meaningful alternative to replace NULL with, such as FALSE, but for the most part it wasn't possible to come up with meaning values without risking confounding the data

1. This approach seemed best for a few reasons. For one, a human should be able to look at a table and understand it. Without descriptors, that is impossible. These instances usually contain categorical data, so it makes sense to separate rows into discrete groups and provide a description for them

2. Removing these codes from otherwise coherent information was an incredibly important step. While in all cases the codes were outside the range of possible values if someone working with the data were to try and take a mean for example, codes throughout the data could drastically throw it off. There were only a few situations where codes could be assigned comparable values, but for the most part we discarded these values to maintain the integrity of the data and its usefulness for analysis

3. Converting to a boolean makes the column much more intuitive to read and much easier to work with. In each column where this is the case, the column is asking a yes or no question. Yes corresponds to TRUE and No corresponds to FALSE. The notable edge cases here is for when subjects refuse to answer or do not know an answer. While in some contexts this might be a response worth analyzing, these two sub categories generally made up a very small amount of the data. As the goal was essentially to make columns as coherent as possible, if they are asking a yes or no question and the participant refuses to answer or doesn't know the answer, that would be considered a NULL value; trying to estimate what they might have said could lead to confounding.

4. This approach was essentially necessary to make certain columns meaningful. For example, time values were stored in one column and units were stored in the other, so it was necessary to unpack the units and apply them to the time values, otherwise they wouldn't make any sense in relation to each other.

5. The justification here is pretty straight-forward. Tables should be as human readable as they can if its a possibility. The variables names they had before had no obvious interpretation, so replacing them with human friendly names made the data easier to work with.

6. While I considered performing mean imputation in a lot of scenarios, for the most part I avoided generating any information that wasn't present in the data set. In many instances, there was a significant portion of the data that was missing. While generating similar information is tantalizing, in most cases there was so much missing data that confounding would have been extremely likely. I figured someone analyzing the data set would be much better off using the non-null values to avoid confounding.

2

Code to descriptor

```
SELECT DIQ010, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ010
ORDER BY DIQ010
```

```
# Convert coded value to descriptor and use a more meaningful column name
DIQ$DiabetesDiagnosis <-
  ifelse(DIQ$DIQ010 == 1, "Yes",
  ifelse(DIQ$DIQ010 == 2, "No",
  ifelse(DIQ$DIQ010 == 3, "Borderline",
  ifelse(DIQ$DIQ010 == 4, "Don't know", NA
  ))))

sample(DIQ$DiabetesDiagnosis, 10)
```

```
SELECT DIQ230, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ230
ORDER BY DIQ230
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$LengthSinceSeenSpecialist <-
  ifelse(DIQ$DIQ230 == 1, "1 year ago or less",
  ifelse(DIQ$DIQ230 == 2, "More than 1 year ago but no more than 2 years ago",
  ifelse(DIQ$DIQ230 == 3, "More than 2 years ago but no more than 5 years ago",
  ifelse(DIQ$DIQ230 == 4, "More than 5 years ago",
  ifelse(DIQ$DIQ230 == 5, "Never", NA
  )))))

sample(DIQ$LengthSinceSeenSpecialist, 10)
```

```sql
SELECT DIQ230, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ230
ORDER BY DIQ230
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$LengthSinceSpecialistSeen <-
  ifelse(DIQ$DIQ230 == 1, "1 year ago or less",
  ifelse(DIQ$DIQ230 == 2, "More than 1 year ago but no more than 2 years ago",
  ifelse(DIQ$DIQ230 == 3, "More than 2 years ago but no more than 5 years ago",
  ifelse(DIQ$DIQ230 == 4, "More than 5 years ago",
  ifelse(DIQ$DIQ230 == 5, "Never", NA
  )))))

sample(DIQ$LengthSinceSpecialistSeen, 10)
```

```sql
SELECT DIQ291, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ291
ORDER BY DIQ291
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$DoctorA1CGoal <-
  ifelse(DIQ$DIQ291 == 1, "Less than 6",
  ifelse(DIQ$DIQ291 == 2, "Less than 7",
  ifelse(DIQ$DIQ291 == 3, "Less than 8",
  ifelse(DIQ$DIQ291 == 4, "Less than 9",
  ifelse(DIQ$DIQ291 == 5, "Less than 10",
  ifelse(DIQ$DIQ291 == 6, "Provider did not specify goal",
  ifelse(DIQ$DIQ291 == 77, "Refused",
  ifelse(DIQ$DIQ291 == 99, "Don't know", NA
  ))))))))

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DoctorA1CGoal[!is.na(DIQ$DoctorA1CGoal)], 10)
```

```sql
SELECT DIQ360, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ360
ORDER BY DIQ360
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$LastPupilDilationExam <-
  ifelse(DIQ$DIQ291 == 1, "Less than 1 month",
  ifelse(DIQ$DIQ291 == 2, "1-12 months",
  ifelse(DIQ$DIQ291 == 3, "13-24 months",
  ifelse(DIQ$DIQ291 == 4, "Greater than 2 years",
  ifelse(DIQ$DIQ291 == 5, "Never",
  ifelse(DIQ$DIQ291 == 7, "Refused",
  ifelse(DIQ$DIQ291 == 9, "Don't know", NA
  )))))))

# Most samples are NA, so we exclude them when sampling
sample(DIQ$LastPupilDilationExam[!is.na(DIQ$LastPupilDilationExam)], 10)
sample(DIQ$LastPupilDilationExam, 10)
```

Code mixed in with values

```r
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID040 >= 2 AND DID040 <= 78 THEN '2-78'
               ELSE DID040
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```r
# Values not related to age should be replaced
DIQ$DiagnosisAge <-
  ifelse(DIQ$DID040 == 666, 0,
  ifelse(DIQ$DID040 == 999, NA,
  ifelse((DIQ$DID040 >= 2 & DIQ$DID040 <= 78), DIQ$DID040, NA)
  ))

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DiagnosisAge[!is.na(DIQ$DiagnosisAge)], 10)
```

```r
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID250 >= 1 AND DID250 <= 60 THEN '1-60'
               WHEN DID250 like '%9999%' THEN NULL
               ELSE DID250
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```r
# Values not related to age should be replaced
DIQ$DoctorVisitsLastYear <-
  ifelse(DIQ$DID250 == '9999', NA, DIQ$DID250)

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DoctorVisitsLastYear[!is.na(DIQ$DoctorVisitsLastYear)], 10)
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN CAST(DIQ280 AS DECIMAL(22,8)) >= 2.0 AND CAST(DIQ280 AS DECIMAL(22,8)) <= 18.5 THEN
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```
# Values not related to age should be replaced
DIQ$LastA1CLevel <-
  ifelse(DIQ$DIQ280 == '999' | DIQ$DIQ280 == '777', NA, DIQ$DIQ280)

# Most samples are NA, so we exclude them when sampling
sample(DIQ$LastA1CLevel[!is.na(DIQ$LastA1CLevel)], 10)
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DIQ300S >= 80 AND DIQ300S <= 201 THEN '80-201'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```
# Values not related to age should be replaced
DIQ$MostRecentSBP <-
  ifelse((DIQ$DIQ300S >= 80 & DIQ$DIQ300S <= 201), DIQ$DIQ300S, NA
  )

# Most samples are NA, so we exclude them when sampling
sample(DIQ$MostRecentSBP[!is.na(DIQ$MostRecentSBP)], 10)
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DIQ300D >= 17 AND DIQ300D <= 251 THEN '17-251'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```
# Values not related to age should be replaced
DIQ$MostRecentDBP <-
  ifelse((DIQ$DIQ300D >= 80 & DIQ$DIQ300D <= 201), DIQ$DIQ300D, NA
  )

# Most samples are NA, so we exclude them when sampling
sample(DIQ$MostRecentDBP[!is.na(DIQ$MostRecentDBP)], 10)
```

```sql
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID310S >= 80 AND DID310S <= 175 THEN '80-175'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```r
# Values not related to age should be replaced
DIQ$DoctorRecommendedSBP <-
  ifelse((DIQ$DID310S >= 80 & DIQ$DID310S <= 175), DIQ$DIQ300S, NA
  )

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DoctorRecommendedSBP[!is.na(DIQ$DoctorRecommendedSBP)], 10)
```

```sql
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID310D >= 18 AND DID310D <= 140 THEN '18-140'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```r
# Values not related to age should be replaced
DIQ$DoctorRecommendedDBP <-
  ifelse((DIQ$DID310D >= 80 & DIQ$DID310D <= 201), DIQ$DID310D, NA
  )

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DoctorRecommendedDBP[!is.na(DIQ$DoctorRecommendedDBP)], 10)
```

Code should be replaced with booleans

```sql
SELECT DIQ160, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ160
ORDER BY DIQ160
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$EverDiagnosedWithPrediabetes <-
  ifelse(DIQ$DIQ160 == 1, TRUE,
  ifelse(DIQ$DIQ160 == 2, FALSE, NA
  ))

sample(DIQ$EverDiagnosedWithPrediabetes[!is.na(DIQ$EverDiagnosedWithPrediabetes)], 10)
```

```sql
SELECT DIQ160, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ160
ORDER BY DIQ160
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$EverDiagnosedWithPrediabetes <-
  ifelse(DIQ$DIQ160 == 1, TRUE,
  ifelse(DIQ$DIQ160 == 2, FALSE, NA
  ))

sample(DIQ$EverDiagnosedWithPrediabetes[!is.na(DIQ$EverDiagnosedWithPrediabetes)], 10)
```

```sql
SELECT DIQ170, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ170
ORDER BY DIQ170
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$EverToldDiabetesHealthRisk <-
  ifelse(DIQ$DIQ170 == 1, TRUE,
  ifelse(DIQ$DIQ170 == 2, FALSE, NA
  ))

sample(DIQ$EverDiagnosedWithPrediabetes[!is.na(DIQ$EverDiagnosedWithPrediabetes)], 10)
```

```sql
SELECT DIQ172, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ172
ORDER BY DIQ172
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$FeelCouldHaveDiabetesRisk <-
  ifelse(DIQ$DIQ172 == 1, TRUE,
  ifelse(DIQ$DIQ172 == 2, FALSE, NA
  ))

sample(DIQ$FeelCouldHaveDiabetesRisk[!is.na(DIQ$FeelCouldHaveDiabetesRisk)], 10)
```

```sql
SELECT DIQ180, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ180
ORDER BY DIQ180
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$HadBloodTestInLast3Years <-
  ifelse(DIQ$DIQ180 == 1, TRUE,
  ifelse(DIQ$DIQ180 == 2, FALSE, NA
  ))

sample(DIQ$HadBloodTestInLast3Years[!is.na(DIQ$HadBloodTestInLast3Years)], 10)
```

```sql
SELECT DIQ050, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ050
ORDER BY DIQ050
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$TakingInsulinNow <-
  ifelse(DIQ$DIQ050 == 1, TRUE,
  ifelse(DIQ$DIQ050 == 2, FALSE, NA
  ))

sample(DIQ$TakingInsulinNow[!is.na(DIQ$TakingInsulinNow)], 10)
```

```sql
SELECT DIQ070, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ070
ORDER BY DIQ070
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$OnPillsToLowerBloodSugar <-
  ifelse(DIQ$DIQ070 == 1, TRUE,
  ifelse(DIQ$DIQ070 == 2, FALSE, NA
  ))

sample(DIQ$OnPillsToLowerBloodSugar[!is.na(DIQ$OnPillsToLowerBloodSugar)], 10)
```

```sql
SELECT DIQ240, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ240
ORDER BY DIQ240
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$HasDiabetesDoctor <-
  ifelse(DIQ$DIQ240 == 1, TRUE,
  ifelse(DIQ$DIQ240 == 2, FALSE, NA
  ))

sample(DIQ$HasDiabetesDoctor[!is.na(DIQ$HasDiabetesDoctor)], 10)
```

```sql
SELECT DIQ275, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ275
ORDER BY DIQ275
```

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$DrCheckedA1CInPastYear <-
  ifelse(DIQ$DIQ275 == 1, TRUE,
  ifelse(DIQ$DIQ275 == 2, FALSE, NA
  ))

sample(DIQ$DrCheckedA1CInPastYear[!is.na(DIQ$DrCheckedA1CInPastYear)], 10)
```

Units needed from one column

```
SELECT DIQ260U, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ260U
ORDER BY DIQ260U
```

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$BloodGlucoseCheckFrequencyUnits <-
  ifelse(DIQ$DIQ260U == 1, "daily",
  ifelse(DIQ$DIQ260U == 2, "weekly",
  ifelse(DIQ$DIQ260U == 3, "monthly",
  ifelse(DIQ$DIQ260U == 4, "yearly", NA
  ))))

# Most samples are NA, so we exclude them when sampling
sample(DIQ$BloodGlucoseCheckFrequencyUnits[!is.na(DIQ$BloodGlucoseCheckFrequencyUnits)], 10)
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID260 >= 1 AND DID260 <= 15 THEN '1-15'
               WHEN DID260 = 0 THEN 'Never'
               ELSE DID260
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```
# Values not related to age should be replaced

DIQ <- DIQ %>%
  mutate(BloodGlucoseCheckFrequency =
           ifelse((BloodGlucoseCheckFrequencyUnits == "daily"),DID260,
           ifelse((BloodGlucoseCheckFrequencyUnits == "weekly"),DID260*7,
           ifelse((BloodGlucoseCheckFrequencyUnits == "monthly"),DID260*31,
           ifelse((BloodGlucoseCheckFrequencyUnits == "yearly"),DID260*365, NA
           )))))
DIQ$BloodGlucoseCheckFrequency <- days(DIQ$BloodGlucoseCheckFrequency)

# Most samples are NA, so we exclude them when sampling
sample(DIQ$BloodGlucoseCheckFrequency[!is.na(DIQ$BloodGlucoseCheckFrequency)], 10)
```

```
SELECT DIQ060U, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ060U
ORDER BY DIQ060U
```

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$DurationTakingInsulinUnits <-
  ifelse(DIQ$DIQ260U == 1, "monthly",
  ifelse(DIQ$DIQ260U == 2, "yearly", NA
  ))
```

```r
# Most samples are NA, so we exclude them when sampling
sample(DIQ$DurationTakingInsulinUnits[!is.na(DIQ$DurationTakingInsulinUnits)], 10)
```

```sql
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID060 >= 1 AND DID060 <= 55 THEN '1-55'
               WHEN DID060 = 666 THEN 'Less than 1 month'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

```r
# Values not related to age should be replaced

DIQ <- DIQ %>%
  mutate(DurationTakingInsulin =
           ifelse((DurationTakingInsulinUnits == "monthly"),DID060,
           ifelse((DurationTakingInsulinUnits == "yearly"),DID060*12, NA
           )))
DIQ$DurationTakingInsulin <- months(DIQ$DurationTakingInsulin)

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DurationTakingInsulin[!is.na(DIQ$DurationTakingInsulin)], 10)
```