MSK Data Quality Test

Meera Y. Patel, MD

4/30/2019

Assigment

Hello!

In this step of the interview process, we're interested in learning about your ability to work with data and explain your findings in a clear, understandable way.

The dataset in this workbook represents one year of patient visits to the fictional adult urological cancer clinic. In preparation to the development of the clinic Activity Dashboard, you have been asked to perform data quality (DQ) assessment and create a report.

Please write a concise report of all your findings, highlighting important DQ factors that may preclude us from performing reliable analysis. This report may include visuals or analyses that would effectively communicate important information.

Since this dataset is not the only piece of information available to the clinic management team, your report can also include any questions that are prompted by the data for the end users.

Any programmatic analysis language (R, Python, SAS, etc.) or Excel tools you would like to use is fine. Please submit your code/Excel tables, etc. along with your report, in a separate file.

As you acknowledged in your confidentiality statement, we expect all work to be completed by you alone. Please don't share these materials with anyone else.

We look forward to seeing your work!

The Knowledge Management Team

Part One: Quality Framework Method

Introduction

This assignment is to apply a data quality framework and report the results on the given clinical dataset. Important factors to take into account prior to analysis is defining the patient population and the timeline bounds of the data such as:

Cohort: adult, urological cancer clinic patients
Timeline: one year
Event: clinic visit

The framework will first standardize the dataset and take a brief overview of the dataset. This will be followed by assigning a datatype to each field in the dataset (*01_dataframe_characterization.R*). The dataframe will be divided according to datatype, which is then subsequently fed into its respective data quality pipeline (*02_qa.R*).

The input data is first imported and prepared via standardization. Standardization is a step towards ensuring that the data quality framework is reproducible and scalable regardless of the data source (*00_config.R*). This entails removing white spaces, converting column names to upper case and replacing punctuation with only underscores, and adding a primary key for internal tracking purposes.

The a 5 observation sample of the raw input data:

```
##   PATIENT_ID PATIENT_DIAGNOSIS    INSURANCE_TYPE AGE SEX VISIT_DATE
## 1    4591390     Kidney Cancer Private Insurance  40   F 2005-11-09
## 2    4574407   Prostate Cancer          Medicare  48   M 2005-04-27
## 3    4588027     Kidney Cancer          Medicaid  84   M 2005-01-13
## 4    4600553 Testicular Cancer          Medicare  32   M 2005-11-10
## 5    4586837    Bladder Cancer Private Insurance  42   M 2005-03-23
##   PT_SCHEDULED_APPT PT_ARRIVE_TIME PT_START_TIME PT_END_TIME PROVIDER_NAME
## 1             11:15          11:11         11:24       11:40   C. Siu Ming
## 2             12:00          12:00         12:04       12:19    L. Svenson
## 3             17:15          17:18         17:40       18:02     W. Plinge
## 4             11:15          11:11         11:24       11:45      J. Smith
## 5             12:00          12:03         12:12       12:47      E. Ahuja
```

A 5 observation sample of the standardized input data:

```
##   PKEY PATIENT_ID PATIENT_DIAGNOSIS INSURANCE_TYPE AGE SEX VISIT_DATE
## 1 2333    4583443   prostate cancer   self insured  32   m 2005-06-01
## 2 3940    4578380   prostate cancer       medicaid  55   m 2005-10-11
## 3 2064    4584589   prostate cancer       medicaid  50   m 2005-05-16
## 4 5114    4611899 testicular cancer       medicare  52   m 2005-12-22
## 5 4769    4582879     kidney cancer       medicaid  72   f 2005-11-30
##   PT_SCHEDULED_APPT PT_ARRIVE_TIME PT_START_TIME PT_END_TIME PROVIDER_NAME
## 1             09:00          09:00         09:15       09:40     s. moreau
## 2             09:15          09:11         09:26       09:48     s. moreau
## 3             16:00          16:03         16:25       16:48     s. moreau
## 4             09:29          09:32         09:41       10:04      j. smith
## 5             15:15          15:11         15:21       15:37   c. siu ming
```

Script Features

There are 3 main common features to all the subprocesses of this framework:
1. **SCRIPT_MAP**: the primary map used for a certain pipeline to drive the qa process
2. **INPUT_DATA**: data the was provided in the assignment
3. **OUTPUT_FLAGS**: observations in the INPUT_DATA that have been flagged for further review

01. Dataframe Characterization (*01_dataframe_characterization.R*)

Dataframe characterization entails surveying the dataset as a whole, and then diving deeper into the data by assigning a datatype for each field:

```
## 'data.frame':    5260 obs. of  12 variables:
##  $ PKEY             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ PATIENT_ID       : chr  "4591553" "4607440" "4590430" "4619481" ...
##  $ PATIENT_DIAGNOSIS: chr  "prostate cancer" "kidney cancer" "bladder cancer" "testicular cancer" ..
##  $ INSURANCE_TYPE   : chr  "private insurance" "medicaid" "medicaid" "self insured" ...
##  $ AGE              : chr  "75" "48" "72" "40" ...
##  $ SEX              : chr  "m" "f" "m" "m" ...
##  $ VISIT_DATE       : chr  "2005-01-03" "2005-01-03" "2005-01-03" "2005-01-03" ...
##  $ PT_SCHEDULED_APPT: chr  "07:59" "08:15" "08:30" "09:00" ...
##  $ PT_ARRIVE_TIME   : chr  "07:55" "08:15" "08:26" "09:03" ...
##  $ PT_START_TIME    : chr  "08:07" "08:18" "08:42" "09:11" ...
##  $ PT_END_TIME      : chr  "08:22" "08:53" "09:17" "09:39" ...
##  $ PROVIDER_NAME    : chr  "l. svenson" "i. petrov" "e. ahuja" "j. smith" ...
```

The distribution of values for each variable is also surveyed:

```
##    PATIENT_ID           PATIENT_DIAGNOSIS           INSURANCE_TYPE
##  4571846:   4   bladder cancer   :1111                      :  63
##  4574810:   4   kidney cancer    :1593    medicaid         :2095
##  4578206:   4   prostate cancer  :1722    medicare         :1492
##  4581714:   4   testicular cancer: 834    private insurance:1055
##  4582013:   4                            self insured     : 555
##  4585255:   4
##  (Other):5236
##       AGE       SEX           VISIT_DATE   PT_SCHEDULED_APPT
##  72     : 549   f: 970   2005-12-05:  68   10:00  : 485
##  54     : 183   m:4290   2005-12-21:  36   09:00  : 423
##  55     : 179            2005-04-22:  32   12:00  : 412
##  58     : 174            2005-04-29:  32   10:59  : 393
##  59     : 156            2005-06-03:  32   07:59  : 308
##  57     : 151            2005-05-06:  31   09:29  : 278
##  (Other):3868            (Other)   :5029   (Other):2961
##  PT_ARRIVE_TIME PT_START_TIME   PT_END_TIME     PROVIDER_NAME
##  10:00  : 171   09:05  :  35   10:37  :  31   j. smith : 834
##  09:56  : 160   12:09  :  32   10:34  :  29   w. plinge: 827
##  09:03  : 156   10:05  :  31   10:57  :  28   s. moreau: 809
##  11:56  : 155   10:11  :  31   10:55  :  27   e. ahuja : 568
##  10:03  : 154   12:07  :  30   10:35  :  26   n. fulano: 543
##  08:56  : 141   10:06  :  29   11:02  :  26   m. dupont: 461
##  (Other):4323   (Other):5072   (Other):5093   (Other)  :1218
```

We then weed out duplicate values from the input data (*INPUT_DATA object*), which is then added to a dataframe of flagged data (*OUTPUT_FLAGS object*) in the same original structure with the addition of *FLAG_REASON* and *FLAG_TYPE variables*. The *FLAG_REASON value* for this data is "duplicate".

A 5 observation sample of the flagged data is:

```
##   PKEY PATIENT_ID PATIENT_DIAGNOSIS    INSURANCE_TYPE AGE SEX VISIT_DATE
## 1 4836    4574810   prostate cancer          medicare  24   m 2005-12-05
## 2 4862    4578206     kidney cancer          medicare  55   m 2005-12-05
## 3 4848    4608321     kidney cancer private insurance  52   m 2005-12-05
## 4 4885    4619036 testicular cancer          medicare  41   m 2005-12-05
## 5 4902    4590150 testicular cancer          medicaid  24   m 2005-12-05
##   PT_SCHEDULED_APPT PT_ARRIVE_TIME PT_START_TIME PT_END_TIME PROVIDER_NAME
## 1             07:59          07:59         08:02       08:25     s. moreau
## 2             10:00          09:56         09:59       10:21     w. plinge
## 3             09:00          09:03         09:07       09:28     w. plinge
## 4             11:44          11:44         11:46       12:08      j. smith
## 5             16:59          16:59         17:01       17:25      j. smith
##   FLAG_REASON
## 1   duplicate
## 2   duplicate
## 3   duplicate
## 4   duplicate
## 5   duplicate
```

For this exercise, the possible datatypes are limited to four: **category**, **number**, **date**, and **time**. In practice, there are additional types such as datetime, string, and boolean. Each datatype follows its assigned rules. An overall overview of the datatypes and their associated rules can be seen in Table 1. These rules are called **Standalone Rules** because it is assessing the data quality in an isolated manner and not relative to other

Table 1: Standalone Rules

COL_DATATYPE	DATATYPE_RULE	DATATYPE_CONSTRAINT	SOFT_FLAG	HARD_FLAG	QA_STEP
category	matches control data	requires control data	not found within control data		a
number	only numeric characters with maximum 1 decimal point		greater or less than 2.5 standard deviations from the mean		b
date	maximum 8 and minimum 4 numeric characters, maximum 2 punctuation characters		greater or less than 2.5 standard deviations from the mean	future date	c
time	maximum 6 numeric characters, maximum 2 punctuation characters			times greater than or equal to 24:00	d

variables found in this data such as relative to visit dates. **Relative Rules** are implemented downstream with increasing complexity, which will be explained later.

Other important definitions include:
Hard Flag: clinically impossible value. For example, a birthdate that takes place in the future is clinically impossible.
Soft Flag: clinically improbable value, such as an extremely high white blood cell count of 30000. Each variable is assigned a datatype and joined with the rules to create a *SCRIPT_MAP object* that is used to guide the remainder of the framework. It is a list that has divided the column names based on datatype to execute scripts against. Each datatype has its own downstream pipeline indicated by the *QA_STEP variable* that this script map will direct the data flow with for **Step 02** in the framework.

```
## $category
##             COL_NAME COL_DATATYPE QA_STEP
## 1               PKEY     category       a
## 2         PATIENT_ID     category       a
## 3  PATIENT_DIAGNOSIS     category       a
## 4     INSURANCE_TYPE     category       a
## 6                SEX     category       a
## 12     PROVIDER_NAME     category       a
##
## $date
##     COL_NAME COL_DATATYPE QA_STEP
## 7 VISIT_DATE         date       c
##
## $number
##   COL_NAME COL_DATATYPE QA_STEP
## 5      AGE       number       b
##
## $time
##             COL_NAME COL_DATATYPE QA_STEP
## 8  PT_SCHEDULED_APPT         time       d
## 9     PT_ARRIVE_TIME         time       d
## 10     PT_START_TIME         time       d
## 11       PT_END_TIME         time       d
```

02. Quality Framework By Datatype

a. Category (*02a_category_qa.R*)

A **category** datatype is defined as a vector of possible values that the content of the column can fall under. Therefore a requisite for a quality process for this datatype is having the necessary control data to compare it to.

The script map for this step is customized to guide this effort:

```
##            COL_NAME COL_DATATYPE CONTROL_EXISTS          CONTROL_OBJ_NAME
## 1              PKEY     category          FALSE
## 2        PATIENT_ID     category          FALSE
## 3 PATIENT_DIAGNOSIS     category           TRUE PATIENT_DIAGNOSIS_CONTROL
## 4    INSURANCE_TYPE     category           TRUE    INSURANCE_TYPE_CONTROL
## 5               SEX     category           TRUE               SEX_CONTROL
## 6     PROVIDER_NAME     category           TRUE     PROVIDER_NAME_CONTROL
```

Though *PKEY* and *PATIENT_ID variables* are categorical, they serve as identifiers in the data and do not require controls. The remaining variables require an associated **control vector**, the object name of which is in the *CONTROL_OBJ_NAME variable*. In this exercise, I've assumed that all the unique values for each column is the control. However, in more realistic cases, the control vectors are iteratively updated with new values or error values can be mapped to an existing control value. The controls are as follows:

```
PATIENT_DIAGNOSIS_CONTROL <- tolower(c("Prostate Cancer","Kidney Cancer",
                                       "Bladder Cancer","Testicular Cancer"))
INSURANCE_TYPE_CONTROL <- tolower(c("Private Insurance","Medicaid",
                                    "Self Insured","Medicare",""))
SEX_CONTROL <- tolower(c("M", "F"))
PROVIDER_NAME_CONTROL <- tolower(c("L. Svenson","I. Petrov", "E. Ahuja",
                                   "J. Smith","N. Fulano","M. Dupont",
                                   "S. Moreau","W. Plinge","C. S. Ming","C. Siu Ming"))
```

If there is a mismatch, the mismatched observations are compiled in an *OUTPUT_FLAGS_02A object* and are removed from the input data. It is also noted at this point that there are two providers "C. S. Ming" and "C. Siu Ming" that may potentially be duplicates. In real-life circumstances, I would check to see if they are the same provider, in which case the control set *PROVIDER_NAME_CONTROL* would not include one of the values. The unmatched value would be flagged and mapped to the correct control value.

b. Number (*02b_number_qa.R*)

The customized script map for the number datatype is:

```
##   COL_NAME COL_DATATYPE
## 1      AGE       number
```

In each variable, a sequential check on the number of numbers, decimals, other punctuation, and letters is used to flag values that do not fall within the rule of only numbers and no more than one decimal point. In this exercise, only one column is designated a number datatype, and analysis on the mentioned character counts returns no observations to flag. All the values in the *AGE variable* consist of only 2 numbers.

c. Date (*02c_date_qa.R*)

Like the previous datatype, a series of sequential checks were conducted on the number of different types of characters. The rules for the date datatype in this exercise were no greater than 2 punctuation marks, no

more than 8 numbers, no less than 4 numbers, values that did not parse using the *lubridate::ymd function*, and parsed dates that occurred in the future.

The script map is very similar to the one for number datatype:

```
##     COL_NAME COL_DATATYPE
## 1 VISIT_DATE         date
```

Applying the rules above on the single column, returns no values to flag. All values in the *VISIT_DATE variable* follow the format of "YYYY-MM-DD" and none of the dates occur in the future.

d. Time (*02d_time_qa.R*)

Like the previous 2 datatypes, a series of sequential checks were conducted on the number of different types of characters. The rules for the time datatype in this exercise were no greater than 2 punctuation marks, no more than 6 numbers, no less than 3 numbers, and values that did not parse using the *lubridate::hm function*.

The script map contains 4 columns this time:

```
##            COL_NAME COL_DATATYPE
## 1 PT_SCHEDULED_APPT         time
## 2    PT_ARRIVE_TIME         time
## 3     PT_START_TIME         time
## 4       PT_END_TIME         time
```

Applying the rules above on all columns, returns no values to flag, however. All data in the 4 columns contain valid hour and minute values as determined by the clinical rules applied.

03. Final Output (*03_final.R*)

The final output is *final_standalone_flag_output.xlsx*. The final **validated data** is de-duplicated. The duplicates and flags from **Step 02a to 02d** are combined into a single **output flags dataframe**. Additionally, there is an overlap between the final validated data and flagged data that is designated in an **overlap** tab. More detailed definitions for the three tabs:
VALIDATED_DATA: all observations that have been de-duplicated and has passed the standalone datatype quality checks.
OUTPUT_FLAGS: observations that have been flagged with the reason in the *FLAG_REASON variable* and flag severity (hard, soft, or NA) in the *FLAG_TYPE variable*.
OVERLAP_INPUT_FLAGS: ideally only one observation is accounted for between the validated data and the flagged data, but an overlap might occur inevitably. In the case of this exercise, the overlap occured because blank values are flagged regardless by default, but a blank value was also included in the control data for *INSURANCE_TYPE variable*. This overlap will require addressing whether blank values should be considered a unique control value, imputed with another value, or considered missing for this specific variable.

04. Beyond Standalone Flags

Conducting a quality check past the standalone flags are beyond the scope of this assignment. However, for a more thorough quality check, some **soft flags** such as values greater than or less than 2.5 standard deviations from the mean for enumerated datatypes and **relative flags** are employed in increasing complexity.

Soft Flags

Soft flags are a means to ensure that the outliers are indeed outliers as opposed to technical or human error. All flags for category data are typically soft because the control data is dynamically updated and can have an

infinite number of control values in theory. However, the highest and lowest values in enumerated datatypes such as number and date may be fed into a pipeline that involves patient record checks to ensure validity and reliability. The only exception would be time datatype, where the data value is circular rather than linear.

Relative Flags

These flags can be relative to the datatype, patient, provider, clinic, diagnosis, and so on. For example, a relative flag for the time datatype in the data here would be making sure that the *PT_START_TIME variable* occurs prior to the *PT_END_TIME variable*. On the other hand, a flag relative to the patient record can be checking if a reported appointment date falls between the patient's date of birth and date of death. These flags can scale in the number of variables being compared against each other at infinite levels that include patient-level, provider-level, diagnosis-level, etc.

Part Two: Analysis

After adjudicating on the flagged and overlap data, the final validated data would be ready for analysis. Assumptions made to this effect include that blank values in the *INSURANCE_TYPE variable* are valid control values and are included in the data used for analysis.

The validated data has 5209 unique observations. The original data had 5260 observations, but 51 were found to be duplicates and excluded.

The variables of the final validated data is converted to the appropriate data class using the script map created in **Step 01 Data Characterization** that designates the datatype.

```
##              COL_NAME COL_DATATYPE
## 1                PKEY     category
## 2          PATIENT_ID     category
## 3  PATIENT_DIAGNOSIS     category
## 4     INSURANCE_TYPE     category
## 5                 SEX     category
## 6       PROVIDER_NAME     category
## 7          VISIT_DATE         date
## 8                 AGE       number
## 9  PT_SCHEDULED_APPT         time
## 10    PT_ARRIVE_TIME         time
## 11     PT_START_TIME         time
## 12       PT_END_TIME         time
```

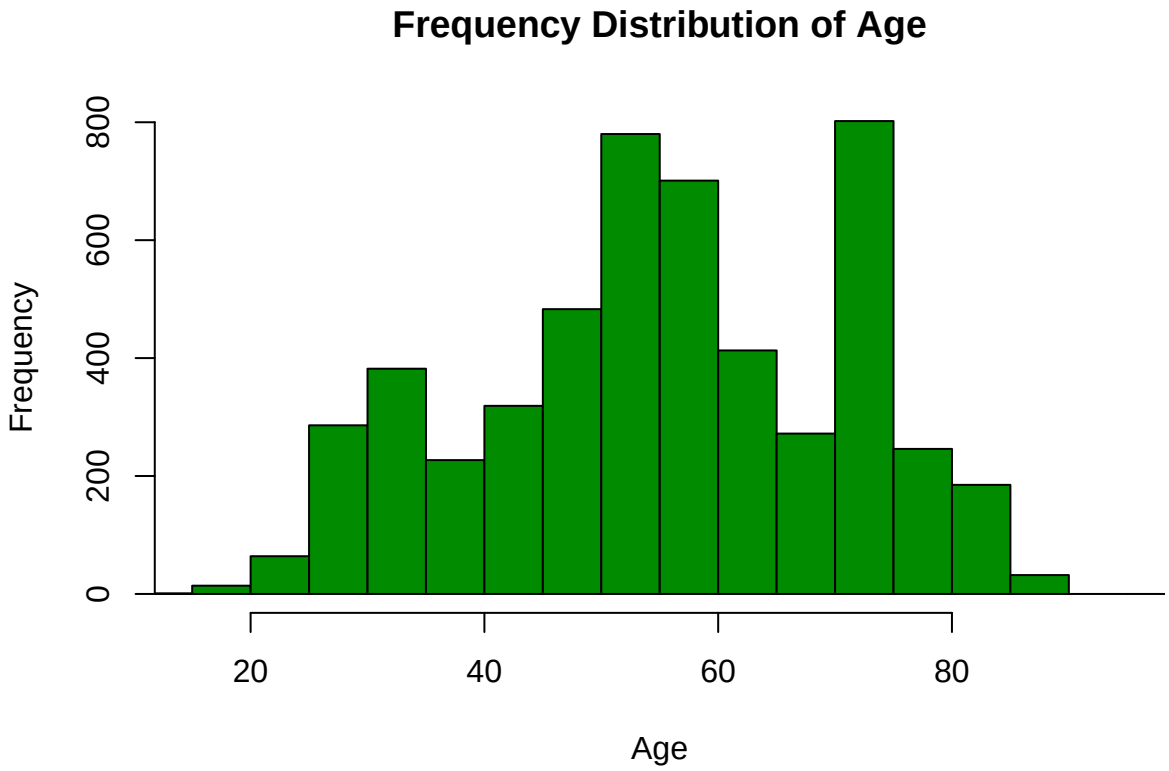
Category datatype columns will be kept as character class while number will be converted to double class, date will be converted to date class, and time will be converted to POSIXct:

```
for (i in 1:nrow(SCRIPT_MAP$date)) {
          x <- SCRIPT_MAP$date$COL_NAME[i]
          x <- enquo(x)
          INPUT_DATA_ANALYSIS <-
          INPUT_DATA_ANALYSIS %>%
                    mutate_at(vars(!!x), list(~ymd(.)))
}

for (i in 1:nrow(SCRIPT_MAP$number)) {
          x <- SCRIPT_MAP$number$COL_NAME[i]
          x <- enquo(x)
          INPUT_DATA_ANALYSIS <-
```

```
          INPUT_DATA_ANALYSIS %>%
                    mutate_at(vars(!!x), list(~as.double(.)))
}

for (i in 1:nrow(SCRIPT_MAP$time)) {
          x <- SCRIPT_MAP$time$COL_NAME[i]
          x <- enquo(x)
          INPUT_DATA_ANALYSIS <-
          INPUT_DATA_ANALYSIS %>%
                    mutate_at(vars(!!x), list(~hm(.)))
}
```

01. Patient-Level

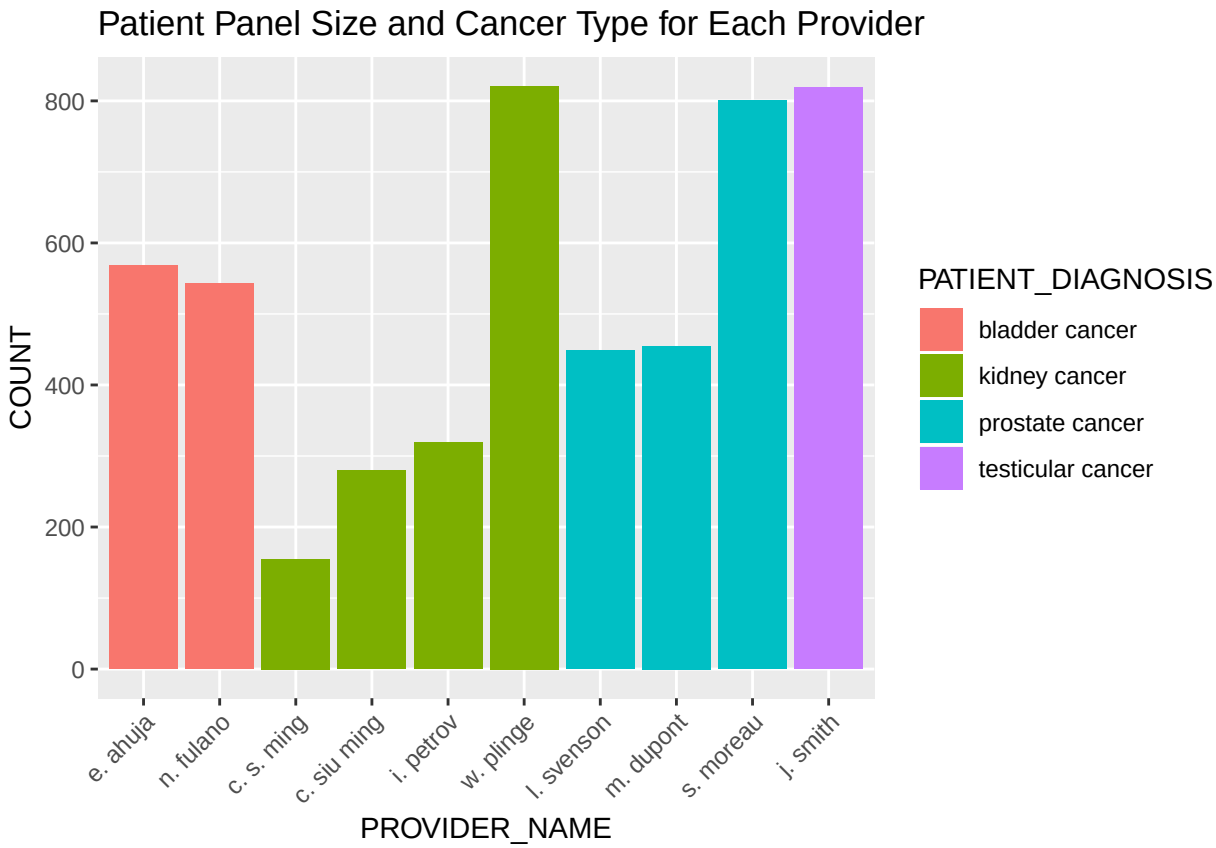
The patient population can be captured with a histogram plotting the distribution of patient qualities. The distribution of age is as follows:



It is common assumption that the more complex the patient, the more clinic time is needed to treat the patient. In this dataset, a patient with multiple primary cancers may be considered a complex patient. However, summary data indicates that all patients in this dataset have one primary cancer diagnosis.

02. Provider

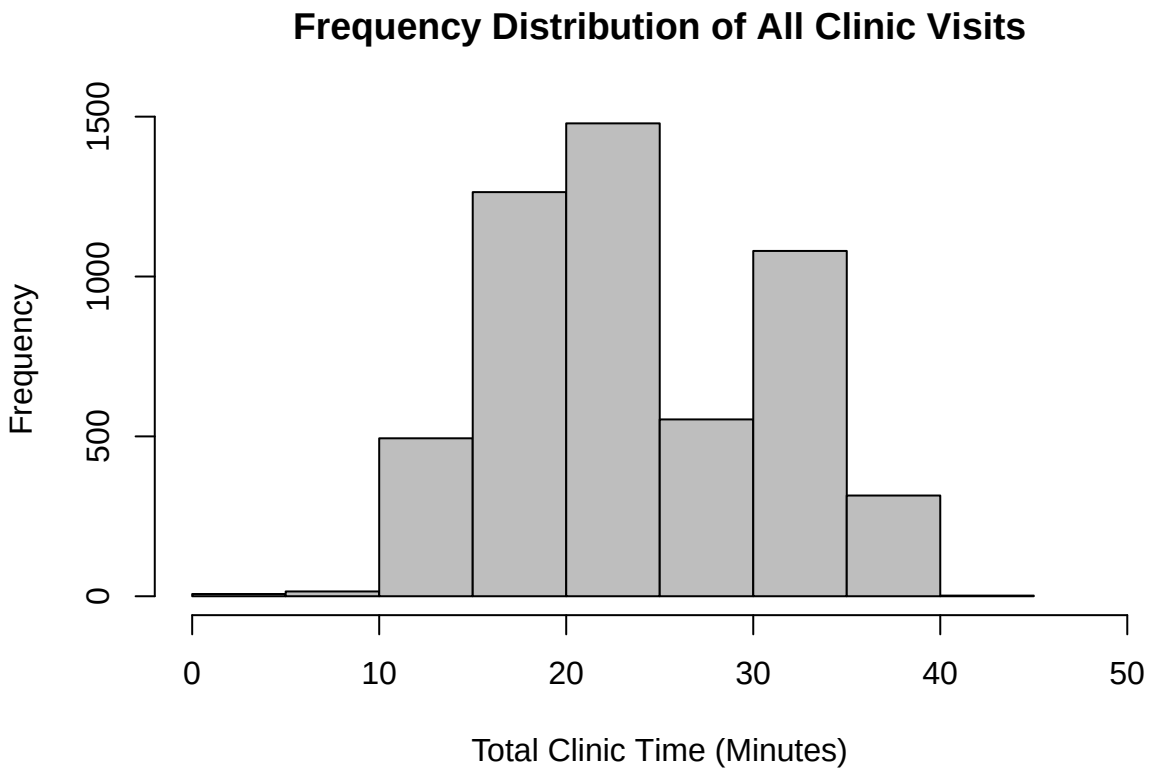
We can also delve into assessing provider performance. Again assuming that "C. S. Ming" and "C. Siu Ming" are two different providers, we can look at the patient population each provider serves. This bar chart was meant to stack patients based on cancer type, but upon visualization, it is clear that each provider treats one cancer type at this clinic.



03. Cancer Type

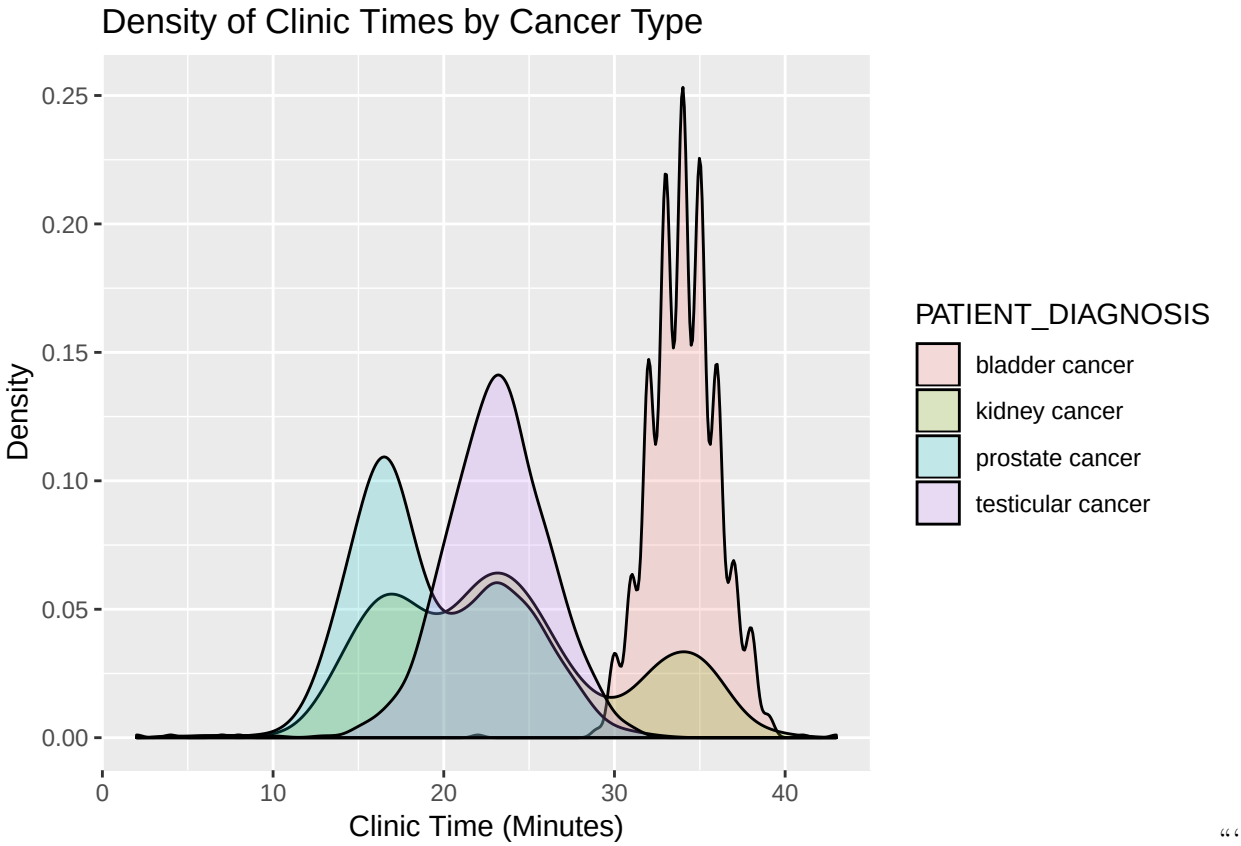
The total clinic time in minutes was calculated based on *PT_START_TIME* and *PT_END_TIME variables*. **Relatively flagged** data was identified when the *PT_START_TIME varible* took place after the *PT_END_TIME variable*. An assumption was made that these values were interchanged to calculate the results rather than excluded.

The frequency distribution was then visualized for clinic time for all appointments.



While the above histogram provides a general idea of the distribution of time dedicated to patients at this clinic, the cancer types represented in this dataset (testicular, kidney, prostate, bladder) vary widely in life expectancy, treatment pathways, and prevalent patient characteristics. For example, prostate cancer is less deadly than the other cancers listed, where many patients may die of other age-related causes before the cancer progresses. Testicular cancer tends to affect younger populations and the long-term survival rate is also higher than other cancers. Bladder and kidney cancers are much more to affect life expectency and require prompt aggressive treatment.

Sieving the clinic time distribution by cancer type shows a different story. Bladder cancer appointments tend to require longer clinic times. Kidney cancer also has a higher density in clinic times greater than 30 minutes, but its distribution has the largest range across times than the other cancers. As expected, prostate and testicular cancers fall on the lower end of the spectrum, since these types of visits are more focused on surveillance than aggressive treatment.



‘‘

Questions For End Users

- How should the blank insurance value be handled? It can be considered a valid control with its own unique definition or it can be imputed/mapped to an agreed upon value.
- Are C. S. Ming and C. Sui Ming the same provider?
- When the start time is after the end time, is it safe to assume that these values were accidentally interchanged at the time of capture?