QBS181 Final

Mark Taylor

Setup

I first begin by making my connection to the database using the odbc package.

```
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)

# Connect to database
con <- DBI::dbConnect(odbc::odbc(),
                      Driver   = "/usr/local/lib/libmsodbcsql.17.dylib",
                      Server   = "qbs181-db.dartmouth.edu",
                      Database = "qbs181",
                      UID      = "mtaylor",
                      PWD      = "mtaylor@qbs181",
                      Port     = 40062)
```

1.

I start by reading in the .csv BP data and display 10 random rows.

```
bpdata = read.csv("~/qbs/qbs181/IC_BP_v2.csv")
sample_n(bpdata,10)
```

```
##                                     ID SystolicValue Diastolicvalue BPAlerts
## 1  0B6DB84D-A7F7-E511-811C-FC15B42886E8           125             91     HTN1
## 2  CC45C5F7-0C16-E611-8128-C4346BB59854           124             76   Normal
## 3  6660B45A-EC4C-E611-80E8-5065F38AF8B1           135             72   Normal
## 4  E950A85A-1B4A-E611-80E8-5065F38B0171           138             78   Normal
## 5  1DEA1762-05E0-E511-8122-C4346BB59854           120             74   Normal
## 6  1FEA1762-05E0-E511-8122-C4346BB59854           110             74    Hypo1
## 7  29F9BF37-E61D-E611-8128-C4346BB59854           123             58    Hypo1
## 8  C886A327-A108-E611-811D-C4346BACD1A8           103             63    Hypo1
## 9  5693A7B0-7148-E611-80E7-5065F38B3241           141            109     HTN2
## 10 6587A327-A108-E611-811D-C4346BACD1A8           108             76    Hypo1
##    ObservedTime
## 1         42654
## 2         42590
## 3         42671
## 4         42645
## 5         42743
## 6         42642
## 7         42711
## 8         42545
## 9         42646
## 10        42628
```

(a)

BPAlerts is a column name in the BP dataset, so to convert to BP Status I rename the column to BPStatus.

```
bpdata <- bpdata %>%
  rename(BPStatus = BPAlerts)
sample_n(bpdata,10)
```

```
##                                     ID SystolicValue Diastolicvalue BPStatus
## 1  61EA1762-05E0-E511-8122-C4346BB59854           116             80   Normal
## 2  45EA1762-05E0-E511-8122-C4346BB59854           155             72     HTN1
## 3  5693A7B0-7148-E611-80E7-5065F38B3241           114             94     HTN1
## 4  E950A85A-1B4A-E611-80E8-5065F38B0171           137             82   Normal
## 5  77FA86B2-2D0B-E611-8120-C4346BAD2660           112             69    Hypo1
## 6  B7FA86B2-2D0B-E611-8120-C4346BAD2660           121             91     HTN1
## 7  CE86A327-A108-E611-811D-C4346BACD1A8           121             69    Hypo1
## 8  EE87A7B0-7148-E611-80E7-5065F38B3241           155             97     HTN1
## 9  29F9BF37-E61D-E611-8128-C4346BB59854           108             58    Hypo1
## 10 E3B436AE-A7E7-E511-8116-C4346BAC02E8           136             86     HTN1
##    ObservedTime
## 1         42590
## 2         42491
## 3         42581
## 4         42600
## 5         42518
## 6         42488
```

```
## 7         42530
## 8         42616
## 9         42556
## 10        42470
```

(b)

I start by defining a function to determine if a patient has Controlled or Uncontrolled BP; this function returns 1 for the former and 0 for the latter. sapply() can then be used to generate a column by applying the function to each row.

```
# function to find if BP is controlled
control.test <- function(status) {
  if (status == "Hypo1" | status == "Normal") {
    return (1)
  }
  else {
    return(0)
  }
}

# We encode controlled as 1 and uncontrolled as 0
bpdata$Controlled <- sapply(bpdata$BPStatus, control.test, USE.NAMES = F)

sample_n(bpdata,10)
```

```
##                                     ID SystolicValue Diastolicvalue BPStatus
## 1  A445C5F7-0C16-E611-8128-C4346BB59854           133             87     HTN1
## 2  1FEA1762-05E0-E511-8122-C4346BB59854           133             72   Normal
## 3  A03C47D1-1FE4-E511-8115-C4346BB5981C           127             93     HTN1
## 4  6587A327-A108-E611-811D-C4346BACD1A8           111             85   Normal
## 5  1089E3CB-5111-E611-811B-C4346BAC02E8           107             71    Hypo1
## 6  B88A4B4F-A3E5-E511-811A-C4346BACD1A8           126             72   Normal
## 7  CE86A327-A108-E611-811D-C4346BACD1A8           142             87     HTN1
## 8  1FEA1762-05E0-E511-8122-C4346BB59854           106             73    Hypo1
## 9  CE86A327-A108-E611-811D-C4346BACD1A8           140             89     HTN1
## 10 E950A85A-1B4A-E611-80E8-5065F38B0171           117             69    Hypo1
##    ObservedTime Controlled
## 1         42511          0
## 2         42437          1
## 3         42489          0
## 4         42647          1
## 5         42503          1
## 6         42737          1
## 7         42528          0
## 8         42738          1
## 9         42523          0
## 10        42693          1
```

(c)

I first write the table I've worked on in R onto the server, then merge it with Demographics by matching IDs. I could have used sqldf() and performed the join in R, but opted to use the server as it is more optimized

for SQL operations like joins.

```
library(DBI)

dbWriteTable(con, SQL("mtaylor.bpdata"), bpdata, overwrite = T)

bpdata <- dbGetQuery(con, "SELECT B.*,
                           D.tri_enrollmentcompletedate as EnrollDate
                           FROM qbs181.mtaylor.bpdata as B
                                LEFT JOIN Demographics as D
                                ON B.ID=D.contactid")

sample_n(bpdata,10)
```

```
##                                     ID SystolicValue Diastolicvalue BPStatus
## 1  C0FDFEE3-D4E4-E511-8123-C4346BB59854            94             62    Hypo1
## 2  6660B45A-EC4C-E611-80E8-5065F38AF8B1           152             71     HTN1
## 3  BDFA86B2-2D0B-E611-8120-C4346BAD2660           129             90     HTN1
## 4  47C234AD-C355-E611-80EC-5065F38B0171           113             78   Normal
## 5  57EA1762-05E0-E511-8122-C4346BB59854           124             82   Normal
## 6  29F9BF37-E61D-E611-8128-C4346BB59854           123             55    Hypo1
## 7  E950A85A-1B4A-E611-80E8-5065F38B0171           179             86     HTN2
## 8  77FA86B2-2D0B-E611-8120-C4346BAD2660           136             87     HTN1
## 9  6D87A327-A108-E611-811D-C4346BACD1A8           121             75   Normal
## 10 57EA1762-05E0-E511-8122-C4346BB59854           124             79   Normal
##    ObservedTime Controlled EnrollDate
## 1         42591          1   3/8/2016
## 2         42592          0  7/19/2016
## 3         42631          0  4/25/2016
## 4         42705          1  7/29/2016
## 5         42488          1   3/3/2016
## 6         42684          1  5/19/2016
## 7         42658          0  7/24/2016
## 8         42521          0  4/25/2016
## 9         42506          1  4/25/2016
## 10        42500          1   3/3/2016
```

(d)

My interpretation of this question is that for each customer, we find the 12-week interval beginning with their date of enrollment and take the averages of the their scores that fall within their 12-week interval.

We start by identifying each customer; we do so by finding the unique ID's.

```
customers <- unique(bpdata$ID)
```

Next we create a function to find the values of interest for each customer. We also compute some values that will be helpful in parts 1.e and 1.f.

We also begin by interpreting the meaning of OberservedTime. Each BP measure has an associated ObservedTime. We assume the lowest ObservedTime for a given customer is their first measure and the greatest is their final measurement. We "normalize" ObservedTime by subtracting the base ObservedTime from each ObservedTime value.

```r
customerData <- function(customer.id) {
    # find rows for the given customer
    customer.data <- bpdata %>%
      filter(ID == customer.id)

    # create a 12-week interval beginning from the customers EnrollDate
    start <- mdy(min(customer.data$EnrollDate))

    inte = interval(start, start + dweeks(12))
    end = int_end(inte)

    # normalize ObservedTime into weeks
    start.ObservedTime = min(customer.data$ObservedTime)
    customer.data$time = customer.data$ObservedTime - start.ObservedTime
    time.observed = max(customer.data$time)


    # average BP values
    sys.avg = mean(customer.data$SystolicValue)
    dia.avg = mean(customer.data$Diastolicvalue)

    # find base values for parts e. and f.
    base <- customer.data %>%
      top_n(-1, ObservedTime)
    # hacky fix, but top_n() returns more than n if there are ties
    base <- base[1,]
    sys.base = base$SystolicValue
    dia.base = base$Diastolicvalue
    controlled.base = base$Controlled

    # find follow-up values for parts e. and f.
    follow <- customer.data %>%
      top_n(1, ObservedTime)
    follow <- follow[1,]
    sys.follow = follow$SystolicValue
    dia.follow = follow$Diastolicvalue
    controlled.follow = follow$Controlled


    row <- list(ID = customer.id,
                Interval = inte,
                "EnrollDate" = start,
                "TimeObserved" = time.observed,
                "SysAvg" = sys.avg,
                "DiaAvg" = dia.avg,
                "SysBase" = sys.base,
                "DiaBase" = dia.base,
                "ControlledBase" = controlled.base,
                "SysFollow" = sys.follow,
                "DiaFollow" = dia.follow,
                "ControlledFollow" = controlled.follow)
    return (as.data.frame(row))
}
```

```
customer.rows <- lapply(customers, customerData)
customer.data <- do.call(rbind,customer.rows)

sample_n(customer.data,10)
```

```
##                                     ID                        Interval
## 1  05E87E9D-5411-E611-811B-C4346BAC02E8 2016-05-12 UTC--2016-08-04 UTC
## 2  3F87A327-A108-E611-811D-C4346BACD1A8 2016-05-12 UTC--2016-08-04 UTC
## 3  27F9BF37-E61D-E611-8128-C4346BB59854 2016-06-03 UTC--2016-08-26 UTC
## 4  87FD86B2-2D0B-E611-8120-C4346BAD2660 2016-04-26 UTC--2016-07-19 UTC
## 5  5FE4047C-FE1C-E611-8122-C4346BAD2660 2016-05-23 UTC--2016-08-15 UTC
## 6  D4FDFEE3-D4E4-E511-8123-C4346BB59854 2016-03-08 UTC--2016-05-31 UTC
## 7  A5764C35-2945-E611-80E6-C4346BDC9111 2016-07-18 UTC--2016-10-10 UTC
## 8  388AE3CB-5111-E611-811B-C4346BAC02E8 2016-05-14 UTC--2016-08-06 UTC
## 9  5060D306-D547-E611-80E6-5065F38BA151 2016-07-19 UTC--2016-10-11 UTC
## 10 F4FDFEE3-D4E4-E511-8123-C4346BB59854 2016-03-11 UTC--2016-06-03 UTC
##    EnrollDate TimeObserved   SysAvg   DiaAvg SysBase DiaBase ControlledBase
## 1  2016-05-12           13 122.5000 76.10000     123      78              1
## 2  2016-05-12            0 120.0000 95.00000     120      95              0
## 3  2016-06-03            0 100.0000 72.00000     100      72              1
## 4  2016-04-26           64 113.0000 78.22222     119      88              1
## 5  2016-05-23            0 131.0000 84.00000     129      83              1
## 6  2016-03-08            8 126.8182 77.54545     111      69              1
## 7  2016-07-18           37 132.5000 91.50000     132      95              0
## 8  2016-05-14           30 128.5000 87.25000     142      91              0
## 9  2016-07-19          181 114.7982 79.39450     121      91              0
## 10 2016-03-11            0 212.0000 97.00000     212      97              0
##    SysFollow DiaFollow ControlledFollow
## 1        113        72                1
## 2        120        95                0
## 3        100        72                1
## 4        112        66                1
## 5        129        83                1
## 6        120        69                1
## 7        121        87                1
## 8        112        82                1
## 9        129        84                1
## 10       212        97                0
```

Here we run into a problem however. We assume the measure with the highest TimeObserved is the last measurement for a patient, however we are interested in not just the last measure but specifically the follow-up measure after 12 weeks. We only have ObservedTime to go off of in determining when measurements were taken in relation to each other, but we have no idea what units are used.

We have two choices: we can assume the last measurement is the measurement after 12 weeks, or we can estimate the units of ObservedTime to filter out results after 12-weeks.

As the question specifically asks for a 12-week interval, in this case we decide to estimate the units then filter out results after 12 weeks of intervention. We are not given enough information to know for certain if the last measured value is the measure after 12 weeks of intervention, so we justify our decision based on the fact that the 12 weeks would be meaningless information unless we use it to cut off values outside of this interval.

```r
summary(customer.data$TimeObserved)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00   30.00   76.91  131.50  334.00
```

As the max value is 334, the unit must be less than weeks. There are 52 weeks in a year, which would mean the max observed value occurs after 6 years ($\frac{334}{52} > 6$); the study started in 2016 so this would be impossible.

The only reasonable choice is days. While even the mean is less than 12 weeks (84 days is 12 weeks), the max is feasible this way. Many patients did not receive measures up to 12 weeks, but loss to follow up is a common problem in studies so this seems reasonable.

We modify our function above to filter out values greater than 12 weeks (84 days).

```r
customerDataInt <- function(customer.id) {
    # find rows for the given customer
    customer.data <- bpdata %>%
      filter(ID == customer.id)

    # create a 12-week interval beginning from the customers EnrollDate
    start <- mdy(min(customer.data$EnrollDate))

    inte = interval(start, start + dweeks(12))
    end = int_end(inte)

    # normalize ObservedTime into weeks
    start.ObservedTime = min(customer.data$ObservedTime)
    customer.data$time = customer.data$ObservedTime - start.ObservedTime

    # filter for range
    int.customer.data <- customer.data %>%
      filter(time <= 84)
    time.observed = max(int.customer.data$time)


    # average BP values
    sys.avg = mean(int.customer.data$SystolicValue)
    dia.avg = mean(int.customer.data$Diastolicvalue)

    # find base values for parts e. and f.
    base <- int.customer.data %>%
      top_n(-1, ObservedTime)
    # hacky fix, but top_n() returns more than n if there are ties
    base <- base[1,]
    sys.base = base$SystolicValue
    dia.base = base$Diastolicvalue
    controlled.base = base$Controlled

    # find follow-up values for parts e. and f.
    follow <- int.customer.data %>%
      top_n(1, ObservedTime)
    follow <- follow[1,]
    sys.follow = follow$SystolicValue
    dia.follow = follow$Diastolicvalue
```

```
    controlled.follow = follow$Controlled


    row <- list(ID = customer.id,
                Interval = inte,
                "EnrollDate" = start,
                "TimeObserved" = time.observed,
                "SysAvg" = sys.avg,
                "DiaAvg" = dia.avg,
                "SysBase" = sys.base,
                "DiaBase" = dia.base,
                "ControlledBase" = controlled.base,
                "SysFollow" = sys.follow,
                "DiaFollow" = dia.follow,
                "ControlledFollow" = controlled.follow)
    return (as.data.frame(row))
}

customer.rows <- lapply(customers, customerDataInt)
customer.data <- do.call(rbind,customer.rows)

sample_n(customer.data,10)
```

```
##                                     ID                        Interval
## 1  5587A327-A108-E611-811D-C4346BACD1A8 2016-05-09 UTC--2016-08-01 UTC
## 2  27F9BF37-E61D-E611-8128-C4346BB59854 2016-06-03 UTC--2016-08-26 UTC
## 3  43FA86B2-2D0B-E611-8120-C4346BAD2660 2016-05-12 UTC--2016-08-04 UTC
## 4  E37A4C35-2945-E611-80E6-C4346BDC9111 2016-11-29 UTC--2017-02-21 UTC
## 5  3B9C2AA6-1624-E611-812A-C4346BB59854 2016-05-30 UTC--2016-08-22 UTC
## 6  8F784C35-2945-E611-80E6-C4346BDC9111 2016-07-11 UTC--2016-10-03 UTC
## 7  A317A194-E92C-E611-80E2-5065F38BA151 2016-06-07 UTC--2016-08-30 UTC
## 8  C886A327-A108-E611-811D-C4346BACD1A8 2016-04-23 UTC--2016-07-16 UTC
## 9  95FA86B2-2D0B-E611-8120-C4346BAD2660 2016-05-12 UTC--2016-08-04 UTC
## 10 E3E77E9D-5411-E611-811B-C4346BAC02E8 2016-05-13 UTC--2016-08-05 UTC
##    EnrollDate TimeObserved   SysAvg   DiaAvg SysBase DiaBase ControlledBase
## 1  2016-05-09            0 114.0000 66.50000     111      70              1
## 2  2016-06-03            0 100.0000 72.00000     100      72              1
## 3  2016-05-12           81 113.2500 66.58333     134      74              1
## 4  2016-11-29           14 117.0588 78.05882     108      51              1
## 5  2016-05-30            0 101.8000 70.80000      72      57              0
## 6  2016-07-11            0 119.0000 79.00000     116      74              1
## 7  2016-06-07            0 125.0000 77.66667     118      73              1
## 8  2016-04-23           65 108.0984 67.80328     116      77              1
## 9  2016-05-12            0 123.0000 90.50000     113      93              0
## 10 2016-05-13           82 137.6667 77.77778     164      88              0
##    SysFollow DiaFollow ControlledFollow
## 1        111        70                1
## 2        100        72                1
## 3         92        62                1
## 4        120        85                1
## 5         72        57                0
## 6        116        74                1
## 7        118        73                1
```

```
## 8          95        59                1
## 9         113        93                0
## 10        132        68                1
```

(e)

To compare, we find the difference between the baseline and follow-up scores (we saved these values in the previous part).

```
customer.data$SysDiff <- customer.data$SysFollow - customer.data$SysBase
customer.data$DiaDiff <- customer.data$DiaFollow - customer.data$DiaBase

customer.data %>%
  select(SysDiff,DiaDiff) %>%
  sample_n(10)
```

```
##    SysDiff DiaDiff
## 1       -1      -6
## 2        0       0
## 3      -26     -25
## 4        0       0
## 5        0      -4
## 6        0       0
## 7        4       2
## 8       -9     -15
## 9       -8      12
## 10      29      14
```

Looking at a sample of the differences shows mostly decreases in both. Increases could be due to random variation or subjects lost to follow-up (so the intervention didn't have time to decrease BP); the zero differences can be explained the same way, either due to chance or more likely because subjects were never followed up on.

Finding the average difference for both is much more informative and shows there is a decrease in both BP measures, indicating the intervention may be helpful.

```
c(mean(customer.data$SysDiff), mean(customer.data$DiaDiff))
```

```
## [1] -1.594406 -1.384615
```

(f)

To find this we filter the subjects initially uncontrolled, filter those that became controlled, and find the number of remaining subjects.

```
customer.data %>%
  filter(ControlledBase == 0) %>%
  filter(ControlledFollow == 1) %>%
  nrow()
```

```
## [1] 29
```

We can also output 10 random rows from these patients.

```
customer.data %>%
  filter(ControlledBase == 0) %>%
  filter(ControlledFollow == 1) %>%
  sample_n(10)
```

```
##                                     ID                         Interval
## 1  B7FA86B2-2D0B-E611-8120-C4346BAD2660 2016-04-26 UTC--2016-07-19 UTC
## 2  8160825E-7EEB-E511-8125-C4346BB59854 2016-03-16 UTC--2016-06-08 UTC
## 3  EFE77E9D-5411-E611-811B-C4346BAC02E8 2016-05-03 UTC--2016-07-26 UTC
## 4  C3FA86B2-2D0B-E611-8120-C4346BAD2660 2016-04-25 UTC--2016-07-18 UTC
## 5  69D67666-0A0E-E611-811E-C4346BACD1A8 2016-04-30 UTC--2016-07-23 UTC
## 6  6D87A327-A108-E611-811D-C4346BACD1A8 2016-04-25 UTC--2016-07-18 UTC
## 7  388AE3CB-5111-E611-811B-C4346BAC02E8 2016-05-14 UTC--2016-08-06 UTC
## 8  47C234AD-C355-E611-80EC-5065F38B0171 2016-07-29 UTC--2016-10-21 UTC
## 9  B88A4B4F-A3E5-E511-811A-C4346BACD1A8 2016-05-09 UTC--2016-08-01 UTC
## 10 CFC6C4CC-B821-E611-811F-C4346BACD1A8 2016-05-31 UTC--2016-08-23 UTC
##    EnrollDate TimeObserved   SysAvg   DiaAvg SysBase DiaBase ControlledBase
## 1  2016-04-26            3 121.5000 87.50000     121      91              0
## 2  2016-03-16           82 126.7812 86.15625     144     105              0
## 3  2016-05-03            3 122.7500 86.00000     119      89              0
## 4  2016-04-25           68 132.4000 80.20000     136      83              0
## 5  2016-04-30           81 118.2000 80.60000     129      85              0
## 6  2016-04-25           84 126.5441 81.10294     140      93              0
## 7  2016-05-14           30 128.5000 87.25000     142      91              0
## 8  2016-07-29           84 121.7250 81.39167     124      93              0
## 9  2016-05-09           48 145.6667 82.09524     160      82              0
## 10 2016-05-31           36 115.5833 82.41667     123      90              0
##    SysFollow DiaFollow ControlledFollow SysDiff DiaDiff
## 1        122        84                1       1      -7
## 2        118        80                1     -26     -25
## 3        118        84                1      -1      -5
## 4        128        80                1      -8      -3
## 5        130        82                1       1      -3
## 6        107        79                1     -33     -14
## 7        112        82                1     -30      -9
## 8        127        75                1       3     -18
## 9        137        81                1     -23      -1
## 10       105        75                1     -18     -15
```

2.

We are able to do this with one complex query. We will start by describing the subqueries and working our way out.

First for Text, we need to limit so that there is only one row for each ID. We do so by finding the max (most recent) TextSentDates in Text, then selecting all columns from another instance of Text where the IDs match.

Then joining Demographics and Conditions is rather straight forward; we match IDs and extract all columns from Demographics and the tri_name from Conditions.

We then join these two subqueries together, before randomly selecting 10 rows by assigning a new random order and taking the top 10 rows.

```
SELECT TOP 10 *
FROM
(SELECT DC.*, T.SenderName, T.TextSentDate
FROM (SELECT T1.*
      FROM Text as T1
      WHERE T1.TextSentDate = (SELECT MAX(T2.TextSentDate)
                               FROM Text as T2
                               WHERE T2.tri_contactId = T1.tri_contactId)) as T
LEFT JOIN (SELECT D.*, C.tri_name
      FROM Conditions as C
      LEFT JOIN Demographics as D
      ON C.tri_patientid = D.contactid) as DC
ON T.tri_contactId = DC.contactid) as DCT
ORDER BY NEWID()
```

Table 1: 10 random rows from merged Demographics, Conditions and Text

contactid	gender	tri_age	parentcustomerid	address1_stateorprovince	TextSentDate
F151413C-72A0-E611-80E9-5065F38A3B91	2	55	Dartmouth-Hitchcock	NH	2016-11-30
A48FA7B0-7148-E611-80E7-5065F38B3241	2	66	Dartmouth-Hitchcock	NH	2017-01-27
B28CA7B0-7148-E611-80E7-5065F38B3241	2	58	Dartmouth-Hitchcock	NH	2017-01-31
120F0F62-D112-E611-811F-C4346BACD1A8	2	50	Dartmouth-Hitchcock	NH	2016-05-14
2E8EA7B0-7148-E611-80E7-5065F38B3241	2	44	Dartmouth-Hitchcock	VT	2017-01-27
3E50F7DC-A90C-E611-811B-C4346BAC02E8	2	41	Dartmouth-Hitchcock	NH	2017-01-27

contactid	gendercode	tri_age	parentcustomerid	tri_imagineaddress1_stateorprovince	tri_enrollmentstatus	createdon	modifiedon	statuscode	SenderName	TextSentDate
9C9BA7B0-7148-E611-80E7-5065F38B3241	2	62	Dartmouth-Hitchcock	167410011	NH	7/12/2016	7/22/2016	2	ActivitySystem Monitoring	2017-01-27
C8B8F1E3-BE43-E611-80E6-5065F38BA151	2	43	Dartmouth-Hitchcock	167410011	NH	7/6/2016	7/18/2016	2	ActivitySystem Monitoring	2017-01-27
642AC3DE-4C0E-E611-811B-C4346BAC02E8	2	30	Dartmouth-Hitchcock	167410011	NH	4/29/2016	5/2/2016	2	ActivitySystem Monitoring	2017-01-27
DCFDFEE3-D4E4-E511-8123-C4346BB59854	2	44	The ImagineCare Testing Team	167410011	ME	3/8/2016	3/8/2016	2	ActivitySystem Monitoring	2017-01-27

3.

First we grab all of the tables from the database into R.

```
demo <- dbGetQuery(con, "SELECT * FROM Demographics")
cond <- dbGetQuery(con, "SELECT * FROM Conditions")
text <- dbGetQuery(con, "SELECT * FROM Text")
```

Here we have to group by ID and select the top row for each ID. As top_n() returns ties, which will have the same ID and date, we use distinct() to filter them out.

```
text.unique <- text %>%
  group_by(tri_contactId) %>%
  top_n(n=1) %>%
  ungroup() %>%
  distinct(tri_contactId,TextSentDate)
```

```
## Selecting by TextSentDate
```

```
demo.cond <- demo %>%
  left_join(cond,by = c("contactid" = "tri_patientid"))

demo.cond.text <- text %>%
  left_join(demo.cond, by = c("tri_contactId" = "contactid"))

sample_n(demo.cond.text, 10)
```

```
##                          tri_contactId SenderName TextSentDate gendercode
```

```
## 1  57FB86B2-2D0B-E611-8120-C4346BAD2660     System   2016-05-23          2
## 2  928AE3CB-5111-E611-811B-C4346BAC02E8     System   2016-05-06          2
## 3  63E3AF86-D86A-E611-80EE-5065F38AF8B1     System   2016-11-03          2
## 4  7787A327-A108-E611-811D-C4346BACD1A8  Clinician   2016-08-09          2
## 5  FAD8C3D3-E9E6-E511-8116-C4346BAC02E8  Clinician   2016-05-22          1
## 6  27EA1762-05E0-E511-8122-C4346BB59854  Clinician   2016-06-19          1
## 7  FA45C5F7-0C16-E611-8128-C4346BB59854  Clinician   2016-09-28          2
## 8  E3E77E9D-5411-E611-811B-C4346BAC02E8   Customer   2016-08-18          1
## 9  809BA7B0-7148-E611-80E7-5065F38B3241     System   2016-09-09          1
## 10 4B23FCAF-2616-E611-8128-C4346BB59854     System   2016-06-03          1
##    tri_age         parentcustomeridname tri_imaginecareenrollmentstatus
## 1       33          Dartmouth-Hitchcock                       167410011
## 2       25          Dartmouth-Hitchcock                       167410003
## 3       32        ProdSquad Testers iOS                       167410011
## 4       60          Dartmouth-Hitchcock                       167410011
## 5       48 The ImagineCare Testing Team                       167410011
## 6       80 The ImagineCare Testing Team                       167410003
## 7       61          Dartmouth-Hitchcock                       167410011
## 8       72          Dartmouth-Hitchcock                       167410011
## 9       33          Dartmouth-Hitchcock                       167410011
## 10      52          Dartmouth-Hitchcock                       167410011
##    address1_stateorprovince tri_imaginecareenrollmentemailsentdate
## 1                        NH                                4/25/2016
## 2                        NH                                 5/3/2016
## 3                        NH                                8/25/2016
## 4                        NH                                4/22/2016
## 5                        VA                                3/10/2016
## 6                        NH                                 3/1/2016
## 7                        NH                                 5/9/2016
## 8                        VT                                 5/3/2016
## 9                        NH                                7/12/2016
## 10                       NH                                 5/9/2016
##    tri_enrollmentcompletedate gender            tri_name
## 1                   4/26/2016      2 Activity Monitoring
## 2                    5/3/2016      2 Activity Monitoring
## 3                   8/25/2016      2        Hypertension
## 4                    5/9/2016      2            Diabetes
## 5                   5/16/2016      1 Activity Monitoring
## 6                        NULL      1 Activity Monitoring
## 7                   5/16/2016      2 Activity Monitoring
## 8                   5/13/2016      1 Activity Monitoring
## 9                   7/18/2016      1 Activity Monitoring
## 10                  5/10/2016      1            Diabetes
```

4.

Here is the link to the GitHub repository: Data_Wrangling_Project_and_Tasks