QBS181 Final

Mark Taylor

Setup

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

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)
```

(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)
```

(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)
```

(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)
```

(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.

```
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)
```

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)
```

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)
```

(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)
```

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))
```

(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()
```

We can also output 10 random rows from these patients.

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

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()
```

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)
```

4.

Here is the link to the GitHub repository: Data_Wrangling_Project_and_Tasks