Coursework Project

Erika Sim

11/29/2021

R Markdown for the 5 questions

The following code chunks helped me to answer the five different queries shown in the PDF given to us. I mainly used DBI to answer the queries.

First, I had to activate my packages by opening up my library for both DBI and dplyr. I am focusing only on DBI to get the queries from the database while using dplyr for certain functions such as piping.

I also set my working directory to the folder that has my data inside.

```
library(DBI)
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(ggplot2)
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.1.2

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
library(RColorBrewer)
colours12 <- brewer.pal(n = 12, name = 'Paired')
memory.limit(size=56000)
```

```
## [1] 56000
```

```
setwd("C:\\Users\\Admin\\Documents\\SIM Coursework Final\\Assignment 3 practice\\Data
↪  Expo 2009")
```

This next portion allows me to connect to the database I have created with the 4 different tables (airports, carriers, ontime, planes) named 'airline_r.db' and I assigned a variable to it known as 'conn':

```
conn <- dbConnect(RSQLite::SQLite(), "airline_r.db")
```

The following codes are to answer all questions:

Question 1

When is the best time of day, day of the week, and time of year to fly to minimize delays?

I split the question into three parts to write my code: best time of day, day of the week and time of year.

Best time of day

```
best_time_of_day <- dbGetQuery(conn, "
                               SELECT
CASE
WHEN CRSDepTime >= 0000 AND CRSDepTime <= 0159 THEN '0000 to 0159'
WHEN CRSDepTime >= 0200 AND CRSDepTime <= 0359 THEN '0200 to 0359'
WHEN CRSDepTime >= 0400 AND CRSDepTime <= 0559 THEN '0400 to 0559'
WHEN CRSDepTime >= 0600 AND CRSDepTime <= 0759 THEN '0600 to 0759'
WHEN CRSDepTime >= 0800 AND CRSDepTime <= 0959 THEN '0800 to 0959'
WHEN CRSDepTime >= 1000 AND CRSDepTime <= 1159 THEN '1000 to 1159'
WHEN CRSDepTime >= 1200 AND CRSDepTime <= 1359 THEN '1200 to 1359'
WHEN CRSDepTime >= 1400 AND CRSDepTime <= 1559 THEN '1400 to 1559'
WHEN CRSDepTime >= 1600 AND CRSDepTime <= 1759 THEN '1600 to 1759'
WHEN CRSDepTime >= 1800 AND CRSDepTime <= 1959 THEN '1800 to 1959'
WHEN CRSDepTime >= 2000 AND CRSDepTime <= 2159 THEN '2000 to 2159'
WHEN CRSDepTime >= 2200 AND CRSDepTime <= 2359 THEN '2200 to 2359'
END AS time,
round(AVG(DepDelay), 3) as avg_delay
FROM ontime
GROUP BY
CASE
WHEN CRSDepTime >= 0000 AND CRSDepTime <= 0159 THEN '0000 to 0159'
WHEN CRSDepTime >= 0200 AND CRSDepTime <= 0359 THEN '0200 to 0359'
```

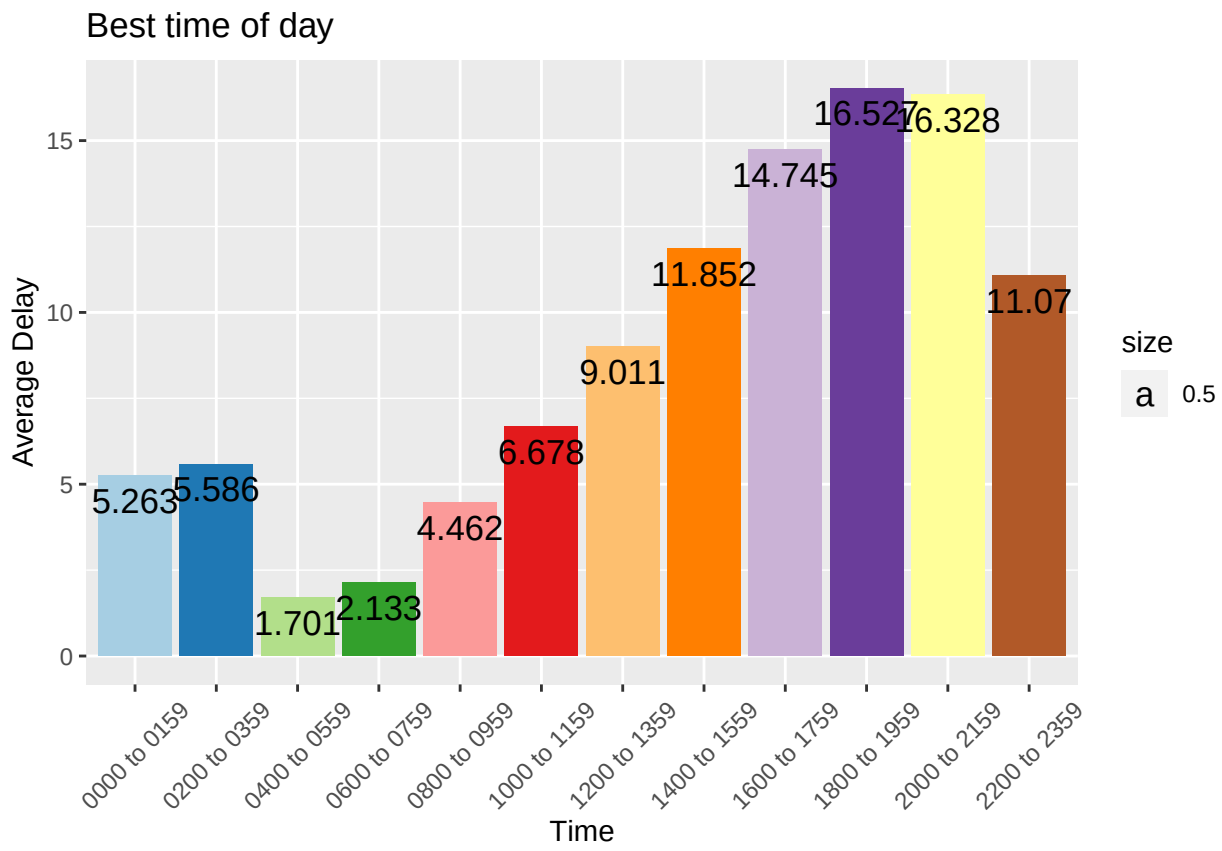
```
WHEN CRSDepTime >= 0400 AND CRSDepTime <= 0559 THEN '0400 to 0559'
WHEN CRSDepTime >= 0600 AND CRSDepTime <= 0759 THEN '0600 to 0759'
WHEN CRSDepTime >= 0800 AND CRSDepTime <= 0959 THEN '0800 to 0959'
WHEN CRSDepTime >= 1000 AND CRSDepTime <= 1159 THEN '1000 to 1159'
WHEN CRSDepTime >= 1200 AND CRSDepTime <= 1359 THEN '1200 to 1359'
WHEN CRSDepTime >= 1400 AND CRSDepTime <= 1559 THEN '1400 to 1559'
WHEN CRSDepTime >= 1600 AND CRSDepTime <= 1759 THEN '1600 to 1759'
WHEN CRSDepTime >= 1800 AND CRSDepTime <= 1959 THEN '1800 to 1959'
WHEN CRSDepTime >= 2000 AND CRSDepTime <= 2159 THEN '2000 to 2159'
WHEN CRSDepTime >= 2200 AND CRSDepTime <= 2359 THEN '2200 to 2359'
END
")
```

```
is.data.frame(best_time_of_day)
```

```
## [1] TRUE
```

I was checking to ensure that it was shown as a data frame as I had an error previously and thought it had to do with the formatting of my result.

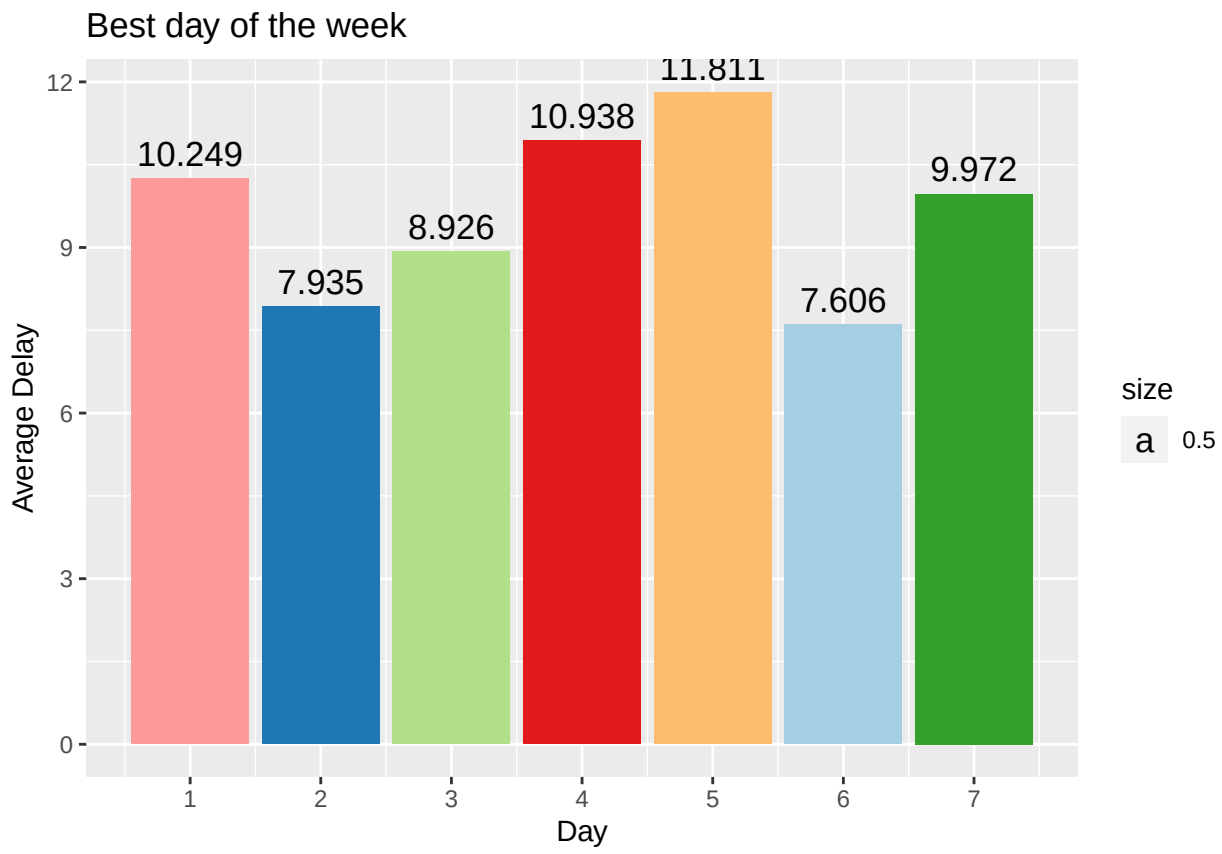
```
ggplot(best_time_of_day, aes(x = time, y = avg_delay)) +
  geom_bar(fill = colours12 , stat = "identity") +
  ggtitle("Best time of day") +
  xlab("Time") +
  ylab("Average Delay") +
  geom_text(aes(label= avg_delay,
                size = 0.5,
                vjust = 1.5)) +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=0.5))
```



From this graph, we can tell that the best time of day to fly would be between 0400 to 0559 hours.

Best day of week

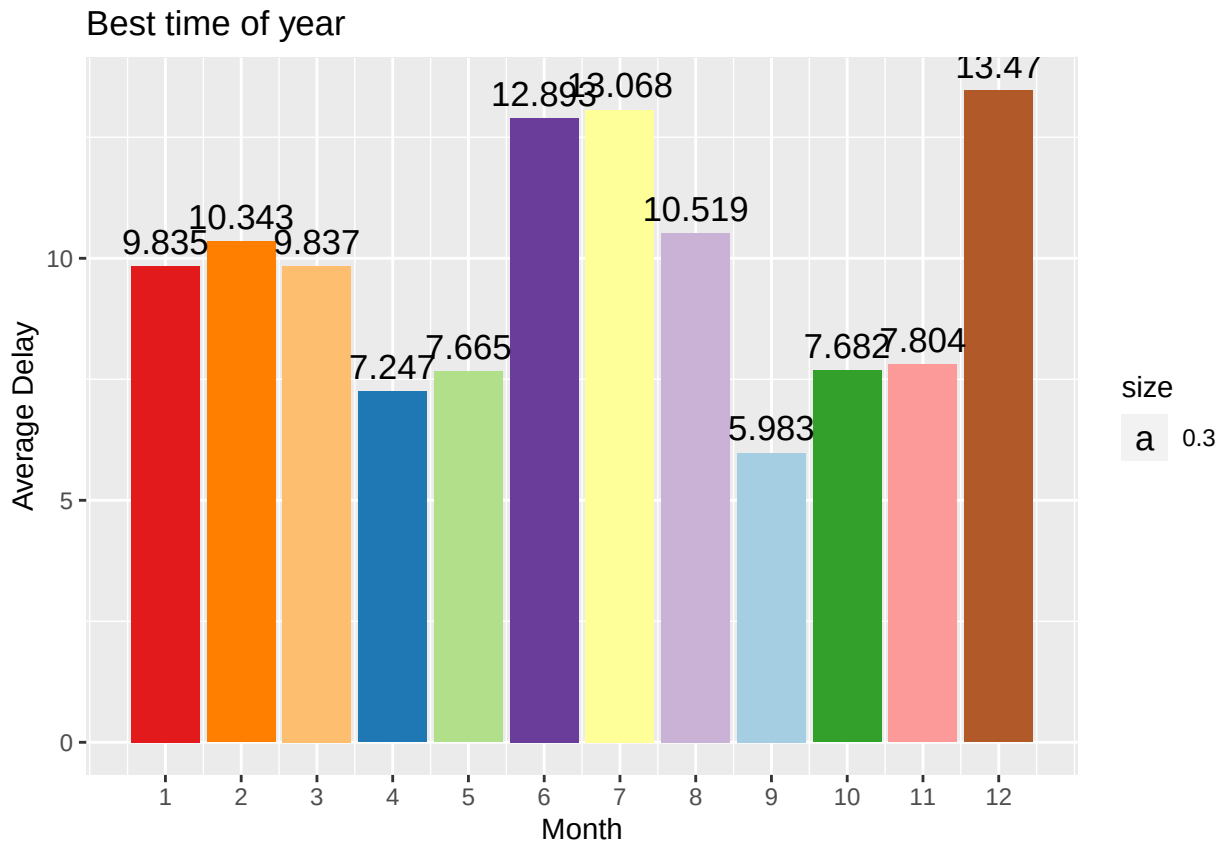
```
best_day_of_week <- dbGetQuery(conn, "
                               SELECT DayofWeek as day, round(AVG(DepDelay), 3) as
↪  avg_delay
                               FROM ontime
                               GROUP BY day
                               ORDER BY avg_delay
                               ")

colours7 <- brewer.pal(n = 7, name = 'Paired')
ggplot(best_day_of_week, aes(x = day, y = avg_delay)) +
  geom_bar(fill = colours7, stat= "identity") +
  scale_x_continuous(breaks = seq(1, 7, by = 1)) +
  ggtitle("Best day of the week") +
  xlab("Day") +
  ylab("Average Delay") +
  geom_text(aes(label= avg_delay,
            size = 0.5,
            vjust = -0.5))
```



From this graph, it is clearly shown that the best day of week to fly would be Day 2 (Tuesday).

Best time of year

```
best_time_of_year <- dbGetQuery(conn, "
                                SELECT Month as month, round(avg(DepDelay), 3) as
↪  avg_delay
                                FROM ontime
                                GROUP BY month
                                ORDER BY avg_delay
                                ")


ggplot(best_time_of_year, aes(x = month, y = avg_delay)) +
  geom_bar(fill = colours12, stat= "identity") +
  ggtitle("Best time of year") +
  xlab("Month") +
  ylab("Average Delay") +
  scale_x_continuous(breaks = seq(1, 12, by = 1)) +
  geom_text(aes(label= avg_delay,
                size = 0.3,
                vjust = -0.5,
                ))
```



From this graph, the best time of year to fly would be Month 1 (January).

Question 2

Do older planes suffer more delays?

I started off by finding the range of years where the planes were manufactured with the following code,

```
years_manufactured <- dbGetQuery(conn, "
                                  SELECT DISTINCT year
                                  FROM planes
                                  ORDER BY year
                                  ")
```

After which I went to find the the median in this range. The assumption I made was that any years later than the median are considered as an older planes. Those above the median year are considered the newer planes.

```
median(years_manufactured$year)
```

```
## [1] "1983"
```

```
older_planes <- dbGetQuery(conn, "
                           SELECT planes.year as planes_year, AVG(ontime.DepDelay) as
↪  avg_delay
                           FROM planes JOIN ontime USING (tailnum)
                           WHERE planes.year < 1983
                           GROUP BY planes_year
                           ORDER BY planes_year
                           ")

older_planes
```

```
##    planes_year avg_delay
## 1               11.586579
## 2         0000  6.038041
## 3         1956 11.485807
## 4         1957  5.455793
## 5         1959 10.605189
## 6         1962 10.081054
## 7         1963 11.168934
## 8         1964 10.966292
## 9         1965 10.717848
## 10        1966 10.478088
## 11        1967  6.017880
## 12        1968  6.595291
## 13        1969  6.006941
## 14        1970  6.324426
## 15        1971  6.104369
## 16        1972 10.394017
## 17        1973  7.085012
## 18        1974 10.880635
## 19        1975  9.432789
## 20        1976  8.779498
## 21        1977  8.356080
## 22        1978  7.640381
## 23        1979  8.831946
## 24        1980  9.397545
## 25        1982 12.562446
```

The result returned had two rows that were invalid or empty so I had to remove them to have a better outlook on the result. I did so by filling the row with NA first then removing the rows from the data frame.

```
older_planes[c(1,2),] <- NA
older_planes <- na.omit(older_planes)
```

I did the same for newer planes. Only difference is that the condition is different since it had to include the median year as shown:

```
newer_planes <- dbGetQuery(conn, "
                           SELECT planes.year as planes_year, AVG(ontime.DepDelay) as
↪  avg_delay
                           FROM planes JOIN ontime USING (tailnum)
                           WHERE planes.year >= 1983
```

```
                        GROUP BY planes_year
                        ORDER BY planes_year
                        ")

newer_planes
```

```
##    planes_year avg_delay
## 1         1983 10.139078
## 2         1984 10.552660
## 3         1985 10.485374
## 4         1986 10.583532
## 5         1987  9.941493
## 6         1988 10.379813
## 7         1989  9.922070
## 8         1990 10.137951
## 9         1991 10.299732
## 10        1992  9.871065
## 11        1993  9.328930
## 12        1994 10.422748
## 13        1995  9.211569
## 14        1996  9.443955
## 15        1997 11.518505
## 16        1998  9.400405
## 17        1999  9.727287
## 18        2000  9.199655
## 19        2001  8.313887
## 20        2002  9.712537
## 21        2003  9.874497
## 22        2004 10.621614
## 23        2005 10.903154
## 24        2006 11.043153
## 25        2007 10.646598
## 26        2008 11.602811
## 27        None 10.550867
```

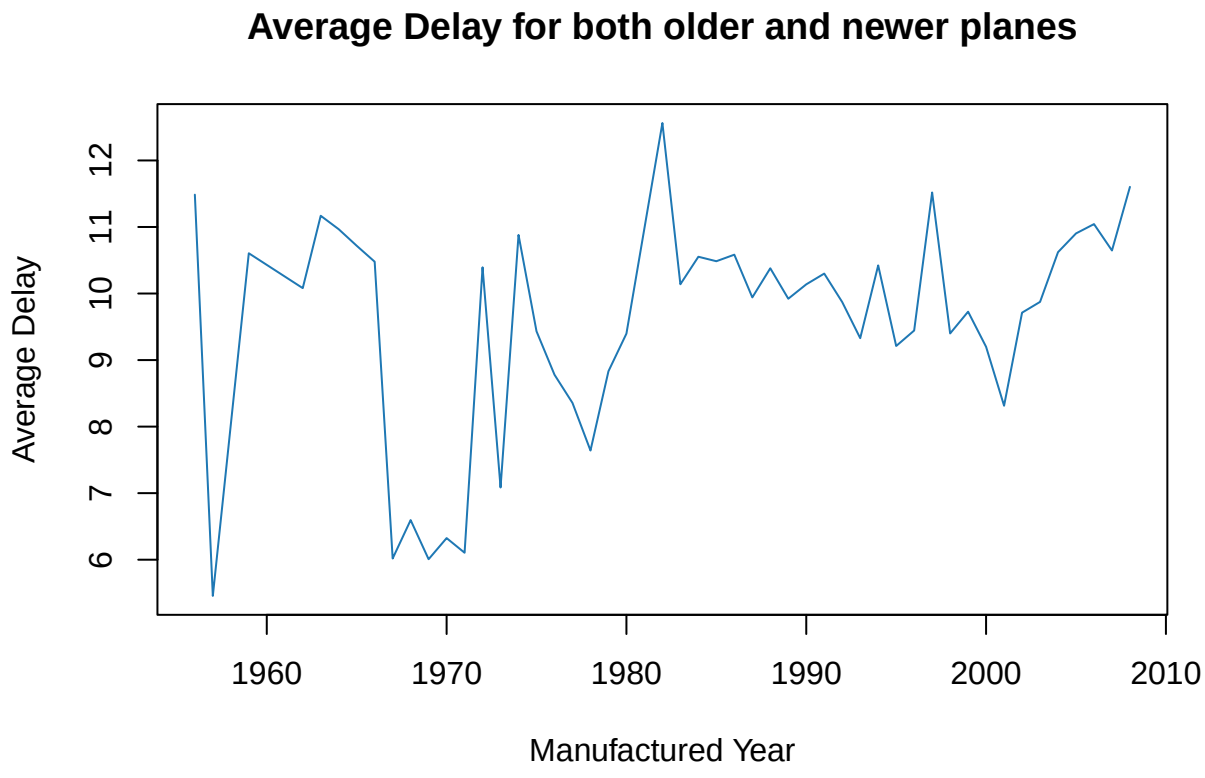
```
newer_planes[27,] <- NA
na.omit(newer_planes)
```

```
##    planes_year avg_delay
## 1         1983 10.139078
## 2         1984 10.552660
## 3         1985 10.485374
## 4         1986 10.583532
## 5         1987  9.941493
## 6         1988 10.379813
## 7         1989  9.922070
## 8         1990 10.137951
## 9         1991 10.299732
## 10        1992  9.871065
## 11        1993  9.328930
## 12        1994 10.422748
## 13        1995  9.211569
```

```
## 14         1996  9.443955
## 15         1997 11.518505
## 16         1998  9.400405
## 17         1999  9.727287
## 18         2000  9.199655
## 19         2001  8.313887
## 20         2002  9.712537
## 21         2003  9.874497
## 22         2004 10.621614
## 23         2005 10.903154
## 24         2006 11.043153
## 25         2007 10.646598
## 26         2008 11.602811
```

After both data frames were done, I went to plot a line graph for both older and newer planes.

```
planes <- rbind(older_planes, newer_planes)

colours<-brewer.pal(n = 12, name = "Paired")

plot(planes$planes_year, planes$avg_delay, type = "l",
     col=colours[2], xlab = 'Manufactured Year', ylab = 'Average Delay', main = 'Average
     ↪  Delay for both older and newer planes')
```



To justify my observation that older planes had lesser average delays, I calculated their mean.

```
mean_delay_for_older_planes <- round(mean(older_planes$avg_delay, na.rm = TRUE), 2)
mean_delay_for_newer_planes <- round(mean(newer_planes$avg_delay, na.rm = TRUE), 2)

mean_delay_for_older_planes
```

```
## [1] 8.93
```

```
mean_delay_for_newer_planes
```

```
## [1] 10.13
```

```
print(paste("Since the average delay of the older and newer planes are",
↪  round(mean_delay_for_older_planes,2), "and", round(mean_delay_for_newer_planes,2),
           "respectively, this shows that older planes does not suffer more delays
           ↪  compared to the newer ones"))
```

```
## [1] "Since the average delay of the older and newer planes are 8.93 and 10.13 respectively, this sho
```

Question 3

How does the number of people flying between different locations change over time?

I started off by assuming the number of trips equals to more people in that flight to that particular state. Therefore, for each time there is a flight to that state, it will count as total trips taken by people.

```
in2004 <- dbGetQuery(conn, "
                     SELECT ontime.Year as year, airports.state as states, COUNT(*)/1000
↪  as total_trips
                     FROM ontime JOIN airports ON ontime.dest = airports.iata
                     WHERE year = 2004
                     GROUP BY states
                     ORDER BY year
                     ");

in2005 <- dbGetQuery(conn, "
                     SELECT ontime.Year as year, airports.state as states, COUNT(*)/1000
↪  as total_trips
                     FROM ontime JOIN airports ON ontime.dest = airports.iata
                     WHERE year = 2005
                     GROUP BY states
                     ORDER BY year
                     ");

in2006 <- dbGetQuery(conn, "
                     SELECT ontime.Year as year, airports.state as states, COUNT(*)/1000
↪  as total_trips
                     FROM ontime JOIN airports ON ontime.dest = airports.iata
                     WHERE year = 2006
                     GROUP BY states
                     ORDER BY year
```

```
                      ");

in2007 <- dbGetQuery(conn, "
                      SELECT ontime.Year as year, airports.state as states, COUNT(*)/1000
↪  as total_trips
                      FROM ontime JOIN airports ON ontime.dest = airports.iata
                      WHERE year = 2007
                      GROUP BY states
                      ORDER BY year
                      ");

in2008 <- dbGetQuery(conn, "
                      SELECT ontime.Year as year, airports.state as states, COUNT(*)/1000
↪  as total_trips
                      FROM ontime JOIN airports ON ontime.dest = airports.iata
                      WHERE year = 2008
                      GROUP BY states
                      ORDER BY year
                      ");
```

The following is to remove NA values as there were 12 unknowns in the database with NULL state.

```
upd2004 <- na.omit(in2004)
upd2005 <- na.omit(in2005)
upd2006 <- na.omit(in2006)
upd2007 <- na.omit(in2007)
upd2008 <- na.omit(in2008)
```

Next, I removed the state of DE in 2006 and 2007 since other years did not have any from there.

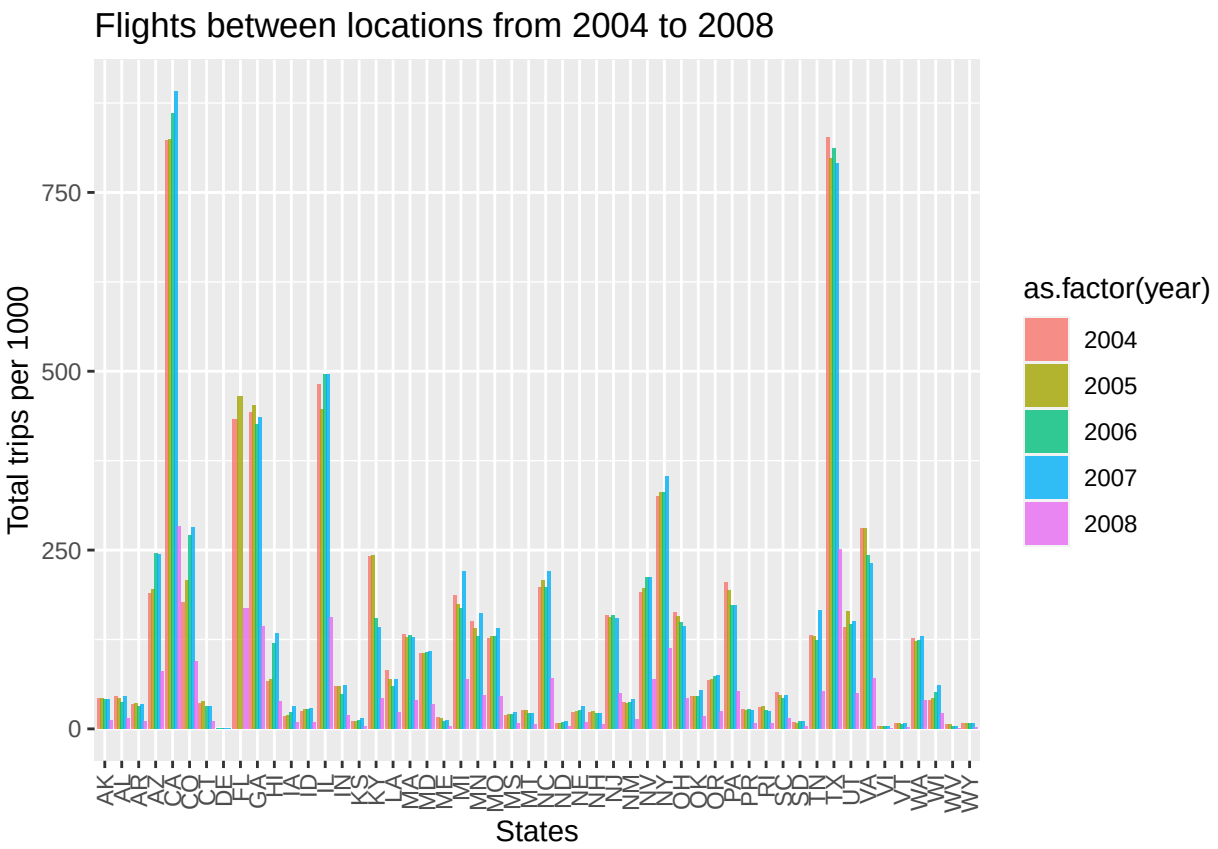
```
upd2006 <- upd2006[-c(9),]
upd2007 <- upd2007[-c(9),]
```

R binded the above data frames together.

```
overallchanges <- rbind(upd2004, upd2005, upd2006, upd2007, upd2008)
```

Lastly, I plotted a bar graph that were side by side for each state to see the flight changes over the 5 years for each state.

```
ggplot(overallchanges, aes(x = states, y = total_trips, fill = as.factor(year))) +
  geom_bar(position = "dodge", stat="identity", alpha = 0.8) +
  ggtitle("Flights between locations from 2004 to 2008") +
  xlab("States") +
  ylab("Total trips per 1000") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Observing from the graph, we can tell that there was mostly a huge drop from 2007 to 2008 due to high fuel prices.

Question 4

Can you detect cascading failures as delays in one airport create delays in others?

I first checked which year had the most number of inbound flights.

```
each_year <- dbGetQuery(conn, "
                        SELECT Year as year, COUNT(*) as total_flights
                        FROM ontime
                        GROUP BY year
                        ORDER BY total_flights DESC
                        ")
```

Next, I counted the number of arrival delays to a destination between two airports where the Origin is from the same airport. This would allow the data to be cleaner and better for comparison on the relevant airports we should focus on. This also creates the edges for Network visualization.

```
relation <- dbGetQuery(conn, "
                        SELECT Origin as origin, Dest as destination, COUNT(ArrDelay)/100
↪  as delayed_arrflights
                        FROM ontime
                        WHERE year = 2007 AND ArrDelay > 0
```

```
                        GROUP BY origin, destination
                        HAVING delayed_arrflights >= 15
                        ")
```

I went on to create the nodes by finding the average delay in each airport from the above 'relation' variable.

```
delay <- dbGetQuery(conn, "
                    SELECT Origin as origin, round(avg(DepDelay), 2) as avg_depdelay
                    FROM ontime JOIN airports ON ontime.dest = airports.iata
                    WHERE Year = 2007 AND DepDelay > 0
                    GROUP BY origin
                    ")
```

The following codes were for me to check whether the vertex names in edge list are listed in nodes ID column.

```
relation[which(! relation$origin %in% delay$origin),]
```

```
## [1] origin             destination        delayed_arrflights
## <0 rows> (or 0-length row.names)
```

```
relation[which(! relation$destination %in% delay$origin),]
```

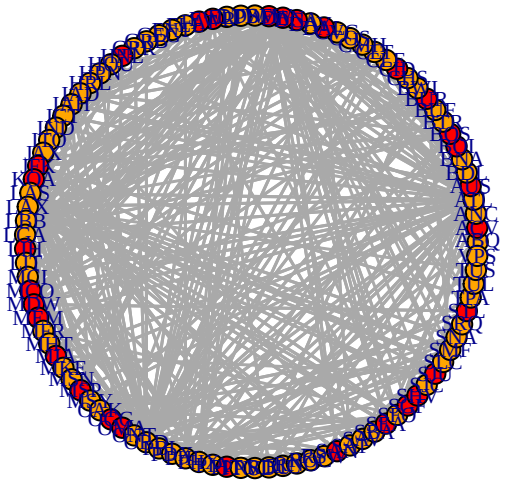
```
## [1] origin             destination        delayed_arrflights
## <0 rows> (or 0-length row.names)
```

Lastly, I plotted out the Network with the following code chunk:

```
net <- graph_from_data_frame(d = relation, vertices = delay, directed = T)
net.new <- delete.vertices(net , which(degree(net)==0))

summary(delay)
```

```
##     origin           avg_depdelay
##  Length:303         Min.   : 8.00
##  Class :character   1st Qu.:29.93
##  Mode  :character   Median :37.31
##                     Mean   :37.81
##                     3rd Qu.:44.26
##                     Max.   :87.75
```

```
circle <- layout_in_circle(net.new)
plot(net.new, edge.arrow.size = 0.1, vertex.size = 9,
     edge.width = 1.5,
     vertex.label.cex = 0.7, vertex.label.size = 0.1,
     vertex.color = ifelse(delay$avg_depdelay>45, "red", "orange"), vertex.frame.color =
     ↪  "black",
     layout = circle)
```



From here, we can see the different concentration of the number of arrival delays to a destination with some of them being colored as red because of high average departure delay of more than 45 minutes.

I further look into the matter by observing the scheduled timings for a particular origin. I had to find the average departure delay of all origin's to DFW and see which had the highest.

```
delays_to_DFW <- dbGetQuery(conn, "
                            SELECT Dest as destination, Origin as origin,
↪  ROUND(AVG(DepDelay), 2) as avg_delay
                            FROM ontime
                            WHERE destination = 'DFW'
                            GROUP BY destination, origin
                            ORDER BY avg_delay DESC
                            ")
```

The result showed MLB.

After which I queried from the database to filter off the respective origins, MLB and DFW, for a better observation on the schedules.

```
departure_delays_from_MLB <- dbGetQuery(conn, "
                             SELECT Origin as origin, Dest as destination, CRSDepTime,
↪  Deptime, CRSArrTime, ArrTime, DepDelay
                             FROM ontime
                             WHERE origin = 'MLB'
                             GROUP BY destination, origin
```

```
                            ORDER BY CRSDepTime
                            ")

departure_delays_from_DFW <- dbGetQuery(conn, "
                            SELECT Origin as origin, Dest as destination, CRSDepTime,
↪  DepTime, DepDelay
                            FROM ontime
                            WHERE origin = 'DFW' AND CRSDepTime >= 2152 AND DepDelay > 0
                            GROUP BY destination
                            ORDER BY CRSDepTime
                            ")

departure_delays_from_MLB
```

```
##   origin destination CRSDepTime DepTime CRSArrTime ArrTime DepDelay
## 1    MLB         DCA        650     645        858     840       -5
## 2    MLB         IAD        700     655        917     855       -5
## 3    MLB         ATL       1020    1005       1203    1143      -15
## 4    MLB         CVG       1140    1140       1402    1402        0
## 5    MLB         LGA       1420    1410       1700    1705      -10
## 6    MLB         MCO       1720    1720       1750    1805        0
## 7    MLB         DFW       1800    2007       2001    2152      127
## 8    MLB         JFK       1840    1904       2110    2138       24
```

```
departure_delays_from_DFW
```

```
##    origin destination CRSDepTime DepTime DepDelay
## 1     DFW         MSY       2152    2154        2
## 2     DFW         OKC       2152    2220       28
## 3     DFW         RNO       2152    2200        8
## 4     DFW         SFO       2152    2217       25
## 5     DFW         SMF       2152    2206       14
## 6     DFW         LAS       2153      17      144
## 7     DFW         MFE       2154    2219       25
## 8     DFW         OAK       2154    2203        9
## 9     DFW         ONT       2154    2212       18
## 10    DFW         STL       2154    2219       25
## 11    DFW         SEA       2155    2325       90
## 12    DFW         MCO       2156    2346      110
## 13    DFW         PHX       2156    2159        3
## 14    DFW         TUS       2156    2228       32
## 15    DFW         ELP       2157    2326       89
## 16    DFW         SAN       2157    2207       10
## 17    DFW         COS       2158    2244       46
## 18    DFW         TUL       2158    2229       31
## 19    DFW         IAH       2200    2210       10
## 20    DFW         ORD       2200    2205        5
## 21    DFW         SAT       2200    2229       29
## 22    DFW         AUS       2201    2238       37
## 23    DFW         LAX       2201    2334       93
## 24    DFW         PDX       2202      43      161
## 25    DFW         ABQ       2207    2233       26
```

```
## 26    DFW         DEN       2210    2237       27
## 27    DFW         MCI       2210    2244       34
## 28    DFW         XNA       2335     229      174
```

The schedule from MLB shows the arrival delay at 2152 hours to DFW which could have possibly caused a departure delay from DFW to MSY, DFW to OKC, etc which was pushed back to 2154 hours and 2220 hours. A 2 and 28 minutes delay on DFW side.

Question 5

Use the available variables to construct a model that predicts delays.

```
library(mlr3)
library(mlr3learners)
```

```
## Warning: package 'mlr3learners' was built under R version 4.1.2
```

```
library(mlr3pipelines)
```

```
## Warning: package 'mlr3pipelines' was built under R version 4.1.2
```

```
library(mlr3tuning)
```

```
## Warning: package 'mlr3tuning' was built under R version 4.1.2
```

```
## Loading required package: paradox
```

```
## Warning: package 'paradox' was built under R version 4.1.2
```

```
library(mlr3viz)
```

```
## Warning: package 'mlr3viz' was built under R version 4.1.2
```

First checked which Delay variable to use.

```
avg_delay <- dbGetQuery(conn, "
                        SELECT Year as year, ROUND(AVG(DepDelay), 2) as avg_depdelay,
↪  ROUND(AVG(ArrDelay), 2) as avg_arrdelay
                        FROM ontime
                        WHERE year = 2007
                        ")
```

Top 5 Origins to use for Machine Learning.

```
origin <- dbGetQuery(conn, "
                     SELECT Origin as origin, ROUND(AVG(DepDelay), 2) as avg_delay
                     FROM ontime
                     WHERE year = 2007 AND DepDelay > 0
                     GROUP BY origin
                     ORDER BY avg_delay DESC
                     LIMIT 5")
```

Loading data and cleaning it.

```
data <- dbGetQuery(conn, "
                   SELECT *
                   FROM ontime
                   WHERE year = 2007 AND DepDelay > 0
                   ")

ndata <- filter(data, Origin == "CMX" | Origin == "ACK" | Origin == "ALO" | Origin ==
↪  "SCE" | Origin == "MCN")

ndata$Origin <- factor(ndata$Origin)
ndata$Dest <- factor(ndata$Dest)
ndata$TailNum <- factor(ndata$TailNum)
```

Setting task and training/test sets.

```
n <- nrow(ndata)
set.seed(10)
train_set <- sample(n, round(0.6 * n))
test_set <- setdiff(1:n, train_set)

# setting up task
task <- TaskRegr$new('delay', backend = ndata, target = 'DepDelay')
task$select(c('CRSDepTime', 'DepTime', 'Dest', 'Origin', 'TailNum'))
task
```

```
## <TaskRegr:delay> (781 x 6)
## * Target: DepDelay
## * Properties: -
## * Features (5):
##   - fct (3): Dest, Origin, TailNum
##   - int (2): CRSDepTime, DepTime
```

```
measure <- msr('regr.mse')
```

Setting up encoder and tuner

```
fencoder <- po("encode", method="treatment",
               affect_columns=selector_type("factor"))

# Some methods require tuning the hyperparameters, and we will later use the following:
tuner <- tnr('grid_search')
terminator <- trm('evals', n_evals = 20)
```

The following is the extension of the results with two different regression models:

Random Forests

```
learner_rf <- lrn('regr.ranger')
learner_rf$param_set$values <- list(min.node.size = 4)
gr_rf <- po('scale') %>>%
  po('imputemean') %>>%
  po(learner_rf)
glrn_rf <- GraphLearner$new(gr_rf)
tune_ntrees <- ParamSet$new (list(
  ParamInt$new('regr.ranger.num.trees', lower = 50, upper = 600)
))
at_rf <- AutoTuner$new(
  learner = glrn_rf,
  resampling = rsmp('cv', folds = 3),
  measure = measure,
  search_space = tune_ntrees,
  terminator = terminator,
  tuner = tuner
)
at_rf$train(task, row_ids = train_set)
```

```
## INFO  [12:03:54.688] [bbotk] Starting to optimize 1 parameter(s) with '<TunerGridSearch>' and '<Term
## INFO  [12:03:54.714] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:54.752] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:54.810] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:55.495] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:55.791] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:55.912] [mlr3] Finished benchmark
## INFO  [12:03:55.938] [bbotk] Result of batch 1:
## INFO  [12:03:55.940] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:55.940] [bbotk]                   172 2380.951             1.07
## INFO  [12:03:55.940] [bbotk]                                        uhash
## INFO  [12:03:55.940] [bbotk]  3e073d37-7a05-412a-bec8-20c9e11b94d8
## INFO  [12:03:55.941] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:55.972] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:55.977] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:56.111] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:56.254] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:56.376] [mlr3] Finished benchmark
## INFO  [12:03:56.400] [bbotk] Result of batch 2:
## INFO  [12:03:56.402] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:56.402] [bbotk]                   417 2467.885              0.4
## INFO  [12:03:56.402] [bbotk]                                        uhash
## INFO  [12:03:56.402] [bbotk]  3332bbb1-d91e-4067-b083-73dccf9078b6
## INFO  [12:03:56.403] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:56.428] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:56.433] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:56.540] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:56.650] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:56.762] [mlr3] Finished benchmark
## INFO  [12:03:56.786] [bbotk] Result of batch 3:
```

```
## INFO  [12:03:56.787] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:56.787] [bbotk]                      50 2782.296            0.31
## INFO  [12:03:56.787] [bbotk]                                      uhash
## INFO  [12:03:56.787] [bbotk]  dd6f11ce-50b8-449a-89fa-9d04cbd11470
## INFO  [12:03:56.788] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:56.814] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:56.818] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:56.945] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:57.073] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:57.203] [mlr3] Finished benchmark
## INFO  [12:03:57.233] [bbotk] Result of batch 4:
## INFO  [12:03:57.234] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:57.234] [bbotk]                      539 2558.582            0.34
## INFO  [12:03:57.234] [bbotk]                                      uhash
## INFO  [12:03:57.234] [bbotk]  d3c518ff-b2a9-4925-867e-2f0b207c39a9
## INFO  [12:03:57.235] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:57.261] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:57.265] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:57.394] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:57.522] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:57.654] [mlr3] Finished benchmark
## INFO  [12:03:57.678] [bbotk] Result of batch 5:
## INFO  [12:03:57.679] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:57.679] [bbotk]                      600 2426.267            0.36
## INFO  [12:03:57.679] [bbotk]                                      uhash
## INFO  [12:03:57.679] [bbotk]  6835598a-9d43-4eb1-937b-59ab065f7fd3
## INFO  [12:03:57.680] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:57.706] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:57.711] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:57.852] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:57.969] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:58.087] [mlr3] Finished benchmark
## INFO  [12:03:58.111] [bbotk] Result of batch 6:
## INFO  [12:03:58.112] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:58.112] [bbotk]                      294 2504.167            0.38
## INFO  [12:03:58.112] [bbotk]                                      uhash
## INFO  [12:03:58.112] [bbotk]  9f264f61-cf93-45c0-ac62-716b9e5d9531
## INFO  [12:03:58.113] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:58.139] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:58.144] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:58.266] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:58.378] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:58.492] [mlr3] Finished benchmark
## INFO  [12:03:58.516] [bbotk] Result of batch 7:
## INFO  [12:03:58.517] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:58.517] [bbotk]                      233 2435.582            0.32
## INFO  [12:03:58.517] [bbotk]                                      uhash
## INFO  [12:03:58.517] [bbotk]  a308dd88-b0ad-41a9-ab13-39d53d8a7d6d
## INFO  [12:03:58.518] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:58.544] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:58.548] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:58.668] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:58.790] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:58.909] [mlr3] Finished benchmark
```

```
## INFO  [12:03:58.933] [bbotk] Result of batch 8:
## INFO  [12:03:58.934] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:58.934] [bbotk]                   356 2464.712            0.34
## INFO  [12:03:58.934] [bbotk]                                       uhash
## INFO  [12:03:58.934] [bbotk]  aba72340-5993-4770-9746-cc7dc0435ee9
## INFO  [12:03:58.935] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:58.960] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:58.965] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:59.074] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:59.181] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:59.297] [mlr3] Finished benchmark
## INFO  [12:03:59.320] [bbotk] Result of batch 9:
## INFO  [12:03:59.321] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:59.321] [bbotk]                   111 2526.937            0.28
## INFO  [12:03:59.321] [bbotk]                                       uhash
## INFO  [12:03:59.321] [bbotk]  7f366cc3-bf56-42cd-bd21-06ca32a584f8
## INFO  [12:03:59.322] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:03:59.348] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:03:59.353] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:03:59.478] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:03:59.603] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:03:59.738] [mlr3] Finished benchmark
## INFO  [12:03:59.761] [bbotk] Result of batch 10:
## INFO  [12:03:59.762] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:03:59.762] [bbotk]                   478 2355.409            0.36
## INFO  [12:03:59.762] [bbotk]                                       uhash
## INFO  [12:03:59.762] [bbotk]  89c4a1c9-062b-43b8-bbdb-f0c9b71c5c91
## INFO  [12:03:59.766] [bbotk] Finished optimizing after 10 evaluation(s)
## INFO  [12:03:59.767] [bbotk] Result:
## INFO  [12:03:59.767] [bbotk]  regr.ranger.num.trees learner_param_vals  x_domain regr.mse
## INFO  [12:03:59.767] [bbotk]                   478         <list[3]> <list[1]> 2355.409
```

```
at_rf$predict(task, row_ids = test_set)$score()
```

```
## regr.mse
## 11509.83
```

Support Vector Machine

```
learner_svm <- lrn("regr.svm")

gr_svm <- po('imputemean', affect_columns=selector_type("numeric")) %>>%
  po('imputemode', affect_columns=selector_type(c("factor"))) %>>%
  fencoder %>>%
  po('scale') %>>%
  po(learner_svm)

glrn_svm <- GraphLearner$new(gr_svm)

glrn_svm$train(task, row_ids = train_set)
```

```
## Warning in svm.default(x = data, y = task$truth()): Variable(s) 'TailNum.80139E' and 'TailNum.80239E
## This happened PipeOp regr.svm's $train()
```

```
glrn_svm$predict(task, row_ids = test_set)$score()
```

```
## regr.mse
## 34188.83
```

Benchmarking the two models and visualised comparison with box plots.

```
# benchmarking the best model
set.seed(1)

lrn_list <- list(
  at_rf,
  glrn_svm
)

# Set the benchmark design and run the comparisons
bm_design <- benchmark_grid(task=task, resamplings=rsmp('cv', folds=3),
                            learners=lrn_list)
bmr <- benchmark(bm_design, store_models=TRUE)
```

```
## INFO  [12:04:00.792] [mlr3] Running benchmark with 6 resampling iterations
## INFO  [12:04:00.796] [mlr3] Applying learner 'imputemean.imputemode.encode.scale.regr.svm' on task '
## INFO  [12:04:01.426] [mlr3] Applying learner 'imputemean.imputemode.encode.scale.regr.svm' on task '
## INFO  [12:04:02.081] [mlr3] Applying learner 'scale.imputemean.regr.ranger.tuned' on task 'delay' (i
## INFO  [12:04:02.158] [bbotk] Starting to optimize 1 parameter(s) with '<TunerGridSearch>' and '<Term
## INFO  [12:04:02.160] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:02.186] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:02.190] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:02.318] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:02.453] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:02.581] [mlr3] Finished benchmark
## INFO  [12:04:02.602] [bbotk] Result of batch 1:
## INFO  [12:04:02.603] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:02.603] [bbotk]                   600 6557.452            0.38
## INFO  [12:04:02.603] [bbotk]                                  uhash
## INFO  [12:04:02.603] [bbotk]  e9ea81ad-b4fd-44d5-b7bf-4b9583612d31
## INFO  [12:04:02.604] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:02.630] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:02.634] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:02.749] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:02.868] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:02.980] [mlr3] Finished benchmark
## INFO  [12:04:03.004] [bbotk] Result of batch 2:
## INFO  [12:04:03.005] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:03.005] [bbotk]                   172 6370.518            0.32
## INFO  [12:04:03.005] [bbotk]                                  uhash
## INFO  [12:04:03.005] [bbotk]  4617eaee-2e8a-4210-84cd-57b9e4145e5f
## INFO  [12:04:03.006] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:03.032] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:03.036] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:03.162] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:03.288] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
```

```
## INFO  [12:04:03.419] [mlr3] Finished benchmark
## INFO  [12:04:03.443] [bbotk] Result of batch 3:
## INFO  [12:04:03.444] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:03.444] [bbotk]                   539 6288.826            0.34
## INFO  [12:04:03.444] [bbotk]                                     uhash
## INFO  [12:04:03.444] [bbotk]  01e42f2b-a94b-4455-a9f8-b12e25f00c5e
## INFO  [12:04:03.445] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:03.471] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:03.476] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:03.595] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:03.714] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:03.844] [mlr3] Finished benchmark
## INFO  [12:04:03.868] [bbotk] Result of batch 4:
## INFO  [12:04:03.869] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:03.869] [bbotk]                   356  6293.86            0.35
## INFO  [12:04:03.869] [bbotk]                                     uhash
## INFO  [12:04:03.869] [bbotk]  9c01024f-eae6-4bcd-836f-5083024f9526
## INFO  [12:04:03.870] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:03.896] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:03.901] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:04.018] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:04.133] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:04.261] [mlr3] Finished benchmark
## INFO  [12:04:04.285] [bbotk] Result of batch 5:
## INFO  [12:04:04.286] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:04.286] [bbotk]                   294 6353.782            0.33
## INFO  [12:04:04.286] [bbotk]                                     uhash
## INFO  [12:04:04.286] [bbotk]  f9be53f5-d2f0-41f5-8705-3859b08b6b15
## INFO  [12:04:04.287] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:04.313] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:04.318] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:04.444] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:04.570] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:04.704] [mlr3] Finished benchmark
## INFO  [12:04:04.730] [bbotk] Result of batch 6:
## INFO  [12:04:04.731] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:04.731] [bbotk]                   478 6287.698            0.35
## INFO  [12:04:04.731] [bbotk]                                     uhash
## INFO  [12:04:04.731] [bbotk]  faa90b5c-cd2f-48c8-ac99-d749a676c6b6
## INFO  [12:04:04.732] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:04.758] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:04.763] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:04.876] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:04.986] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:05.100] [mlr3] Finished benchmark
## INFO  [12:04:05.125] [bbotk] Result of batch 7:
## INFO  [12:04:05.133] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:05.133] [bbotk]                   111 6297.666            0.31
## INFO  [12:04:05.133] [bbotk]                                     uhash
## INFO  [12:04:05.133] [bbotk]  3fab3e6c-7a51-47ea-bb26-5023d810d4e5
## INFO  [12:04:05.135] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:05.160] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:05.164] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:05.274] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
```

```
## INFO  [12:04:05.386] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:05.498] [mlr3] Finished benchmark
## INFO  [12:04:05.523] [bbotk] Result of batch 8:
## INFO  [12:04:05.525] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:05.525] [bbotk]                    50 6107.083            0.33
## INFO  [12:04:05.525] [bbotk]                                       uhash
## INFO  [12:04:05.525] [bbotk]  21fa5789-abec-4555-bf5e-fc031a84fff8
## INFO  [12:04:05.525] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:05.559] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:05.563] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:05.690] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:05.815] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:05.943] [mlr3] Finished benchmark
## INFO  [12:04:05.969] [bbotk] Result of batch 9:
## INFO  [12:04:05.970] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:05.970] [bbotk]                    417 6261.994            0.36
## INFO  [12:04:05.970] [bbotk]                                       uhash
## INFO  [12:04:05.970] [bbotk]  448ec650-93d5-4e4e-9d4c-dcb637153200
## INFO  [12:04:05.971] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:05.997] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:06.009] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:06.125] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:06.241] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:06.359] [mlr3] Finished benchmark
## INFO  [12:04:06.385] [bbotk] Result of batch 10:
## INFO  [12:04:06.386] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:06.386] [bbotk]                    233 6299.902            0.36
## INFO  [12:04:06.386] [bbotk]                                       uhash
## INFO  [12:04:06.386] [bbotk]  e1dc956a-9947-4239-8e7a-8b78b75caca9
## INFO  [12:04:06.390] [bbotk] Finished optimizing after 10 evaluation(s)
## INFO  [12:04:06.390] [bbotk] Result:
## INFO  [12:04:06.391] [bbotk]  regr.ranger.num.trees learner_param_vals  x_domain regr.mse
## INFO  [12:04:06.391] [bbotk]                     50          <list[3]> <list[1]> 6107.083
## INFO  [12:04:06.537] [mlr3] Applying learner 'scale.imputemean.regr.ranger.tuned' on task 'delay' (it
## INFO  [12:04:06.616] [bbotk] Starting to optimize 1 parameter(s) with '<TunerGridSearch>' and '<Termi
## INFO  [12:04:06.618] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:06.645] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:06.649] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:06.769] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:06.899] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:07.024] [mlr3] Finished benchmark
## INFO  [12:04:07.046] [bbotk] Result of batch 1:
## INFO  [12:04:07.047] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:07.047] [bbotk]                    356 3370.874            0.33
## INFO  [12:04:07.047] [bbotk]                                       uhash
## INFO  [12:04:07.047] [bbotk]  a3e7fe52-1260-4384-a76b-54b5bc539a17
## INFO  [12:04:07.048] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:07.075] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:07.080] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:07.219] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:07.358] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:07.490] [mlr3] Finished benchmark
## INFO  [12:04:07.514] [bbotk] Result of batch 2:
## INFO  [12:04:07.515] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
```

```
## INFO  [12:04:07.515] [bbotk]                    539 3363.766            0.39
## INFO  [12:04:07.515] [bbotk]                                      uhash
## INFO  [12:04:07.515] [bbotk]  f2bf3d9a-4a4c-4b90-83df-d8c9931ecc56
## INFO  [12:04:07.516] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:07.543] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:07.548] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:07.675] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:07.809] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:07.936] [mlr3] Finished benchmark
## INFO  [12:04:07.960] [bbotk] Result of batch 3:
## INFO  [12:04:07.961] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:07.961] [bbotk]                    417 3342.333            0.34
## INFO  [12:04:07.961] [bbotk]                                      uhash
## INFO  [12:04:07.961] [bbotk]  26556b82-8a84-4f1f-86e9-61fcc1a0f071
## INFO  [12:04:07.962] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:07.989] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:07.993] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:08.105] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:08.267] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:08.381] [mlr3] Finished benchmark
## INFO  [12:04:08.405] [bbotk] Result of batch 4:
## INFO  [12:04:08.406] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:08.406] [bbotk]                     50     3429            0.37
## INFO  [12:04:08.406] [bbotk]                                      uhash
## INFO  [12:04:08.406] [bbotk]  a61f7150-c1cc-45b6-96d6-df7e50c5a458
## INFO  [12:04:08.407] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:08.435] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:08.439] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:08.553] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:08.666] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:08.799] [mlr3] Finished benchmark
## INFO  [12:04:08.824] [bbotk] Result of batch 5:
## INFO  [12:04:08.825] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:08.825] [bbotk]                    111 3258.291            0.33
## INFO  [12:04:08.825] [bbotk]                                      uhash
## INFO  [12:04:08.825] [bbotk]  5b084d2e-18bf-4148-96ac-5538c89b35e9
## INFO  [12:04:08.826] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:08.852] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:08.856] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:08.984] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:09.108] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:09.249] [mlr3] Finished benchmark
## INFO  [12:04:09.277] [bbotk] Result of batch 6:
## INFO  [12:04:09.278] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:09.278] [bbotk]                    478 3252.846            0.39
## INFO  [12:04:09.278] [bbotk]                                      uhash
## INFO  [12:04:09.278] [bbotk]  6a18b2da-f40a-472b-9ccb-d4e88930b8de
## INFO  [12:04:09.279] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:09.305] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:09.309] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:09.425] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:09.540] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:09.655] [mlr3] Finished benchmark
## INFO  [12:04:09.688] [bbotk] Result of batch 7:
```

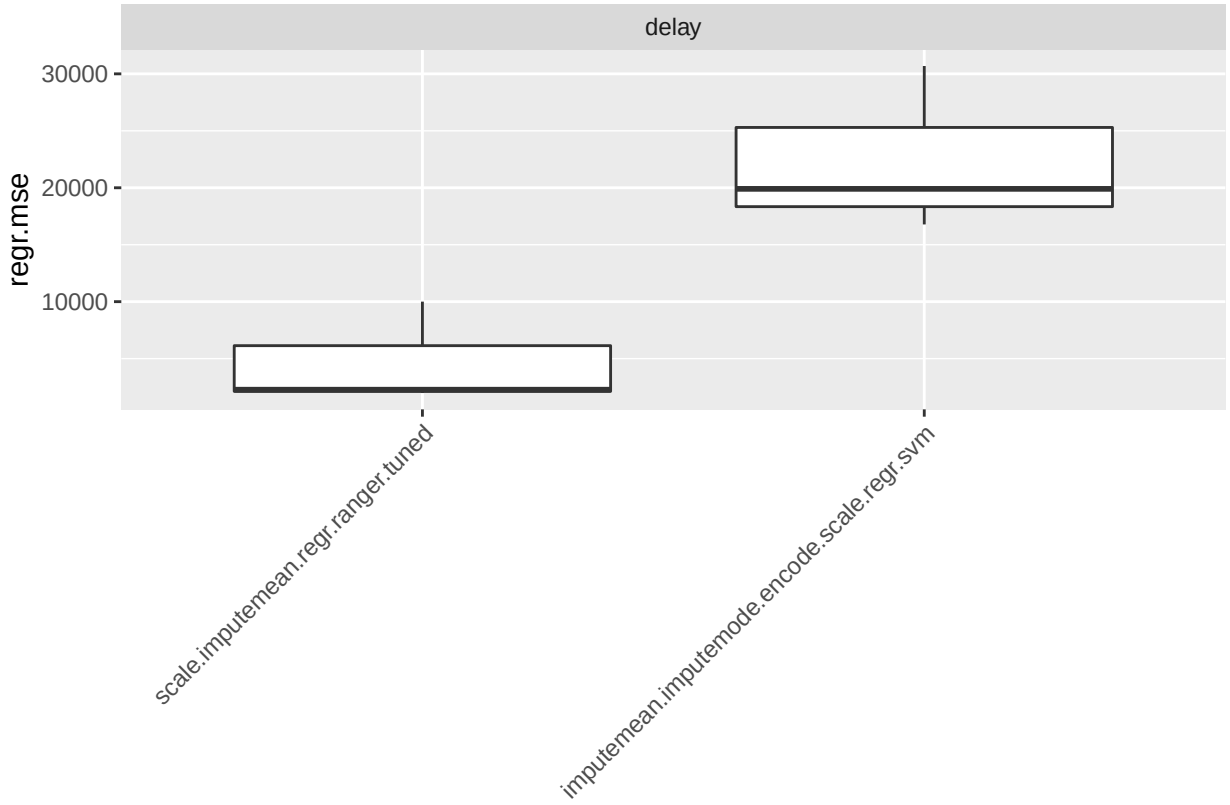
```
## INFO  [12:04:09.689] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:09.689] [bbotk]                   233 3309.567             0.33
## INFO  [12:04:09.689] [bbotk]                                      uhash
## INFO  [12:04:09.689] [bbotk]  df2620d1-d711-49d5-9d31-a8c20e4785ad
## INFO  [12:04:09.690] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:09.726] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:09.731] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:09.861] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:09.991] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:10.120] [mlr3] Finished benchmark
## INFO  [12:04:10.144] [bbotk] Result of batch 8:
## INFO  [12:04:10.145] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:10.145] [bbotk]                   600 3287.444             0.36
## INFO  [12:04:10.145] [bbotk]                                      uhash
## INFO  [12:04:10.145] [bbotk]  4e321c0f-28d0-4c18-8157-9f7c9d43174d
## INFO  [12:04:10.146] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:10.181] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:10.186] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:10.317] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:10.435] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:10.553] [mlr3] Finished benchmark
## INFO  [12:04:10.577] [bbotk] Result of batch 9:
## INFO  [12:04:10.578] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:10.578] [bbotk]                   294 3374.503             0.34
## INFO  [12:04:10.578] [bbotk]                                      uhash
## INFO  [12:04:10.578] [bbotk]  c0fefeb5-bc0b-45a7-9967-cce06322d778
## INFO  [12:04:10.579] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:10.606] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:10.610] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:10.744] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:10.858] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:10.972] [mlr3] Finished benchmark
## INFO  [12:04:10.996] [bbotk] Result of batch 10:
## INFO  [12:04:10.997] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:10.997] [bbotk]                   172 3126.669              0.3
## INFO  [12:04:10.997] [bbotk]                                      uhash
## INFO  [12:04:10.997] [bbotk]  ebac2b60-98a1-4df3-967f-c0ea04a385bc
## INFO  [12:04:11.001] [bbotk] Finished optimizing after 10 evaluation(s)
## INFO  [12:04:11.002] [bbotk] Result:
## INFO  [12:04:11.003] [bbotk]  regr.ranger.num.trees learner_param_vals  x_domain regr.mse
## INFO  [12:04:11.003] [bbotk]                   172          <list[3]> <list[1]> 3126.669
## INFO  [12:04:11.161] [mlr3] Applying learner 'imputemean.imputemode.encode.scale.regr.svm' on task '
## INFO  [12:04:11.782] [mlr3] Applying learner 'scale.imputemean.regr.ranger.tuned' on task 'delay' (i
## INFO  [12:04:11.877] [bbotk] Starting to optimize 1 parameter(s) with '<TunerGridSearch>' and '<Term
## INFO  [12:04:11.879] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:11.906] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:11.910] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:12.041] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:12.170] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:12.298] [mlr3] Finished benchmark
## INFO  [12:04:12.330] [bbotk] Result of batch 1:
## INFO  [12:04:12.331] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:12.331] [bbotk]                   600  7653.12             0.38
## INFO  [12:04:12.331] [bbotk]                                      uhash
```

```
## INFO  [12:04:12.331] [bbotk]  0cc45387-32f0-4edf-bfdd-4f6b62269104
## INFO  [12:04:12.333] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:12.365] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:12.370] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:12.480] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:12.592] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:12.703] [mlr3] Finished benchmark
## INFO  [12:04:12.727] [bbotk] Result of batch 2:
## INFO  [12:04:12.728] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:12.728] [bbotk]                    111 8028.733            0.32
## INFO  [12:04:12.728] [bbotk]                                  uhash
## INFO  [12:04:12.728] [bbotk]  2277c147-0aad-47fe-b26e-ec91dbb62831
## INFO  [12:04:12.729] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:12.767] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:12.773] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:12.886] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:12.996] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:13.106] [mlr3] Finished benchmark
## INFO  [12:04:13.130] [bbotk] Result of batch 3:
## INFO  [12:04:13.132] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:13.132] [bbotk]                     50 7891.584            0.29
## INFO  [12:04:13.132] [bbotk]                                  uhash
## INFO  [12:04:13.132] [bbotk]  c261a068-8efa-45fe-b09f-670961f4c0c3
## INFO  [12:04:13.132] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:13.159] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:13.163] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:13.300] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:13.417] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:13.536] [mlr3] Finished benchmark
## INFO  [12:04:13.560] [bbotk] Result of batch 4:
## INFO  [12:04:13.561] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:13.561] [bbotk]                    294 7839.924            0.33
## INFO  [12:04:13.561] [bbotk]                                  uhash
## INFO  [12:04:13.561] [bbotk]  599c20ce-4e03-44e8-8b71-2e66ac185a34
## INFO  [12:04:13.562] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:13.589] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:13.593] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:13.739] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:13.865] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:13.993] [mlr3] Finished benchmark
## INFO  [12:04:14.017] [bbotk] Result of batch 5:
## INFO  [12:04:14.018] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:14.018] [bbotk]                    539 7852.103            0.37
## INFO  [12:04:14.018] [bbotk]                                  uhash
## INFO  [12:04:14.018] [bbotk]  45d1a308-5388-48bc-9790-a8deea03ccfd
## INFO  [12:04:14.019] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:14.045] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:14.050] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:14.192] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:14.315] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:14.443] [mlr3] Finished benchmark
## INFO  [12:04:14.467] [bbotk] Result of batch 6:
## INFO  [12:04:14.468] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:14.468] [bbotk]                    478 7739.584            0.39
```

```
## INFO  [12:04:14.468] [bbotk]                                  uhash
## INFO  [12:04:14.468] [bbotk]  c760713f-8398-430f-ad91-ead87bac8eae
## INFO  [12:04:14.469] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:14.495] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:14.499] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:14.626] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:14.741] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:14.855] [mlr3] Finished benchmark
## INFO  [12:04:14.879] [bbotk] Result of batch 7:
## INFO  [12:04:14.880] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:14.880] [bbotk]                    172 7830.886            0.32
## INFO  [12:04:14.880] [bbotk]                                  uhash
## INFO  [12:04:14.880] [bbotk]  37412e73-839e-4667-92ac-3fce14e545b6
## INFO  [12:04:14.881] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:14.907] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:14.912] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:15.041] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:15.169] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:15.290] [mlr3] Finished benchmark
## INFO  [12:04:15.314] [bbotk] Result of batch 8:
## INFO  [12:04:15.315] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:15.315] [bbotk]                    356 7793.762            0.37
## INFO  [12:04:15.315] [bbotk]                                  uhash
## INFO  [12:04:15.315] [bbotk]  3f552a26-ffb1-43f2-bc6c-cd2483bc4f3b
## INFO  [12:04:15.316] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:15.342] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:15.347] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:15.462] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:15.595] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:15.712] [mlr3] Finished benchmark
## INFO  [12:04:15.737] [bbotk] Result of batch 9:
## INFO  [12:04:15.738] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:15.738] [bbotk]                    233 7695.874            0.35
## INFO  [12:04:15.738] [bbotk]                                  uhash
## INFO  [12:04:15.738] [bbotk]  27016e19-8df3-490a-9c9c-ff21734d5108
## INFO  [12:04:15.739] [bbotk] Evaluating 1 configuration(s)
## INFO  [12:04:15.766] [mlr3] Running benchmark with 3 resampling iterations
## INFO  [12:04:15.771] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 3/3
## INFO  [12:04:15.893] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 2/3
## INFO  [12:04:16.034] [mlr3] Applying learner 'scale.imputemean.regr.ranger' on task 'delay' (iter 1/3
## INFO  [12:04:16.157] [mlr3] Finished benchmark
## INFO  [12:04:16.181] [bbotk] Result of batch 10:
## INFO  [12:04:16.182] [bbotk]  regr.ranger.num.trees regr.mse runtime_learners
## INFO  [12:04:16.182] [bbotk]                    417 7645.239            0.38
## INFO  [12:04:16.182] [bbotk]                                  uhash
## INFO  [12:04:16.182] [bbotk]  5d008178-ef7a-42a6-a191-15c2a1c9dbf9
## INFO  [12:04:16.187] [bbotk] Finished optimizing after 10 evaluation(s)
## INFO  [12:04:16.187] [bbotk] Result:
## INFO  [12:04:16.188] [bbotk]  regr.ranger.num.trees learner_param_vals  x_domain regr.mse
## INFO  [12:04:16.188] [bbotk]                    417          <list[3]> <list[1]> 7645.239

## Warning in svm.default(x = data, y = task$truth()): Variable(s) 'TailNum.80129E' and 'TailNum.80139E
## This happened PipeOp regr.svm's $train()
```

```
## Warning in svm.default(x = data, y = task$truth()): Variable(s) 'TailNum.80239E' and 'TailNum.80309E
## This happened PipeOp regr.svm's $train()

## Warning in svm.default(x = data, y = task$truth()): Variable(s) 'TailNum.80359E' and 'TailNum.80409E
## This happened PipeOp regr.svm's $train()

## INFO  [12:04:16.344] [mlr3] Finished benchmark
```

```
# Visualise comparisons with boxplots
autoplot(bmr) + theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



From the graph, Random Forests has clear dominance over Support Vector Machine.