Data Cleaning and (preliminary) EDA

Optimizing HVAC Operation for Occupant Comfort and Energy Savings

Caleb Neale

Spring 2021

Load libraries

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(lubridate)
```

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

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

```r
library(fpp3)
```

```
## -- Attaching packages -------------------------------------------- fpp3 0.4.0 --

## v tsibble     1.0.0     v feasts      0.1.7
## v tsibbledata 0.2.0     v fable       0.3.0

## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

Import Data and convert to tibble

```r
read_and_clean <- function(csv_path){
  df <- read.csv(csv_path, sep=";", row.names = NULL)
  colnames(df) <- c("series", 'time','value')

  # NAs will be induced by following line, seems like this occurs when the value in the 'value' column
  df$value <- as.numeric(df$value)
  df<- df[-1,]
  df <- as_tibble(df)
  return(df)
}

co2 <- read_and_clean('co2.csv')
occupied_status <- read_and_clean('occupied_status.csv')
occupied_status$value <- as.factor(occupied_status$value)
supply_air_flow <- read_and_clean('supply_air_flow.csv')
supply_fan <- read_and_clean('supply_fan.csv')
supply_fan$value <- as.factor(supply_fan$value)
temperature <- read_and_clean('temperature.csv')
```

Check for NaN

```r
sum(is.na(occupied_status$value))
```

```
## [1] 0
```

```r
sum(is.na(co2$value))
```

```
## [1] 2136
```

```r
sum(is.na(supply_air_flow$value)) # lot of NAs (over 32000)
```

```
## [1] 32334
```

```r
sum(is.na(supply_fan$value))
```

```
## [1] 0
```

```r
sum(is.na(temperature$value)) # lot of NAs (over 15000)
```

```
## [1] 15664
```

Convert time data to datetime format

```r
convert_to_datetime <- function(df){
  df$time <- gsub("-04:00$", "-0400", df$time)
  df$time <- gsub("-05:00$", "-0500", df$time)
  df$time <- strptime(df$time, format ="%Y-%m-%dT%H:%M:%S%z")
  df$time <- as.POSIXct(df$time)
  return(df)
}

co2 <- convert_to_datetime(co2)
occupied_status <- convert_to_datetime(occupied_status)
supply_air_flow <- convert_to_datetime(supply_air_flow)
supply_fan <- convert_to_datetime(supply_fan)
temperature <- convert_to_datetime(temperature)
```

Investigate data in series columns

```r
co2 %>% count(series)
```

```
## # A tibble: 6 x 2
##   series                                          n
## * <chr>                                       <int>
## 1 co2_ppm.mean {location_specific: 203 Olsson}  1337
## 2 co2_ppm.mean {location_specific: 211 Olsson}  1337
## 3 co2_ppm.mean {location_specific: 213 Olsson}  1337
## 4 co2_ppm.mean {location_specific: 217 Olsson}  1337
## 5 co2_ppm.mean {location_specific: 221 Olsson}  1337
## 6 co2_ppm.mean {location_specific: 225 Olsson}  1337
```

```r
supply_air_flow %>% count(value)
```

```
## # A tibble: 21,203 x 2
##     value     n
##  *  <dbl> <int>
##  1 -11        1
##  2 -10.8      1
##  3 -10.6      1
##  4 -10.4      1
##  5 -10.4      2
##  6 -10.2      1
##  7  -9.83     1
##  8  -9.81     1
##  9  -9.72     1
## 10  -9.67     1
## # ... with 21,193 more rows
```

```r
supply_fan %>% count(series)
```

```
## # A tibble: 2 x 2
##   series                                                 n
## * <chr>                                              <int>
## 1 supply_fan_status {device_id: 0202EquipmentAHU2E}   292
## 2 supply_fan_status {device_id: 0202EquipmentAHU2W}   268
```

```r
temperature %>% count(series)
```

```
## # A tibble: 44 x 2
##    series                         n
##  * <chr>                      <int>
##  1 Temperature C - 201 Olsson  1337
##  2 Temperature C - 203 Olsson  1337
##  3 Temperature C - 204 Olsson  1337
##  4 Temperature C - 208 Olsson  1337
##  5 Temperature C - 211 Olsson  1337
##  6 Temperature C - 213 Olsson  1337
##  7 Temperature C - 217 Olsson  1337
##  8 Temperature C - 218 Olsson  1337
##  9 Temperature C - 221 Olsson  1337
## 10 Temperature C - 225 Olsson  1337
## # ... with 34 more rows
```

```r
occupied_status %>% count(series)
```

```
## # A tibble: 2 x 2
##   series                                        n
## * <chr>                                     <int>
## 1 occupied {device_id: 0202EquipmentAHU2E}     79
## 2 occupied {device_id: 0202EquipmentAHU2W}    300
```

Co2 data is only provided for 6 rooms: Olsson 203, 211, 213, 217, 221, 225. Investigating whether additional rooms are available.

Supply_air_flow contains data for 45 rooms in Olsson hall, as well as set-point data for each room. (Note: Investigate documentation for definition of set-point data)

Supply_fan_status is given for both HVAC units; the nature of the time intervals of the supply generating process is still under investigation.

Temperature is given for 44 rooms in Olsson, all but the generic "2nd floor" label which was fond in the supply_air_flow table. There is no set-point data provided here.

Occupied_status is given for both HVAC units. As the status is not given by room, I'm looking for documentation which shows what occupied_status means in the system. The nature of the time intervals of the supply generating process is still under investigation.

Create rooms column by parsing from series column

```r
co2$room = regmatches(x= co2$series, m=regexpr("([0-9]{3})", co2$series))

supply_air_flow$room = str_match(supply_air_flow$series, "C[0-9]{3}|[0-9]{3}")

temperature$room = str_match(temperature$series, "C[0-9]{3}|[0-9]{3}")
```

There exists a mapping from HVAC unit to rooms which could be used to relate observations on the room level and observations on the system level (e.g. which rooms are receiving supply at a given time based on supply_status data)

Create room assignment vectors for each HVAC unit

```
AHU_2E <- c(241, 243, 245, 247, 249, 251, 253, 257, 255, 259, 263, 261, 240, "C244", 244, 260, 213, 217

AHU_2W <- c(269, 267, 265, 273, 271, 275, 277, 279, 281, 283, 285, 274, 286, 204, 208, 272, 270, "C260"

# Check for duplicates/overlap

table(AHU_2E)
```

```
## AHU_2E
##  213  217  218  220  223  225  229  231  240  241  243  244  245  247  249  251
##    1    1    1    1    1    2    1    1    1    1    1    1    1    1    1    1
##  253  254  255  256  257  258  259  260  261  263 C210 C211 C214 C216 C227 C230
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## C244 C250 T210 T212 T218
##    1    1    1    1    1
```

```
table(AHU_2W)
```

```
## AHU_2W
##  201  203  204  208  211  265  267  269  270  271  272  273  274  275  276  277
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##  279  281  283  285  286 C200 C201 C260 C270 C280
##    1    1    1    1    1    1    1    1    1    1
```

```
table(c(AHU_2E, AHU_2W))
```

```
##
##  201  203  204  208  211  213  217  218  220  223  225  229  231  240  241  243
##    1    1    1    1    1    1    1    1    1    1    2    1    1    1    1    1
##  244  245  247  249  251  253  254  255  256  257  258  259  260  261  263  265
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##  267  269  270  271  272  273  274  275  276  277  279  281  283  285  286 C200
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## C201 C210 C211 C214 C216 C227 C230 C244 C250 C260 C270 C280 T210 T212 T218
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
```

Parse equipment names from series column

```
occupied_status$equiment <- str_match(occupied_status$series, "AHU2[EW]")
supply_fan$equipment <- str_match(supply_fan$series, "AHU2[EW]")
```

Final Table Designs

Table for analysis of system dynamics and energy consumption

Key: HVAC unit Index: time (3 hour intervals? pending documentation) Observations: supply air flow (aggregated by HVAC unit), supply air flow set-point (aggregated by HVAC unit), supply fan, energy use (calculated)

Data cleaning tasks: - Unstack supply__air__flow data such that there is a column for value and a column for set-point for each room at each time-stamp - Aggregate flow and set-point values for each HVAC unit to create an HVAC unit value at each time-stamp - For every three hour interval, assign the supply fan status column to the most recent value from supply__fan for each AHU - Calculate energy consumption and input into final column

Unstack supply__air__flow

```
supply_air_flow %>% filter(grepl("Setpoint", series)) -> supply_air_flow_setpoints

supply_air_flow %>% filter(!grepl("Setpoint", series)) -> supply_air_flow


inner_join(supply_air_flow_setpoints,supply_air_flow, by=c("room", "time")) %>% select(c("time", "value

colnames(unstacked_supply_air) <- c("time", "setpoint", "value", "room")
```

Aggregate flow and set point by HVAC unit

```
unstacked_supply_air <- as_tsibble(unstacked_supply_air, key= room, index = time)
library(dplyr)

unstacked_supply_air %>% filter(room %in% AHU_2E) -> air_supply_AHU_2E
unstacked_supply_air %>% filter(room %in% AHU_2W) -> air_supply_AHU_2W

aggregated_AHU_2E <- aggregate(cbind(air_supply_AHU_2E$setpoint, air_supply_AHU_2E$value), by=list(time

aggregated_AHU_2W <- aggregate(cbind(air_supply_AHU_2W$setpoint, air_supply_AHU_2W$value), by=list(time

colnames(aggregated_AHU_2E) = c("time", "setpoint", "air_supply")
colnames(aggregated_AHU_2W) = c("time", "setpoint", "air_supply")

inner_join(aggregated_AHU_2E, aggregated_AHU_2W, by=c("time"), suffix=c(".AHU2E", ".AHU2W")) -> unit_su
```

Feed forward supply__fan data

```
# create df where columns are time, AHU2E status, AHU2W status
supply_fan %>% filter(equipment == "AHU2E") -> AHU2E_supply_fan
supply_fan %>% filter(equipment == "AHU2W") -> AHU2W_supply_fan
```

```r
unstacked_fan = full_join(AHU2E_supply_fan, AHU2W_supply_fan, by=c("time"))
unstacked_fan = as.data.frame(unstacked_fan)

unstacked_fan %>% select(2,3,6) -> unstacked_fan
colnames(unstacked_fan) = c("time", "AHU2E_status", "AHU2W_status")
unstacked_fan$time = as.POSIXlt(unstacked_fan$time)

# up fill dataset
unstacked_fan %>% fill(AHU2E_status, .direction ="up") -> unstacked_fan
unstacked_fan %>% fill(AHU2W_status, .direction ="up") -> unstacked_fan


# time_difference = function(time, times){
#   differences = as.data.frame(times - time)
#   colnames(differences) = c("value")
#   pos_differences = filter(temp, value > 0)$value
#   pos_diff_index = which(as.numeric(pos_differences) == min(as.numeric(pos_differences)))
#   rel_time = times[which(temp$value == pos_differences[pos_diff_index])]
#   unit_supply_air[unit_supply_air$time == rel_time,"AHU2W_fan_status"] = unstacked_fan[unstacked_fan$
#
#   unit_supply_air[unit_supply_air$time == rel_time,"AHU2E_fan_status"] = unstacked_fan[unstacked_fan$
# }

times = unit_supply_air$time

# for (time in unstacked_fan$time) {
#  time_difference(time, times)
# }

# at that found time, add the values of AHU2E status + AHU2W status

temp = as.data.frame(times - unstacked_fan$time[100])
colnames(temp) = c("value")
pos_differences = filter(temp, value > 0)$value
pos_diff_index = which(as.numeric(pos_differences) == min(as.numeric(pos_differences)))
rel_time = times[which(temp$value == pos_differences[pos_diff_index])]

unit_supply_air[unit_supply_air$time == rel_time,"AHU2W_fan_status"] = unstacked_fan[unstacked_fan$time

unit_supply_air[unit_supply_air$time == rel_time,"AHU2E_fan_status"] = unstacked_fan[unstacked_fan$time
```

Table for analysis of system dynamics and room comfort

Key: room, Index: time Observations: c02, occupied status, supply air flow, supply fan, temperature, supply air flow set-point

Convert to tsibble objects

```r
co2 <- as_tsibble(co2, key= series, index = time)
occupied_status <- as_tsibble(occupied_status, key= series, index = time)
```

```r
supply_air_flow <- as_tsibble(supply_air_flow, key= room, index = time)
supply_fan <- as_tsibble(supply_fan, key= series, index = time)
temperature <- as_tsibble(temperature, key= series, index = time)
```