Data Cleaning and (preliminary) EDA

Optimizing HVAC Operation for Occupant Comfort and Energy Savings

Caleb Neale

Spring 2021

HVAC System Summary

System diagram + data accessible through: http://icoweb.fm.virginia.edu/anyglass/pubdisplay/UVa/Customers/LinkLab0202/Home.gdfx

Acronyms decoded using below resources:

- https://www.abraxasenergy.com/energy-resources/toolbox/hvac-acronyms/
- https://github.com/uva-eng-time-series-sp21/neale-caleb/blob/main/HVAC_System_Documentation/141602%20-%20M0.0.pdf

For each air handling unit we have:

- Temperature data (deg F):
- RA-T; Return Air
- SA-T; Supply Air
- PH-T; Pre-Heat Air
- MA-T; Mixed Air
- Fans (logical on/off, Variable Frequency Drive perecntage):
  - R-FN; Return Fan
  - S-FN; Supply Fan
- Humidity:
  - RA-H; Return Air (% rel. humidity)
- Ducts:
  - EXH-D; Exhaust (%, open? documentation unclear)
  - OA-D; Outside Air (%, open? documentation unclear)
- Heating/Cooling:
  - PH-V; Pre-Heat Valve (% Open)
  - CHW-V; Chilled Water Valve (% Open)
- Occupied (logical 0/1)

For each room we have:

- HW-V; Hot Water Valve (%, open? documentation unclear)
- SA-T; Supply Air Temperature (73.3 deg F)
- SA-F; Supply Air Flow (CFM)
- SA-F-SP; Supply Air Flow Set-Point
- ZN-T; Unconfirmed but appears to be temperature set-point (deg F)
- Temperature (deg C)
- co2 (only select rooms, PPM)

Load libraries

```
library(tidyverse)
library(lubridate)
library(fpp3)
```

Import Data and convert to tibble

```
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

```
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
```

```
supply_air_flow %>% count(series)
```

```
## # A tibble: 89 x 2
##    series                          n
##  * <chr>                       <int>
##  1 Supply Air Flow (201 Olsson)  1337
##  2 Supply Air Flow (203 Olsson)  1337
##  3 Supply Air Flow (204 Olsson)  1337
##  4 Supply Air Flow (208 Olsson)  1337
##  5 Supply Air Flow (211 Olsson)  1337
##  6 Supply Air Flow (213 Olsson)  1337
##  7 Supply Air Flow (217 Olsson)  1337
##  8 Supply Air Flow (218 Olsson)  1337
##  9 Supply Air Flow (221 Olsson)  1337
## 10 Supply Air Flow (225 Olsson)  1337
## # ... with 79 more rows
```

```
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
```

```
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
```

```
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.

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)

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

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

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

```
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 in the plans for the HVAC system (HVAC_System_Documentation folder in file "141605 - M0.3.pdf", tables on right side) 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"

rooms_tbl <- rbind(tibble('room' = AHU_2E, 'equipment' = "AHU_2E"), tibble('room' = AHU_2W, 'equipment'

# Check for duplicates/overlap
rooms_tbl %>% group_by(room) %>% count() %>% filter(n>1) -> dupes

print(dupes)
```

```
## # A tibble: 0 x 2
## # Groups:   room [0]
## # ... with 2 variables: room <chr>, n <int>
```

No duplicates.

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]")
```

Table Designs

Table for room-specific indoor environmental quality data

Index: date-time, 3 hour intervals

Keys: room

Observations:

- CO_2 levels (ppm, mean)
- temperature (degrees C)
- supply air flow (ft^3 s-1),
- supply air flow set-point (ft^3 s-1),

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
```

```
co2 %>%
  rename('co2_ppm_mean' = 'value') %>%
  select(time, room, co2_ppm_mean) -> co2_ts

temperature %>%
  rename('temperature_C' = 'value') %>%
  select(time, room, temperature_C) -> temperature_ts

supply_air_flow %>%
  rename('supply_air_flow_cfs' = 'value') %>%
  select(time, room, supply_air_flow_cfs) -> supply_air_flow_ts
```

```
supply_air_flow_setpoints %>%
  rename('supply_air_flow_setpoint_cfs' = 'value') %>%
  select(time, room, supply_air_flow_setpoint_cfs) -> supply_air_flow_setpoints_ts

full_join(co2_ts, temperature_ts, by=c("time", "room")) %>%
  full_join(supply_air_flow_ts, by=c("time", "room")) %>%
  full_join(supply_air_flow_setpoints_ts, by=c("time", "room")) -> ieq_tbl

print(ieq_tbl)
```

```
## # A tibble: 60,165 x 6
##    time                room[,1] co2_ppm_mean temperature_C supply_air_flow~
##    <dttm>              <chr>           <dbl>         <dbl>             <dbl>
##  1 2020-08-31 23:00:00 203              434.          22.7              101.
##  2 2020-09-01 02:00:00 203              431.          22.7              100.
##  3 2020-09-01 05:00:00 203              433.          22.4              172.
##  4 2020-09-01 08:00:00 203              442.          22.0              234.
##  5 2020-09-01 11:00:00 203              439.          21.9              227.
##  6 2020-09-01 14:00:00 203              435.          21.9              247.
##  7 2020-09-01 17:00:00 203              430.          21.9              207.
##  8 2020-09-01 20:00:00 203              436.          22.2              102.
##  9 2020-09-01 23:00:00 203              444.          22.3              101.
## 10 2020-09-02 02:00:00 203              446.          22.4              102.
## # ... with 60,155 more rows, and 1 more variable:
## #   supply_air_flow_setpoint_cfs <dbl>
```

Table of equipment operational data

Index: date-time, irregular time intervals

Keys: HVAC equipment code

Observations:

- supply fan (logical)
- occupancy status (logical)
- aggregated air flow (ft^3/min)

```
supply_fan %>%
  rename('supply_fan_status' = 'value') %>%
  select(time, equipment, supply_fan_status) -> supply_fan_irreg_ts

occupied_status %>%
  rename('occupied' = 'value',
         'equipment' = 'equiment') %>%
  select(time, equipment, occupied) -> occupied_irreg_ts

equipment_ops_irreg_ts <- full_join(supply_fan_irreg_ts, occupied_irreg_ts, by=c("time", "equipment"))

print(equipment_ops_irreg_ts %>% arrange(time))
```

```
## # A tibble: 728 x 4
```

```
##    time                equipment[,1] supply_fan_status occupied
##    <dttm>              <chr>         <fct>             <fct>
##  1 2020-09-01 06:06:05 AHU2E         1                 <NA>
##  2 2020-09-01 06:06:05 AHU2W         <NA>              1
##  3 2020-09-01 08:27:17 AHU2E         1                 1
##  4 2020-09-01 08:27:17 AHU2W         1                 1
##  5 2020-09-01 08:27:44 AHU2E         1                 1
##  6 2020-09-01 08:27:44 AHU2W         1                 1
##  7 2020-09-01 19:01:10 AHU2W         0                 <NA>
##  8 2020-09-01 19:06:10 AHU2E         0                 <NA>
##  9 2020-09-01 19:06:10 AHU2W         1                 0
## 10 2020-09-02 06:06:14 AHU2E         1                 <NA>
## # ... with 718 more rows
```

Q: Do rooms that share HVAC equipment also share the same supply air flow and setpoints?

```
ieq_tbl %>%
  filter(room %in% (rooms_tbl %>% filter(equipment == "AHU_2E") %>% pull(room))) %>%
  select(time, room, supply_air_flow_cfs, supply_air_flow_setpoint_cfs) %>%
  arrange(time, room)
```

```
## # A tibble: 28,077 x 4
##    time                room[,1] supply_air_flow_cfs supply_air_flow_setpoint_cfs
##    <dttm>              <chr>                  <dbl>                         <dbl>
##  1 2020-08-31 23:00:00 213                   -5.44                          100
##  2 2020-08-31 23:00:00 217                    0.778                          50
##  3 2020-08-31 23:00:00 218                    0.056                          50
##  4 2020-08-31 23:00:00 225                   -3                              50
##  5 2020-08-31 23:00:00 231                    0                             100
##  6 2020-08-31 23:00:00 240                    6.33                          100
##  7 2020-08-31 23:00:00 241                   10.6                            33.3
##  8 2020-08-31 23:00:00 243                   10.6                            33.3
##  9 2020-08-31 23:00:00 245                   10.6                            33.3
## 10 2020-08-31 23:00:00 247                   -0.444                          50
## # ... with 28,067 more rows
```

A: Clearly not. Rooms can share the same HVAC equipment yet have different setpoints and different realized supply air flows.

```
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")
```

** Q: How does energy consumption related to reported values of system operations – esp. supply air flow, supply fan status, occupied status? **

Formulae SEER = BTU/Watt-Hours = total change in heat energy in conditioned space divided by energy consumed to condition space

For heating: BTU/hr = C * del(T) * M

Where: - C is the specific heat of air (approx. 1 kJ/Kg*K) - *del(T) is the change in temperature - M is the mass of air = rho*V, using an approximate density of 1.2 Kg/mˆ3 - Given our data an adjustment will also have to be made to convert from cubic feet per **minute** to BTU per **hour**.

For cooling, an additional term can be included in calculating BTU related to the removal of humidity: BTU/hr = C * del(T) * M + (0.68 * CFM * del(w_gr))

Note about above: This concept makes sense but I have been unable to find sourcing outside of a blog post on an HVAC website.

- CFM = cubic feet per minute
- del(w_gr) = change in humidiy ratio in grains

I have been unable to locate a data source on the differential between supply and return air humidity so I will be excluding this term for now, with the potential for revisiting later as there is one measure of relative humidity as a part of return air that is available to glean information from.

Data Data to be exported from icoweb platform:

- Temperature differential (del(T)) can be calculated by finding the difference between relevant points in the AHU, which I believe to be supply and return air (RA-T minus SA-T)

Aggregate flow and set point by HVAC unit

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

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
```

Calculate estimated energy usage

Feed forward supply_fan data

I did finally get this to work, but further research into the factors which contribute to energy consumption suggest that supply fan status is not a significant contributor to energy consumption.

```
# 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

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)

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


time_difference = function(time, times){
  if(!is.na(time)){
    differences = as.data.frame(times - time)
    colnames(differences) = c("value")
    pos_differences = dplyr::filter(differences, value > 0)$value
    pos_diff_index = which(as.numeric(pos_differences) == min(as.numeric(pos_differences)))
    rel_time = times[which(differences$value == pos_differences[pos_diff_index])]

    unit_supply_air[unit_supply_air$time == rel_time,"AHU2W_fan_status"] <<- unstacked_fan[unstacked_fa
    unit_supply_air[unit_supply_air$time == rel_time,"AHU2E_fan_status"] <<- unstacked_fan[unstacked_fa
  }
  else{
    print("fail")
  }
}

times = unit_supply_air$time


for (i in 1:length(unstacked_fan$time)) {
  time = unstacked_fan$time[i]
  time_difference(time, times)
}

# doum fill dataset
unit_supply_air %>% fill(AHU2E_fan_status, .direction ="down") -> unit_supply_air
unit_supply_air %>% fill(AHU2W_fan_status, .direction ="down") -> unit_supply_air
```

Table for analysis of system dynamics and room comfort

This analysis is being set aside for a later time

Key: room, Index: time

Observations:

- c02,
- occupied status,
- supply air flow,
- supply fan,
- temperature,
- supply air flow set-point

Convert to tsibble objects

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