MAT 8444 Project Code - Time Series Analysis of Electricity Demand and Related Data

Duane Stanton

May 10, 2019

```
# upfront code for code chunk behavior
# eval=F sets the code to not be evaluated in-file
knitr::opts_chunk$set(eval=F, fig.align="center", message=F, warning=F)
```

```
# Plot styling setup
theme_ds2 <- function(base_size = 12.5){
  theme_bw(base_size = base_size) %+replace%
    theme(
      #line = element_line(color = "#000000"),
      #text = element_text(color = "#000000"),
      axis.title = element_text(color = "#000000", face = "plain"),
      axis.text = element_text(color="#000000", size = 11),
      axis.line = element_line(size = 0.5, linetype = 1, color = "#505050"),
      #axis.line.y = element_line(size = 0.5, linetype = 1, color = "#505050"),
      #legend.key = element_rect(color="#D2E2F9", fill = "#FFFFFF", linetype = 0),
      legend.title = element_text(size = 11.5, face = "plain"),
      legend.title.align = 0.5,
      legend.text = element_text(size = 11),
      legend.background = element_rect(size = 0.1, color = "#EAEAEA",
                                       fill = "#FFFFFF", linetype = 0),
      panel.grid = element_line(color = "#505050", linetype = 0, size = 0.5),
      #panel.grid.minor.x = element_line(size = 0.35),
      #panel.grid.minor.y = element_line(size = 0.35),
      panel.border = element_rect(color = "#EAEAEA", fill = NA),
      panel.background = element_rect(fill = "#FFFFFF"),
      #plot.title = element_text(color = "#000000"),
      plot.background = element_rect(color = NA, fill = "#FFFFFF"),
      #plot.title = element_text(hjust = 0, face = "plain"),
      plot.subtitle = element_text(hjust = 0, size = 12, face = "plain"),
      plot.caption=element_text(size=11, hjust=1, face="italic", color="black")
      #strip.text = element_text(size = 12),
      #strip.background = element_rect(fill = NA),
      )
}

color_set8 <- c("#00205B", "#13B5EA", "#079500", "#9ECC14",
                "#CDB87D", "#EEB31B", "#E27400", "#894600")
```

Data Processing Code

A note on the datafiles

The initial datafiles were processed from their "as-downloaded" form (or "as manually updated" form for the NY County-level temperature data); datafiles sent along with this code file are the finished product of the data processing. I've retained copies of the initial files and can send them if they are preferred to the processed versions.

Processing U.S. Electric System Operating Data

ISO-NE

```
library(readxl)

path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/eia_electric_sys_op_data
## eia_electric_sys_op_data.xlsx

isoneHrDyAhdDmd <-
  read_xlsx(path      = path,
            sheet     = "ISONE",
            range     = "A6:B31878",
            col_names = c("MDY_H", "dayAheadDemand.MWh"),
            col_types = c("guess", "numeric")
            )

isoneHrDmd <-
  read_xlsx(path      = path,
            sheet     = "ISONE",
            range     = "C6:D31864",
            col_names = c("MDY_H", "demand.MWh"),
            col_types = c("guess", "numeric")
            )

isoneHrNetGen <-
  read_xlsx(path      = path,
            sheet     = "ISONE",
            range     = "E6:F31853",
            col_names = c("MDY_H", "netGen.MWh"),
            col_types = c("guess", "numeric")
            )

library(lubridate)
library(tidyverse)

isoneHrDyAhdDmd <-
  isoneHrDyAhdDmd %>%
  mutate(hour = str_split(MDY_H, " ") %>%
           sapply("[[", 2) %>%
           gsub(pattern = "H", replacement = "") %>%
           as.integer(),
```

```
          date = str_split(MDY_H, " ") %>%
            sapply("[[", 1)
          ) %>%
  mutate(date = as.Date(date, format = "%m/%d/%Y")) %>%
  arrange(date, hour) %>%
  mutate(hour  = hour,
          day   = day(date),
          month = month(date),
          year  = year(date)
          )

isoneHrDmd <-
  isoneHrDmd %>%
  mutate(hour = str_split(MDY_H, " ") %>%
            sapply("[[", 2) %>%
            gsub(pattern = "H", replacement = "") %>%
            as.integer(),
          date = str_split(MDY_H, " ") %>%
            sapply("[[", 1)
          ) %>%
  mutate(date = as.Date(date, format = "%m/%d/%Y")) %>%
  arrange(date, hour) %>%
  mutate(hour  = hour,
          day   = day(date),
          month = month(date),
          year  = year(date)
          )

isoneHrNetGen <-
  isoneHrNetGen %>%
  mutate(hour = str_split(MDY_H, " ") %>%
            sapply("[[", 2) %>%
            gsub(pattern = "H", replacement = "") %>%
            as.integer(),
          date = str_split(MDY_H, " ") %>%
            sapply("[[", 1)
          ) %>%
  mutate(date = as.Date(date, format = "%m/%d/%Y")) %>%
  arrange(date, hour) %>%
  mutate(hour  = hour,
          day   = day(date),
          month = month(date),
          year  = year(date)
          )

isoneHR_all <-
  isoneHrDyAhdDmd %>%
  full_join(isoneHrDmd, by = c("hour", "day", "month", "year",
                               "date", "MDY_H")) %>%
  full_join(isoneHrNetGen, by = c("hour", "day", "month", "year",
                                  "date", "MDY_H")) %>%
  select(MDY_H, date, hour, day, month, year, demand.MWh,
          netGen.MWh, dayAheadDemand.MWh)
```

```r
# plot summary of missingness (NA) by variable
isoneHR_all %>%
  mutate(na_demand.MWh         = is.na(demand.MWh),
         na_netGen.MWh         = is.na(netGen.MWh),
         na_dayAheadDemand.MWh = is.na(dayAheadDemand.MWh)) %>%
  select(date, na_demand.MWh, na_netGen.MWh, na_dayAheadDemand.MWh) %>%
  gather(key=whichVar, value=isNA, -date) %>%
  group_by(date, whichVar) %>%
  summarize(isNA = sum(isNA) %>% as.integer()) %>%
  filter(isNA > 0) %>%
  ggplot(aes(x=date, y=isNA, color=whichVar, size=isNA)) +
  geom_vline(xintercept = c(as.Date("2015-07-01"), as.Date("2019-01-01"),
                            as.Date("2019-02-18")), color="grey") +
  geom_jitter(alpha=0.7, width=0, height=1.5) +
  labs(title="ISONE 'is.na()' cases",
       x="Date", y=NULL, color=NULL, size="# missing",
       caption="Y-axis not exact due to jittering \n  6+ NA in
                \n 2016: 24 May \n 2017: 5 & 7 Dec,
                \n 2018: 11 Sep, 10-11 & 17 Oct, 4 Nov, 5 Dec
                \n 2019: 5 & 17-18 Feb") +
  scale_x_date(limits = c(as.Date("2015-01-01"), tail(isoneHR_all$date, 1))) +
  scale_color_manual(values=color_set8[3:4]) +
  scale_size_area() +
  guides(size = guide_legend(reverse=T)) +
  theme_ds2() + theme(legend.position = c(0.15, 0.5),
                      axis.text.y = element_blank(),
                      axis.ticks.y = element_blank())

# 117 NA's for demand.MWh and 127 for netGen.MWh
#   the following uses "dayAheadDemand.MWh" from 24hrs ago for demand.MWh
#   and imputes the mean of "same time yesterday" and "same time tomorrow"
#   when both exist for netGen.MWh,
#   and uses whichever does exist in cases where both don't exist,
#   again for netGen.MWh

isoneHR_all2 <-
  isoneHR_all %>%
  # first shift dayAheadDemand.MWh ahead by 24hrs to align with the forecast period
  #   and align same-time-yesterday/same-time-tomorrow netGen.MWh
  #   with current period
  #   note: still one missing netGen.MWh using the planned approach,
  #   so also including two-days-apart netGen.MWh
  mutate(
    XdayAheadDemand = c(rep(0,24),
                        isoneHR_all$dayAheadDemand.MWh[1:(nrow(isoneHR_all)-24)]),
    yes.netGen.MWh  = c(rep(0,24),
                        isoneHR_all$netGen.MWh[1:(nrow(isoneHR_all)-24)]),
    tom.netGen.MWh  = c(isoneHR_all$netGen.MWh[25:nrow(isoneHR_all)],
                        rep(0,24)),
    yes2.netGen.MWh = c(rep(0,48),
                        isoneHR_all$netGen.MWh[1:(nrow(isoneHR_all)-48)]),
    tom2.netGen.MWh = c(isoneHR_all$netGen.MWh[49:nrow(isoneHR_all)],
```

```
                        rep(0,48)),
    # now carry out imputation
    demand.MWh = if_else(!is.na(demand.MWh), demand.MWh,
                   if_else(!is.na(XdayAheadDemand), XdayAheadDemand, -999)),
    netGen.MWh = if_else(!is.na(netGen.MWh), netGen.MWh,
                   if_else(!is.na(yes.netGen.MWh) & !is.na(tom.netGen.MWh),
                           (yes.netGen.MWh + tom.netGen.MWh)/2,
                     if_else(!is.na(yes.netGen.MWh) & is.na(tom.netGen.MWh),
                             yes.netGen.MWh,
                       if_else(is.na(yes.netGen.MWh) & !is.na(tom.netGen.MWh),
                               tom.netGen.MWh,
                         if_else(!is.na(yes2.netGen.MWh) & !is.na(tom2.netGen.MWh),
                                 (yes2.netGen.MWh + tom2.netGen.MWh)/2,
                                 -999)))))
    )

#summary(isoneHR_all2) no -999 or NA's in the desired variables now

isoneHR_all <-
  isoneHR_all2 %>%
  select(-XdayAheadDemand, -yes.netGen.MWh, -tom.netGen.MWh,
         -yes2.netGen.MWh, -tom2.netGen.MWh)

#write.csv(isoneHR_all,
#          file = "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/isoneHR_all.cs
#          row.names = F)
## isoneHR_all.csv
```

NYISO

```
library(readxl)

path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/eia_electric_sys_op_data
## eia_electric_sys_op_data.xlsx

nyisoHrDyAhdDmd <-
  read_xlsx(path      = path,
            sheet     = "NYISO",
            range     = "A6:B31662",
            col_names = c("MDY_H", "dayAheadDemand.MWh"),
            col_types = c("guess", "numeric")
            )

nyisoHrDmd <-
  read_xlsx(path      = path,
            sheet     = "NYISO",
            range     = "C6:D31816",
            col_names = c("MDY_H", "demand.MWh"),
            col_types = c("guess", "numeric")
            )

nyisoHrNetGen <-
```

```
  read_xlsx(path      = path,
            sheet     = "NYISO",
            range     = "E6:F31806",
            col_names = c("MDY_H", "netGen.MWh"),
            col_types = c("guess", "numeric")
            )

library(lubridate)
library(tidyverse)

nyisoHrDyAhdDmd <-
  nyisoHrDyAhdDmd %>%
  mutate(hour = str_split(MDY_H, " ") %>%
           sapply("[[", 2) %>%
           gsub(pattern = "H", replacement = "") %>%
           as.integer(),
         date = str_split(MDY_H, " ") %>%
           sapply("[[", 1)
         ) %>%
  mutate(date = as.Date(date, format = "%m/%d/%Y")) %>%
  arrange(date, hour) %>%
  mutate(hour  = hour,
         day   = day(date),
         month = month(date),
         year  = year(date)
         )

nyisoHrDmd <-
  nyisoHrDmd %>%
  mutate(hour = str_split(MDY_H, " ") %>%
           sapply("[[", 2) %>%
           gsub(pattern = "H", replacement = "") %>%
           as.integer(),
         date = str_split(MDY_H, " ") %>%
           sapply("[[", 1)
         ) %>%
  mutate(date = as.Date(date, format = "%m/%d/%Y")) %>%
  arrange(date, hour) %>%
  mutate(hour  = hour,
         day   = day(date),
         month = month(date),
         year  = year(date)
         )

nyisoHrNetGen <-
  nyisoHrNetGen %>%
  mutate(hour = str_split(MDY_H, " ") %>%
           sapply("[[", 2) %>%
           gsub(pattern = "H", replacement = "") %>%
           as.integer(),
         date = str_split(MDY_H, " ") %>%
           sapply("[[", 1)
         ) %>%
```

```
  mutate(date = as.Date(date, format = "%m/%d/%Y")) %>%
  arrange(date, hour) %>%
  mutate(hour  = hour,
         day   = day(date),
         month = month(date),
         year  = year(date)
         )

nyisoHR_all <-
  nyisoHrDyAhdDmd %>%
  full_join(nyisoHrDmd, by = c("hour", "day", "month", "year",
                               "date", "MDY_H")) %>%
  full_join(nyisoHrNetGen, by = c("hour", "day", "month", "year",
                                  "date", "MDY_H")) %>%
  select(MDY_H, date, hour, day, month, year, demand.MWh, netGen.MWh,
         dayAheadDemand.MWh)

nyisoHR_all %>% group_by(date) %>% summarize(nObs = n()) %>%
                arrange(nObs) %>% head(20)

# four days have nonexistent data entries:
#   01-Jul-2018 is missing 5h to 23h [19 entries]
#   02-Jul-2018 is missing 0h to 4h  [5 entries]
#   11-Nov-2018 is missing 6h to 23h [18 entries]
#   12-Nov-2018 is missing 0h to 5h  [6 entries]
# need to insert empty rows here for missing-data imputation further on
missing_ny_hours <-
  tibble(
   MDY_H = c(paste("07/01/2018", paste0(5:23, "H")),
             paste("07/02/2018", paste0(0:4, "H")),
             paste("11/11/2018", paste0(6:23, "H")),
             paste("11/12/2018", paste0(0:5, "H"))),
   date  = c(rep("07/01/2018", 19), rep("07/02/2018", 5),
             rep("11/11/2018", 18), rep("11/12/2018", 6)) %>%
           as.Date(format = "%m/%d/%Y"),
   hour  = c(5:23, 0:4, 6:23, 0:5),
   day   = c(rep(1, 19), rep(2, 5), rep(11, 18), rep(12, 6)),
   month = c(rep(7, 24), rep(11, 24)),
   year  = 2018,
   demand.MWh         = NA,
   netGen.MWh         = NA,
   dayAheadDemand.MWh = NA
  )

nyisoHR_all <-
  nyisoHR_all %>%
  bind_rows(missing_ny_hours)

# plot summary of missingness (NA) by variable
nyisoHR_all %>%
  mutate(na_demand.MWh         = is.na(demand.MWh),
         na_netGen.MWh         = is.na(netGen.MWh),
         na_dayAheadDemand.MWh = is.na(dayAheadDemand.MWh)) %>%
```

```
  select(date, na_demand.MWh, na_netGen.MWh, na_dayAheadDemand.MWh) %>%
  gather(key=whichVar, value=isNA, -date) %>%
  group_by(date, whichVar) %>%
  summarize(isNA = sum(isNA) %>% as.integer()) %>%
  filter(isNA > 0) %>%
  ggplot(aes(x=date, y=isNA, color=whichVar, size=isNA)) +
  geom_vline(xintercept = c(as.Date("2015-07-01"),
                            as.Date("2019-01-01"),
                            as.Date("2019-02-18")), color="grey") +
  geom_jitter(alpha=0.7, width=0, height=1.5) +
  labs(title="NYISO 'is.na()' cases",
       x="Date", y=NULL, color=NULL, size="# missing",
       caption="Y-axis not exact due to jittering \n All NA in
                \n 2015: 1-2 Jul,
                \n 2018: 23-25 & 30 Jun, 1-3 Jul, 10-13 Nov, 25-27 Dec,
                \n 2019: 2-4 Jan, 14-18 Feb") +
  scale_x_date(limits = c(as.Date("2015-01-01"), as.Date("2019-02-18"))) +
  scale_color_manual(values=color_set8[c(5, 1:2)]) +
  scale_size_area() +
  guides(size = guide_legend(reverse=T)) +
  theme_ds2() + theme(legend.position = c(0.5, 0.5),
                      axis.text.y = element_blank(),
                      axis.ticks.y = element_blank())

# 158+43 NA's for demand.MWh (and 3 '0') and
# 168+43 each for netGen.MWh (2 '0') and
#     dayAheadDemand.MWh (23 '0', not at the end which we might expect)
#   the following uses "dayAheadDemand.MWh" from 24hrs ago for demand.MWh OR
#     imputes the mean of "same time yesterday" and
#     "same time tomorrow" when both exist for demand.MWh
#   also imputes the mean of "same time yesterday" and
#     "same time tomorrow" when both exist for netGen.MWh,
#     and uses whichever does exist in cases where both don't exist,
#     again for netGen.MWh
#   also imputes the mean of "same time yesterday" and "same time tomorrow"
#     when both exist for dayAheadDemand.MWh,
#     and uses whichever does exist in cases where both don't exist,
#     again for dayAheadDemand.MWh

#     note: still 120 missing dayAheadDemand.MWh with this approach,
#           so also bringing in two-days out yes./tom. values
#

#     note: STILL 72 missing dayAheadDemand.MWh with this approach;
#           at this point I'm bringing in the imputeTS package for na.kalman
#           (Kalman smoothing) with auto.arima model, and
#           na.ma for moving-average weighting of 11-12 Nov 2018
#           also using this imputation approach for remaining NA in
#           demand.MWh and netGen.MWh

library(imputeTS)

nyisoHR_all2 <-
```

```
nyisoHR_all %>%
# first shift dayAheadDemand.MWh ahead by 24hrs to align with the forecast period
#   and align same-time-yesterday/same-time-tomorrow netGen.MWh /
#   dayAheadDemand.MWh with current period
mutate(
  XdayAheadDemand = c(rep(0,24),
                      nyisoHR_all$dayAheadDemand.MWh[1:(nrow(nyisoHR_all)-24)]),
  yes.demand.MWh  = c(rep(0,24),
                      nyisoHR_all$demand.MWh[1:(nrow(nyisoHR_all)-24)]),
  tom.demand.MWh  = c(nyisoHR_all$demand.MWh[25:nrow(nyisoHR_all)],
                      rep(0,24)),
  yes.netGen.MWh  = c(rep(0,24),
                      nyisoHR_all$netGen.MWh[1:(nrow(nyisoHR_all)-24)]),
  tom.netGen.MWh  = c(nyisoHR_all$netGen.MWh[25:nrow(nyisoHR_all)],
                      rep(0,24)),
  yes.dayAheadDemand = c(rep(0,24),
                         nyisoHR_all$dayAheadDemand.MWh[1:(nrow(nyisoHR_all)-24)]),
  tom.dayAheadDemand = c(nyisoHR_all$dayAheadDemand.MWh[25:nrow(nyisoHR_all)],
                         rep(0,24)),
  yes2.dayAheadDemand = c(rep(0,48),
                          nyisoHR_all$dayAheadDemand.MWh[1:(nrow(nyisoHR_all)-48)]),
  tom2.dayAheadDemand = c(nyisoHR_all$dayAheadDemand.MWh[49:nrow(nyisoHR_all)],
                          rep(0,48)),
  # now carry out imputation
  demand.MWh = if_else(!is.na(demand.MWh) & demand.MWh != 0, demand.MWh,
                 if_else(!is.na(XdayAheadDemand) & (XdayAheadDemand > 0),
                         XdayAheadDemand,
                         # na.kalman drastically underestimates 11 Nov 2018
                         # trying na.ma for moving average of 12+ obs
                         # in either direction of NA
                    if_else(date != as.Date("2018-11-11") &
                            date != as.Date("2018-11-12"),
                            na.kalman(demand.MWh, model="auto.arima"),
                            # default exponential weighting for na.ma
                            # >> more weight to closer obs.
                            na.ma(demand.MWh, k=12) ) ) ),
  # apparently 4h on 4 Oct 2015, only 56 MWh of electricity was generated
  # next lowest nonzero value is 8431 MWh (7h on 10 Sep 2018)
  netGen.MWh = if_else(!is.na(netGen.MWh) & netGen.MWh != 0, netGen.MWh,
                 if_else(!is.na(yes.netGen.MWh) & (yes.netGen.MWh > 0) &
                         !is.na(tom.netGen.MWh) & (tom.netGen.MWh > 0),
                         (yes.netGen.MWh + tom.netGen.MWh)/2,
                   if_else(!is.na(yes.netGen.MWh) & yes.netGen.MWh > 0 &
                             (is.na(tom.netGen.MWh)|tom.netGen.MWh<=0),
                           yes.netGen.MWh,
                    if_else((is.na(yes.netGen.MWh)|yes.netGen.MWh<=0)
                            & !is.na(tom.netGen.MWh) & (tom.netGen.MWh > 0),
                            tom.netGen.MWh, na.kalman(netGen.MWh,
                                                      model="auto.arima"))))),
  dayAheadDemand.MWh = if_else(!is.na(dayAheadDemand.MWh) &
                               (dayAheadDemand.MWh > 0),
                               dayAheadDemand.MWh,
                          if_else(!is.na(yes.dayAheadDemand) &
```

```
                                    !is.na(tom.dayAheadDemand),
                                    (yes.dayAheadDemand + tom.dayAheadDemand)/2,
                                if_else(!is.na(yes.dayAheadDemand) &
                                         (yes.dayAheadDemand > 0) &
                                         (is.na(tom.dayAheadDemand)|
                                          tom.dayAheadDemand<=0),
                                         yes.dayAheadDemand,
                                if_else((is.na(yes.dayAheadDemand)|
                                         yes.dayAheadDemand<=0) &
                                        (!is.na(tom.dayAheadDemand)|
                                         tom.dayAheadDemand<=0) &
                                        (tom.dayAheadDemand > 0),
                                         tom.dayAheadDemand,
                                         na.kalman(dayAheadDemand.MWh,
                                                   model="auto.arima")))))
  )

nyisoHR_all <-
  nyisoHR_all2 %>%
  select(-XdayAheadDemand, -yes.demand.MWh, -tom.demand.MWh,
         -yes.netGen.MWh, -tom.netGen.MWh, -yes.dayAheadDemand,
         -tom.dayAheadDemand, -yes2.dayAheadDemand, -tom2.dayAheadDemand)

#write.csv(nyisoHR_all,
#          file = "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/nyisoHR_all.cs
#          row.names = F)
## nyisoHR_all.csv
```

County temperature processing

```
# note: this was a "processing" code chunk I used for all NY county files
#       I just updated the 'path' filename e.g. "NY NOAA Counties20_23"
#       if I was processing the 20th through 23rd county data subsets

path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/NY NOAA Counties59_60.csv
## NY NOAA Counties59_60.csv

dat <- read.csv(path)

library(dplyr)

# filter to only entries with temp data
dat <-
  dat %>%
  filter(!is.na(TAVG) | !is.na(TMIN) | !is.na(TMAX))
#write.csv(dat, file = path, row.names = F)
```

```
library(dplyr)

# load the data
path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/NY NOAA Counties60.csv"
```

```
dat  <- read.csv(path)
## NY NOAA Counties60.csv

# group by county, take mean avg, high, and low temp by date
#   lots of missing avg values, so may need to take simple average of
#   high and low for consistency [didn't end up doing that-see below]
dat <-
  dat %>%
  group_by(county, DATE) %>%
  summarize(meanTAVG = mean(TAVG, na.rm = T),
            meanTMAX = mean(TMAX, na.rm = T),
            meanTMIN = mean(TMIN, na.rm = T)
            )

# take overall mean of meanTAVG, meanTMAX, meanTMIN vals by date
datDayMean <-
  dat %>%
  group_by(DATE) %>%
  summarize(tempAvg.F = mean(meanTAVG, na.rm = T),
            tempMax.F = mean(meanTMAX, na.rm = T),
            tempMin.F = mean(meanTMIN, na.rm = T)
            )
# no missing entries for tempAvg.F (overall) by date - useful for next steps

# find average temp per day for each county
#   if is.na(meanTAVG), try arithmetic mean of meanTMAX and meanTMIN
#   if one of these is missing, go to next step
#     where that county-date entry will be assigned overall tempAvg.F

dat2 <-
  dat %>%
  mutate(
    tempAvg.F = if_else(!is.na(meanTAVG), meanTAVG,
                  if_else(!is.na(meanTMAX) & !is.na(meanTMIN),
                          (meanTMAX + meanTMIN)/2,
                          -999)
                  ) %>% as.double(),
    tempMax.F = if_else(!is.na(meanTMAX), meanTMAX, -999) %>% as.double(),
    tempMin.F = if_else(!is.na(meanTMIN), meanTMIN, -999) %>% as.double()
    )

# 43 avg entries currently -999; assign these to the statewide tempAvg.F
# for that date
datSubAvg <- dat2 %>% filter(tempAvg.F == -999)

datSubAvgMatch <-
  datSubAvg %>%
  rename(OLDtempAvg.F = tempAvg.F) %>%
  inner_join(datDayMean, by = "DATE") %>%
  select(county, DATE, tempAvg.F)

# 14 max entries currently -999; same general process with statewide tempMax.F
datSubMax <- dat2 %>% filter(tempMax.F == -999)
```

```r
datSubMaxMatch <-
  datSubMax %>%
  rename(OLDtempMax.F = tempMax.F) %>%
  inner_join(datDayMean, by = "DATE") %>%
  select(county, DATE, tempMax.F)

# 29 min entries currently -999; same general process with statewide tempMin.F
datSubMin <- dat2 %>% filter(tempMin.F == -999)

datSubMinMatch <-
  datSubMin %>%
  rename(OLDtempMin.F = tempMin.F) %>%
  inner_join(datDayMean, by = "DATE") %>%
  select(county, DATE, tempMin.F)

# filter to datasets not missing avg / max / min temp
dat2NoMiss.Avg <-
  dat2 %>%
  filter(tempAvg.F != -999)

dat2NoMiss.Max <-
  dat2 %>%
  filter(tempMax.F != -999)

dat2NoMiss.Min <-
  dat2 %>%
  filter(tempMin.F != -999)

dat3.Avg <-
  dat2NoMiss.Avg %>%
  full_join(datSubAvgMatch, by = c("county", "DATE")) %>%
  mutate(
    tempAvg.F = if_else(!is.na(tempAvg.F.x), tempAvg.F.x, tempAvg.F.y) %>%
      as.integer()
    ) %>%
  select(county, DATE, tempAvg.F)

dat3.Max <-
  dat2NoMiss.Max %>%
  full_join(datSubMaxMatch, by = c("county", "DATE")) %>%
  mutate(
    tempMax.F = if_else(!is.na(tempMax.F.x), tempMax.F.x, tempMax.F.y) %>%
      as.integer()
    ) %>%
  select(county, DATE, tempMax.F)

dat3.Min <-
  dat2NoMiss.Min %>%
  full_join(datSubMinMatch, by = c("county", "DATE")) %>%
  mutate(
    tempMin.F = if_else(!is.na(tempMin.F.x), tempMin.F.x, tempMin.F.y) %>%
      as.integer()
    ) %>%
```

```
  select(county, DATE, tempMin.F)


# putting it all together
library(purrr)
dat3 <-
  list(dat3.Avg, dat3.Max, dat3.Min) %>%
  reduce(full_join, by = c("county", "DATE"))

  dat3.Avg %>%
  full_join(dat3.Max, dat3.Min, by = c("county", "DATE"))

newFilename <- "NY NOAA county temps"
newPath     <- gsub(pattern     = "NY NOAA Counties60",
                    replacement = newFilename,
                    x           = path)

#write.csv(dat3, file = newPath, row.names = F)
```

Weighting county-level temperature data by county population levels (NY only)

```
# loading U.S. Census Bureau population estimates by county
path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/2015.16.17_PEP_2017_PEPA

countyPop <- read.csv(file = path)

library(dplyr)
# filter to only NY data and process the data a bit
countyPop.NY <-
  countyPop %>%
  select(
    countyState        = GEO.display.label,
    totPopEst01Jul2015 = est72015sex0_age999,
    totPopEst01Jul2016 = est72016sex0_age999,
    totPopEst01Jul2017 = est72017sex0_age999
    ) %>%
  filter(grepl("New York", countyState)) %>%
  # countyPop loads popEst vars as factor because 1st row of source file
  # is descriptor text
  mutate(
    totPopEst01Jul2015 = totPopEst01Jul2015 %>% as.character() %>% as.numeric(),
    totPopEst01Jul2016 = totPopEst01Jul2016 %>% as.character() %>% as.numeric(),
    totPopEst01Jul2017 = totPopEst01Jul2017 %>% as.character() %>% as.numeric()
    )

# compute mean population levels by year for next step
totPopEst2015.mean <- countyPop.NY$totPopEst01Jul2015 %>% mean()
totPopEst2016.mean <- countyPop.NY$totPopEst01Jul2016 %>% mean()
totPopEst2017.mean <- countyPop.NY$totPopEst01Jul2017 %>% mean()

countyPop.NY_popWt <-
```

```
  countyPop.NY %>%
  mutate(
    popWt2015 = totPopEst01Jul2015 / totPopEst2015.mean,
    popWt2016 = totPopEst01Jul2016 / totPopEst2016.mean,
    popWt2017 = totPopEst01Jul2017 / totPopEst2017.mean,
    county    = as.character(countyState),
    county    = gsub(" County, New York", "", county)
  )

# visualizing how county population weights changed over time for top 20 counties
# as of 2017
library(tidyr)
library(ggplot2)

popWtOrd.2017 <-
  countyPop.NY_popWt %>%
  arrange(desc(popWt2017)) %>%
  select(county) %>%
  unlist()

# plot relative population weighting of top 20 counties as of 2017
countyPop.NY_popWt %>%
  select(county, popWt2015, popWt2016, popWt2017) %>%
  gather(key = "popWt.Yr", value = "popWt", -county) %>%
  mutate(popWt.Yr = gsub("popWt", "", popWt.Yr) %>% as.factor()) %>%
  mutate(county = factor(county, levels = popWtOrd.2017)) %>%
  filter(county %in% levels(county)[1:20]) %>%
  ggplot(aes(x = county, y=popWt, group=popWt.Yr, fill = popWt.Yr)) +
  geom_col(position = "dodge") +
  geom_hline(yintercept = 1, color = "darkgrey", size = 1) +
  labs(title = "Population weighting of NY Counties",
       subtitle = "Top 20 of 62 counties shown",
       x = "NY county", y = "Population weight \n (1 = statewide mean)",
       fill = NULL,
       caption = "Kings County 2017: 2.65mn ; Ulster County 2017: 0.18mn
                  \n County pop. mean \u2248 0.32mn each year"
       ) +
  scale_fill_manual(values = color_set8[6:8]) +
  theme_ds2() +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5),
        legend.position = c(0.75, 0.7),
        legend.direction = "horizontal")

# map showing county-level weight by year for each county
library(USAboundaries)
library(USAboundariesData)
library(sf)

us_counties(states = "New York", resolution = "high") %>%
  full_join(countyPop.NY_popWt %>%
            select(county, popWt2015, popWt2016, popWt2017) %>%
            gather(key = "year", value = "popWt", -county) %>%
            mutate(year = gsub("popWt", "", year) %>% as.integer()),
```

```
            by = c("name" = "county")) %>%
  ggplot() +
  geom_sf(aes(fill = popWt)) +
  facet_wrap(year ~ ., nrow = 2) +
  labs(title = "Population weighting of NY state counties by year") +
  scale_fill_viridis_c() +
  theme_ds2() +
  theme(legend.text = element_text(size = 9.5),
        legend.position = c(0.75, 0.24),
        panel.spacing.x = unit(1, "lines")
        )

# Note: some counties missing from temp data; need to see if
#       they have high pop. wts
path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/NY NOAA county temps.csv
tempDat <- read.csv(path)
## NY NOAA county temps.csv

tempDat.counties <-
  tempDat %>%
  group_by(county) %>%
  summarize(nObs = n()) %>%
  select(county, nObs)

temp_popMatch <-
  tempDat.counties %>%
  full_join(countyPop.NY_popWt, by = c("county" = "countyState"))

# Kings County (highest pop. wt) is missing, but is right next to Queens County;
#   assigning temp data to match for these
# Bronx County (5th highest pop. wt) also missing,
#   but right next to New York County; assigning these to match
# Richmond County (10th pop. wt) also missing, is next to Kings County;
#   assigning to also match Queens County
# Rockland County also missing, is next to Westchester County;
#   assigning these to match
# Montgomery County also missing, is next to Schenectady County;
#   assigning these to match
# Cortland County also missing, is next to Tompkins County;
#   assigning these to match
# Seneca County also missing, is next to Yates County;
#   assigning these to match

# need to assign Queens County data to neighboring Kings County
tempDat.Kings <-
  tempDat %>%
  filter(county == "Queens County, New York") %>%
  mutate(county = gsub("Queens", "Kings", county))

# need to assign Bronx County data to neighboring New York County
tempDat.Bronx <-
  tempDat %>%
  filter(county == "New York County, New York") %>%
```

```
  mutate(county = gsub("New York County", "Bronx County", county))

# need to assign Queens County data to neighbor-neighboring Richmond County
tempDat.Richmond <-
  tempDat %>%
  filter(county == "Queens County, New York") %>%
  mutate(county = gsub("Queens", "Richmond", county))

# need to assign Westchester County data to neighboring Rockland County
tempDat.Rockland <-
  tempDat %>%
  filter(county == "Westchester County, New York") %>%
  mutate(county = gsub("Westchester", "Rockland", county))

# need to assign Schenectady County data to neighboring Montgomery County
tempDat.Montgomery <-
  tempDat %>%
  filter(county == "Schenectady County, New York") %>%
  mutate(county = gsub("Schenectady", "Montgomery", county))

# need to assign Tompkins County data to neighboring Cortland County
tempDat.Cortland <-
  tempDat %>%
  filter(county == "Tompkins County, New York") %>%
  mutate(county = gsub("Tompkins", "Cortland", county))

# need to assign Yates County data to neighboring Seneca County
tempDat.Seneca <-
  tempDat %>%
  filter(county == "Yates County, New York") %>%
  mutate(county = gsub("Yates", "Seneca", county))

tempDat <-
  tempDat %>%
  bind_rows(tempDat.Kings, tempDat.Bronx, tempDat.Richmond, tempDat.Rockland,
            tempDat.Montgomery, tempDat.Cortland, tempDat.Seneca)

# rechecking county match
tempDat.counties <-
  tempDat %>%
  group_by(county) %>%
  summarize(nObs = n()) %>%
  select(county, nObs)

temp_popMatch <-
  tempDat.counties %>%
  full_join(countyPop.NY_popWt, by = c("county" = "countyState"))

# good to proceed; matching popWt to county temp data by year
#   first need to create popWt values for 2018 and 2019 (==popWt for 2017)
dat2 <-
  countyPop.NY_popWt %>%
  mutate(popWt2018 = popWt2017,
```

```
         popWt2019 = popWt2017,
         county = paste0(county, " County, New York")) %>%
  select(county, popWt2015, popWt2016, popWt2017, popWt2018, popWt2019) %>%
  gather(key = "popWt.Yr", value = "popWt", -county) %>%
  mutate(popWt.Yr = gsub("popWt", "", popWt.Yr))

tempDat <-
  tempDat %>%
  mutate(year = substring(DATE, 1, 4))

tempDat2 <-
  tempDat %>%
  full_join(dat2, by = c("county", "year" = "popWt.Yr")) %>%
  mutate(
    tempAvg.F.wtd = tempAvg.F * popWt,
    tempMax.F.wtd = tempMax.F * popWt,
    tempMin.F.wtd = tempMin.F * popWt
  ) %>%
  group_by(DATE) %>%
  # divide sum of weighted temp data by 62 == # counties
  #   (== sum of weight measure per year)
  # note: some smaller (low-weighted) counties don't have all available dates;
  #       I'm not positive but this may lead to lower weighting
  #       BUT should be very very minor since top-weighted countied have full data
  summarize(
    tempAvg.F = sum(tempAvg.F.wtd / 62),
    tempMax.F = sum(tempMax.F.wtd / 62),
    tempMin.F = sum(tempMin.F.wtd / 62)
  )

#write.csv(tempDat2,
#          file = "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/popWtdTempF_NY
#          row.names = F
#          )
## popWtdTempF_NY.csv
```

Combining ISO electrical data, NOAA weather data (NY only), Economic index data

```
library(dplyr)

## New England data first
neISO <-
  read.csv("C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/isoneHR_all.csv")
## isoneHR_all.csv

# convert to daily values
neISO.day <-
  neISO %>%
  group_by(date) %>%
  summarize(
```

```
    demand.MWh          = sum(demand.MWh),
    netGen.MWh          = sum(netGen.MWh),
    dayAheadDemand.MWh = sum(dayAheadDemand.MWh)
    )

# load population data - need to weight economic index data by state populations
path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/2015.16.17_PEP_2017_PEPA
countyPop <- read.csv(file = path)

countyPop <-
  # row 1 is variable descriptor data - removing this
  countyPop[-1,] %>%
  select(
    countyState = GEO.display.label,
    popEst2015  = est72015sex0_age999,
    popEst2016  = est72016sex0_age999,
    popEst2017  = est72017sex0_age999
    ) %>%
  mutate(
    countyState = countyState %>% as.character(),
    popEst2015  = popEst2015 %>% as.character() %>% as.integer(),
    popEst2016  = popEst2016 %>% as.character() %>% as.integer(),
    popEst2017  = popEst2017 %>% as.character() %>% as.integer(),
    popEst2018  = popEst2017,
    state       = countyState %>% strsplit(split = ", ") %>% sapply("[[", 2)
    )

statePop.NE <-
  countyPop %>%
  group_by(state) %>%
  summarize(
    popEst2015  = sum(popEst2015),
    popEst2016  = sum(popEst2016),
    popEst2017  = sum(popEst2017),
    popEst2018  = sum(popEst2018)
  ) %>%
  filter(state != "New York")

meanPop2015.NE <- mean(statePop.NE$popEst2015)
meanPop2016.NE <- mean(statePop.NE$popEst2016)
meanPop2017.NE <- mean(statePop.NE$popEst2017)
meanPop2018.NE <- mean(statePop.NE$popEst2018)

library(tidyr)

statePop.NE <-
  statePop.NE %>%
  mutate(
    popWt2015 = popEst2015 / meanPop2015.NE,
    popWt2016 = popEst2016 / meanPop2016.NE,
    popWt2017 = popEst2017 / meanPop2017.NE,
    popWt2018 = popEst2018 / meanPop2018.NE
  ) %>%
```

```
  select(-popEst2015, -popEst2016, -popEst2017, -popEst2018) %>%
  gather(key = "year", value = "popWt", -state) %>%
  mutate(year = gsub("popWt", "", year))

# load economic index data
path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/coincident-revised.xls"
library(readxl)
econIndex <- read_excel(path = path)
## coincident-revised.xls

econIndex.NE <-
  econIndex %>%
  select(Date, CT, MA, ME, NH, RI, VT) %>%
  gather(key = "state", value = "indexVal", -Date) %>%
  mutate(
    year  = substring(Date, 1, 4),
    month = substring(Date, 6, 7),
    state = state.name[match(state,state.abb)]
    ) %>%
  filter(Date >= "2015-06-30" & Date <= "2019-02-18")

pop_econ.NE <-
  econIndex.NE %>%
  full_join(statePop.NE, by = c("state", "year")) %>%
  mutate(indexVal.wtd = indexVal * popWt)

wtdEconIndex.NE <-
  pop_econ.NE %>%
  group_by(year, month) %>%
  # divide by 6 == divide by # of New England states
  summarize(econIndex.wtd = sum(indexVal.wtd) / 6)

# combine ISO data and econ index data for New England and save
neISO.day2 <-
  neISO.day %>%
  mutate(
    year  = substring(date, 1, 4),
    month = substring(date, 6, 7)
  ) %>%
  full_join(wtdEconIndex.NE, by = c("year", "month"))

#write.csv(neISO.day2,
#          file = "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/isoneDY_all.cs
#          row.names = F)
## isoneDY_all.csv

# same process for New York
path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/nyisoHR_all.csv"
## nyisoHR_all.csv

#nyISO <- read.csv(path)

nyISO <-
```

```
  nyisoHR_all %>%
  mutate(date = as.character(date))

nyISO.day <-
  nyISO %>%
  group_by(date) %>%
  summarize(
    # na.rm = T not needed here since there's no missingness
    demand.MWh         = sum(demand.MWh),
    netGen.MWh         = sum(netGen.MWh),
    dayAheadDemand.MWh = sum(dayAheadDemand.MWh)
    )

path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/popWtdTempF_NY.csv"
nyTempF <- read.csv(path)
## popWtdTempF_NY.csv

nyISO.day2 <-
  nyISO.day %>%
  full_join(nyTempF, by = c("date" = "DATE"))

path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/coincident-revised.xls"
library(readxl)
econIndex <- read_excel(path = path)
## coincident-revised.xls

econIndex.NY <-
  econIndex %>%
  select(Date, NY) %>%
  gather(key = "state", value = "indexVal", -Date) %>%
  mutate(
    year  = substring(Date, 1, 4),
    month = substring(Date, 6, 7),
    state = state.name[match(state,state.abb)]
    ) %>%
  filter(Date >= "2015-06-30" & Date <= "2019-02-18")

nyISO.day3 <-
  nyISO.day2 %>%
  mutate(
    year  = substring(date, 1, 4),
    month = substring(date, 6, 7)
    ) %>%
  full_join(econIndex.NY %>% select(indexVal, year, month),
            by = c("year", "month")
            ) %>%
  rename(econIndex.NY = indexVal)

#write.csv(nyISO.day3,
#          file = "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/nyisoDY_all.cs
#          row.names = F)
## nyisoDY_all.csv
```

Follow-up work with daily datasets (revising MWh to GWh, addressing some NA)

```
library(dplyr)

path <- "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/"

isone.Day <- read.csv(file = paste0(path, "isoneDY_all.csv"))
nyiso.Day <- read.csv(file = paste0(path, "nyisoDY_all.csv"))

summary(isone.Day) # NA's for 2019 month econ.index data
                   # - it's just not available yet
summary(nyiso.Day) # last row is fully NA

summary(nyISO.day3)

#nyisoDY_all <-
#  nyiso.Day[1:(nrow(nyiso.Day)-1),]

nyisoDY_all <-
  nyISO.day3[1:(nrow(nyISO.day3)-1),]

#write.csv(nyisoDY_all,
#          file = "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/nyisoDY_all.cs
#          row.names = F)
## nyisoDY_all.csv

# transforming megawatt-hour data to gigawatt-hour data
isone.Day <-
  isone.Day %>%
  mutate(
    demand.GWh         = demand.MWh/1000,
    netGen.GWh         = netGen.MWh/1000,
    dayAheadDemand.GWh = dayAheadDemand.MWh/1000
    ) %>%
  select(
    date, demand.GWh, netGen.GWh, dayAheadDemand.GWh, econIndex.wtd, year, month
    )

#write.csv(isone.Day,
#          file = "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/isoneDY_all.cs
#          row.names = F)
## isoneDY_all.csv

nyiso.Day <-
  nyisoDY_all %>%
  mutate(
    demand.GWh         = demand.MWh/1000,
    netGen.GWh         = netGen.MWh/1000,
    dayAheadDemand.GWh = dayAheadDemand.MWh/1000
    ) %>%
  select(
    date, demand.GWh, netGen.GWh, dayAheadDemand.GWh,
```

```
    tempAvg.F, tempMax.F, tempMin.F, econIndex.NY, year, month
    ) %>%
  # not at all sure why there are NA dayAheadDemand.GWh again (13 total)
  #   reapplying na.kalman here
  mutate(dayAheadDemand.GWh = if_else(!is.na(dayAheadDemand.GWh),
                                      dayAheadDemand.GWh,
                                      na.kalman(dayAheadDemand.GWh,
                                                model="auto.arima"))
         )

#write.csv(nyiso.Day,
#          file = "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/nyisoDY_all.cs
#          row.names = F)
## nyisoDY_all.csv
```

Report Code

Note: This code is in the exact same order as it appears in the report file.

```
# loading packages used in this analysis
library(astsa)
library(cowplot)
library(dplyr)
library(forecast)
library(ggplot2)
library(ggfortify) # use this with forecast package if using at all
library(MASS)
library(stringr)
library(tidyr)

# ensuring MASS::select() doesn't override dplyr::select()
select <- dplyr::select

# loading the data and filtering to pre-2019 observations
path.nyiso <-
  "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/nyisoDY_all.csv"
## nyisoDY_all.csv
path.isone <-
  "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/isoneDY_all.csv"
## isoneDY_all.csv

nyiso <-
  read.csv(path.nyiso) %>%
  mutate(date = as.Date(date)) %>%
  filter(year < 2019)

isone <-
  read.csv(path.isone) %>%
  mutate(date = as.Date(date)) %>%
  filter(year < 2019)

# quick glimpse of the datasets
```

```
#head(nyiso, 2) %>%
#  bind_rows(tail(nyiso, 2))
#head(isone, 2) %>%
#  bind_rows(tail(isone, 2))
```

Detailed Analysis

Exploratory Data Analysis

NYISO data

```
# setting some x-axis label and gridline defaults
monthly_x_setup <-
  list(
    theme_ds2(),
    scale_x_date(date_breaks      ="3 months",
                 date_minor_breaks="1 month",
                 date_labels      ="%b-%Y"),
    theme(axis.text.x=element_text(angle=45, hjust=1, vjust=1),
          panel.grid.major.x=element_line(color="lightgrey", linetype=1),
          panel.grid.minor.x=element_line(color="lightgrey", linetype=1))
    )

# time series plot of daily demand and net generation
nyiso %>%
  select(date, demand.GWh, netGen.GWh) %>%
  gather(key="metric", value="value", -date) %>%
  ggplot(aes(x=date, y=value, color=metric)) +
  geom_line() +
  annotate("text", x=as.Date("2016-01-01"), y=520, label="Demand",
           color=color_set8[1], hjust = 0.5, size = 4.5) +
  annotate("text", x=as.Date("2015-06-01"), y=280, label="Net Generation",
           color=color_set8[2], hjust = 0, size = 4.5) +
  labs(title="NYISO daily electricity demand and net generation",
       subtitle = "Demand - Net Generation = Imports",
       x="Date", y="Quantity of electrical power \n (gigawatt-hours)",
       caption="1 GWh = 9,090 Nissan Leaf engines at full capacity for 1 hour",
       color=NULL) +
  scale_color_manual(values=color_set8[1:2], guide=F) +
  monthly_x_setup
```

```
# time series plot of daily demand and scaled daily temperature
ggplot() +
  geom_line(
    data = nyiso %>%
           select(date, demand.GWh),
    aes(x=date, y=demand.GWh), color=color_set8[1]
    ) +
  geom_line(
    data =
      nyiso %>%
      select(date, tempAvg.F, tempMax.F, tempMin.F) %>%
```

```
      mutate(tempAvg.F = tempAvg.F * 8,
             tempMax.F = tempMax.F * 8,
             tempMin.F = tempMin.F * 8) %>%
      gather(key="metric", value="value", -date),
    aes(x=date, y=value, color=metric), alpha=0.6
    ) +
  annotate("text", x=as.Date("2015-10-30"), y=180, label="Demand",
           color=color_set8[1], hjust = 0.9, size = 4.5) +
  annotate("text", x=as.Date("2016-08-15"), y=180, label="Max. temp.",
           color=color_set8[6], hjust = 0.5, size = 4.5) +
  annotate("text", x=as.Date("2016-08-15"), y=130, label="Mean temp.",
           color=color_set8[7], hjust = 0.5, size = 4.5) +
  annotate("text", x=as.Date("2016-08-15"), y=80, label="Min. temp.",
           color=color_set8[8], hjust = 0.5, size = 4.5) +
  labs(title="NYISO daily electricity demand and temperature summaries",
       x="Date", y="Electricity demand (gigawatt-hours)",
       color=NULL) +
  scale_color_manual(values=color_set8[c(7,6,8)], guide=F) +
  scale_y_continuous(sec.axis=sec_axis(~. / 8,  name="Temperature (F)")) +
  monthly_x_setup
```

```
nyiso <-
  nyiso %>%
  mutate(absTempAvgMinus50.F = abs(tempAvg.F - 50))

# showing max / mean / min temperatures are highly correlated
nyiso %>%
  select(tempMax.F, tempAvg.F, tempMin.F) %>%
  cor() %>%
  round(4)
```

```
# evaluating the new variable in a schmancy new plot
secAxisBreaks <- seq(0, 50, 10)

nyiso %>%
  select(date, demand.GWh, absTempAvgMinus50.F) %>%
  mutate(absTempAvgMinus50.F = absTempAvgMinus50.F * 7 +
                               mean(nyiso$demand.GWh)) %>%
  gather(key="metric", value="value", -date) %>%
  ggplot(aes(x=date, y=value, color=metric)) +
  geom_line() +
  annotate("text", x=as.Date("2016-10-01"), y=340, label="Demand",
           color=color_set8[1], hjust = 0.5, size = 4.5) +
  annotate("text", x=as.Date("2016-11-01"), y=720,
           label="Absolute temperature \n deviation from 50F",
           color=color_set8[7], hjust = 0.5, size = 4.5) +
  labs(title="NYISO daily electricity demand and |mean temp. dist. from 50F|",
       x="Date", y="Electricity demand (gigawatt-hours)",
       color=NULL) +
  scale_color_manual(values=color_set8[c(7,1)], guide=F) +
  scale_y_continuous(sec.axis=sec_axis(~(. / 7 - mean(nyiso$demand.GWh)/7),
                                       breaks=secAxisBreaks,
                                       labels=secAxisBreaks,
```

```
                                        name  ="Absolute temperature
                                          \n deviation from 50F")) +
  monthly_x_setup
```

```
# considering electricity demand and economic activity index
nyiso %>%
  mutate(econIndex.NY = econIndex.NY * 4)%>%
  select(date, demand.GWh, econIndex.NY) %>%
  gather(key="metric", value="value", -date) %>%
  ggplot(aes(x=date, y=value, color=metric)) +
  geom_line() +
  annotate("text", x=as.Date("2016-10-01"), y=340, label="Demand",
           color=color_set8[1], hjust = 0.5, size = 4.5) +
  annotate("text", x=as.Date("2017-02-01"), y=550, label="Economic \n index",
           color=color_set8[8], hjust = 0.5, size = 4.5) +
  labs(title="NYISO daily electricity demand and monthly economic activity index",
       x="Date", y="Electricity demand (gigawatt-hours)",
       color=NULL) +
  scale_color_manual(values=color_set8[c(1,8)], guide=F) +
  scale_y_continuous(sec.axis=sec_axis(~. / 4,
                                        name  ="Economic activity index
                                          \n (100 = 2007 annual mean)")) +
  monthly_x_setup
```

```
# ACF and PCF plots of demand.GWh - with GGPlot2 styling
# default lags (aligns with monthly span reasonably well)
acfPlot <-
  autoplot(acf(nyiso$demand.GWh, plot=F)) +
  geom_hline(yintercept = 0) +
  labs(title="ACF and PACF for nyiso$demand.GWh") +
  theme_ds2()

pacfPlot <-
  autoplot(pacf(nyiso$demand.GWh, plot=F)) +
  geom_hline(yintercept = 0) +
  labs(title=expression("lag.max "%~~%" 31"), y="Partial ACF") +
  theme_ds2()

cowplot::plot_grid(acfPlot, pacfPlot, ncol=2)

# default lag.max=366 (aligns with annual span)
acfPlot <-
  autoplot(acf(nyiso$demand.GWh, plot=F, lag.max=366)) +
  geom_hline(yintercept = 0) +
  labs(title="ACF and PACF for nyiso$demand.GWh") +
  theme_ds2()

pacfPlot <-
  autoplot(pacf(nyiso$demand.GWh, plot=F, lag.max=366)) +
  geom_hline(yintercept = 0) +
  labs(title="lag.max=366", y="Partial ACF") +
  theme_ds2()
```

```
cowplot::plot_grid(acfPlot, pacfPlot, ncol=2)
```

```
# comparing actual and predicted demand (both from the original dataset)
tibble(
  date   = rep(nyiso$date[1:{nrow(nyiso)-1}], 2),
  metric = rep(c("demand.GWh", "predictedDemand.GWh"), each = {nrow(nyiso)-1}),
  value  = c(nyiso$demand.GWh[1:{nrow(nyiso)-1}],
             nyiso$dayAheadDemand.GWh[2:nrow(nyiso)])
  ) %>%
  ggplot(aes(x=date, y=value, color=metric)) +
  geom_line() +
  annotate("text", x=as.Date("2016-02-01"), y=550, label="Demand",
           color=color_set8[1], hjust = 0.5, size = 4.5) +
  annotate("text", x=as.Date("2017-02-01"), y=550,
           label="'Day ahead' \n prediction",
           color=color_set8[5], hjust = 0.5, size = 4.5) +
  labs(title="NYISO daily electricity demand and aligned 'day ahead' demand",
       x="Date", y="Electricity demand (gigawatt-hours)",
       color=NULL) +
  scale_color_manual(values=color_set8[c(1,5)], guide=F) +
  monthly_x_setup
```

```
# comparing ccf for average temperature or absolute deviance from 50F for demand
par(mfrow=c(1,2))
ccf2(x=nyiso$tempAvg.F, y=nyiso$demand.GWh)
box(bty="7", col="lightgrey")

ccf2(x=nyiso$absTempAvgMinus50.F, y=nyiso$demand.GWh)
box(bty="7", col="lightgrey")
```

ISONE data

```
# time series plot of daily demand and net generation
isone %>%
  select(date, demand.GWh, netGen.GWh) %>%
  mutate(ximports.GWh = (demand.GWh - netGen.GWh)+250) %>%
  gather(key="metric", value="value", -date) %>%
  ggplot(aes(x=date, y=value, color=metric)) +
  geom_line() +
  annotate("text", x=as.Date("2016-02-01"), y=450, label="Demand",
           color=color_set8[3], hjust = 0.5, size = 4.5) +
  annotate("text", x=as.Date("2016-02-01"), y=200, label="Net Generation",
           color=color_set8[4], hjust = 0.5, size = 4.5) +
  labs(title="ISONE daily electricity demand and net generation",
       subtitle = "Demand - Net Generation = Imports (dark grey)",
       x="Date", y="Quantity of electrical power \n (gigawatt-hours)",
       color=NULL) +
  scale_color_manual(values=c(color_set8[3:4], "grey50"), guide=F) +
  scale_y_continuous(sec.axis=sec_axis(~. - 250,
                                       name  ="Electric imports (GWh)")) +
  monthly_x_setup
```

```
# ACF and PCF plots of demand.GWh - with GGPlot2 styling
# default lags (aligns with monthly span reasonably well)
acfPlot <-
  autoplot(acf(isone$demand.GWh, plot=F)) +
  geom_hline(yintercept = 0) +
  labs(title="ACF and PACF for isone$demand.GWh") +
  theme_ds2()

pacfPlot <-
  autoplot(pacf(isone$demand.GWh, plot=F)) +
  geom_hline(yintercept = 0) +
  labs(title=expression("lag.max "%~~%" 31"), y="Partial ACF") +
  theme_ds2()

cowplot::plot_grid(acfPlot, pacfPlot, ncol=2)

# default lag.max=366 (aligns with annual span)
acfPlot <-
  autoplot(acf(isone$demand.GWh, plot=F, lag.max=366)) +
  geom_hline(yintercept = 0) +
  labs(title="ACF and PACF for nyiso$demand.GWh") +
  theme_ds2()

pacfPlot <-
  autoplot(pacf(isone$demand.GWh, plot=F, lag.max=366)) +
  geom_hline(yintercept = 0) +
  labs(title="lag.max=366", y="Partial ACF") +
  theme_ds2()

cowplot::plot_grid(acfPlot, pacfPlot, ncol=2)
```

NYISO analysis

Spectral analysis

```
# spectral analysis
# creating the time series
meanCtrDemand.GWh <- nyiso$demand.GWh - mean(nyiso$demand.GWh)

nyisoDemand.GWh <- ts(meanCtrDemand.GWh, start=nyiso$date[1], frequency=365.25)

# computing and plotting the scaled periodograms
P = Mod(2*fft(nyisoDemand.GWh)/length(nyisoDemand.GWh))^2
Fr = 0:(length(nyisoDemand.GWh)-1)/length(nyisoDemand.GWh)

tibble(Fr, P) %>%
  # show just the first half since there's symmetry at Fr=0.5
  filter(Fr <= 0.5) %>%
  ggplot(aes(x=Fr %>% as.numeric(), y = P %>% as.numeric(),
             color = P %>% as.numeric())
         ) +
```

```
  geom_line(color=color_set8[1]) +
  geom_point(alpha=0.9, size=3) +
  labs(title="nyiso$demand.GWh scaled periodogram",
       x="Frequency", y="Scaled periodogram") +
  scale_color_viridis_c(direction=-1, guide=F) +
  theme_ds2()
```

```
# plot summarizing the "most important" frequencies
# first filter to only Frequencies <= 0.5 because of symmetry about 0.5
Fr.red <- Fr[Fr<=0.5]
P.red  <- P[Fr<=0.5]

fractionDat <-
  tibble(
    Freq = as.character(fractions(Fr.red)) %>%
           factor(levels=as.character(fractions(Fr.red))[order(P.red,
                                                               decreasing=T)]),
    Pdgm = P.red
    ) %>%
  arrange(desc(Pdgm)) %>%
  head(10) %>%
# want the top 10 Freq fractions to have common denominator 1280
# need to scale 3rd (1/320 >> 4/1280), 5th (1/256 >> 5/1280),
#               6th (1/640 >> 2/1280)
#               8th (183/640 >> 366/1280), and 10th (3/640 >> 6/1280)
# requires converting Freq to character then 'resetting' Freq factor levels after
  mutate(
    Freq = as.character(Freq),
    Freq = if_else(Freq == "1/320", "4/1280",
              if_else(Freq == "1/256", "5/1280",
                 if_else(Freq == "1/640", "2/1280",
                    if_else(Freq == "183/640", "366/1280",
                       if_else(Freq == "3/640", "6/1280", Freq))))
              ),
    # returning to factor - the pasted c() string are the rescaled numerators
    Freq = factor(Freq, levels=paste(c(7,3,4,183,5,2,13,366,11,6),1280, sep="/"))
    )

fractionDat %>%
  ggplot(aes(x=Freq, y=Pdgm)) +
  geom_col(fill=color_set8[1]) +
  annotate("text", x=4, y=2850, hjust=0,
           label="Top 4 frequencies in annual terms:") +
  annotate("text", x=4, y=2500, hjust=0,
           label=expression(
                  omega*" = 7/1280"%~~%"0.00547 ; 2/365.25"%~~%"0.00548")) +
  annotate("text", x=4, y=2000, hjust=0,
           label=expression(
                  omega*" = 3/1280"%~~%"0.00234 ; 0.85/365.25"%~~%"0.00233")) +
  annotate("text", x=4, y=1750, hjust=0,
           label=expression(
                  omega*" = 4/1280"%~~%"0.00313 ; 1.14/365.25"%~~%"0.00312")) +
  annotate("text", x=4, y=1250, hjust=0,
```

```
          label=expression(
                omega*" = 183/1280"%~~%"0.14297 ; 52.22/365.25"%~~%"0.14297")) +
  labs(title="Top 10 frequencies by scaled periodogram value",
       subtitle="nyisoDemand.GWh",
       x="Frequency", y="Scaled Periodogram") +
  theme(axis.text.x = element_text(angle=45, hjust=1, vjust=1))
```

Regression modelling - frequencies, temperatures, weekdays, months (plus ARMA errors)

```
# sin/cos variables based on class code example for Philly weather
# from 3/27 class (7th set of class R code, around line 206 in .R file)

nyiso2 <-
  nyiso %>%
  mutate(
    # creating a time index, quadratic time term, weekday/month variables,
    # key frequency sine/cosine pairs, and absolute temp deviance from 50F
    dayIndex      = 1:nrow(nyiso),
    dayIndex2.ctr = (dayIndex - mean(dayIndex))^2,
    # need an indicator variable for each weekday and each month
    # (except ref levels)
    # Sunday is the reference level of weekday
    weekday       = weekdays(date),
    isSun         = (weekday == "Sunday") %>% as.integer(),
    isMon         = (weekday == "Monday") %>% as.integer(),
    isTue         = (weekday == "Tuesday") %>% as.integer(),
    isWed         = (weekday == "Wednesday") %>% as.integer(),
    isThu         = (weekday == "Thursday") %>% as.integer(),
    isFri         = (weekday == "Friday") %>% as.integer(),
    isSat         = (weekday == "Saturday") %>% as.integer(),
    # wasn't able to incorporate month(s) into the model
    # January is the reference level of month
    month         = months(date),
    isJan         = (month == "January") %>% as.integer(),
    isFeb         = (month == "February") %>% as.integer(),
    isMar         = (month == "March") %>% as.integer(),
    isApr         = (month == "April") %>% as.integer(),
    isMay         = (month == "May") %>% as.integer(),
    isJun         = (month == "June") %>% as.integer(),
    isJul         = (month == "July") %>% as.integer(),
    isAug         = (month == "Aug") %>% as.integer(),
    isSep         = (month == "Sep") %>% as.integer(),
    isOct         = (month == "Oct") %>% as.integer(),
    isNov         = (month == "Nov") %>% as.integer(),
    isDec         = (month == "Dec") %>% as.integer(),
    isDecJanFeb   = isDec + isJan + isFeb,
    isMarAprMay   = isMar + isApr + isMay,
    isJunJulAug   = isJun + isJul + isAug,
    isSepOctNov   = isSep + isOct + isNov,
    prevDayDemand = lag(demand.GWh),
    sin.1xYr      = sin(2*pi*(dayIndex)*(1/365.25)),
```

```
    cos.1xYr      = cos(2*pi*(dayIndex)*(1/365.25)),
    sin.2xYr      = sin(2*pi*(dayIndex)*(2/365.25)),
    cos.2xYr      = cos(2*pi*(dayIndex)*(2/365.25)),
    # next 2 weren't stat. sig. using either 52 or 60
    sin.2Mo       = sin(2*pi*(dayIndex)*(52/365.25)),
    cos.2Mo       = cos(2*pi*(dayIndex)*(52/365.25)),
    absTempAvgMinus50.F = abs(tempAvg.F - 50)
    )

# fitting the model
nyisoDemandGWh.lm <-
  lm(demand.GWh ~ dayIndex + dayIndex2.ctr + prevDayDemand +
                  isMon + isTue + isWed + isThu + isFri + isSat +
                  sin.1xYr + cos.1xYr + sin.2xYr + cos.2xYr +
                  sin.2Mo + cos.2Mo + absTempAvgMinus50.F,
                  data=nyiso2)

# no 'stargazing'
options(show.signif.stars = F)

#summary(nyisoDemandGWh.lm)

# sin/cos.2Mo terms aren't stat. sig. - updating the model
nyisoDemandGWh.lm <- update(nyisoDemandGWh.lm,
                            .~. -sin.2Mo -cos.2Mo)

# summary of the 'final' model
summary(nyisoDemandGWh.lm)
```

```
# ACF and PCF plots of demand.GWh - with GGPlot2 styling
acfPlot <-
  autoplot(acf(nyisoDemandGWh.lm$residuals, plot=F)) +
  geom_hline(yintercept = 0) +
  labs(subtitle="ACF & PACF for nyisoDemandGWh.lm$residuals") +
  theme_ds2()

pacfPlot <-
  autoplot(pacf(nyisoDemandGWh.lm$residuals, plot=F)) +
  geom_hline(yintercept = 0) +
  labs(title=NULL, y="Partial ACF") +
  theme_ds2()

cowplot::plot_grid(acfPlot, pacfPlot, ncol=1)
```

```
# determining a good ARMA model for errors using the AIC matrix approach
uprLim = 5
aicMat = matrix(double((uprLim+1)^2), uprLim+1, uprLim+1)
for (i in 0:uprLim){
  for (j in 0:uprLim){
    aicMat[i+1,j+1] = sarima(nyisoDemandGWh.lm$residuals, i, 0, j,
                             details = F)$AIC
    rownames(aicMat) <- paste0("p:", 0:uprLim)
    colnames(aicMat) <- paste0("q:", 0:uprLim)
```

```
    }
  }
# identify the row, column index of the minimum value
which(aicMat == min(aicMat), arr.ind = T)

aicMat %>% round(3)
```

```
sarima(nyisoDemandGWh.lm$residuals, p=0, d=0, q=1)
```

Predicting from the model

```
# decent online reference - just tucking this away in the comments
# https://otexts.com/fpp2/

# loading the data, filtering to Jan through 14 Feb 2019 observations,
# computing needed covariates
path.nyiso <-
  "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/nyisoDY_all.csv"
## nyisoDY_all.csv

nyisoEarly2019 <-
  read.csv(path.nyiso) %>%
  mutate(date = as.Date(date)) %>%
  filter(year == 2019, date <= as.Date("2019-02-14"))

nyisoEarly2019 <-
  nyisoEarly2019 %>%
  mutate(
    # creating only the variables needed for forecasting
    intercept     = 1,
    dayIndex      = (nrow(nyiso2)+1):(nrow(nyiso2) + nrow(nyisoEarly2019)),
    dayIndex2.ctr = (dayIndex - mean(dayIndex))^2,
    # need an indicator variable for each weekday and each month
    # (except ref levels)
    # Sunday is the reference level of weekday
    weekday       = weekdays(date),
    isSun         = (weekday == "Sunday") %>% as.integer(),
    isMon         = (weekday == "Monday") %>% as.integer(),
    isTue         = (weekday == "Tuesday") %>% as.integer(),
    isWed         = (weekday == "Wednesday") %>% as.integer(),
    isThu         = (weekday == "Thursday") %>% as.integer(),
    isFri         = (weekday == "Friday") %>% as.integer(),
    isSat         = (weekday == "Saturday") %>% as.integer(),
    month         = months(date),
    # first entry will be NA for prevDayDemand - need to carry in 31 Dec 2018
    prevDayDemand = if_else(dayIndex==(nrow(nyiso2)+1),
                            nyiso2$demand.GWh[nrow(nyiso2)],
                            lag(demand.GWh)),
    sin.1xYr      = sin(2*pi*(dayIndex)*(1/365.25)),
    cos.1xYr      = cos(2*pi*(dayIndex)*(1/365.25)),
    sin.2xYr      = sin(2*pi*(dayIndex)*(2/365.25)),
```

```
    cos.2xYr      = cos(2*pi*(dayIndex)*(2/365.25)),
    absTempAvgMinus50.F = abs(tempAvg.F - 50)
    )

betaHats <- coef(nyisoDemandGWh.lm)

# column names for model covariates from the regression component
modelCovariates <-
  (summary(nyisoDemandGWh.lm)$terms %>% as.character())[3] %>%
  strsplit(split = "\\+") %>%
  unlist() %>%
  str_trim()

predXMat <-
  nyisoEarly2019 %>%
  select(intercept, modelCovariates) %>%
  as.matrix()

lmPred    <- predXMat %*% betaHats
```

```
errorPred <- sarima.for(nyisoDemandGWh.lm$residuals,
                        n.ahead = nrow(nyisoEarly2019),
                        p=0, d=0, q=1)
```

```
modelPred <- lmPred + errorPred$pred

nyisoEarly2019 <-
  nyisoEarly2019 %>%
  mutate(modEst = modelPred %>% as.numeric())

# plotting the results: actual vs. fitted "in-model" values plus predictions
nyiso.demandAllDat <-
  nyiso2 %>%
  # don't have 'day 1' estimate due to missing prevDayDemand
  mutate(modEst = c(NA, unname(nyisoDemandGWh.lm$fitted.values))) %>%
  full_join(nyisoEarly2019) %>%
  mutate(modelResid = demand.GWh - modEst)

nyiso.demandAllPlotDat <-
  nyiso.demandAllDat %>%
  select(date, demand.GWh, modEst) %>%
  gather(key="metric", value="value", -date) %>%
  mutate(qqch = if_else(metric == "demand.GWh" & date <= as.Date("2018-12-31"),
                        "Obs., in-sample",
                  if_else(metric == "demand.GWh" & date > as.Date("2018-12-31"),
                          "Obs., out-of-sample",
                    if_else(metric == "modEst" & date <= as.Date("2018-12-31"),
                            "Model, in-sample", "Model, out-of-sample")))
         )

# full view
ggplot(nyiso.demandAllPlotDat, aes(x=date, y=value)) +
  geom_line(data=nyiso.demandAllPlotDat %>% filter(grepl("Obs.", qqch)),
```

```
            aes(color=qqch)) +
  geom_point(data=nyiso.demandAllPlotDat %>% filter(grepl("Model", qqch),
                                                    date >= as.Date("2015-07-02")),
             aes(fill=qqch), shape=21,
             alpha=0.8) +
  labs(title="NYISO demand.GWh observed vs model fit",
       subtitle="1 Jul 2015 through 14 Feb 2019",
       x="Date", y="Electrical power demand \n (gigawatt-hours)",
       caption="No model estimate for 1 Jul 2015 (no 'previous day demand')",
       color=NULL, fill=NULL) +
  scale_color_manual(values=color_set8[c(1,7)]) +
  scale_fill_manual(values=color_set8[c(2,6)]) +
  monthly_x_setup +
  theme(legend.direction = "vertical",
        legend.margin = margin(c(0,0,0,0)),
        legend.position = "top")

# focusing on Oct 2018 forward
ggplot(nyiso.demandAllPlotDat,
       aes(x=date, y=value)) +
  geom_line(data=nyiso.demandAllPlotDat %>% filter(grepl("Obs.", qqch),
                                                   date >= as.Date("2018-10-01")),
            aes(color=qqch)) +
  geom_point(data=nyiso.demandAllPlotDat %>% filter(grepl("Model", qqch),
                                                    date >= as.Date("2018-10-01")),
             aes(fill=qqch), shape=21,
             alpha=0.8) +
  labs(title="NYISO demand.GWh observed vs model fit",
       subtitle="1 Oct 2018 through 14 Feb 2019",
       x="Date", y="Electrical power demand \n (gigawatt-hours)",
       color=NULL, fill=NULL) +
  scale_color_manual(values=color_set8[c(1,7)]) +
  scale_fill_manual(values=color_set8[c(2,6)]) +
  monthly_x_setup +
  # need to revise the the breaks in the scale_x_date part of monthly_x_setup
  scale_x_date(date_breaks      ="1 month",
               date_minor_breaks="1 month",
               date_labels      ="%b-%Y") +
  theme(legend.direction = "vertical",
        legend.margin = margin(c(0,0,0,0)),
        legend.position = "top")
```

```
# in- vs out-of-sample RMSE for model vs. 'aligned day-ahead demand'
#   'aligned day-ahead demand' might technically be considered to be all in-sample
# to allow for apples-to-apples comparison,
#   need to compare 02 Jul 2015 through 14 Feb 2019
#   (not 01 Jul 2015 - missing prevDayDemand for the model)

inSampRMSE <-
  nyiso.demandAllDat %>%
  mutate(alignedDayAheadDemand.GWh = lag(dayAheadDemand.GWh)) %>%
  filter(date >= as.Date("2015-07-02"),
         date <= as.Date("2018-12-31")) %>%
```

```
  mutate(demandVSalignedDayAheadDemand = demand.GWh - alignedDayAheadDemand.GWh)

inSampRMSE <-
  tibble(source = c("model", "alignedDayAheadDemand"),
         where  = "in-sample",
         RMSE   = c(sqrt(sum(inSampRMSE$modelResid^2)/nrow(inSampRMSE)),
                    sqrt(sum(inSampRMSE$demandVSalignedDayAheadDemand^2)/
                         nrow(inSampRMSE))
                    ))

outSampRMSE <-
  nyiso.demandAllDat %>%
  mutate(alignedDayAheadDemand.GWh = lag(dayAheadDemand.GWh)) %>%
  filter(date > as.Date("2018-12-31"),
         date <= as.Date("2019-02-14")) %>%
  mutate(demandVSalignedDayAheadDemand = demand.GWh - alignedDayAheadDemand.GWh)

outSampRMSE <-
  tibble(source = c("model", "alignedDayAheadDemand"),
         where  = "out-of-sample",
         RMSE   = c(sqrt(sum(outSampRMSE$modelResid^2)/nrow(outSampRMSE)),
                    sqrt(sum(outSampRMSE$demandVSalignedDayAheadDemand^2)/
                         nrow(outSampRMSE))
                    ))

inSampRMSE %>%
  full_join(outSampRMSE, by=c("source", "where", "RMSE"))
```

```
# checking residuals
nyiso.demandAllDat %>%
  mutate(absResid = abs(modelResid)) %>%
  arrange(desc(absResid)) %>%
  mutate(date = format(date, format="%d %b %Y")) %>%
  select(date, prevDayDemand, weekday, demand.GWh, modEst, modelResid) %>%
  mutate_if(is.numeric, round, digits=1) %>%
  head(10)
```

ISONE analysis

Spectral analysis

```
# spectral analysis

# reminder: isone is filtered to 01 Jul 2015 to 31 Dec 2018

# creating the time series
meanCtrDemand.GWh <- isone$demand.GWh - mean(isone$demand.GWh)

isoneDemand.GWh <- ts(meanCtrDemand.GWh, start=isone$date[1], frequency=365.25)

# computing and plotting the scaled periodograms
```

```
P = Mod(2*fft(isoneDemand.GWh)/length(isoneDemand.GWh))^2
Fr = 0:(length(isoneDemand.GWh)-1)/length(isoneDemand.GWh)

tibble(Fr, P) %>%
  # show just the first half since there's symmetry at Fr=0.5
  filter(Fr <= 0.5) %>%
  ggplot(aes(x=Fr %>% as.numeric(), y = P %>% as.numeric(),
             color = P %>% as.numeric())
         ) +
  geom_line(color=color_set8[3]) +
  geom_point(alpha=0.9, size=3) +
  labs(title="isone$demand.GWh scaled periodogram",
       x="Frequency", y="Scaled periodogram") +
  scale_color_viridis_c(direction=-1, guide=F) +
  theme_ds2()
```

```
# plot summarizing the "most important" frequencies
# first filter to only Frequencies <= 0.5 because of symmetry about 0.5
Fr.red <- Fr[Fr<=0.5]
P.red  <- P[Fr<=0.5]

fractionDat <-
  tibble(
    Freq = as.character(fractions(Fr.red)) %>%
           factor(levels=as.character(fractions(Fr.red))[order(P.red,
                                                               decreasing=T)]),
    Pdgm = P.red
    ) %>%
  arrange(desc(Pdgm)) %>%
  head(10) %>%
# want the top 10 Freq fractions to have common denominator 1280
# interestingly, only 10th differs from NYISO top 10 (not exact same order/Pdgm)
# need to scale 6th (1/256 >> 5/1280), 7th (3/640 >> 6/1280),
  #             8th (183/640 >> 366/1280),
#               9th (1/320 >> 4/1280), and 10th (3/320 >> 12/1280)
# requires converting Freq to character then 'resetting' Freq factor levels after
  mutate(
    Freq = as.character(Freq),
    Freq = if_else(Freq == "1/256", "5/1280" ,
              if_else(Freq == "3/640", "6/1280",
                 if_else(Freq == "183/640", "366/1280",
                    if_else(Freq == "1/320", "4/1280",
                       if_else(Freq == "3/320", "12/1280", Freq))))
              ),
    # returning to factor - the pasted c() string are the rescaled numerators
    Freq = factor(Freq, levels=paste(c(7,183,3,11,13,5,6,366,4,12),1280, sep="/"))
    )

fractionDat %>%
  ggplot(aes(x=Freq, y=Pdgm)) +
  geom_col(fill=color_set8[3]) +
  annotate("text", x=4, y=2000, hjust=0,
           label="Top 3 frequencies in annual terms:") +
```

```
  annotate("text", x=4, y=1700, hjust=0,
           label=expression(
                  omega*" = 7/1280"%~~%"0.00547 ; 2/365.25"%~~%"0.00548")) +
  annotate("text", x=4, y=1400, hjust=0,
           label=expression(
                  omega*" = 183/1280"%~~%"0.14297 ; 52.22/365.25"%~~%"0.14297")) +
  annotate("text", x=4, y=1100, hjust=0,
           label=expression(
                  omega*" = 3/1280"%~~%"0.00234 ; 0.85/365.25"%~~%"0.00233")) +
  labs(title="Top 10 frequencies by scaled periodogram value",
       subtitle="isoneDemand.GWh",
       x="Frequency", y="Scaled Periodogram") +
  theme(axis.text.x = element_text(angle=45, hjust=1, vjust=1))
```

Regression modelling - frequencies, temperatures, weekdays, months (plus ARMA errors)

```
# sin/cos variables based on class code example for Philly weather
# from 3/27 class (7th set of class R code, around line 206 in .R file)

isone2 <-
  isone %>%
  mutate(
    # creating a time index, quadratic time term, weekday/month variables,
    # and key frequency sine/cosine pairs
    # dayIndex and dayIndex2.ctr have greatest p-vals (0.22-0.30) in first model
    dayIndex      = 1:nrow(isone),
    dayIndex2.ctr = (dayIndex - mean(dayIndex))^2,
    # need an indicator variable for each weekday and each month
    # (except ref levels)
    # Sunday is the reference level of weekday
    weekday       = weekdays(date),
    isSun         = (weekday == "Sunday") %>% as.integer(),
    isMon         = (weekday == "Monday") %>% as.integer(),
    isTue         = (weekday == "Tuesday") %>% as.integer(),
    isWed         = (weekday == "Wednesday") %>% as.integer(),
    isThu         = (weekday == "Thursday") %>% as.integer(),
    isFri         = (weekday == "Friday") %>% as.integer(),
    isSat         = (weekday == "Saturday") %>% as.integer(),
    month         = months(date),
    prevDayDemand = lag(demand.GWh),
    sin.1xYr      = sin(2*pi*(dayIndex)*(1/365.25)),
    cos.1xYr      = cos(2*pi*(dayIndex)*(1/365.25)),
    sin.2xYr      = sin(2*pi*(dayIndex)*(2/365.25)),
    cos.2xYr      = cos(2*pi*(dayIndex)*(2/365.25)),
    #
    sin.2Mo       = sin(2*pi*(dayIndex)*(52/365.25)),
    cos.2Mo       = cos(2*pi*(dayIndex)*(52/365.25))
    )

# fitting the model
isoneDemandGWh.lm <-
```

```
  lm(demand.GWh ~ dayIndex + dayIndex2.ctr + prevDayDemand +
                  isMon + isTue + isWed + isThu + isFri + isSat +
                  sin.1xYr + cos.1xYr + sin.2xYr + cos.2xYr +
                  sin.2Mo + cos.2Mo,
                  data=isone2)


#summary(isoneDemandGWh.lm)

# sin/cos.2Mo terms aren't stat. sig. - updating the model
isoneDemandGWh.lm <- update(isoneDemandGWh.lm,
                            .~. -dayIndex2.ctr -sin.2Mo -cos.2Mo)

# summary of the 'final' model
summary(isoneDemandGWh.lm)
```

```
# ACF and PCF plots of demand.GWh - with GGPlot2 styling
acfPlot <-
  autoplot(acf(isoneDemandGWh.lm$residuals, plot=F)) +
  geom_hline(yintercept = 0) +
  labs(subtitle="ACF & PACF for isoneDemandGWh.lm$residuals") +
  theme_ds2()

pacfPlot <-
  autoplot(pacf(isoneDemandGWh.lm$residuals, plot=F)) +
  geom_hline(yintercept = 0) +
  labs(title=NULL, y="Partial ACF") +
  theme_ds2()

cowplot::plot_grid(acfPlot, pacfPlot, ncol=1)
```

```
# determining a good ARMA model for errors using the AIC matrix approach
uprLim = 5
aicMat = matrix(double((uprLim+1)^2), uprLim+1, uprLim+1)
for (i in 0:uprLim){
  for (j in 0:uprLim){
    aicMat[i+1,j+1] = sarima(isoneDemandGWh.lm$residuals, i, 0, j,
                             details = F)$AIC
    rownames(aicMat) <- paste0("p:", 0:uprLim)
    colnames(aicMat) <- paste0("q:", 0:uprLim)
    }
  }
# identify the row, column index of the minimum value
which(aicMat == min(aicMat), arr.ind = T)

aicMat %>% round(3)
```

```
sarima(isoneDemandGWh.lm$residuals, p=2, d=0, q=1)
```

Predicting from the model

```
# decent online reference - just tucking this away in the comments
# https://otexts.com/fpp2/

# loading the data, filtering to Jan through 14 Feb 2019 observations,
# computing needed covariates
path.isone <-
  "C:/Users/Duane/Documents/Academic/Villanova/5. Spring 19/Project Data/isoneDY_all.csv"
## isoneDY_all.csv

isoneEarly2019 <-
  read.csv(path.isone) %>%
  mutate(date = as.Date(date)) %>%
  filter(year == 2019, date <= as.Date("2019-02-14"))

isoneEarly2019 <-
  isoneEarly2019 %>%
  mutate(
    # creating only the variables needed for forecasting
    intercept     = 1,
    dayIndex      = (nrow(isone2)+1):(nrow(isone2) + nrow(isoneEarly2019)),
    # need an indicator variable for each weekday and each month (except ref levels)
    # Sunday is the reference level of weekday
    weekday       = weekdays(date),
    isSun         = (weekday == "Sunday") %>% as.integer(),
    isMon         = (weekday == "Monday") %>% as.integer(),
    isTue         = (weekday == "Tuesday") %>% as.integer(),
    isWed         = (weekday == "Wednesday") %>% as.integer(),
    isThu         = (weekday == "Thursday") %>% as.integer(),
    isFri         = (weekday == "Friday") %>% as.integer(),
    isSat         = (weekday == "Saturday") %>% as.integer(),
    month         = months(date),
    # first entry will be NA for prevDayDemand - need to carry in 31 Dec 2018
    prevDayDemand = if_else(dayIndex==(nrow(isone2)+1),
                            isone2$demand.GWh[nrow(isone2)],
                            lag(demand.GWh)),
    sin.1xYr      = sin(2*pi*(dayIndex)*(1/365.25)),
    cos.1xYr      = cos(2*pi*(dayIndex)*(1/365.25)),
    sin.2xYr      = sin(2*pi*(dayIndex)*(2/365.25)),
    cos.2xYr      = cos(2*pi*(dayIndex)*(2/365.25))
    )

betaHats <- coef(isoneDemandGWh.lm)

# column names for model covariates from the regression component
modelCovariates <-
  (summary(isoneDemandGWh.lm)$terms %>% as.character())[3] %>%
  strsplit(split = "\\+") %>%
  unlist() %>%
  str_trim()

predXMat <-
```

```
  isoneEarly2019 %>%
  select(intercept, modelCovariates) %>%
  as.matrix()

lmPred    <- predXMat %*% betaHats
```

```
errorPred <- sarima.for(isoneDemandGWh.lm$residuals,
                        n.ahead = nrow(isoneEarly2019),
                        p=2, d=0, q=1)
```

```
modelPred <- lmPred + errorPred$pred

isoneEarly2019 <-
  isoneEarly2019 %>%
  mutate(modEst = modelPred %>% as.numeric())

# plotting the results: actual vs. fitted "in-model" values plus predictions
isone.demandAllDat <-
  isone2 %>%
  # don't have 'day 1' estimate due to missing prevDayDemand
  mutate(modEst = c(NA, unname(isoneDemandGWh.lm$fitted.values))) %>%
  full_join(isoneEarly2019) %>%
  mutate(modelResid = demand.GWh - modEst)

isone.demandAllPlotDat <-
  isone.demandAllDat %>%
  select(date, demand.GWh, modEst) %>%
  gather(key="metric", value="value", -date) %>%
  mutate(qqch = if_else(metric == "demand.GWh" & date <= as.Date("2018-12-31"),
                        "Obs., in-sample",
                  if_else(metric == "demand.GWh" & date > as.Date("2018-12-31"),
                          "Obs., out-of-sample",
                    if_else(metric == "modEst" & date <= as.Date("2018-12-31"),
                            "Model, in-sample", "Model, out-of-sample")))
         )

# full view
ggplot(isone.demandAllPlotDat, aes(x=date, y=value)) +
  geom_line(data=isone.demandAllPlotDat %>% filter(grepl("Obs.", qqch)),
            aes(color=qqch)) +
  geom_point(data=isone.demandAllPlotDat %>% filter(grepl("Model", qqch),
                                                    date >= as.Date("2015-07-02")),
             aes(fill=qqch), shape=21,
             alpha=0.8) +
  labs(title="ISONE demand.GWh observed vs model fit",
       subtitle="1 Jul 2015 through 14 Feb 2019",
       x="Date", y="Electrical power demand \n (gigawatt-hours)",
       caption="No model estimate for 1 Jul 2015 (no 'previous day demand')",
       color=NULL, fill=NULL) +
  scale_color_manual(values=color_set8[c(3,7)]) +
  scale_fill_manual(values=color_set8[c(4,6)]) +
  monthly_x_setup +
  theme(legend.direction = "vertical",
```

```
        legend.margin = margin(c(0,0,0,0)),
        legend.position = "top")

# focusing on Oct 2018 forward
ggplot(isone.demandAllPlotDat,
       aes(x=date, y=value)) +
  geom_line(data=isone.demandAllPlotDat %>% filter(grepl("Obs.", qqch),
                                                   date >= as.Date("2018-10-01")),
            aes(color=qqch)) +
  geom_point(data=isone.demandAllPlotDat %>% filter(grepl("Model", qqch),
                                                    date >= as.Date("2018-10-01")),
             aes(fill=qqch), shape=21,
             alpha=0.8) +
  labs(title="ISONE demand.GWh observed vs model fit",
       subtitle="1 Oct 2018 through 14 Feb 2019",
       x="Date", y="Electrical power demand \n (gigawatt-hours)",
       color=NULL, fill=NULL) +
  scale_color_manual(values=color_set8[c(3,7)]) +
  scale_fill_manual(values=color_set8[c(4,6)]) +
  monthly_x_setup +
  # need to revise the the breaks in the scale_x_date part of monthly_x_setup
  scale_x_date(date_breaks      ="1 month",
               date_minor_breaks="1 month",
               date_labels      ="%b-%Y") +
  theme(legend.direction = "vertical",
        legend.margin = margin(c(0,0,0,0)),
        legend.position = "top")
```

```
# in- vs out-of-sample RMSE for model vs. 'aligned day-ahead demand'
#   'aligned day-ahead demand' might technically be considered to be all in-sample
# to allow for apples-to-apples comparison,
#   need to compare 02 Jul 2015 through 14 Feb 2019
#   (not 01 Jul 2015 - missing prevDayDemand for the model)

inSampRMSE <-
  isone.demandAllDat %>%
  mutate(alignedDayAheadDemand.GWh = lag(dayAheadDemand.GWh)) %>%
  filter(date >= as.Date("2015-07-02"),
         date <= as.Date("2018-12-31")) %>%
  mutate(demandVSalignedDayAheadDemand = demand.GWh - alignedDayAheadDemand.GWh)

inSampRMSE <-
  tibble(source = c("model", "alignedDayAheadDemand"),
         where  = "in-sample",
         RMSE   = c(sqrt(sum(inSampRMSE$modelResid^2)/nrow(inSampRMSE)),
                    sqrt(sum(inSampRMSE$demandVSalignedDayAheadDemand^2)/
                         nrow(inSampRMSE))
                    ))

outSampRMSE <-
  isone.demandAllDat %>%
  mutate(alignedDayAheadDemand.GWh = lag(dayAheadDemand.GWh)) %>%
  filter(date > as.Date("2018-12-31"),
```

```
        date <= as.Date("2019-02-14")) %>%
  mutate(demandVSalignedDayAheadDemand = demand.GWh - alignedDayAheadDemand.GWh)

outSampRMSE <-
  tibble(source = c("model", "alignedDayAheadDemand"),
         where  = "out-of-sample",
         RMSE   = c(sqrt(sum(outSampRMSE$modelResid^2)/nrow(outSampRMSE)),
                    sqrt(sum(outSampRMSE$demandVSalignedDayAheadDemand^2)/
                         nrow(outSampRMSE))
                    ))

inSampRMSE %>%
  full_join(outSampRMSE, by=c("source", "where", "RMSE"))
```

```
# checking residuals
isone.demandAllDat %>%
  mutate(absResid = abs(modelResid)) %>%
  arrange(desc(absResid)) %>%
  mutate(date = format(date, format="%d %b %Y")) %>%
  select(date, prevDayDemand, weekday, demand.GWh, modEst, modelResid) %>%
  mutate_if(is.numeric, round, digits=1) %>%
  head(10)
```