Low Carbon London Notebook

UK Data Service

July 10, 2017

Introduction

We explore the Low Carbon London dataset. This project sampled 5,567 London households between November 2011 and Feburary 2014, with reading taken at half-hourly intervals. The dataset contains: - energy consumption (kWh/hh), - unique household identifier, - date and time, and - CACI Acorn group. The dataset below contains sample data representing a single household.

It can be a good idea to have a look at the first few rows of the data to get a sense of what we're dealing with.

```
url = paste0("https://files.datapress.com/london/dataset/",
             "smartmeter-energy-use-data-in-london-households/",
             "UKPN-LCL-smartmeter-sample.csv")
scan(url, what = "", sep = ",", n = 12)
```

```
##  [1] "LCLid"                  "stdorToU"
##  [3] "DateTime"               "KWH/hh (per half hour) "
##  [5] "Acorn"                  "Acorn_grouped"
##  [7] "MAC003718"              "Std"
##  [9] "17/10/2012 13:00:00"    "0.09"
## [11] "ACORN-A"                "Affluent"
```

We can now read in the data.

```
col_names = c("id", "time_of_use", "datetime", "kwh_hh", "acorn", "acorn_group")
col_types = cols(id = col_character(),
                 time_of_use = col_character(),
                 datetime = col_datetime("%d/%m/%Y %H:%M:%S"),
                 kwh_hh = col_double(),
                 acorn = col_character(),
                 acorn_group = col_character())
lcl = read_csv(url, col_names = col_names, col_types = col_types, skip = 1)
```

Sampling rate

First we ensure that all the readings are in fact half-hourly. This is to allow the use of time series analysis methods later, most of which assume equally- spaced time intervals in the series.

```
dates = select(lcl, datetime)

## Wipe duplicated and missing dates.
lcl2 = lcl %>%
  filter(!duplicated(datetime), !is.na(datetime))
```

```
## Compute time differences between rows. We also store row numbers to see
## which rows are dropped later on.
lcl_diffs = lcl2 %>%
  mutate(ind = row_number()) %>%
  select(datetime, ind) %>%
  distinct() %>%
```

```
  mutate(diff = datetime - lag(datetime)) %>%
  slice(-1)
```

Observe that the sum of the time difference between rows 2981 and 2982 is in fact 30 minutes, so removing row 2981 should do the trick for 2 out of the 4 problematic rows.

```
## Which households have non-hh readings?
lcl_diffs %>%
  filter(diff != 30)
```

```
## # A tibble: 4 x 3
##               datetime   ind           diff
##                 <dttm> <int>         <time>
## 1 2012-12-09 07:30:00  2533 60.000000 mins
## 2 2012-12-18 15:24:01  2981 24.016667 mins
## 3 2012-12-18 15:30:00  2982  5.983333 mins
## 4 2013-02-19 20:00:00  6014 60.000000 mins
```

```
lcl_diffs %>%
  slice(2979:2982)
```

```
## # A tibble: 4 x 3
##               datetime   ind           diff
##                 <dttm> <int>         <time>
## 1 2012-12-18 15:00:00  2980 30.000000 mins
## 2 2012-12-18 15:24:01  2981 24.016667 mins
## 3 2012-12-18 15:30:00  2982  5.983333 mins
## 4 2012-12-18 16:00:00  2983 30.000000 mins
```

```
lcl_diffs2 = lcl2 %>%
  slice(-2981) %>%
  mutate(ind = row_number()) %>%
  select(datetime, ind) %>%
  distinct() %>%
  mutate(diff = datetime - lag(datetime)) %>%
  slice(-1)
```

This seems to have had the desired effect. We now have to tackle two other rows.

```
lcl_diffs2 %>%
  filter(diff != 30)
```

```
## # A tibble: 2 x 3
##               datetime   ind    diff
##                 <dttm> <int>  <time>
## 1 2012-12-09 07:30:00  2533 60 mins
## 2 2013-02-19 20:00:00  6013 60 mins
```

Since the gaps in time are both 2 half-hour units, we can simply take the average of the surrounding values.

```
slice(lcl_diffs2, 2531:2533)
```

```
## # A tibble: 3 x 3
##               datetime   ind    diff
##                 <dttm> <int>  <time>
## 1 2012-12-09 06:30:00  2532 30 mins
## 2 2012-12-09 07:30:00  2533 60 mins
## 3 2012-12-09 08:00:00  2534 30 mins
```

```
slice(lcl_diffs, 6014:6016)
```

```
## # A tibble: 3 x 3
##              datetime   ind    diff
##                <dttm> <int>  <time>
## 1 2013-02-19 20:30:00  6015 30 mins
## 2 2013-02-19 21:00:00  6016 30 mins
## 3 2013-02-19 21:30:00  6017 30 mins
```

```
## Simple function for linear interpolation of equally-spaced series.
insert_row = function(data, ind) {
  sub = ind:(ind + 1)
  new_kwh = mean(data$kwh_hh[sub])
  new_time = mean(data$datetime[sub])
  new_row = data %>%
    slice(ind) %>%
    mutate(kwh_hh = new_kwh, datetime = new_time)
  data %>%
    head(ind) %>%
    rbind(new_row, slice(data, ind + 1:nrow(data)))
}
```

```
lcl2 %>%
  insert_row(2532) %>%
  slice(2532:2534)
```

```
## # A tibble: 3 x 6
##          id time_of_use            datetime kwh_hh   acorn acorn_group
##       <chr>       <chr>              <dttm>  <dbl>   <chr>       <chr>
## 1 MAC003718         Std 2012-12-09 06:30:00  0.112 ACORN-A    Affluent
## 2 MAC003718         Std 2012-12-09 07:00:00  0.142 ACORN-A    Affluent
## 3 MAC003718         Std 2012-12-09 07:30:00  0.172 ACORN-A    Affluent
```

```
lcl2 %>%
  insert_row(6013) %>%
  slice(6013:6015)
```

```
## # A tibble: 3 x 6
##          id time_of_use            datetime kwh_hh   acorn acorn_group
##       <chr>       <chr>              <dttm>  <dbl>   <chr>       <chr>
## 1 MAC003718         Std 2013-02-19 19:00:00 0.4010 ACORN-A    Affluent
## 2 MAC003718         Std 2013-02-19 19:30:00 0.3225 ACORN-A    Affluent
## 3 MAC003718         Std 2013-02-19 20:00:00 0.2440 ACORN-A    Affluent
```

Check that the series is equidistant:

```
lcl3 = lcl2 %>%
  insert_row(2532) %>%
  insert_row(6014) %>%
  slice(-2982)

lcl_diffs3 = lcl3 %>%
  select(datetime) %>%
  distinct() %>%
  mutate(diff = datetime - lag(datetime)) %>%
  slice(-1)
```

```
all(lcl_diffs3$diff == 30)
```

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

Time

We augment our dataset with useful time information: year, month, day, hour, minute, and weekday.

```
lcl4 = lcl3 %>%
  mutate(wday = wday(datetime, lab = TRUE),
         year = year(datetime),
         month = month(datetime),
         day = day(datetime),
         hour = hour(datetime),
         minute = minute(datetime)) %>%
  arrange(datetime)
```

Add 2012 UK Seasons. Recall that: - Spring: March to June - Summer: July to September - Autumn: October to December - Winter: January to March

```
brks = ymd(c("20120922", "20121221", "20130320", "20130621", "20130922", "20131221"))
labs = c("Autumn 12", "Winter 12-13", "Spring 13", "Summer 13", "Autumn 13")
lcl_seas = lcl4 %>%
  mutate(date = as.Date(datetime),
         seas = cut(date, breaks = brks, labels = labs))

lcl_seas %>%
  select(seas, datetime, kwh_hh) %>%
  head()
```

```
## # A tibble: 6 x 3
##        seas            datetime kwh_hh
##      <fctr>              <dttm>  <dbl>
## 1 Autumn 12 2012-10-17 13:00:00  0.090
## 2 Autumn 12 2012-10-17 13:30:00  0.160
## 3 Autumn 12 2012-10-17 14:00:00  0.212
## 4 Autumn 12 2012-10-17 14:30:00  0.145
## 5 Autumn 12 2012-10-17 15:00:00  0.104
## 6 Autumn 12 2012-10-17 15:30:00  0.122
```

If we were to plot this, we need a "change-of-season" dates. But since we have 48 rows of each day, there will be many matching cut-off dates (in fact $48 \times 4 = 192$ matches). To overcome this issue, we could just take the first match.

```
matches = lcl_seas %>%
  mutate(n = row_number()) %>%
  filter(date %in% brks) %>%
  select(n)

matches
```

```
## # A tibble: 192 x 1
##        n
##    <int>
##  1  3095
##  2  3096
```

```
##  3  3097
##  4  3098
##  5  3099
##  6  3100
##  7  3101
##  8  3102
##  9  3103
## 10  3104
## # ... with 182 more rows
```

```
## Get the first match.
n = matches$n
i = c(0, which(diff(n) > 1)) + 1
j = n[i]
```

Missing readings

Let's move on to missing readings. Fortunately there doesn't seem to be any for this household.

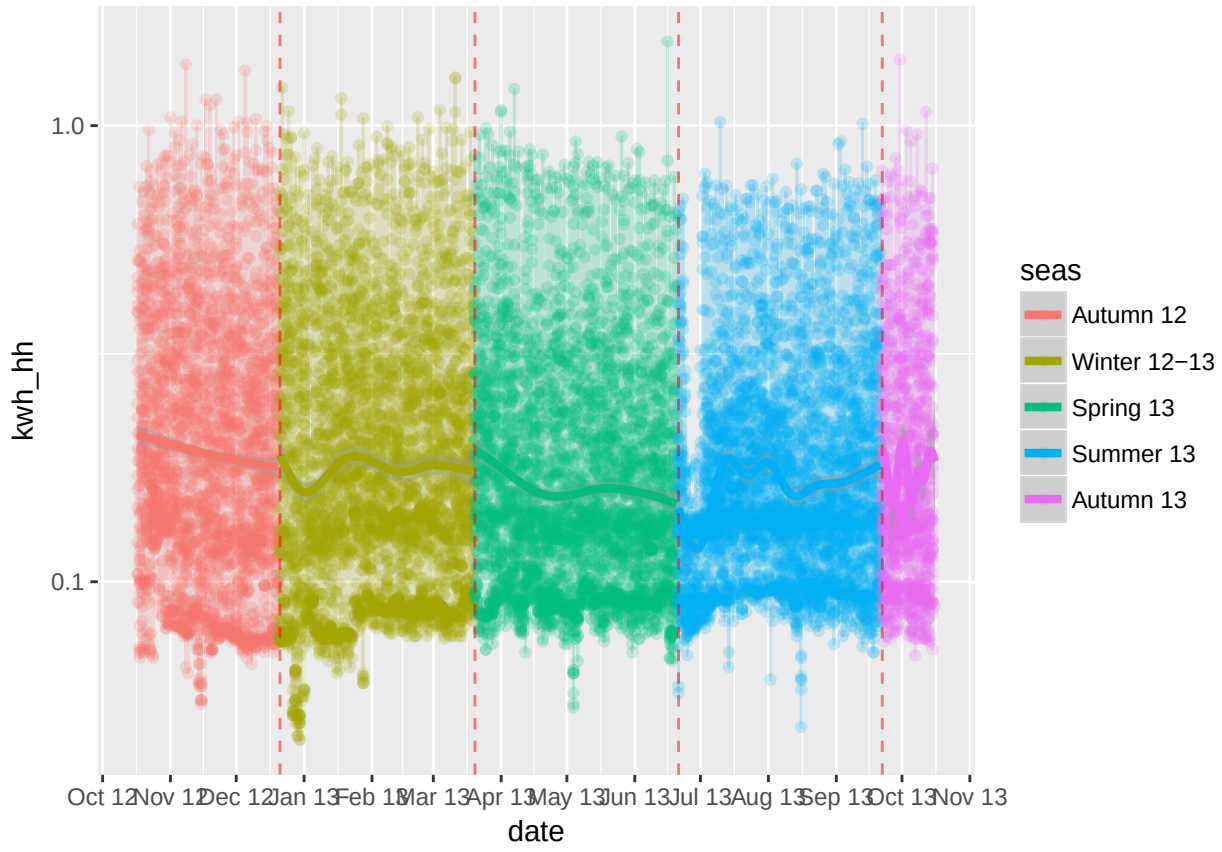
```
lcl_seas %>%
  filter(is.na(kwh_hh))
```

```
## # A tibble: 0 x 14
## # ... with 14 variables: id <chr>, time_of_use <chr>, datetime <dttm>,
## #   kwh_hh <dbl>, acorn <chr>, acorn_group <chr>, wday <ord>, year <dbl>,
## #   month <dbl>, day <int>, hour <int>, minute <int>, date <date>,
## #   seas <fctr>
```

Plots

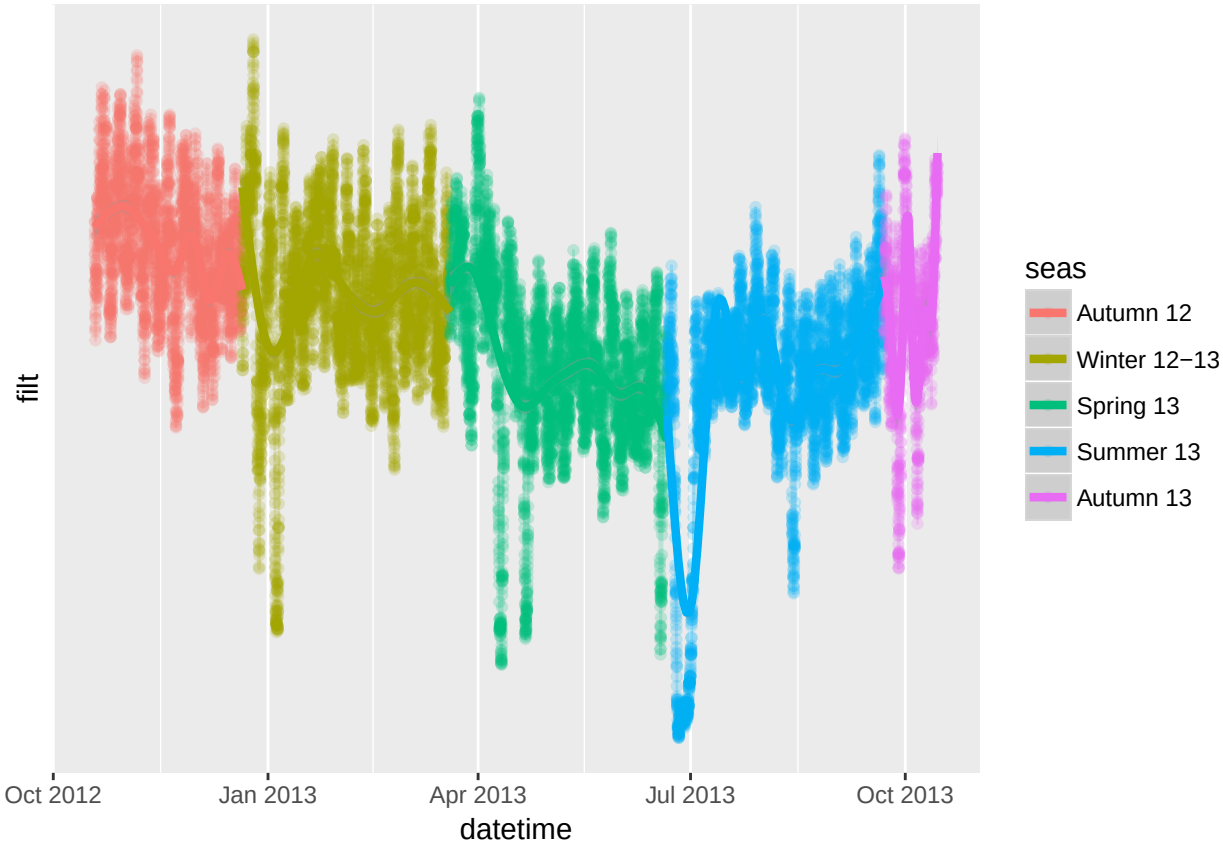
```
lcl_seas %>%
  ggplot(aes(date, kwh_hh, col = seas)) +
  geom_line(alpha = 0.2) +
  geom_point(alpha = 0.2) +
  geom_smooth(span = 1, lwd = 1.5) +
  scale_y_log10() +
  scale_x_date(date_labels = "%b %y",
               date_breaks = "1 month") +
  geom_vline(xintercept = as.numeric(lcl_seas$date[j]),
             linetype = 2, colour = 2, alpha = 0.5)
```

```
## `geom_smooth()` using method = 'gam'
```




```
## Smooth with a 2-sided 115-pt weighted moving average.
wgts = 1 / c(rep(16, 31), rep(8, 21), rep(4, 11), rep(8, 21), rep(16, 31))
lcl_seas %>%
  mutate(filt = stats::filter(kwh_hh, filter = wgts, sides = 2)) %>%
  filter(!is.na(filt)) %>%
  ggplot(aes(datetime, filt, col = seas)) +
  geom_line(alpha = 0.2) +
  geom_point(alpha = 0.2) +
  geom_smooth(span = 0.75, lwd = 1.5) +
  scale_y_log10() +
  geom_vline(xintercept = as.numeric(lcl_seas$date[j]),
             linetype = 2, colour = 2, alpha = 0.5)
```

```
## `geom_smooth()` using method = 'gam'
```

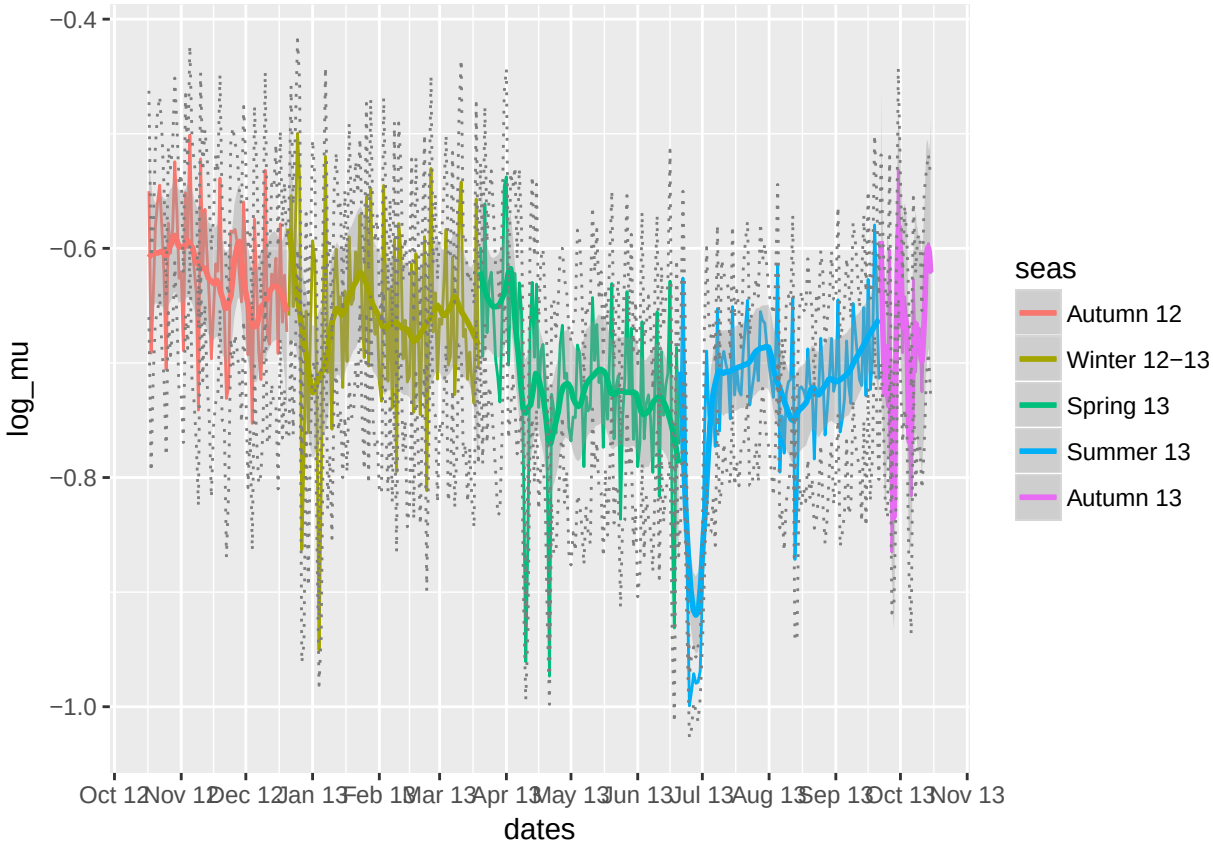



```
## Statistical summaries.
lcl_seas2 = lcl_seas %>%
  group_by(year, month, day) %>%
  summarize(mu = mean(kwh_hh),
            log_mu = log10(mu),
            sd = sd(kwh_hh),
            n = n(),
            seas = unique(seas),
            wday = unique(wday),
            ci_low = mu - 1.96 * sd / sqrt(n),
            ci_high = mu + 1.96 * sd / sqrt(n),
            log_low = log10(ci_low),
            log_high = log10(ci_high)) %>%
  filter(!is.na(log_low), !is.na(log_high))
```

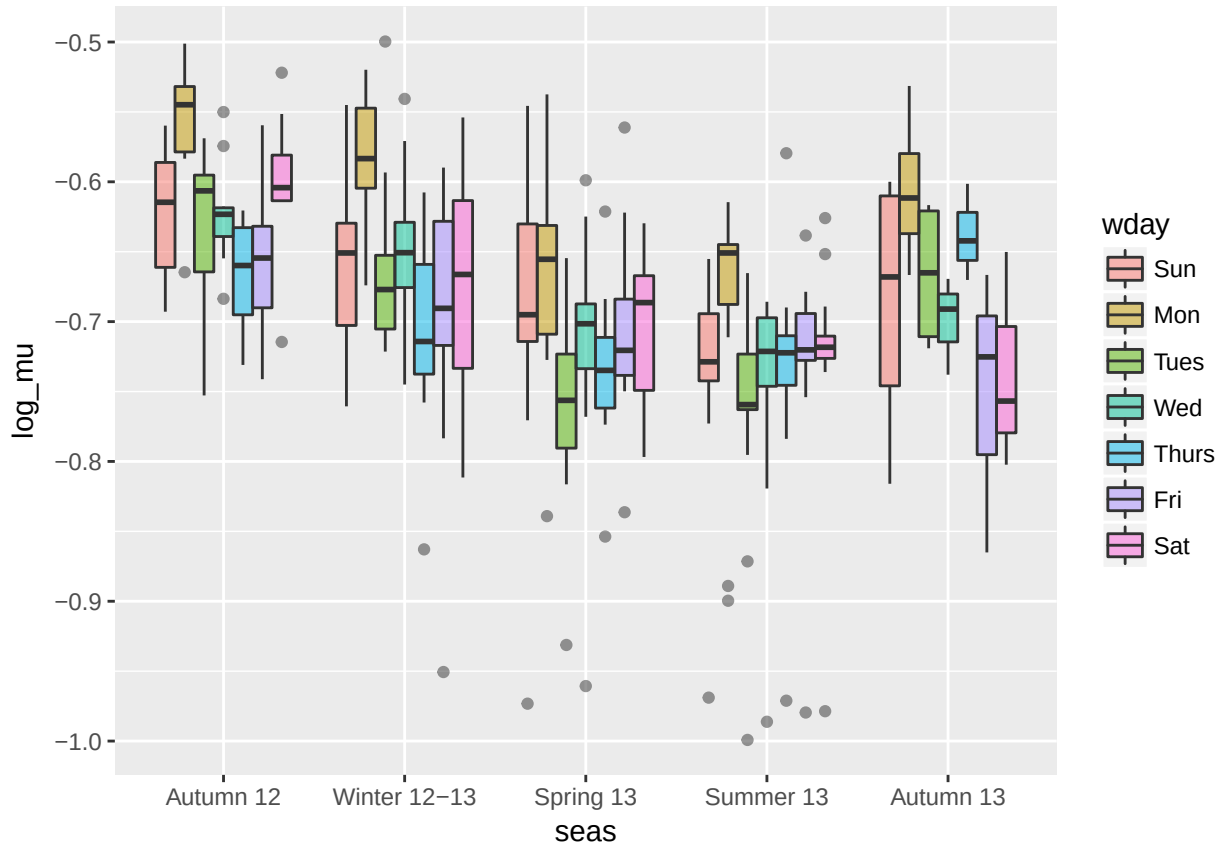
```
dates = with(lcl_seas2, ymd(paste0(year,
                                   sprintf("%02d", month),
                                   sprintf("%02d", day))))

lcl_seas2 %>%
  ggplot(aes(dates, log_mu, col = seas)) +
  geom_line() +
  geom_smooth(span = 0.25) +
  geom_line(aes(dates, log10(ci_low)), col = "gray50", linetype = "dotted") +
  geom_line(aes(dates, log10(ci_high)), col = "gray50", linetype = "dotted") +
  scale_x_date(date_labels = "%b %y", date_breaks = "1 month")
```

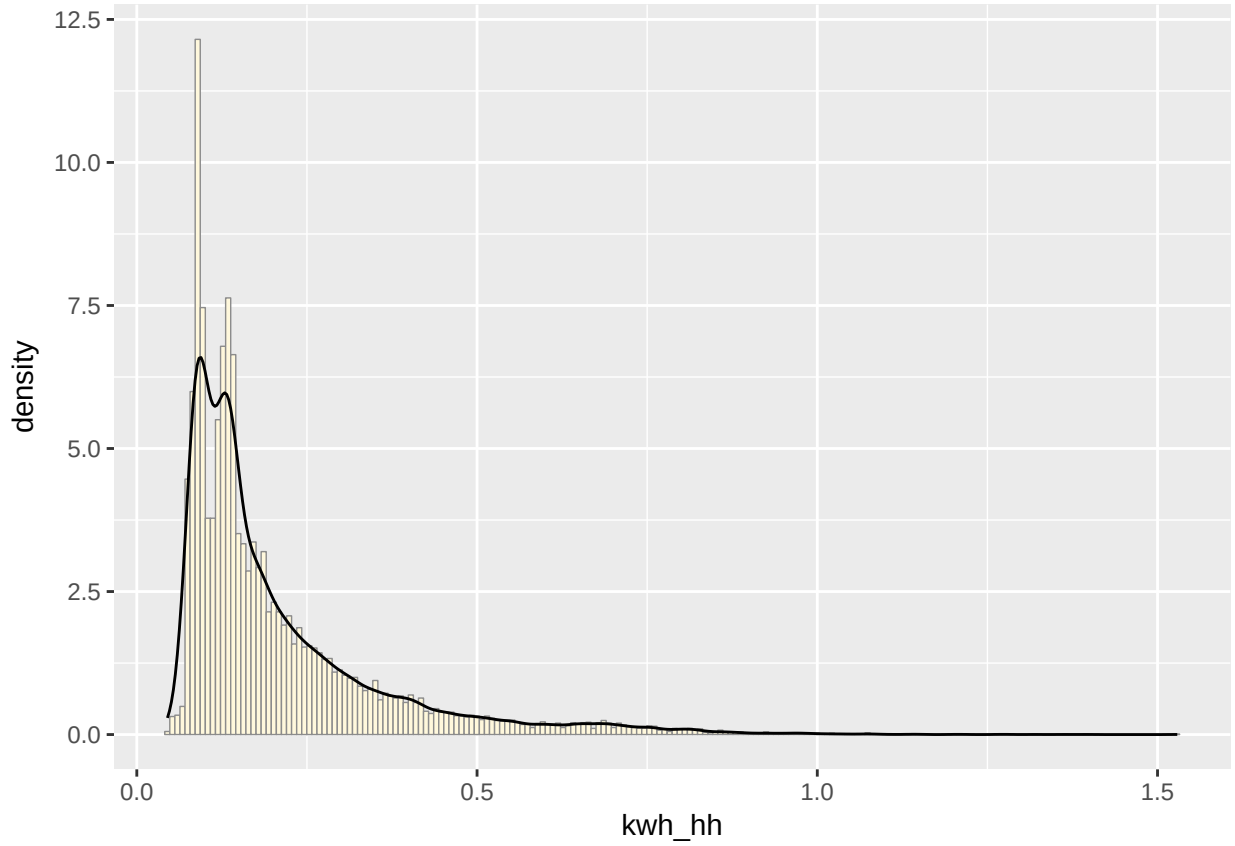
```
## `geom_smooth()` using method = 'loess'
```



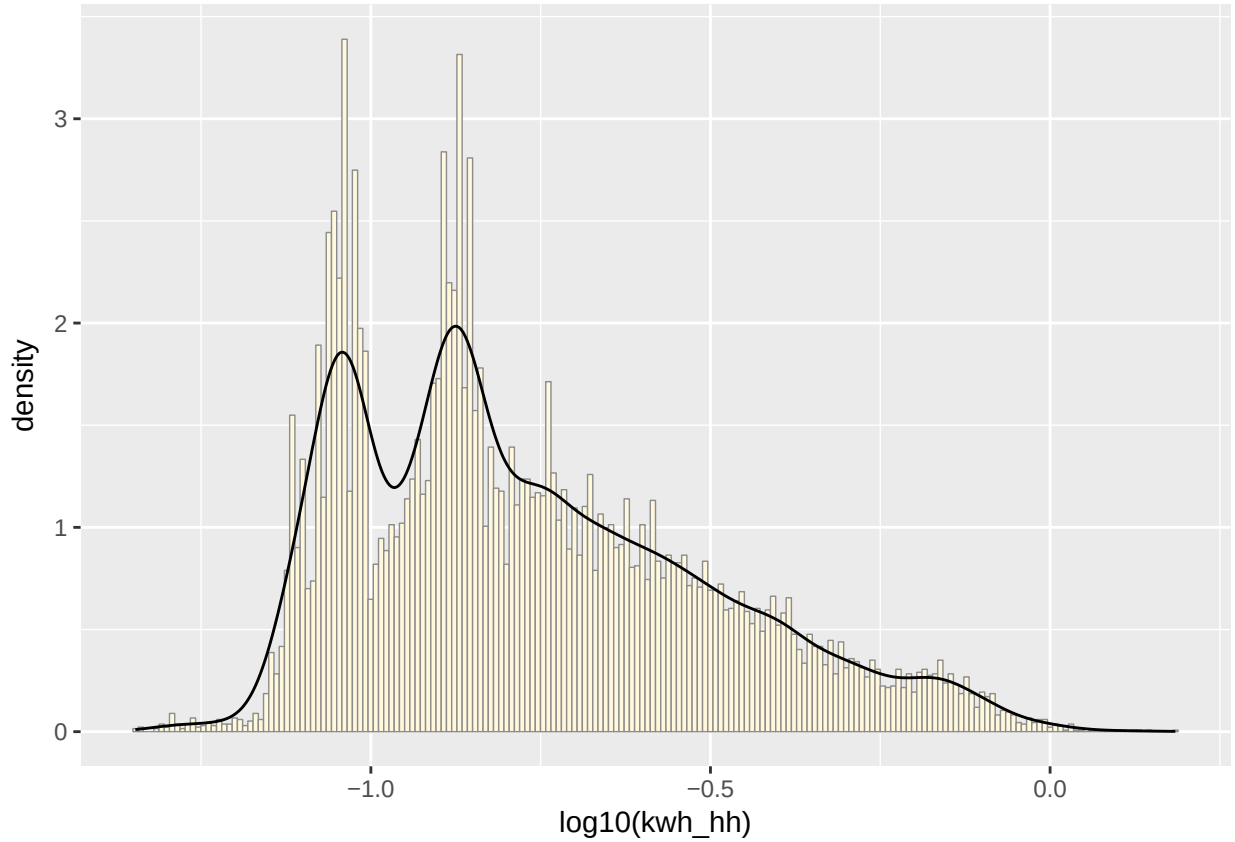

```
## Weekdays
lcl_seas2 %>%
  ggplot(aes(seas, log_mu, bg = wday)) +
  geom_boxplot(alpha = .5)
```



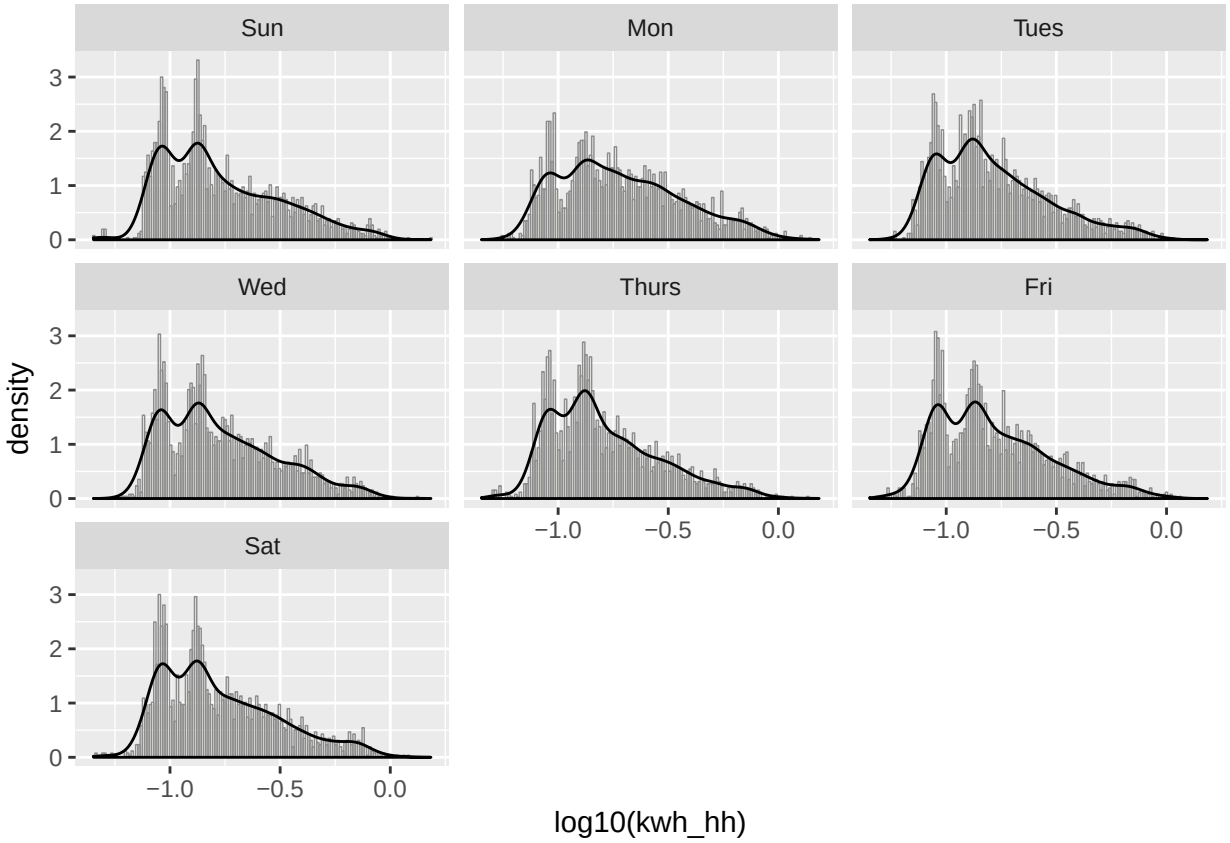

```
lcl_seas %>%
  ggplot(aes(kwh_hh, ..density..)) +
  geom_histogram(fill = "cornsilk",
                 colour = "grey55",
                 size = .25,
                 bins = 200) +
  geom_line(stat = "density")
```



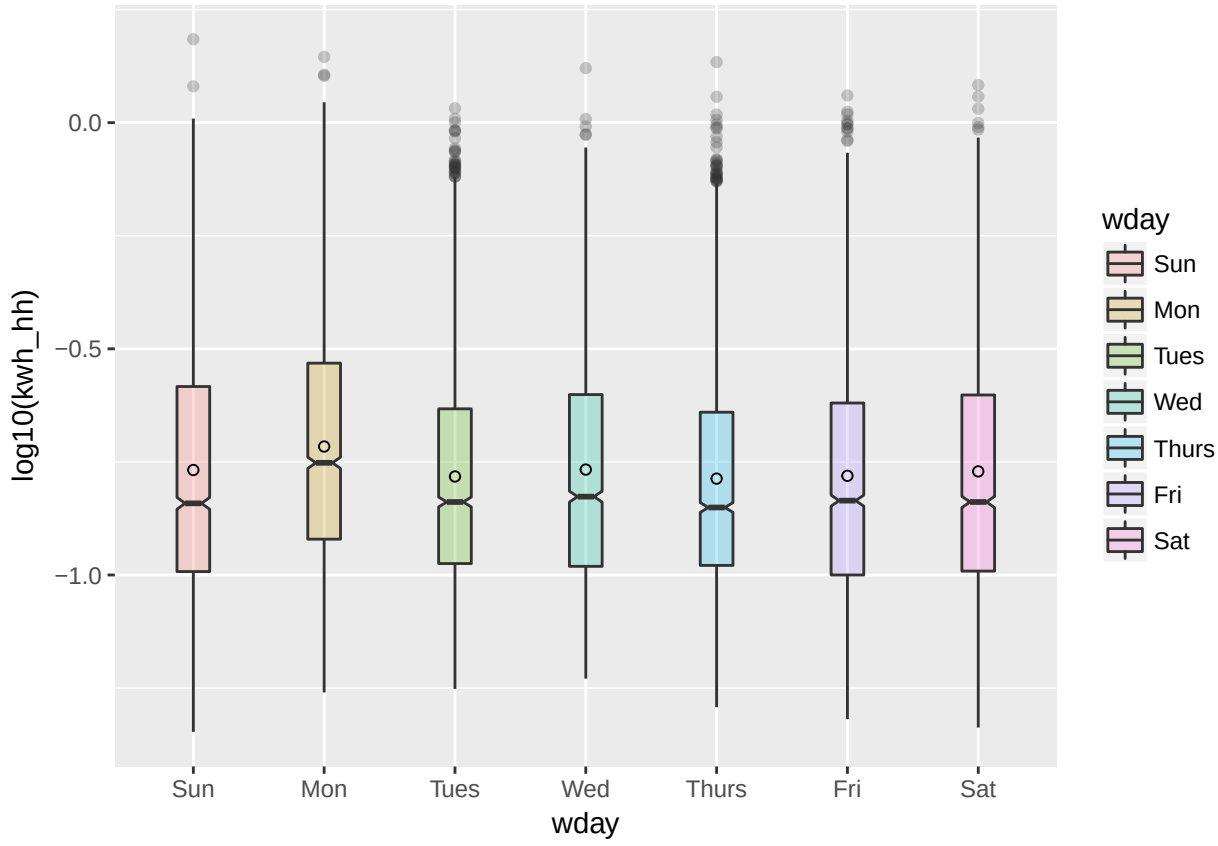

```
lcl_seas %>%
  ggplot(aes(log10(kwh_hh), ..density..)) +
  geom_histogram(fill = "cornsilk",
                 colour = "grey55",
                 size = .25,
                 bins = 200) +
  geom_line(stat = "density")
```



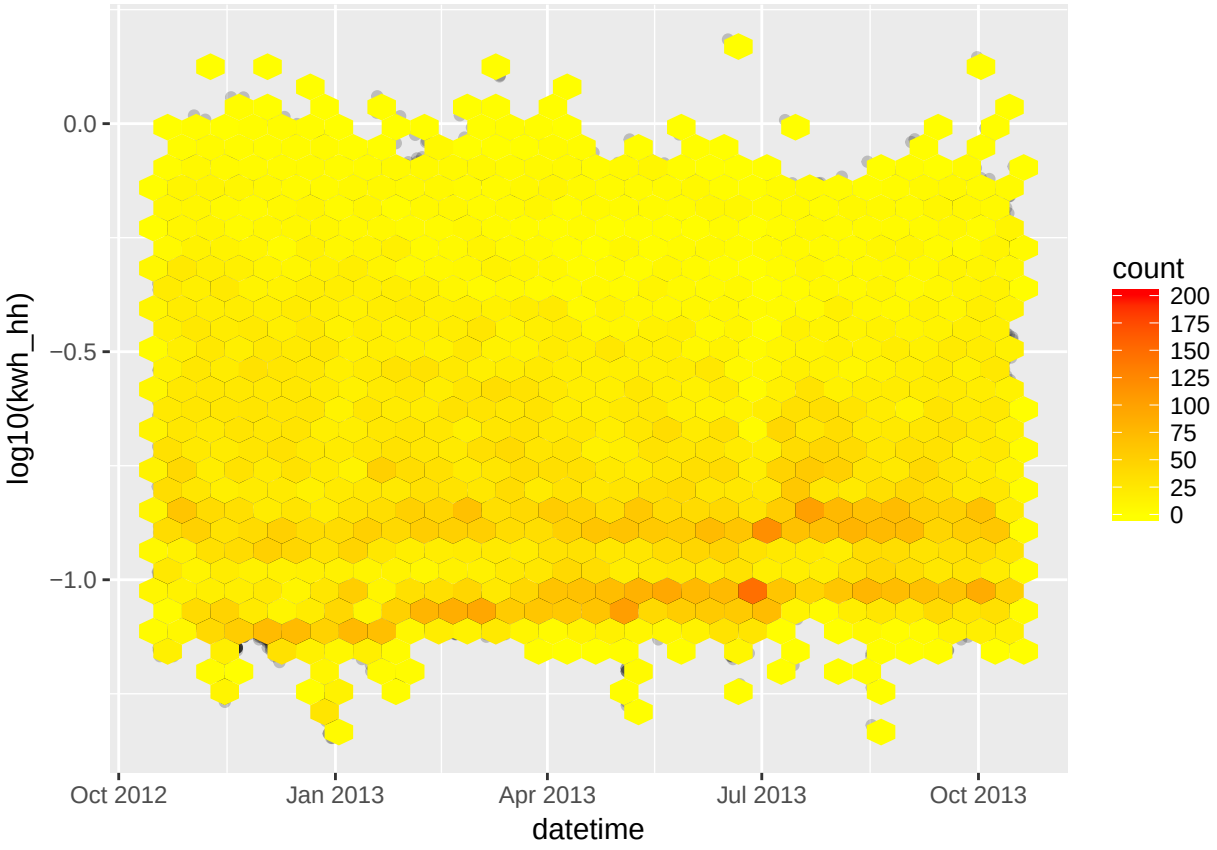

```
lcl_seas %>%
  ggplot(aes(log10(kwh_hh), ..density..)) +
  geom_histogram(fill = "cornsilk",
                 colour = "grey55",
                 alpha = 0.1,
                 size = .25,
                 bins = 150) +
  geom_density(alpha = .1) +
  facet_wrap(~ wday)
```




```
lcl_seas %>%
  ggplot(aes(wday, log10(kwh_hh), fill = wday)) +
  geom_boxplot(width = .25, alpha = .25, notch = TRUE) +
  stat_summary(fun.y = "mean", geom = "point", shape = 1, fill = "white")
```




```
## Ignore time for a second.
lcl_seas %>%
  ggplot(aes(datetime, log10(kwh_hh))) +
  geom_point(alpha = 0.2) +
  stat_binhex() +
  scale_fill_gradient(low = "yellow", high = "red",
                      breaks = seq(0, 200, length = 9),
                      limits = c(0, 200))
```


0.0
−0.5
−1.0
log10(kwh_hh)
Oct 2012
Jan 2013
Apr 2013
Jul 2013
Oct 2013
datetime
count
200
175
150
125
100
75
50
25
0