Figures: Snowflake Workload Timeseries Patterns

Murray Stokely

2024-10-18

Contents

1 Introduction

This file includes figures and analysis for Section 2.2 on the Snowflake Workload User Demand Patterns. All figures and analysis are generated from the public dataset.

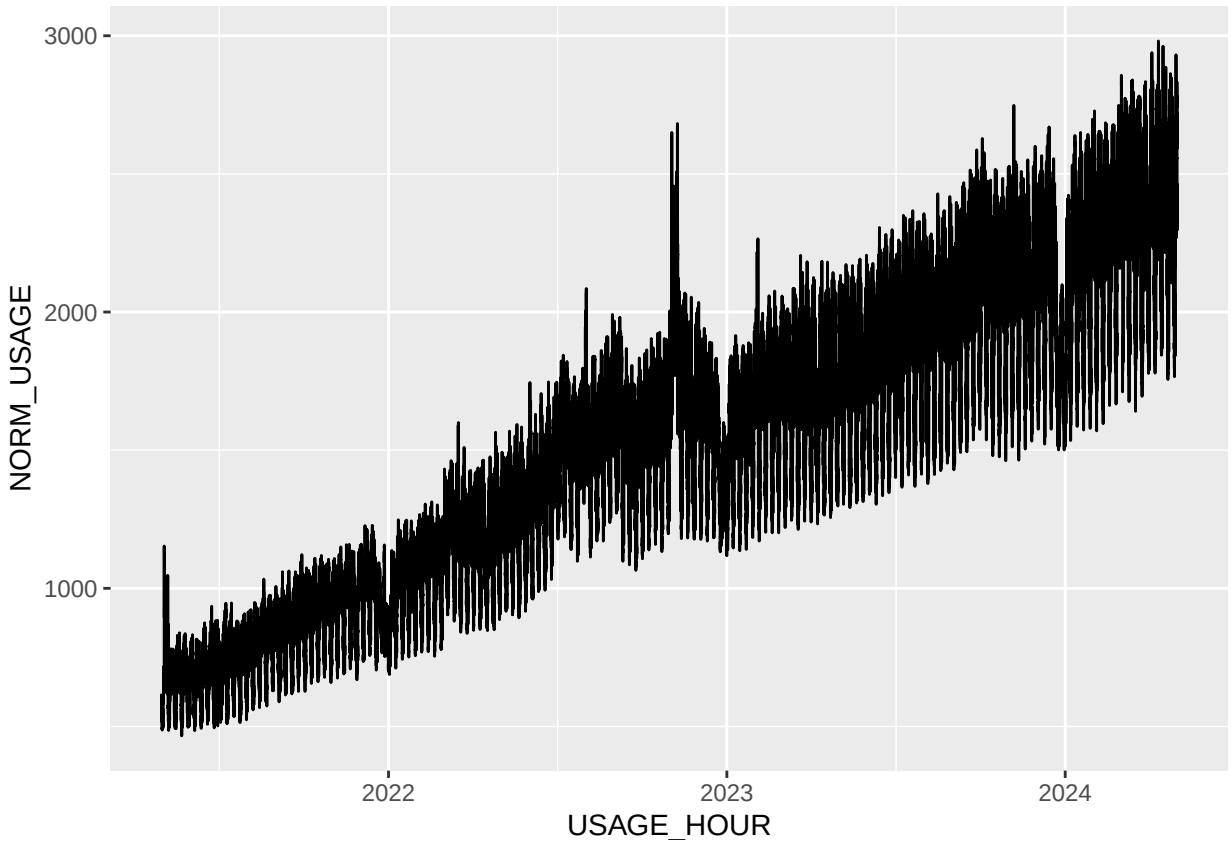
2 Full-Granularity Aggregated Hourly Timeseries

All regions and SKU types aggregated together at full hourly granularity and plotted over the full time range of the data set.

```
data <- read_parquet("../hourly_normalized.parquet")
data.all <- data %>% group_by(USAGE_HOUR) %>% summarise(across(NORM_USAGE, sum))
head(data.all)
```

```
## # A tibble: 6 x 2
##   USAGE_HOUR          NORM_USAGE
##   <dttm>                   <dbl>
## 1 2021-05-01 00:00:00        618
## 2 2021-05-01 01:00:00        561
## 3 2021-05-01 02:00:00        526
## 4 2021-05-01 03:00:00        549
## 5 2021-05-01 04:00:00        555
## 6 2021-05-01 05:00:00        569
```

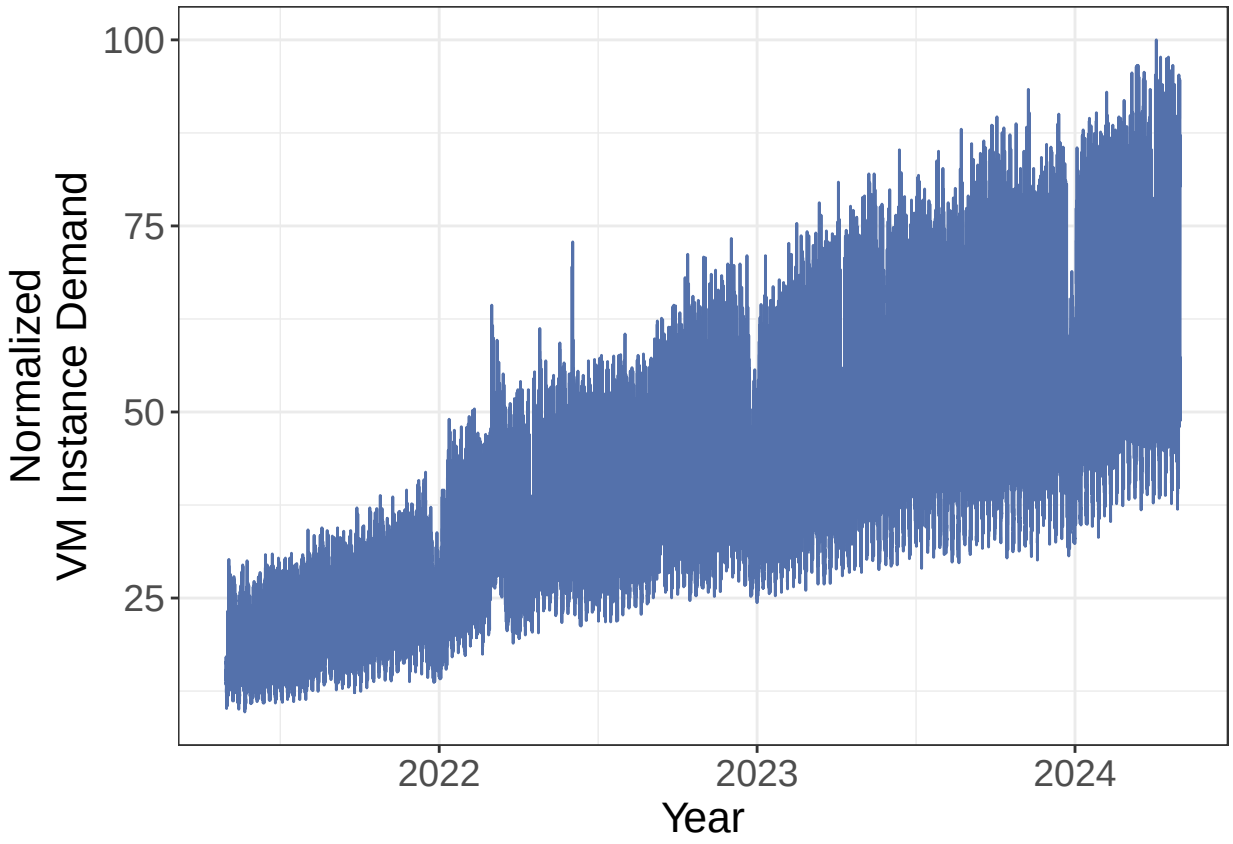
```
ggplot(data.all, aes(x=USAGE_HOUR, y=NORM_USAGE)) + geom_line()
```



2.1 Normalized Full-Granularity Aggregated Hourly Timeseries

All regions and SKU types aggregated together at full hourly granularity and plotted over the full time range of the data set. Normalized with the largest data point at 100.

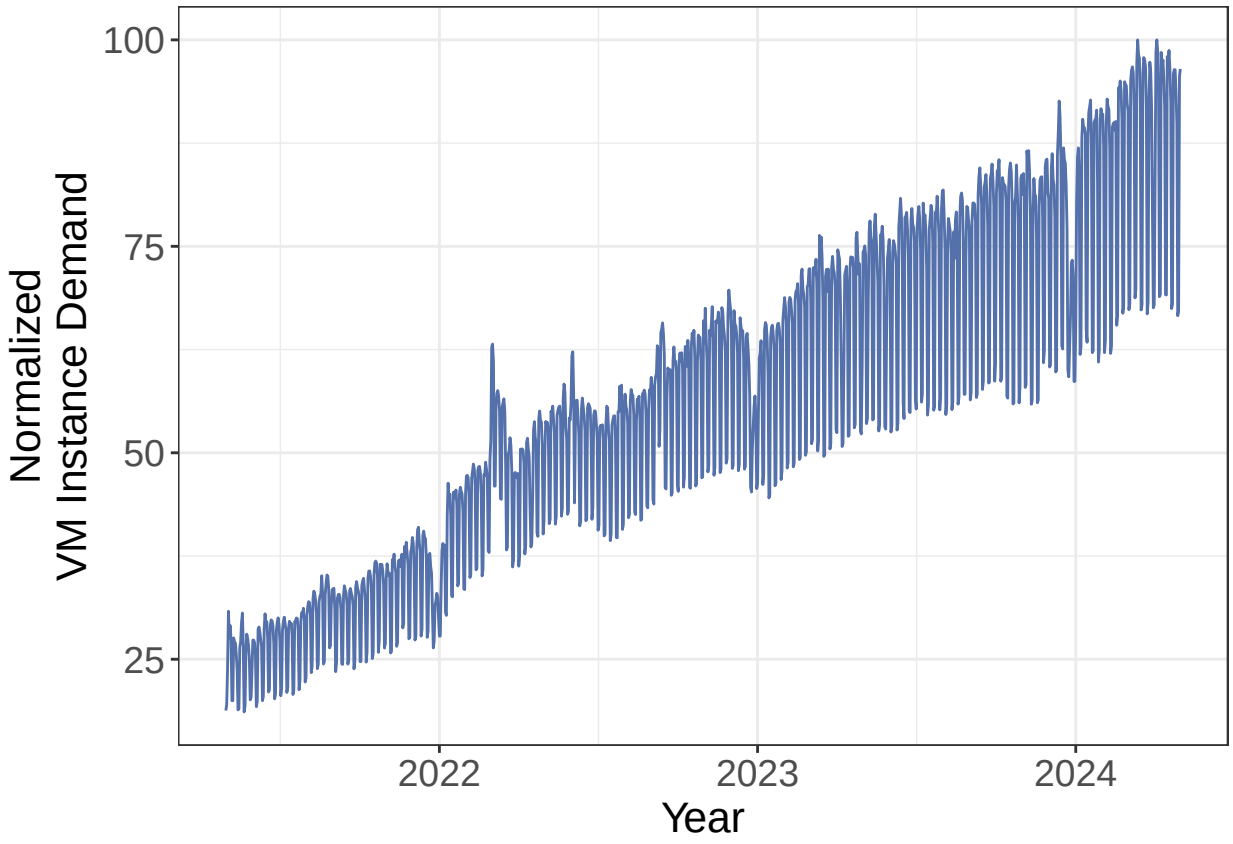
```
data.p1 <- data %>% subset(REGION_NUM == 2) %>% group_by(USAGE_HOUR) %>%
  summarise(across(NORM_USAGE, sum))
data.p1$TOTAL_COMPUTE = data.p1$NORM_USAGE
data.p1$day <- as.Date(data.p1$USAGE_HOUR)
data.p1$TOTAL_COMPUTE = (100*data.p1$TOTAL_COMPUTE) / max(data.p1$TOTAL_COMPUTE)
p1 <- ggplot(data.p1, aes(x=USAGE_HOUR, y=TOTAL_COMPUTE)) +
  geom_line(color="#5471AB") + theme_bw() +
  theme(axis.title.x=element_text(size=16), axis.title.y=element_text(size=16),
        axis.text=element_text(size=14)) +
  xlab("Year") + ylab("Normalized\nVM Instance Demand")
old.mai <- par("mai")
par("mai"= c(1,0.9,2,0.4))
p1
```



2.2 Normalized Timeseries of the Daily Means

Plot of the normalized daily means.

```
data.p2 <- data.p1 %>% group_by(day) %>%
  summarize(DAILY_MEAN = mean(TOTAL_COMPUTE))
data.p2$DAILY_MEAN = (100*data.p2$DAILY_MEAN) / max(data.p2$DAILY_MEAN)
p2 <- ggplot(data.p2, aes(x=day, y=DAILY_MEAN)) + geom_line(color="#5471AB") +
  theme_bw() + theme(axis.title.x=element_text(size=16),
                     axis.title.y=element_text(size=16),
                     axis.text=element_text(size=14)) +
  xlab("Year") + ylab("Normalized\nVM Instance Demand")
old.mai <- par("mai")
par("mai"= c(1,0.9,2,0.4))
p2
```



3 Weekly Pattern Analysis

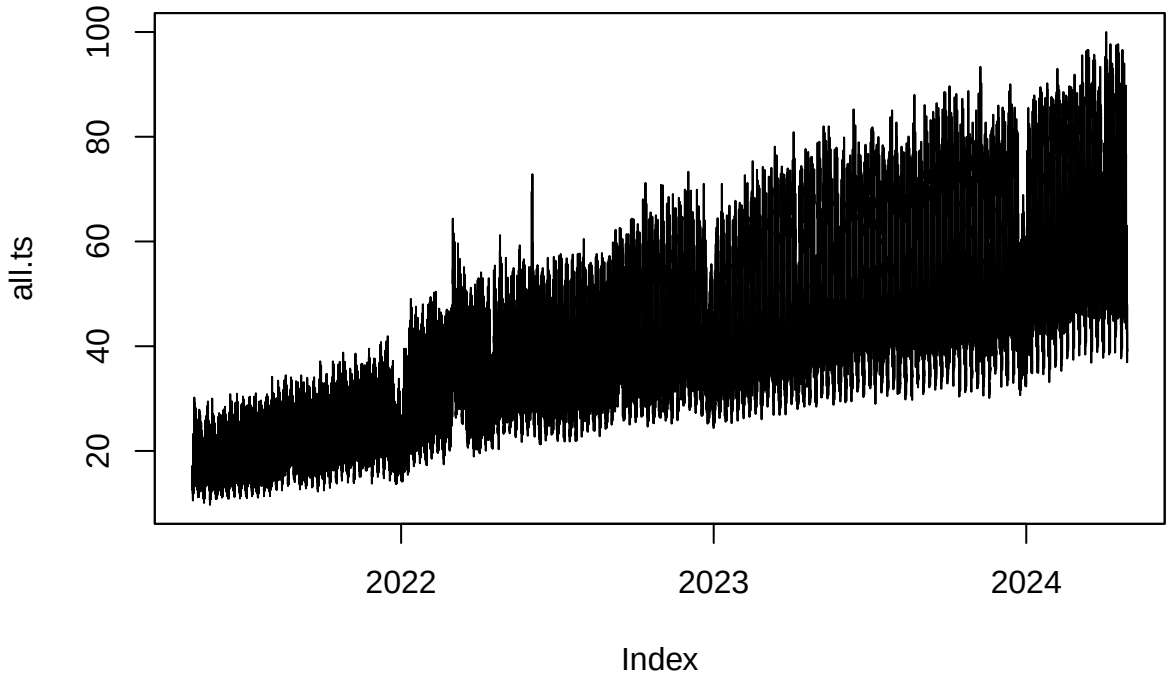
We trim the start and end of our timeseries toalign our dataset with Sunday through Saturday weeks so we can split it up in 7 day chunks and look at the distribution of weekly patterns.

We then generate timeseries of the maximum, minimum, and mean of each day.

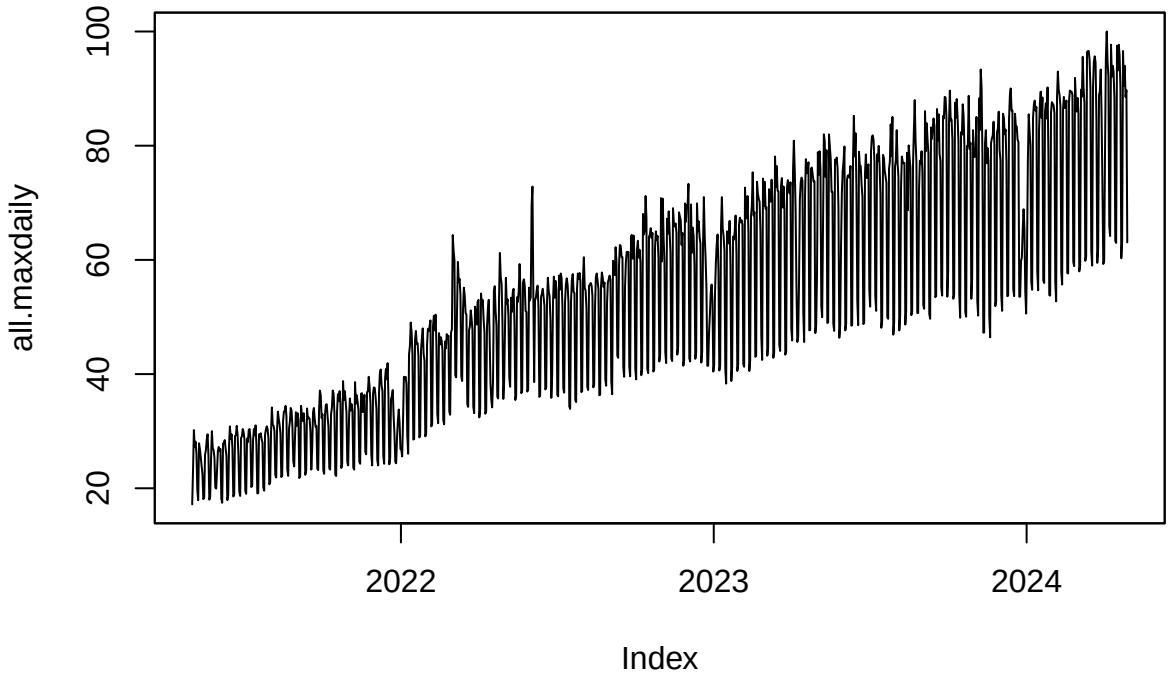
```
all <- data.p1 %>% subset(USAGE_HOUR >= as.POSIXct("2021-05-02", tz="UTC")) %>%
  subset(USAGE_HOUR < as.POSIXct("2024-04-28", tz="UTC"))
range(all$USAGE_HOUR)
```

```
## [1] "2021-05-02 00:00:00 UTC" "2024-04-27 23:00:00 UTC"
```

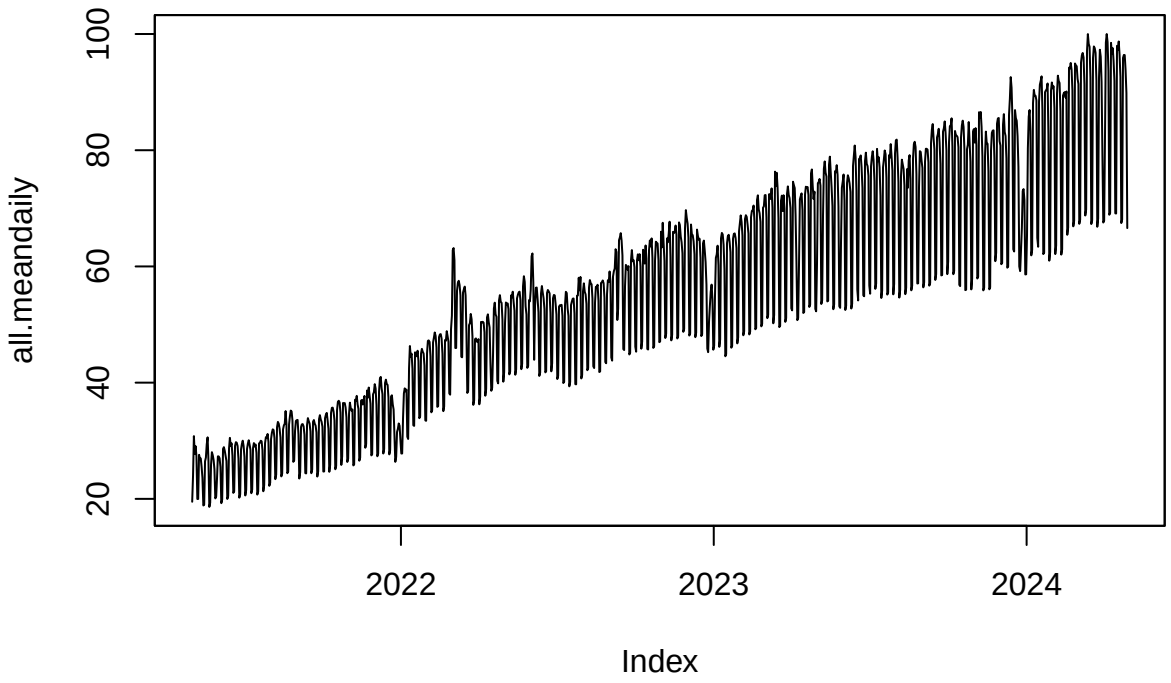
```
all.ts <- zoo(all$TOTAL_COMPUTE, all$USAGE_HOUR)
# Compute the maximumes over each day.
all.maxdaily <- rollapply(all.ts, width=24, by=24, FUN=max)
all.meandaily <- rollapply(all.ts, width=24, by=24, FUN=mean)
all.mindaily <- rollapply(all.ts, width=24, by=24, FUN=min)
plot(all.ts)
```




```
plot(all.maxdaily)
```




```
all.meandaily <- 100*all.meandaily/(max(all.meandaily))
plot(all.meandaily)
```



Next we plot the first two weeks of our dataset to examine the different weekly periodicity.

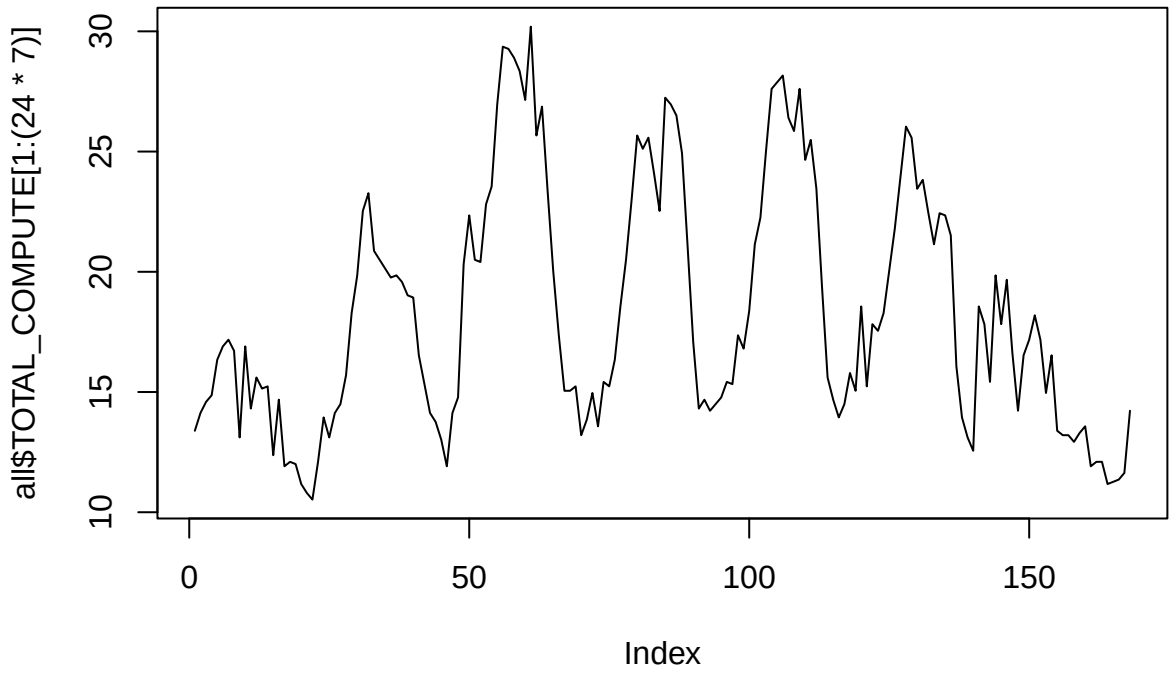
```
range(all$USAGE_HOUR)
```

```
## [1] "2021-05-02 00:00:00 UTC" "2024-04-27 23:00:00 UTC"
```

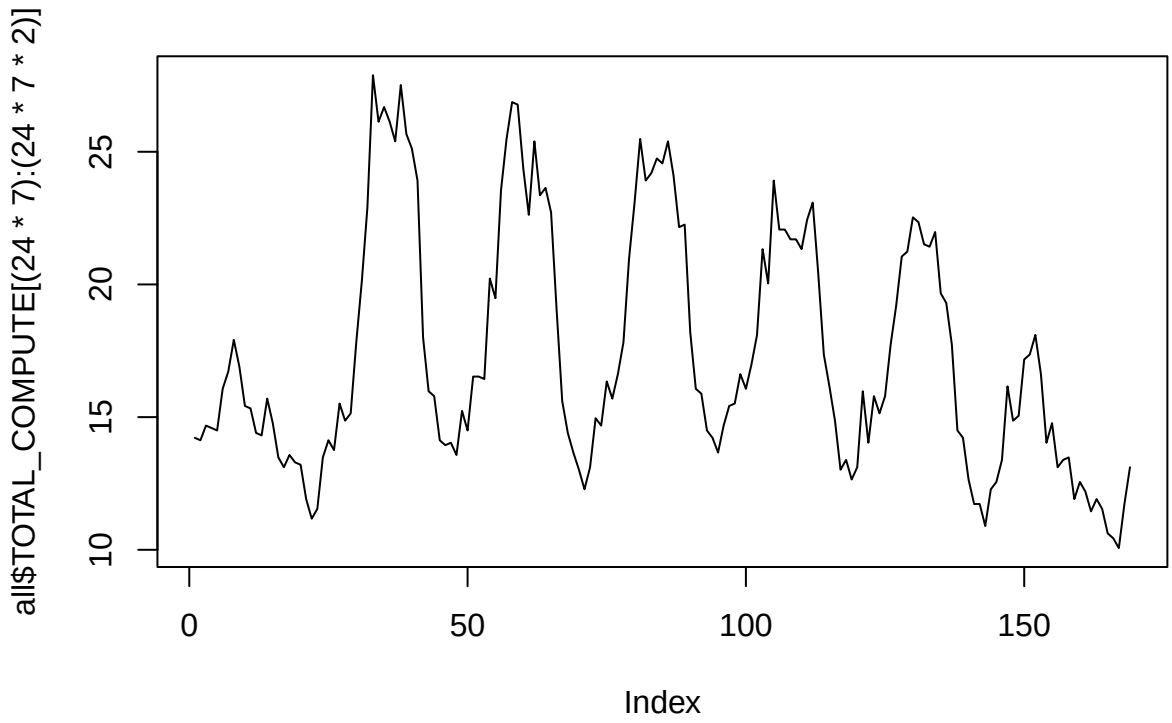
```
head(all)
```

```
## # A tibble: 6 x 4
##   USAGE_HOUR          NORM_USAGE TOTAL_COMPUTE day
##   <dttm>                   <dbl>         <dbl> <date>
## 1 2021-05-02 00:00:00        145          13.4 2021-05-02
## 2 2021-05-02 01:00:00        153          14.1 2021-05-02
## 3 2021-05-02 02:00:00        158          14.6 2021-05-02
## 4 2021-05-02 03:00:00        161          14.9 2021-05-02
## 5 2021-05-02 04:00:00        177          16.3 2021-05-02
## 6 2021-05-02 05:00:00        183          16.9 2021-05-02
```

```
plot(all$TOTAL_COMPUTE[1:(24*7)], type="l")
```



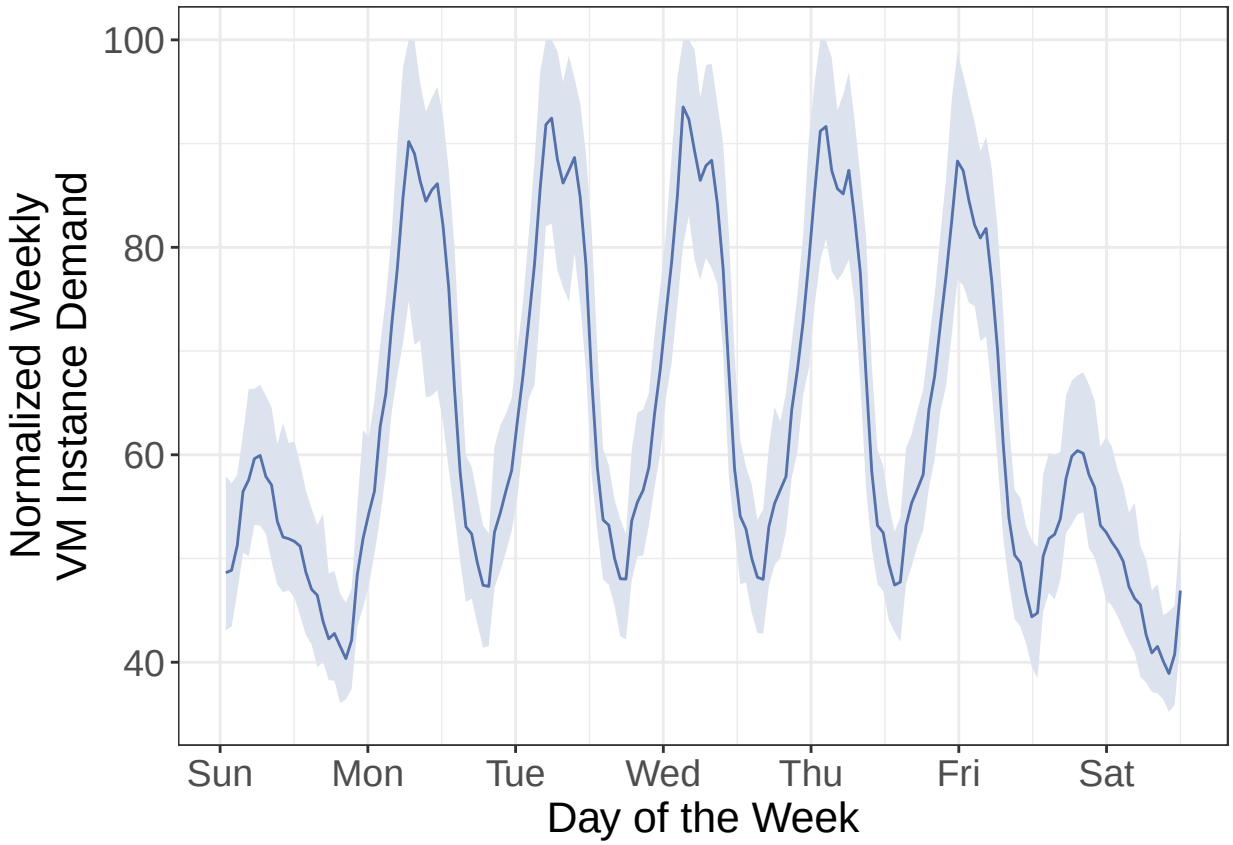

```
plot(all$TOTAL_COMPUTE[(24*7):(24*7*2)], type="l")
```



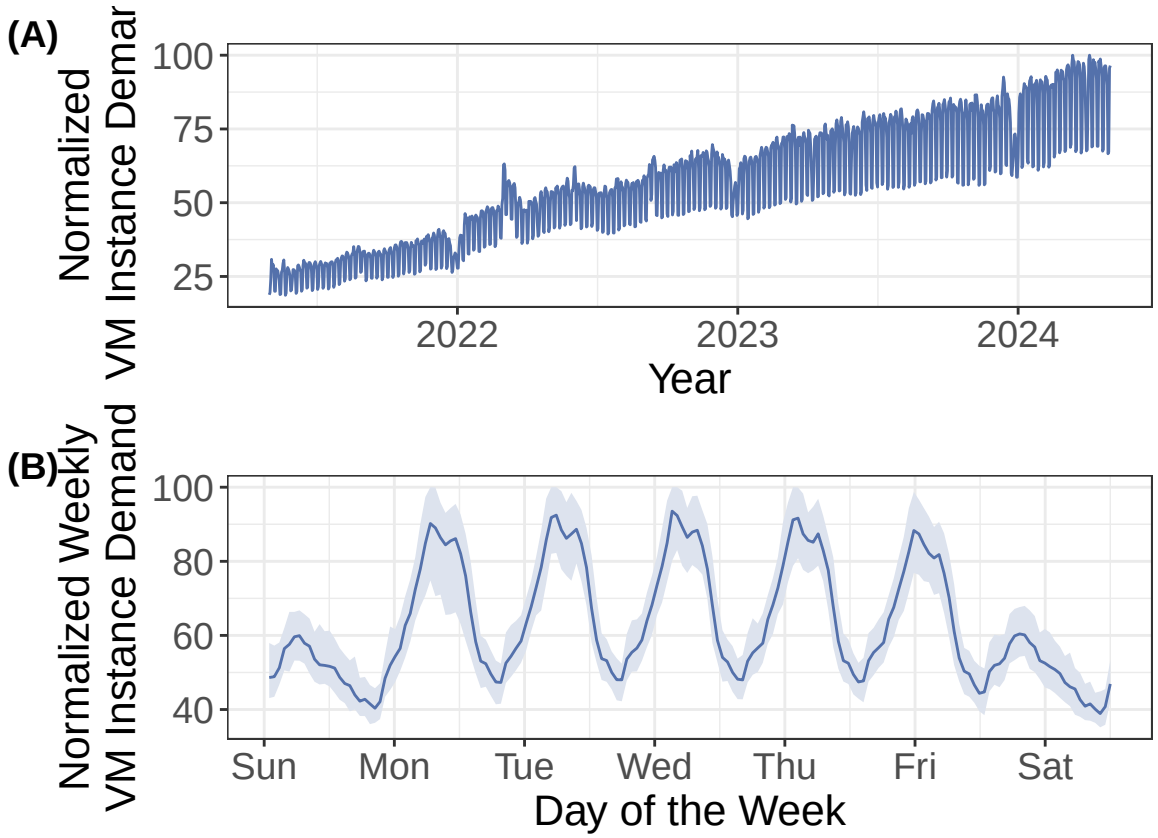
Now we break up the timeseries into 7 day partitions, calculate the quantiles at each time offset within the partitions to compute the range of weekly patterns.

```
weeks <- rollapply(all$TOTAL_COMPUTE, width=24*7, by=24*7,
                   FUN=function(x) return(x))

# Now we have a matrix.  Each row is one of the 1 week periods of time.
# Each 1-week period should be normalized separately to the peak demand that week.
```

```
weeks.norm <- apply(weeks, 1, function(x) { return ((100*x)/max(x)) })

# Now for each week, return a data frame and concat them together.
tmp.df <- do.call(rbind, lapply(1:ncol(weeks.norm), function(i) return (
  data.frame(x=weeks.norm[,i], y=1:length(weeks.norm[,i]),
             dow=rep(c("Su", "M", "T", "W", "R", "F", "S"), each=24, times=1)))))

# Now we generate a single data.frame of hourly quantiles.  One row for
# each hour in the week (24*7 = 168 rows) with a column for each of 7 different
# percentiles.

hourly.quantiles <- tmp.df %>% group_by(y) %>%
  summarize(p90 = quantile(x, probs=0.9),
            p95 = quantile(x, probs=0.95),
            p99 = quantile(x, probs=0.99),
            p10=quantile(x, probs=0.1),
            p5=quantile(x, probs=0.05),
            p1=quantile(x, probs=0.01),
            med=quantile(x, probs=0.5))

# Nice x-axis labels for the days of the week.
days <- c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat")

p3 <- ggplot(hourly.quantiles, aes(x=y, y=med)) +
  geom_ribbon(aes(ymin=p5, ymax=p95), fill="#DDE3EE") +
  geom_line(color="#5471AB") + theme_bw() +
  theme(axis.text = element_text(size=14),
        axis.title.x = element_text(size=16),
        axis.title.y = element_text(size=16)) +
  xlab("Day of the Week") + ylab("Normalized Weekly\nVM Instance Demand") +
  scale_x_continuous(breaks=seq(from=0, to=24*14, length.out=14),
                     labels=rep(days, 2))

p3
```




```
plot_grid(p2, p3, labels=c("(A)", "(B)"), scale=0.9, ncol=1)
```

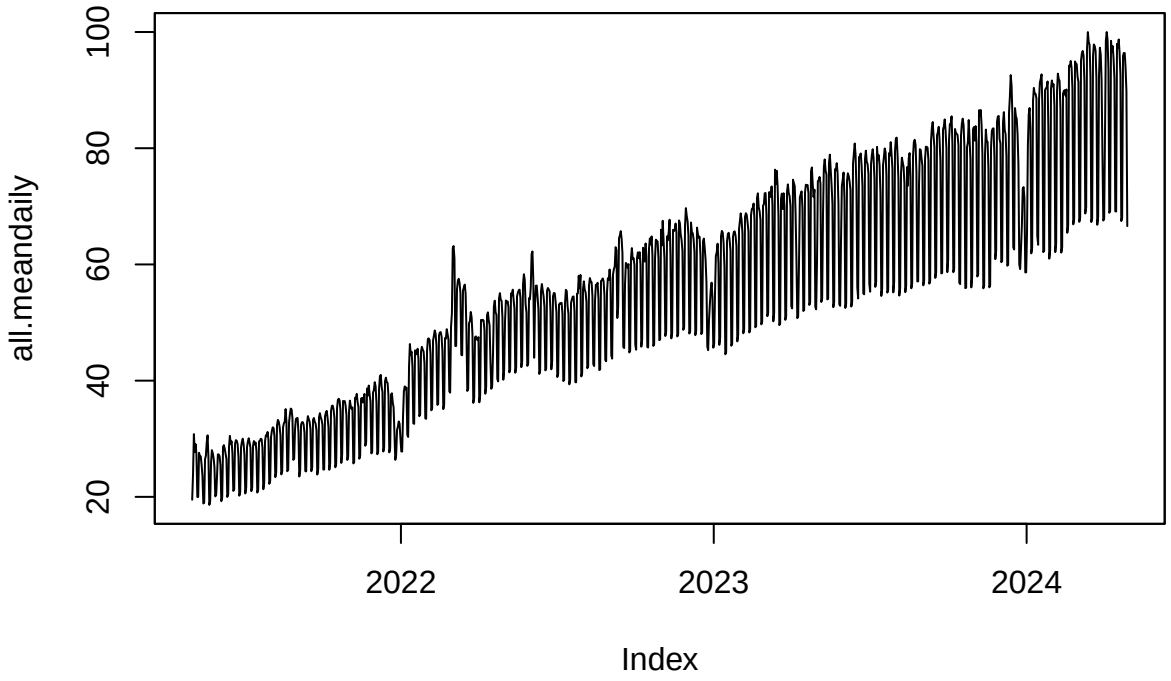



```
pdf("dailypattern.pdf", width=7, height=5.5)
plot_grid(p2, p3, labels=c("(A)", "(B)"), scale=0.9, ncol=1)
dev.off()
```

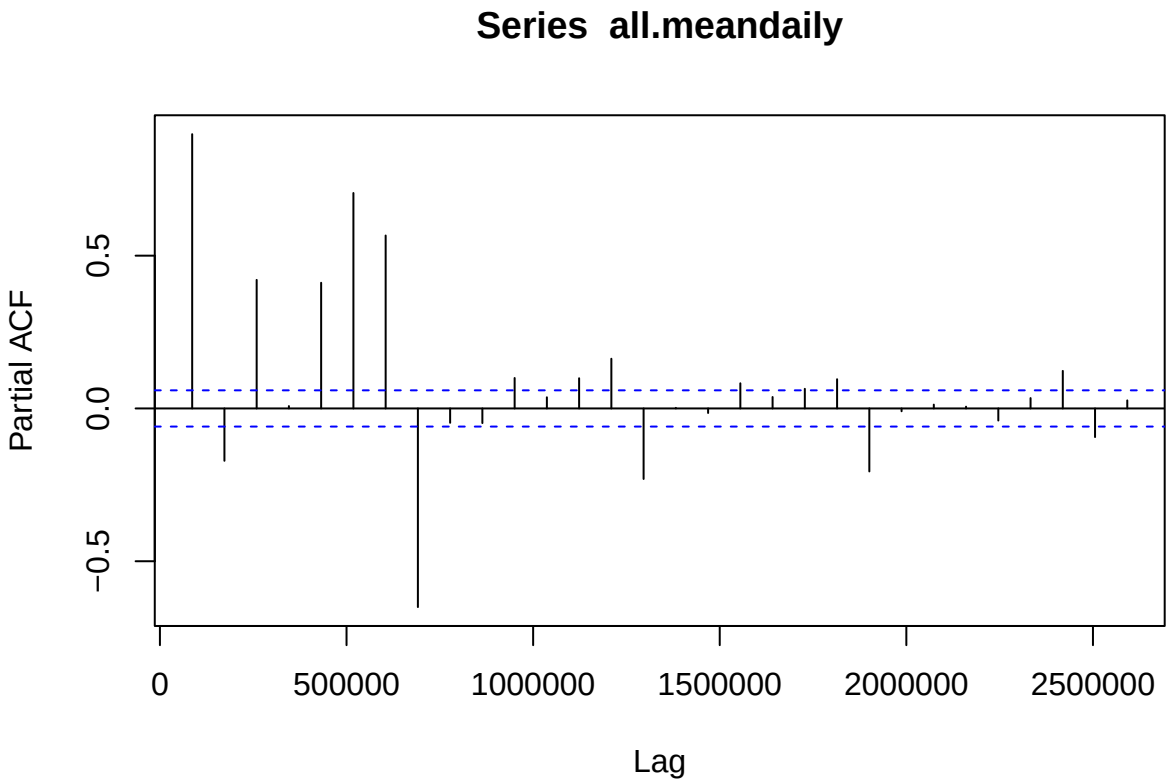
```
png("dailypattern.png", width=700, height=550)
plot_grid(p2, p3, labels=c("(A)", "(B)"), scale=0.9, ncol=1)
dev.off()
```

4 Autocorrelation Analysis

```
plot(all.meandaily)
```

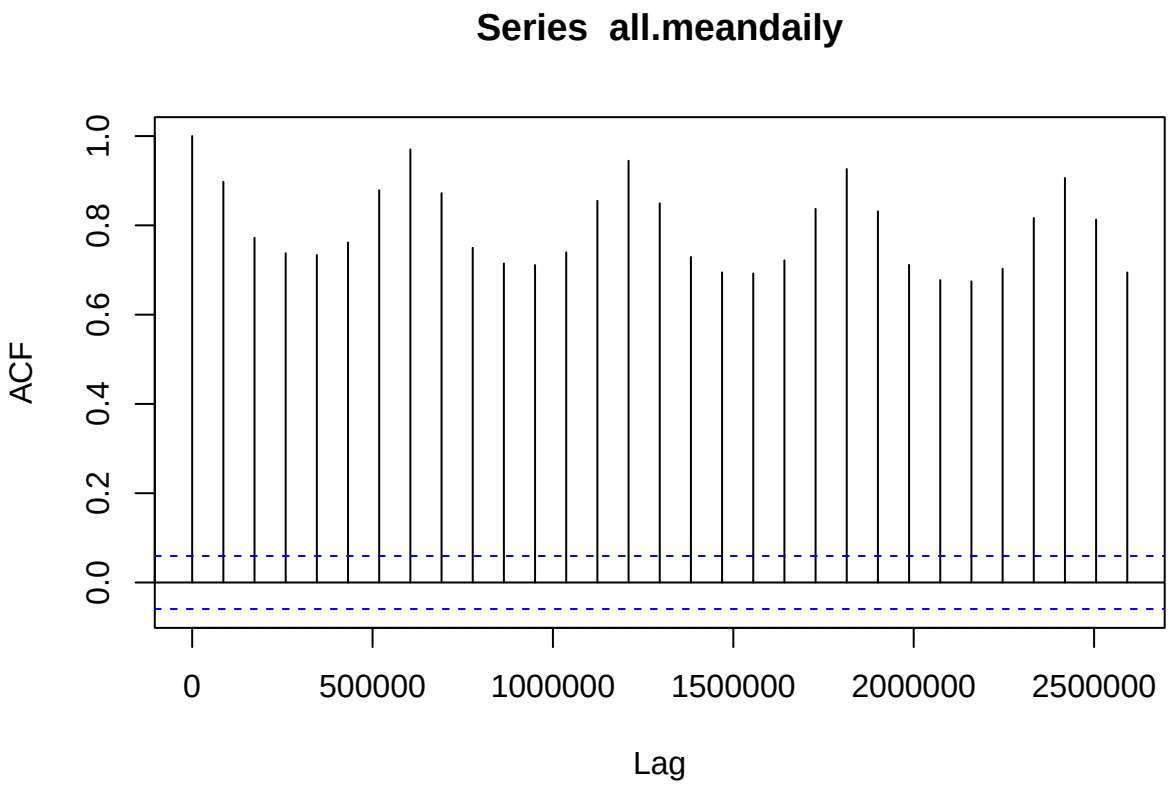



```
pacf(all.meandaily)
```




```
#data.zoo <- zoo(all$NORM_USAGE, all$USAGE_HOUR)
#data.sub
#data.sub.zoo <- zoo(data.sub$TOTAL_COST, data.sub$day)
#data.sub.zoo
#plot(data.zoo)
#pacf(data.sub.zoo)
```

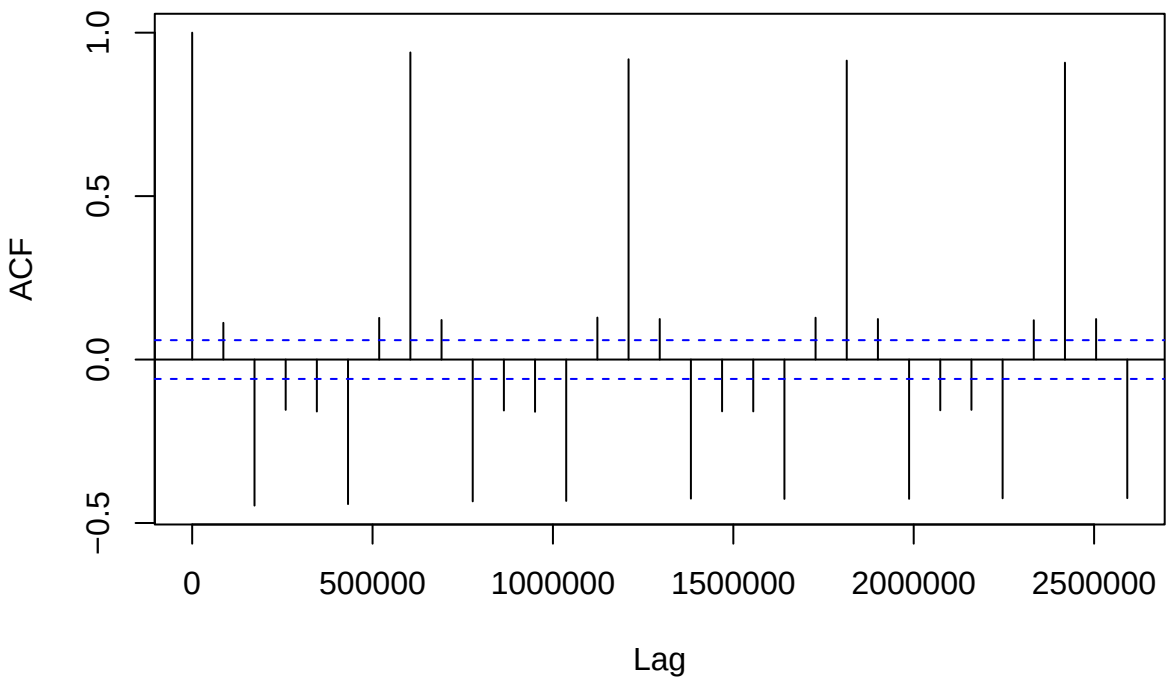
```r
#acf(data.sub.zoo)
z = acf(all.meandaily)
```




```r
z
```

```
##
## Autocorrelations of series 'all.meandaily', by lag
##
##       0   86400  172800  259200  345600  432000  518400  604800  691200  777600
##   1.000   0.898   0.772   0.738   0.734   0.762   0.879   0.970   0.872   0.750
##  864000  950400 1036800 1123200 1209600 1296000 1382400 1468800 1555200 1641600
##   0.715   0.711   0.740   0.855   0.945   0.849   0.729   0.695   0.692   0.722
## 1728000 1814400 1900800 1987200 2073600 2160000 2246400 2332800 2419200 2505600
##   0.837   0.926   0.831   0.711   0.677   0.675   0.703   0.817   0.906   0.813
## 2592000
##   0.694
```

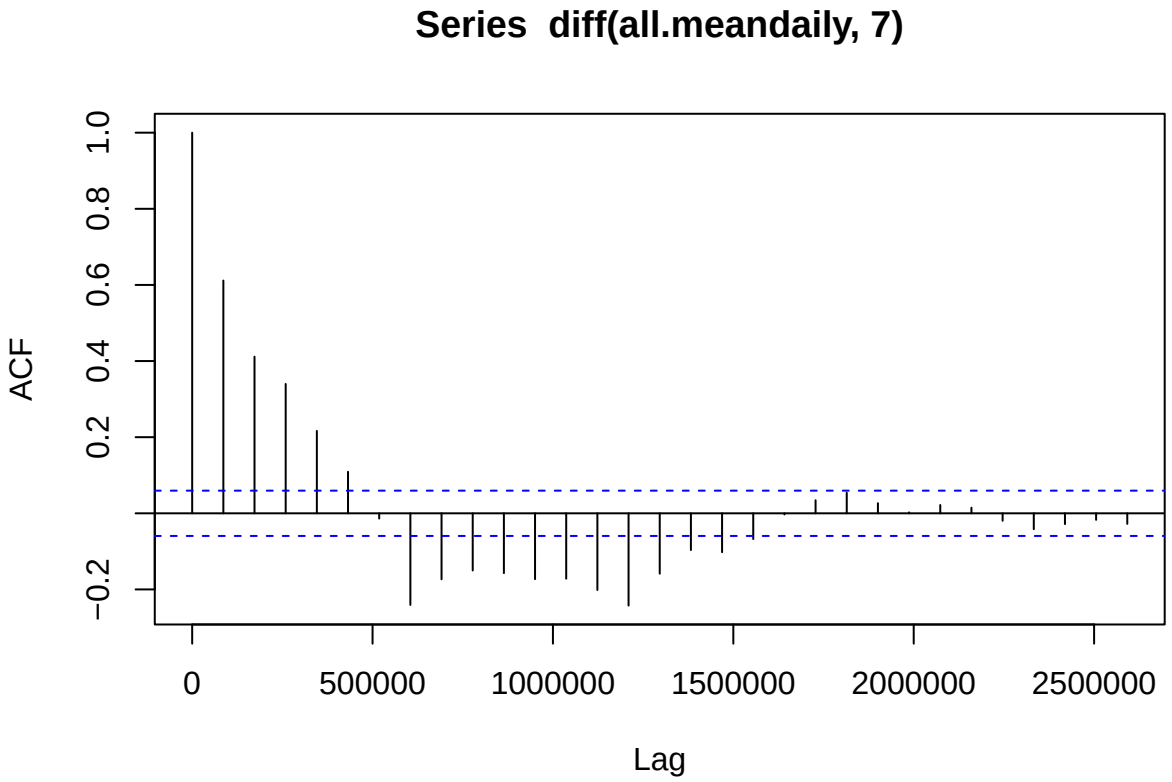
```r
y = acf(diff(all.meandaily))
```




```
y
```

```
##
## Autocorrelations of series 'diff(all.meandaily)', by lag
##
##       0   86400  172800  259200  345600  432000  518400  604800  691200  777600
##   1.000   0.112  -0.447  -0.154  -0.159  -0.442   0.127   0.940   0.121  -0.434
##  864000  950400 1036800 1123200 1209600 1296000 1382400 1468800 1555200 1641600
##  -0.156  -0.160  -0.432   0.128   0.919   0.124  -0.425  -0.158  -0.159  -0.426
## 1728000 1814400 1900800 1987200 2073600 2160000 2246400 2332800 2419200 2505600
##   0.128   0.915   0.124  -0.426  -0.155  -0.154  -0.424   0.120   0.908   0.124
## 2592000
##  -0.424
```

```
x = acf(diff(all.meandaily,7))
```




```
x
```

```
##
## Autocorrelations of series 'diff(all.meandaily, 7)', by lag
##
##       0   86400  172800  259200  345600  432000  518400  604800  691200  777600
##   1.000   0.612   0.412   0.340   0.217   0.109  -0.014  -0.241  -0.173  -0.150
##  864000  950400 1036800 1123200 1209600 1296000 1382400 1468800 1555200 1641600
##  -0.157  -0.173  -0.172  -0.202  -0.242  -0.159  -0.097  -0.102  -0.068  -0.003
## 1728000 1814400 1900800 1987200 2073600 2160000 2246400 2332800 2419200 2505600
##   0.034   0.054   0.027   0.003   0.022   0.015  -0.020  -0.042  -0.028  -0.017
## 2592000
##  -0.028
```

5 Hourly daily maximum vs daily minimum

```
all.maxweekly <- rollapply(all.meandaily, width=7, by=7, FUN=max)
all.minweekly <- rollapply(all.meandaily, width=7, by=7, FUN=min)
head(all.maxweekly)
```

```
## 2021-05-05 11:00:00 2021-05-12 11:00:00 2021-05-19 11:00:00 2021-05-26 11:00:00
##            30.80386            27.58307            30.59486            28.03859
## 2021-06-02 11:00:00 2021-06-09 11:00:00
##            27.32047            28.90139
```

```
head(all.minweekly)
```

```
## 2021-05-05 11:00:00 2021-05-12 11:00:00 2021-05-19 11:00:00 2021-05-26 11:00:00
##            19.50161            18.86388            18.61736            19.32476
## 2021-06-02 11:00:00 2021-06-09 11:00:00
```

```
##            19.26045            19.87138
```

```
head(all.maxweekly / all.minweekly)
```

```
## 2021-05-05 11:00:00 2021-05-12 11:00:00 2021-05-19 11:00:00 2021-05-26 11:00:00
##            1.579555            1.462216            1.643351            1.450915
## 2021-06-02 11:00:00 2021-06-09 11:00:00
##            1.418475            1.454423
```

```
mean(all.maxweekly / all.minweekly)
```

```
## [1] 1.427439
```

```
head(all.maxdaily)
```

```
## 2021-05-02 11:00:00 2021-05-03 11:00:00 2021-05-04 11:00:00 2021-05-05 11:00:00
##            17.17452            23.26870            30.19391            27.23915
## 2021-05-06 11:00:00 2021-05-07 11:00:00
##            28.16251            26.03878
```

```
head(all.mindaily)
```

```
## 2021-05-02 11:00:00 2021-05-03 11:00:00 2021-05-04 11:00:00 2021-05-05 11:00:00
##            10.52632            11.91136            13.20406            13.57341
## 2021-05-06 11:00:00 2021-05-07 11:00:00
##            13.94275            12.55771
```

```
head(all.maxdaily/all.mindaily)
```

```
## 2021-05-02 11:00:00 2021-05-03 11:00:00 2021-05-04 11:00:00 2021-05-05 11:00:00
##            1.631579            1.953488            2.286713            2.006803
## 2021-05-06 11:00:00 2021-05-07 11:00:00
##            2.019868            2.073529
```

```
mean(all.maxdaily/all.mindaily)
```

```
## [1] 1.955282
```

Average daily maximum is 34% higher than average daily minimum.

6 Holiday Effect Analysis

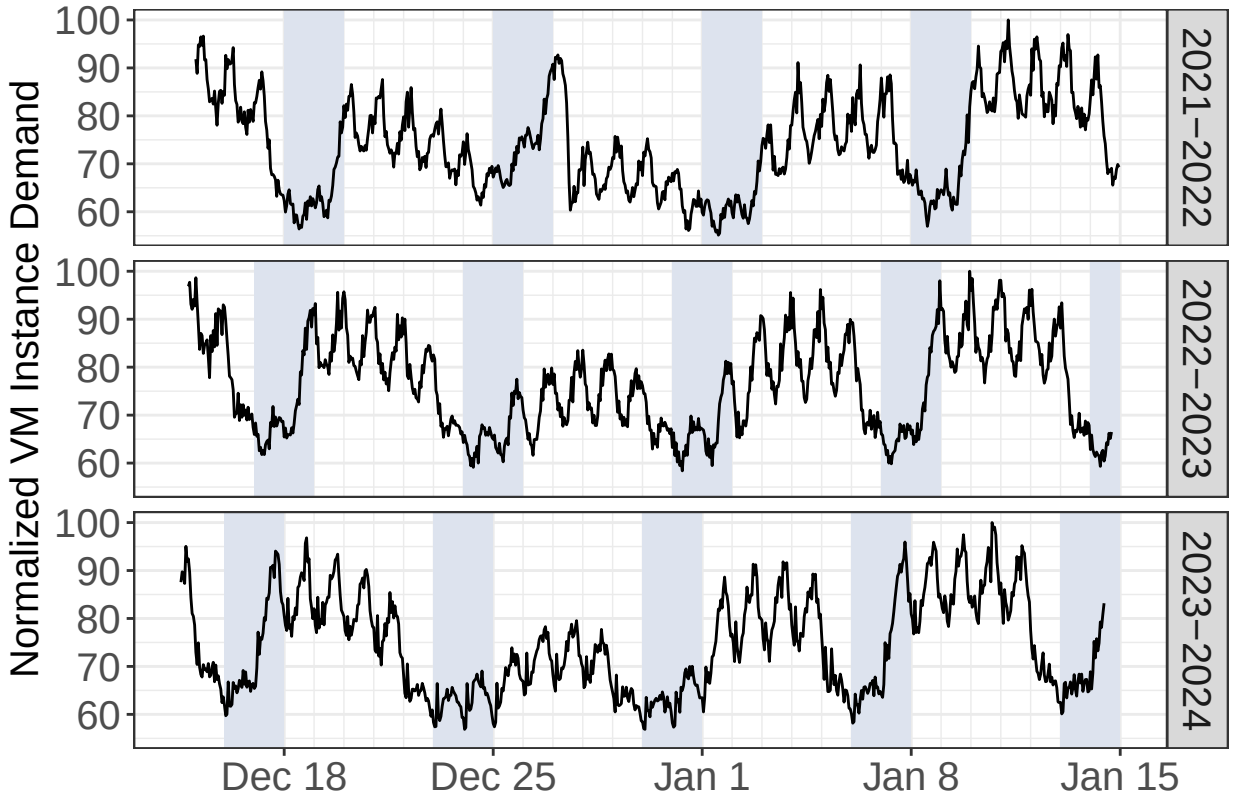
```
holidays.2022 <- subset(data.all, (USAGE_HOUR > as.POSIXct("2021-12-15") &
                                     USAGE_HOUR < as.POSIXct("2022-01-15")))
# Normalize this holiday period to 100 over the holidays.
holidays.2022$NORM_USAGE <- (100 * holidays.2022$NORM_USAGE) /
  max(holidays.2022$NORM_USAGE)
label.1 <- "2021-2022"
label.2 <- "2022-2023"
label.3 <- "2023-2024"
holidays.2022$year = label.1
holidays.2022$weekend.start = as.POSIXct("2021-12-18")
holidays.2022$weekend.end = as.POSIXct("2021-12-20")
holidays.2022$ts <- holidays.2022$USAGE_HOUR

highlights <- data.frame(year=c(rep(label.1, 4), rep(label.2, 5),
                                rep(label.3, 5)),
                         weekend.start=c(as.POSIXct("2021-12-18"),
```

```
                                        as.POSIXct("2021-12-25"),
                                        as.POSIXct("2022-01-01"),
                                        as.POSIXct("2022-01-08"),
                                        as.POSIXct("2021-12-17"),
                                        as.POSIXct("2021-12-24"),
                                        as.POSIXct("2021-12-31"),
                                        as.POSIXct("2022-01-07"),
                                        as.POSIXct("2022-01-14"),
                                        as.POSIXct("2021-12-16"),
                                        as.POSIXct("2021-12-23"),
                                        as.POSIXct("2021-12-30"),
                                        as.POSIXct("2022-01-06"),
                                        as.POSIXct("2022-01-13")),
                          weekend.end=c(as.POSIXct("2021-12-20"),
                                        as.POSIXct("2021-12-27"),
                                        as.POSIXct("2022-01-03"),
                                        as.POSIXct("2022-01-10"),
                                        as.POSIXct("2021-12-19"),
                                        as.POSIXct("2021-12-26"),
                                        as.POSIXct("2022-01-02"),
                                        as.POSIXct("2022-01-09"),
                                        as.POSIXct("2022-01-15"),
                                        as.POSIXct("2021-12-18"),
                                        as.POSIXct("2021-12-25"),
                                        as.POSIXct("2022-01-01"),
                                        as.POSIXct("2022-01-08"),
                                        as.POSIXct("2022-01-15")))

holidays.2023 <- subset(data.all, (USAGE_HOUR > as.POSIXct("2022-12-15") &
                                     USAGE_HOUR < as.POSIXct("2023-01-15")))
holidays.2023$ts <- holidays.2023$USAGE_HOUR - dyears(1)

holidays.2023$NORM_USAGE <- (100 * holidays.2023$NORM_USAGE) /
  max(holidays.2023$NORM_USAGE)
holidays.2023$year = label.2
holidays.2023$weekend.start = as.POSIXct("2021-12-17")
holidays.2023$weekend.end = as.POSIXct("2021-12-19")

holidays.2024 <- subset(data.all, (USAGE_HOUR > as.POSIXct("2023-12-15") &
                                     USAGE_HOUR < as.POSIXct("2024-01-15")))
holidays.2024$ts <- holidays.2024$USAGE_HOUR - dyears(2)

holidays.2024$NORM_USAGE <- (100 * holidays.2024$NORM_USAGE) /
  max(holidays.2024$NORM_USAGE)
holidays.2024$year = label.3
holidays.2024$weekend.start = as.POSIXct("2021-12-16")
holidays.2024$weekend.end = as.POSIXct("2021-12-18")

holidays.all <- rbind(holidays.2022, holidays.2023, holidays.2024)
head(holidays.all)
```

```
## # A tibble: 6 x 6
##   USAGE_HOUR          NORM_USAGE year    weekend.start       weekend.end
##   <dttm>                   <dbl> <chr>   <dttm>              <dttm>
```

```
## 1 2021-12-15 09:00:00       91.8 2021-2~ 2021-12-18 00:00:00 2021-12-20 00:00:00
## 2 2021-12-15 10:00:00       88.9 2021-2~ 2021-12-18 00:00:00 2021-12-20 00:00:00
## 3 2021-12-15 11:00:00       94.8 2021-2~ 2021-12-18 00:00:00 2021-12-20 00:00:00
## 4 2021-12-15 12:00:00       94.5 2021-2~ 2021-12-18 00:00:00 2021-12-20 00:00:00
## 5 2021-12-15 13:00:00       96.5 2021-2~ 2021-12-18 00:00:00 2021-12-20 00:00:00
## 6 2021-12-15 14:00:00       94.7 2021-2~ 2021-12-18 00:00:00 2021-12-20 00:00:00
## # i 1 more variable: ts <dttm>
```

```r
holidayPlot <- function() {
  highlights$ts <- min(holidays.all$ts)
  highlights$NORM_USAGE <- min(holidays.all$NORM_USAGE)
  ggplot(data=holidays.all, aes(x=ts, y=NORM_USAGE))  +
    geom_rect(data=highlights, aes(xmin=weekend.start, xmax=weekend.end,
                                   ymin=-Inf, ymax=Inf),
              fill="#DDE3EE",alpha=1.0) +
    geom_line() + facet_grid(rows = vars(year)) + theme_bw() +
    theme(axis.text=element_text(size=15), axis.title=element_text(size=15),
          strip.text=element_text(size=15), legend.text = element_text(size=15),
          legend.title = element_text(size=15)) +
    ylab("Normalized VM Instance Demand") + xlab("") +
    scale_x_continuous(breaks=c(as.POSIXct("2021-12-18"), as.POSIXct("2021-12-25"),
                                as.POSIXct("2022-01-01"), as.POSIXct("2022-01-08"),
                                as.POSIXct("2022-01-15")),
                       labels=c("Dec 18", "Dec 25", "Jan 1", "Jan 8", "Jan 15"),
                       minor_breaks=seq(from=as.POSIXct("2021-12-19"),
                                        to=as.POSIXct("2022-01-15"), by=86400))
}
holidayPlot()
```



```
## pdf
##   2
```

```
## pdf
##   2
```