Figures: Optimal Commitment Levels

Murray Stokely

2024-10-18

Contents

1 Introduction

This file includes figures and analysis for Section 3.2 on Setting Optimal Commitment Level for Periodic Demand.

We start by restricting to a 2-week subset of the data from January 2024, aggregate the different regions and SKUs together to a single timeseries of total VM demand, and normalize this to a 100 unit peak over that time window.

```
data <- read_parquet("../hourly_normalized.parquet")
data <- data %>%
  subset(USAGE_HOUR >= as.POSIXct("2024-01-07", tz="UTC")) %>%
  subset(USAGE_HOUR < as.POSIXct("2024-01-21", tz="UTC"))
dim(data)
```

```
## [1] 10850     4
```

```
head(data)
```

```
## # A tibble: 6 x 4
##   USAGE_HOUR          REGION_NUM INSTANCE_TYPE NORM_USAGE
##   <dttm>                   <dbl> <chr>              <dbl>
## 1 2024-01-07 00:00:00          4 C                      7
## 2 2024-01-07 00:00:00          1 D                      8
## 3 2024-01-07 00:00:00          3 D                      5
## 4 2024-01-07 00:00:00          1 L                     15
## 5 2024-01-07 00:00:00          2 D                      6
## 6 2024-01-07 00:00:00          3 E                      5
```

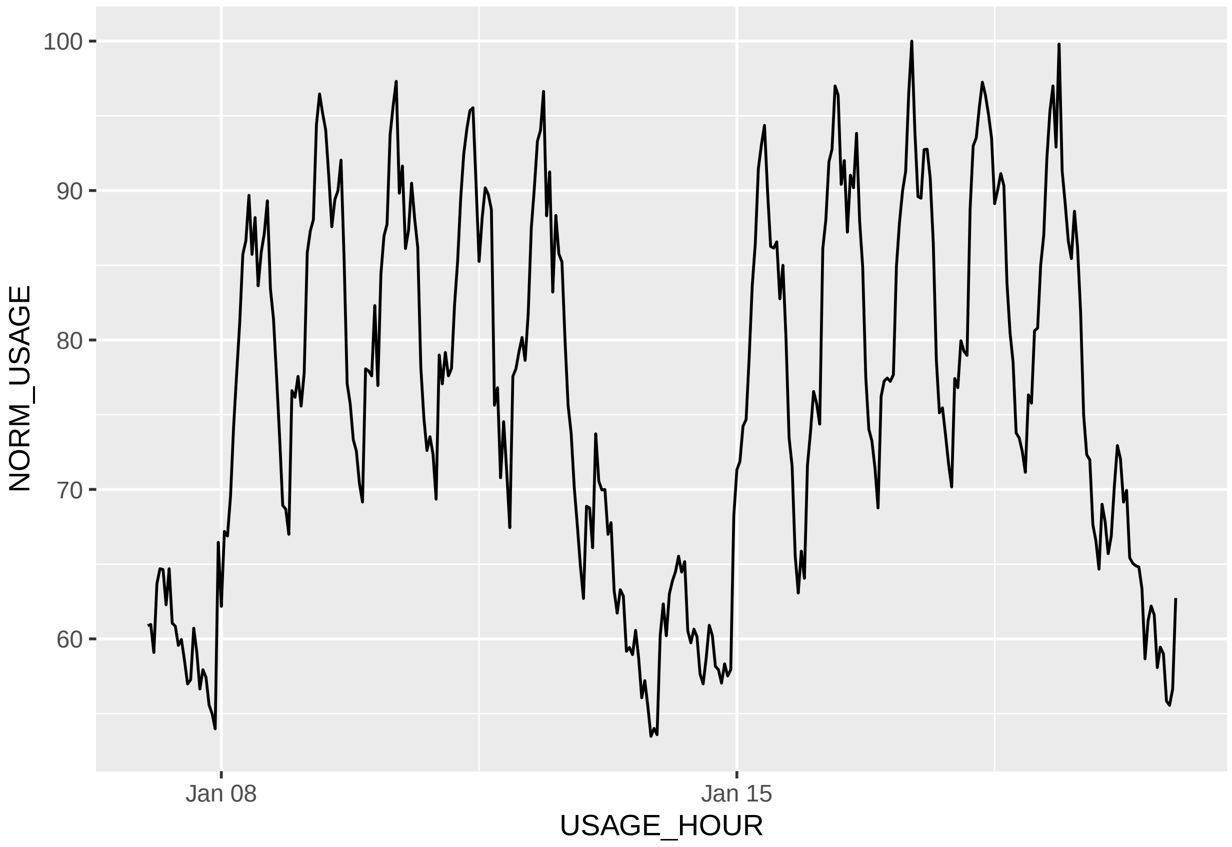
```
range(data$USAGE_HOUR)
```

```
## [1] "2024-01-07 00:00:00 UTC" "2024-01-20 23:00:00 UTC"
```

```
data.all <- data %>% group_by(USAGE_HOUR) %>% summarise(across(NORM_USAGE, sum))
data.all$NORM_USAGE = 100*data.all$NORM_USAGE / max(data.all$NORM_USAGE)
head(data.all)
```

```
## # A tibble: 6 x 2
##   USAGE_HOUR          NORM_USAGE
##   <dttm>                   <dbl>
## 1 2024-01-07 00:00:00       60.9
## 2 2024-01-07 01:00:00       61.0
## 3 2024-01-07 02:00:00       59.1
## 4 2024-01-07 03:00:00       63.7
## 5 2024-01-07 04:00:00       64.7
## 6 2024-01-07 05:00:00       64.6
```

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



2 Visualization

In order to illustrate the impact of savings plans on the amount of VM demand that is (1) covered by a savings plan, (2) purchased with on-demand rates, and (3) wasted as an unused savings plan, we introduce a simple 3-color area visualization with time on the x-axis and normalized cost on the y-axis.

Note that since the y-axis is cost, instead of VM instance hours, the more of the demand that is covered at expensive on-demand rates means the higher the y-axis will be.

```
GenerateBoxes <- function(df, on.demand.premium=2.1) {
  # GenerateBoxes - Create data.frame with pricing
  # Args:
  #   df: A data.frame with columns
  #     USAGE_HOUR: POSIXct hourly time
  #     NORM_USAGE: The VM demand at that hour
  # 3 boxes each time range - ununused, used with sp/ri coverage,
  #   used above sp/ri coverage level.
```

```r
  data.boxes <- data.frame(
    xmin = rep(head(df$USAGE_HOUR, -1), 3),
    xmax = rep(tail(df$USAGE_HOUR, -1), 3),
    Pricing = c(rep("SP Commitment", length(head(df$USAGE_HOUR, -1))),
                rep("Unused Commitment", length(head(df$USAGE_HOUR, -1))),
                rep("On Demand", length(head(df$USAGE_HOUR, -1)))),
    ymin = c(rep(0, length(head(df$USAGE_HOUR, -1))),
             head(ifelse(df$NORM_USAGE < df$SPRI_LEVEL, df$NORM_USAGE,
                         df$SPRI_LEVEL), -1),
             head(df$SPRI_LEVEL, -1)),
    ymax = c(head(ifelse(df$NORM_USAGE < df$SPRI_LEVEL, df$NORM_USAGE,
                         df$SPRI_LEVEL), -1),
             head(df$SPRI_LEVEL,-1),
             head(ifelse(df$NORM_USAGE > df$SPRI_LEVEL,
                         df$SPRI_LEVEL +
                           (df$NORM_USAGE - df$SPRI_LEVEL)*on.demand.premium,
                         df$SPRI_LEVEL), -1)))

  return(data.boxes)
}
```

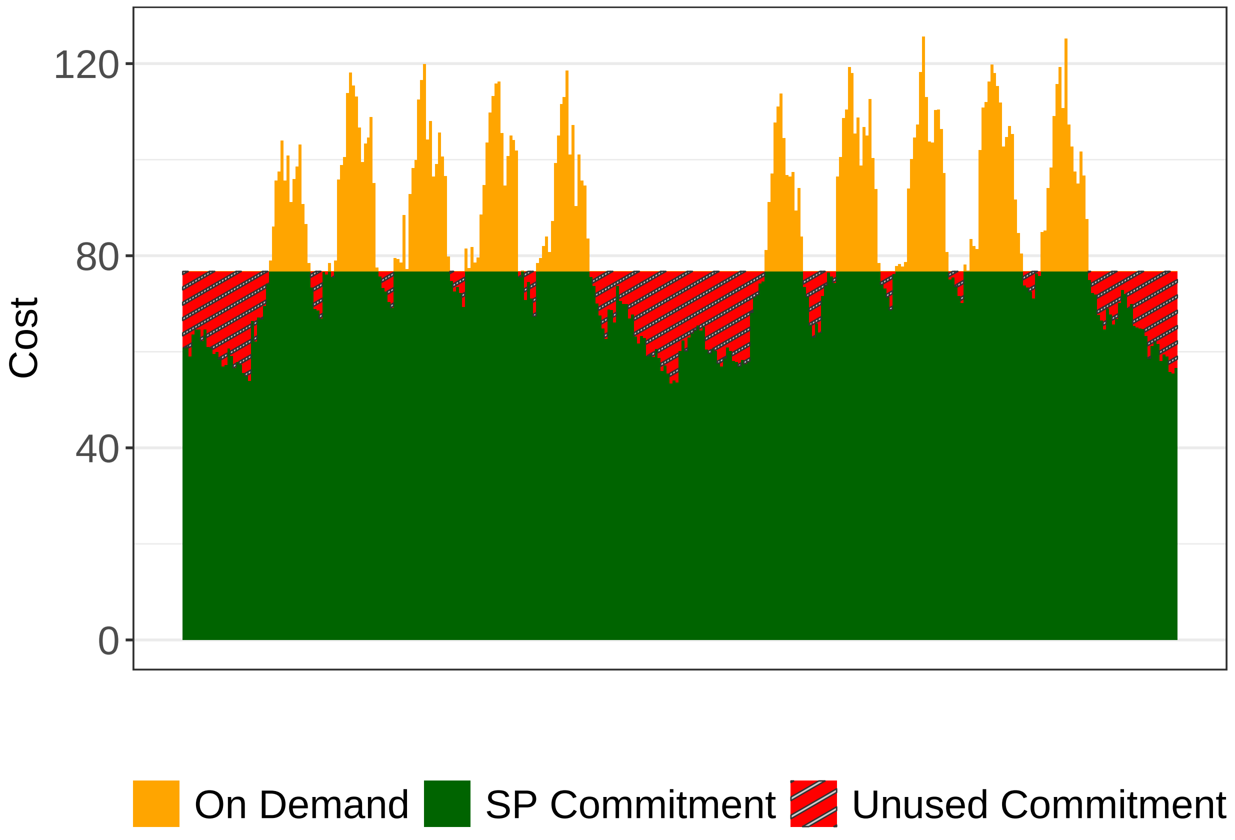
```r
AnnotateSPRILevel <- function(df, sp.level) {
  df$SPRI_LEVEL = sp.level
  return(df)
}

PlotBoxes <- function(df) {
  days <- c("S", "M", "T", "W", "R", "F", "S")
  return(ggplot(df) +
           geom_rect_pattern(aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax,
                                 fill=Pricing, pattern=Pricing),
                             colour=NA, pattern_size=0.25,
                             pattern_spacing=0.02) +
           theme_bw() +
           theme(axis.text=element_text(size=15),
                 axis.title=element_text(size=15),
                 legend.text = element_text(size=15),
                 legend.title = element_blank(),
                 legend.position="bottom",
                 strip.text=element_text(size=15)) +
           ylab("Cost") + xlab("") +
           scale_pattern_manual(values=c("none", "none", "stripe")) +
           scale_x_continuous(breaks=seq(from=as.POSIXct("2024-06-02 12:00:00"),
                                         to=as.POSIXct("2024-06-15 12:00:00"),
                                         length.out=14),
                              labels=rep(days,2)) +
           scale_fill_manual(values=c("orange", "darkgreen", "red")))
}
```

```r
# Set the Savings Plan commitment level to halfway between the max and min
# demand over the time period.
data.all <- AnnotateSPRILevel(data.all,
                              (max(data.all$NORM_USAGE) +
                                 min(data.all$NORM_USAGE))/2)
```

```
df <- GenerateBoxes(data.all)
head(df)
```

```
##                  xmin                xmax       Pricing ymin     ymax
## 1 2024-01-07 00:00:00 2024-01-07 01:00:00 SP Commitment    0 60.88054
## 2 2024-01-07 01:00:00 2024-01-07 02:00:00 SP Commitment    0 60.96467
## 3 2024-01-07 02:00:00 2024-01-07 03:00:00 SP Commitment    0 59.08581
## 4 2024-01-07 03:00:00 2024-01-07 04:00:00 SP Commitment    0 63.68480
## 5 2024-01-07 04:00:00 2024-01-07 05:00:00 SP Commitment    0 64.69434
## 6 2024-01-07 05:00:00 2024-01-07 06:00:00 SP Commitment    0 64.63825
```

```
PlotBoxes(df)
```



3 Optimization

The final step is to iterate over a number of possible SP commitment levels to compute the minimum cost option given the VM demand curve.

```
TotalCostPremium <- function(df) {
  # TotalCost - Computes the Total Cost Premium
  #
  # Args:
  #   df - A data.frame with 3 areas - unused, covered, on-demand.
  # Returns
  #   Cost premium

  l1 <- subset(df, Pricing == "SP Commitment")
  l2 <- subset(df, Pricing == "Unused Commitment")
```

```r
  l3 <- subset(df, Pricing == "On Demand")
  plan.cost <- sum(l1$ymax - l1$ymin)
  unused.cost <- sum(l2$ymax - l2$ymin)
  ondemand.cost <- sum(l3$ymax - l3$ymin)
  total.cost <- plan.cost + unused.cost + ondemand.cost
  return(total.cost)
}


# findOptimalSPLevel - Iteratively identify lowest cost Savings Plan level
# Args:
#   df: a data.frame containing 3 columns
#.     USAGE_HOUR - timestamp
#.     NORM_USAGE - normalized usage
#.  steps: The number of steps to iterate through the possible SP levels
# Returns:
#   A number corresponding to the optimal SP level to minimize cost.
findOptimalSPLevel <- function(df, steps=9) {
  min.demand <- min(df$NORM_USAGE)
  max.demand <- max(df$NORM_USAGE)
  total.days <- as.numeric(max(df$USAGE_HOUR) - min(df$USAGE_HOUR))

  fulldf <- NULL
  plts <- list()
  i<-1
  for (sp.level in seq(min.demand, max.demand, length=steps)) {
    df.ann <- AnnotateSPRILevel(df, sp.level)
    df.boxes <- GenerateBoxes(df.ann)
    df.boxes$sp.level.label <- sp.level
    cost.premium  <- TotalCostPremium(df.boxes)
    df.boxes$cost.premium <- cost.premium
    df.boxes$panel.title <- paste0(i, ". c=", round(sp.level, 1), " ",
                                   "C(c)=", round(cost.premium, 0))
    i <- i + 1

    if (is.null(fulldf)) {
      fulldf <- df.boxes
    } else {
      fulldf <- rbind(fulldf, df.boxes)
    }
  }
  return(fulldf)
}

findMinSPLevel <- function(df, steps=9) {
  min.demand <- min(df$NORM_USAGE)
  max.demand <- max(df$NORM_USAGE)
  total.days <- as.numeric(max(df$USAGE_HOUR) - min(df$USAGE_HOUR))

  lowest.cost <- NULL
  lowest.sp <- NULL
  i<-1
  for (sp.level in seq(min.demand, max.demand, length=steps)) {
    df.ann <- AnnotateSPRILevel(df, sp.level)
    df.boxes <- GenerateBoxes(df.ann)
    df.boxes$sp.level.label <- sp.level
    cost.premium  <- TotalCostPremium(df.boxes)
    if (is.null(lowest.cost)) {
      lowest.cost <- cost.premium
      lowest.sp <- sp.level
    } else if (cost.premium < lowest.cost) {
      lowest.cost <- cost.premium
      lowest.sp <- sp.level
    }
  }
  return(lowest.sp)
}

FindCost <- function(df, sp.level) {
  df.ann <- AnnotateSPRILevel(df, sp.level)
  df.boxes <- GenerateBoxes(df.ann)
  df.boxes$sp.level.label <- sp.level
  return(TotalCostPremium(df.boxes))
}

fulldf.1 <- findOptimalSPLevel(data.all)
lowest.1 <- findMinSPLevel(data.all, 100)
cost.1 <- FindCost(data.all, lowest.1)
print(lowest.1)
```

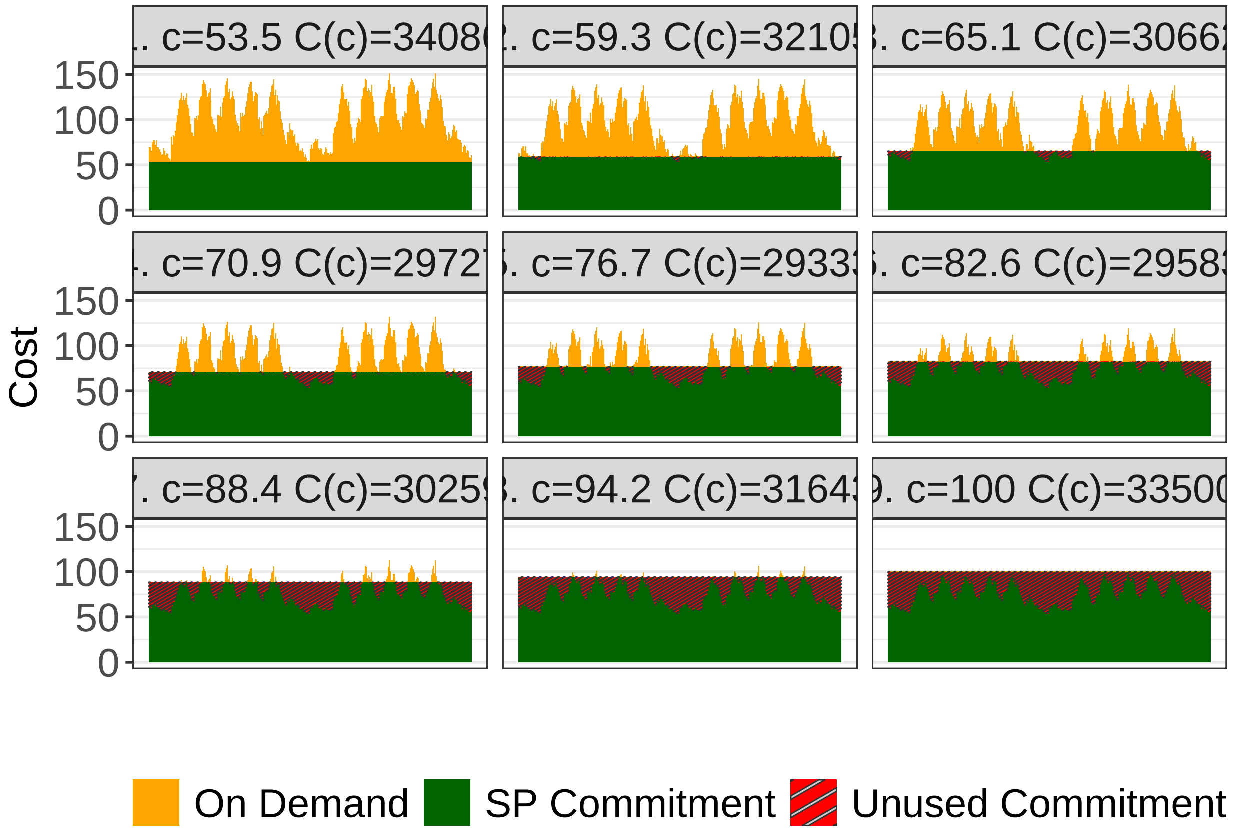
```
## [1] 77.44353
```

```r
print(cost.1)
```

```
## [1] 29326.44
```

And generate output PDF and PNGs:

```r
Plot3x3 <- function() {
  days <- c("S", "M", "T", "W", "R", "F", "S")
  ggplot(fulldf.1) +
    geom_rect_pattern(aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax,
                          fill=Pricing, pattern=Pricing),
                      colour=NA, pattern_size=0.25, pattern_spacing=0.02) +
    theme_bw() +
    theme(axis.text=element_text(size=15), axis.title=element_text(size=15),
          legend.text = element_text(size=15), legend.title = element_blank(),
          legend.position="bottom", strip.text=element_text(size=15)) +
    ylab("Cost") + xlab("") +
    scale_pattern_manual(values=c("none", "none", "stripe")) +
    scale_x_continuous(breaks=seq(from=as.POSIXct("2024-06-02 12:00:00"),
                                  to=as.POSIXct("2024-06-15 12:00:00"),
                                  length.out=14),
                      labels=rep(days,2)) +
    scale_fill_manual(values=c("orange", "darkgreen", "red")) +
    facet_wrap(~panel.title)
}
Plot3x3()
```



. c=53.5 C(c)=3408
. c=59.3 C(c)=3210
. c=65.1 C(c)=3066
. c=70.9 C(c)=2972
. c=76.7 C(c)=2933
. c=82.6 C(c)=2958
. c=88.4 C(c)=3025
. c=94.2 C(c)=3164
. c=100 C(c)=3350
Cost
150
100
50
0
On Demand
SP Commitment
Unused Commitment