Summary Report for Assignment 1

Henghui Qi

To write the object function, I processed the input "period" firstly. "Year", "start.date", and "end.date" were got from different situations of the length of "period". For example, the length of the annually period '2019A' is 5 and the length of the quarterly period '2019Q3' is 6. The start month and the end month of a quarter were calculated from the quarter number (3 for "Q3" for instance). The exact last day of the end month was calculated with the help of the first day of the next month.

```
l = nchar(period)#length of the string period
if(is.na(as.numeric(substr(period,1,4)))){stop("wrong input of period")}
Year = as.numeric(substr(period,1,4))
if(l==5){#annually
  Year = as.numeric(substr(period,1,4))
  start.date = ymd(paste0(Year,"0101"))
  end.date = ymd(paste0(Year,"1231"))

}else if(l==6){#quarterly
  Year = as.numeric(substr(period,1,4))
  Quarter = as.numeric(substr(period,6,6))
  start_month = 3*(Quarter-1) + 1
  end_month = start_month + 2
  start.date = ymd(paste0(Year,month.name[start_month],"01"))
  end.date = ymd(paste0(Year,month.name[end_month],"01"))%m+% months(1)%m-% days(1)
```

Then, I processed "peak.type". Given that peak types "flat" and "7x8" do not take holidays into account by their definitions, I calculated the result under these two situations.

```
if(peak.type == "flat"){
  include = 1
  hours = length(days)*24

}else if(peak.type=="7x8"){
  include = 1
  hours = length(days)*16
```

With other peak types, holidays' dates were obtained with *holidayNERC()* and "Year" I got from the "period" processing above. After that, I assigned "weekend" to 1 and 7 for Eastern markets and only 1 for Western markets which takes Saturday as a weekday. Then non-holiday weekdays are obtained by holidays and the "period" (with *wday()*). The hours can be calculated under "onpeak", "offpeak", and "2x16H" situations.

```
if(iso %in% c("PJMISO", "MISO", "ERCOT", "SPPISO", "NYISO")){#Eastern market
  weekend = c(1,7)
}else if(iso %in% c("WECC", "CAISO")){#Western market
  weekend = 1
}else {stop("wrong input of iso")}

non_holiday_weekdays =
  non_holidays[non_holidays%in%days[!weekdays%in%weekend]]
```

```
if(peak.type=="onpeak"){
  hours = length(non_holiday_weekdays)*16
}else if(peak.type=="offpeak"){
  include = 1
  hours = length(days)*24 - length(non_holiday_weekdays)*16
}else if(peak.type=="2x16H"){
  hours = length(unique(c(days[weekdays%in%weekend], holidays)))*16
}else {stop("wrong input of peak.type")}
```

Whether daylight-saving setting should be considered was marked as “include”. “Flat”, “7x8”, “offpeak” turned “include” to 1. If the second Sunday of March is in the “period” then the result hours should minus one and if the first Sunday of November is in the “period” then the result hours should plus one.

```
if(iso!="MISO"&&include==1){#daylight-saving setting
  march = seq(ymd(paste0(Year, "0301")), ymd(paste0(Year, "0331")), by = "day")
  mar_weekdays = wday(march)
  if(march[mar_weekdays==1][2]%in%days){hours = hours - 1}#it begins at 2:00 a.m. on the second Sunday of March
  november = seq(ymd(paste0(Year, "1101")), ymd(paste0(Year, "1130")), by = "day")
  nov_weekdays = wday(november)
  if(november[nov_weekdays==1][1]%in%days){hours = hours + 1}#it ends at 2:00 a.m. on the first Sunday of November
}
```

The return of the function is of correct format.

```
# test the type of the return
num.hours.ercot.onpeak.may19 <- get.hours("ERCOT", "onpeak", "2019May")
num.hours.ercot.onpeak.may19
```

```
## $iso
## [1] "ERCOT"
##
## $peak.type
## [1] "onpeak"
##
## $start.date
## [1] "2019-05-01"
##
## $end.date
## [1] "2019-05-31"
##
## $num.hour
## [1] 352
```

Wrong input situations are identified and showed.

```
# test wrong input situation
get.hours("wrong", "onpeak", "2019May")
```

```
## Error in get.hours("wrong", "onpeak", "2019May"): wrong input of iso
```

```
get.hours("ERCOT", "wrong", "2019May")
```

```
## Error in get.hours("ERCOT", "wrong", "2019May"): wrong input of peak.type
```

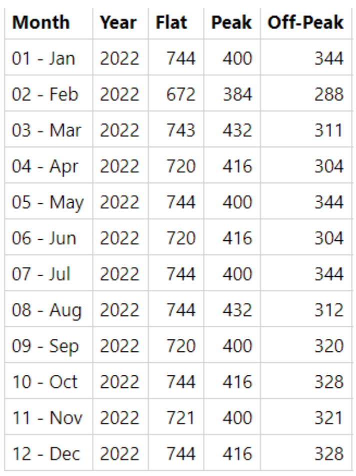
```
get.hours("ERCOT", "onpeak", "wrong")
```

```
## Warning in get.hours("ERCOT", "onpeak", "wrong"): 强制改变过程中产生了NA
```

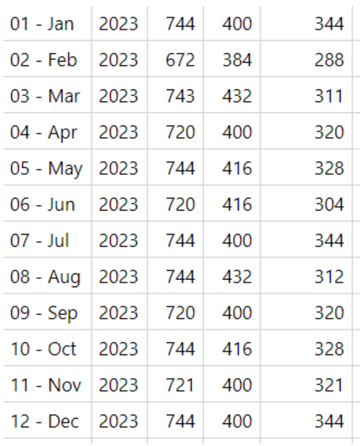
```
## Error in get.hours("ERCOT", "onpeak", "wrong"): wrong input of period
```

The test result of CAISO in 2022 to 2024 is consistent with the Power Calendar: https://www.energygps.com/HomeTools/PowerCalendar.
Annually:

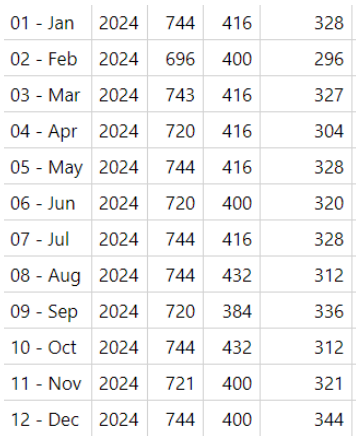
```
##     Month Year flat onpeak offpeak
## 1     Jan 2022  744    400     344
## 2     Feb 2022  672    384     288
## 3     Mar 2022  743    432     311
## 4     Apr 2022  720    416     304
## 5     May 2022  744    400     344
## 6     Jun 2022  720    416     304
## 7     Jul 2022  744    400     344
## 8     Aug 2022  744    432     312
## 9     Sep 2022  720    400     320
## 10    Oct 2022  744    416     328
## 11    Nov 2022  721    400     321
## 12    Dec 2022  744    416     328

## 13    Jan 2023  744    400     344
## 14    Feb 2023  672    384     288
## 15    Mar 2023  743    432     311
## 16    Apr 2023  720    400     320
## 17    May 2023  744    416     328
## 18    Jun 2023  720    416     304
## 19    Jul 2023  744    400     344
## 20    Aug 2023  744    432     312
## 21    Sep 2023  720    400     320
## 22    Oct 2023  744    416     328
## 23    Nov 2023  721    400     321
## 24    Dec 2023  744    400     344

## 25    Jan 2024  744    416     328
## 26    Feb 2024  696    400     296
## 27    Mar 2024  743    416     327
## 28    Apr 2024  720    416     304
## 29    May 2024  744    416     328
## 30    Jun 2024  720    400     320
## 31    Jul 2024  744    416     328
## 32    Aug 2024  744    432     312
## 33    Sep 2024  720    384     336
## 34    Oct 2024  744    432     312
## 35    Nov 2024  721    400     321
## 36    Dec 2024  744    400     344
```



Month	Year	Flat	Peak	Off-Peak
01 - Jan	2022	744	400	344
02 - Feb	2022	672	384	288
03 - Mar	2022	743	432	311
04 - Apr	2022	720	416	304
05 - May	2022	744	400	344
06 - Jun	2022	720	416	304
07 - Jul	2022	744	400	344
08 - Aug	2022	744	432	312
09 - Sep	2022	720	400	320
10 - Oct	2022	744	416	328
11 - Nov	2022	721	400	321
12 - Dec	2022	744	416	328



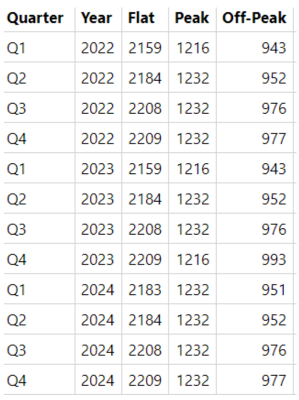
01 - Jan	2023	744	400	344
02 - Feb	2023	672	384	288
03 - Mar	2023	743	432	311
04 - Apr	2023	720	400	320
05 - May	2023	744	416	328
06 - Jun	2023	720	416	304
07 - Jul	2023	744	400	344
08 - Aug	2023	744	432	312
09 - Sep	2023	720	400	320
10 - Oct	2023	744	416	328
11 - Nov	2023	721	400	321
12 - Dec	2023	744	400	344



01 - Jan	2024	744	416	328
02 - Feb	2024	696	400	296
03 - Mar	2024	743	416	327
04 - Apr	2024	720	416	304
05 - May	2024	744	416	328
06 - Jun	2024	720	400	320
07 - Jul	2024	744	416	328
08 - Aug	2024	744	432	312
09 - Sep	2024	720	384	336
10 - Oct	2024	744	432	312
11 - Nov	2024	721	400	321
12 - Dec	2024	744	400	344

Quarterly:

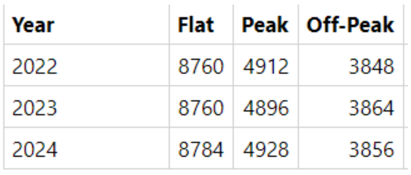
```
##    Quarter Year flat onpeak offpeak
## 1       Q1 2022 2159   1216     943
## 2       Q2 2022 2184   1232     952
## 3       Q3 2022 2208   1232     976
## 4       Q4 2022 2209   1232     977
## 5       Q1 2023 2159   1216     943
## 6       Q2 2023 2184   1232     952
## 7       Q3 2023 2208   1232     976
## 8       Q4 2023 2209   1216     993
## 9       Q1 2024 2183   1232     951
## 10      Q2 2024 2184   1232     952
## 11      Q3 2024 2208   1232     976
## 12      Q4 2024 2209   1232     977
```



Quarter	Year	Flat	Peak	Off-Peak
Q1	2022	2159	1216	943
Q2	2022	2184	1232	952
Q3	2022	2208	1232	976
Q4	2022	2209	1232	977
Q1	2023	2159	1216	943
Q2	2023	2184	1232	952
Q3	2023	2208	1232	976
Q4	2023	2209	1216	993
Q1	2024	2183	1232	951
Q2	2024	2184	1232	952
Q3	2024	2208	1232	976
Q4	2024	2209	1232	977

Yearly:

```
##   Year flat onpeak offpeak
## 1 2022 8760   4912    3848
## 2 2023 8760   4896    3864
## 3 2024 8784   4928    3856
```



Year	Flat	Peak	Off-Peak
2022	8760	4912	3848
2023	8760	4896	3864
2024	8784	4928	3856

Assignment1

Henghui Qi

2022/5/15

```
library(lubridate)
```

```
##
## 载入程辑包：'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(timeDate)

get.hours = function(iso, peak.type, period){
  #process date
  l = nchar(period)#length of the string period
  if(is.na(as.numeric(substr(period,1,4)))){stop("wrong input of period")}
  Year = as.numeric(substr(period,1,4))
  if(l==5){#annually
    Year = as.numeric(substr(period,1,4))
    start.date = ymd(paste0(Year,"0101"))
    end.date = ymd(paste0(Year,"1231"))

  }else if(l==6){#quarterly
    Year = as.numeric(substr(period,1,4))
    Quarter = as.numeric(substr(period,6,6))
    start_month = 3*(Quarter-1) + 1
    end_month = start_month + 2
    start.date = ymd(paste0(Year,month.name[start_month],"01"))
    end.date = ymd(paste0(Year,month.name[end_month],"01"))%m+% months(1)%m-% days(1)

  }else if(l==7){#monthly
    Year = as.numeric(substr(period,1,4))
    Month = which(month.abb==substr(period,5,7))
    start.date = ymd(paste0(Year,month.name[Month],"01"))
    end.date = ymd(paste0(Year,month.name[Month],"01"))%m+% months(1)%m-% days(1)

  }else if(l>7&&l<11){#daily
    period = ymd(period)
    Year = year(period)
    Month = month(period)
    Day = day(period)
    start.date = period
    end.date = period
  }else {stop("wrong input of period")}

  #_______________________________________________________
  #process peak type
  include = 0#whether this kind of peak type take daylight-saving into consideration
  days = seq(start.date, end.date, by = "days")
  if(peak.type == "flat"){
    include = 1
    hours = length(days)*24

  }else if(peak.type=="7x8"){
    include = 1
    hours = length(days)*16

  }else{
    interval = interval(start.date, end.date)
    weekdays = wday(days)
    year_holidays = as.Date(holidayNERC(Year))
    holidays = year_holidays[year_holidays%within%interval]
    non_holidays = days[!days%in%holidays]

    #process ISO
    if(iso %in% c("PJMISO", "MISO", "ERCOT", "SPPISO", "NYISO")){#Eastern market
      weekend = c(1,7)
    }else if(iso %in% c("WECC", "CAISO")){#Western market
      weekend = 1
    }else {stop("wrong input of iso")}

    non_holiday_weekdays =
      non_holidays[non_holidays%in%days[!weekdays%in%weekend]]
    if(peak.type=="onpeak"){
      hours = length(non_holiday_weekdays)*16
    }else if(peak.type=="offpeak"){
      include = 1
      hours = length(days)*24 - length(non_holiday_weekdays)*16
    }else if(peak.type=="2x16H"){
      hours = length(unique(c(days[weekdays%in%weekend], holidays)))*16
    }else {stop("wrong input of peak.type")}
  }

  if(iso!="MISO"&&include==1){#daylight-saving setting
    march = seq(ymd(paste0(Year, "0301")), ymd(paste0(Year, "0331")), by = "day")
    mar_weekdays = wday(march)
    if(march[mar_weekdays==1][2]%in%days){hours = hours - 1}#it begins at 2:00 a.m. on the sec
ond Sunday of March
    november = seq(ymd(paste0(Year, "1101")), ymd(paste0(Year, "1130")), by = "day")
    nov_weekdays = wday(november)
    if(november[nov_weekdays==1][1]%in%days){hours = hours + 1}#it ends at 2:00 a.m. on the fi
rst Sunday of November
  }

  #______________________________________________________
  #present the result
  result = list(iso = iso,
                peak.type = peak.type,
                start.date = start.date,
                end.date = end.date,
                num.hour = hours)
  return(result)
}
```

```
# test the type of the return
num.hours.ercot.onpeak.may19 <- get.hours("ERCOT", "onpeak", "2019May")
num.hours.ercot.onpeak.may19
```

```
## $iso
## [1] "ERCOT"
##
## $peak.type
## [1] "onpeak"
##
## $start.date
## [1] "2019-05-01"
##
## $end.date
## [1] "2019-05-31"
##
## $num.hour
## [1] 352
```

```
# test wrong input situation
get.hours("wrong", "onpeak", "2019May")
```

```
## Error in get.hours("wrong", "onpeak", "2019May"): wrong input of iso
```

```
get.hours("ERCOT", "wrong", "2019May")
```

```
## Error in get.hours("ERCOT", "wrong", "2019May"): wrong input of peak.type
```

```
get.hours("ERCOT", "onpeak", "wrong")
```

```
## Warning in get.hours("ERCOT", "onpeak", "wrong"): 强制改变过程中产生了NA
```

```
## Error in get.hours("ERCOT", "onpeak", "wrong"): wrong input of period
```

```
# test CAISO of 2022 to 2024
## monthly
peaktype = c("flat", "onpeak", "offpeak")
CAISO_calender_monthly = data.frame(Month = month.abb,
                                    Year = rep(2022:2024, each = 12),
                                    flat = 0,
                                    onpeak = 0,
                                    offpeak = 0)
for(y in 1:3){
  for(i in 1:12){
    period = paste0(y+2022-1, month.abb[i])
    for(j in 1:3){
      peak = peaktype[j]
      result = get.hours("CAISO", peak, period)
      CAISO_calender_monthly[(y-1)*12+i, j+2] = result$num.hour
      }
    }
}
print(CAISO_calender_monthly)
```

```
##    Month Year flat onpeak offpeak
## 1    Jan 2022  744    400     344
## 2    Feb 2022  672    384     288
## 3    Mar 2022  743    432     311
## 4    Apr 2022  720    416     304
## 5    May 2022  744    400     344
## 6    Jun 2022  720    416     304
## 7    Jul 2022  744    400     344
## 8    Aug 2022  744    432     312
## 9    Sep 2022  720    400     320
## 10   Oct 2022  744    416     328
## 11   Nov 2022  721    400     321
## 12   Dec 2022  744    416     328
## 13   Jan 2023  744    400     344
## 14   Feb 2023  672    384     288
## 15   Mar 2023  743    432     311
## 16   Apr 2023  720    400     320
## 17   May 2023  744    416     328
## 18   Jun 2023  720    416     304
## 19   Jul 2023  744    400     344
## 20   Aug 2023  744    432     312
## 21   Sep 2023  720    400     320
## 22   Oct 2023  744    416     328
## 23   Nov 2023  721    400     321
## 24   Dec 2023  744    400     344
## 25   Jan 2024  744    416     328
## 26   Feb 2024  696    400     296
## 27   Mar 2024  743    416     327
## 28   Apr 2024  720    416     304
## 29   May 2024  744    416     328
## 30   Jun 2024  720    400     320
## 31   Jul 2024  744    416     328
## 32   Aug 2024  744    432     312
## 33   Sep 2024  720    384     336
## 34   Oct 2024  744    432     312
## 35   Nov 2024  721    400     321
## 36   Dec 2024  744    400     344
```

```
## Quarterly
CAISO_calender_quarterly = data.frame(Quarter = rep(paste0("Q",1:4), 3),
                                    Year = rep(2022:2024, each = 4),
                                    flat = 0,
                                    onpeak = 0,
                                    offpeak = 0)
for(y in 1:3){
  for(i in 1:4){
    period = paste0(y+2022-1, paste0("Q",i))
    for(j in 1:3){
      peak = peaktype[j]
      result = get.hours("CAISO", peak, period)
      CAISO_calender_quarterly[(y-1)*4+i, j+2] = result$num.hour
      }
    }
}
print(CAISO_calender_quarterly)
```

```
##    Quarter Year flat onpeak offpeak
## 1       Q1 2022 2159   1216     943
## 2       Q2 2022 2184   1232     952
## 3       Q3 2022 2208   1232     976
## 4       Q4 2022 2209   1232     977
## 5       Q1 2023 2159   1216     943
## 6       Q2 2023 2184   1232     952
## 7       Q3 2023 2208   1232     976
## 8       Q4 2023 2209   1216     993
## 9       Q1 2024 2183   1232     951
## 10      Q2 2024 2184   1232     952
## 11      Q3 2024 2208   1232     976
## 12      Q4 2024 2209   1232     977
```

```
## Yearly
CAISO_calender_yearly = data.frame(Year = 2022:2024,
                                    flat = 0,
                                    onpeak = 0,
                                    offpeak = 0)
for(y in 1:3){
  period = paste0(y+2022-1, "A")
  for(j in 1:3){
    peak = peaktype[j]
    result = get.hours("CAISO", peak, period)
    CAISO_calender_yearly[y, j+1] = result$num.hour
    }
  }
print(CAISO_calender_yearly)
```

```
##   Year flat onpeak offpeak
## 1 2022 8760   4912    3848
## 2 2023 8760   4896    3864
## 3 2024 8784   4928    3856
```