Question 7.1

Describe a situation or problem from your job, everyday life, current events, etc., for which exponential smoothing would be appropriate. What data would you need? Would you expect the value of α (the first smoothing parameter) to be closer to 0 or 1, and why?

In the conversion of solar energy to electricity via the photovoltaic process, intuitively, the quantity/quality of sunlight is the most impactful variable for production. In both project planning and on-going operations, solar irradiance (typically noted in Watts / m^2) is a key component in projecting (planning) or confirming (on-going operations) the quantity of electricity generated. Sophisticated, laboratory calibrated meters can provide detailed, specific information for the solar irradiance experienced by a particular site/location.

However, as might also be expected for any measured variable, there can be a degree of variability within the measured quantity. Some of the variability can be plausibly explained (e.g. cloud movement, storms), while other variability could be erroneous (e.g. dirty measuring device, communication failure). In both instances, the variability can be attributed to some degree of randomness.

An exponential smoothing approach to the data could help to smooth peaks and valleys resulting from any or all of these phenomena. Initially, an alpha of a 0.7-0.8 would be suggested, as the system is rather predictable (i.e. the sun rises and sets each day), though unforeseen randomness experienced a particular location would still need to "determined"; this alpha value could then be refined as additional data is collected and compared against redundant measurements (if available) to better account for the level of randomness experienced by the system overall, and determine if further smoothing of the data is warranted.

Question 7.2

Using the 20 years of daily high temperature data for Atlanta (July through October) from Question 6.2 (file temps.txt), build and use an exponential smoothing model to help make a judgment of whether the unofficial end of summer has gotten later over the 20 years. (Part of the point of this assignment is for you to think about how you might use exponential smoothing to answer this question. Feel free to combine it with other models if you'd like to. There's certainly more than one reasonable approach.)

Note: in R, you can use either HoltWinters (simpler to use) or the smooth package's es function (harder to use, but more general). If you use es, the Holt-Winters model uses model="AAM" in the function call (the first and second constants are used "A"dditively, and the third (seasonality) is used "M"ultiplicatively; the documentation doesn't make that clear).

Several exponential smoothing models were built in R using the Holt-Winters model for each instance. The twenty years of temperature data were smoothed both together (using the seasonality component) and individually (using just the alpha coefficient). To analyze the data using the seasonality component of the Holt-Winters model, the data was stacked together, converted into a ts object, and then run through Holt-Winters model, using the multiplicative selection, but allowing alpha, beta, and gamma to "self"-determine based on minimizing the sum of squares optimization approach native to the function. Smoothing parameters via this approach were: alpha=0.615, beta=0.0, gamma=0.549.

The season factor of the model output was then broken back into annual components (minus the initial year as it not a part of the fitted output) for analysis on a per year basis with a descending CUSUM algorithm. The mean for the CUSUM algorithm was calculated using the first 90 data points (i.e. ~summer solstice to ~autumnal equinox, or "summer"). Values of T and C were iteratively adjusted to obtain non-reversing descending change detection across all years. For the season factor approach, T=0.12 and C=0.05.

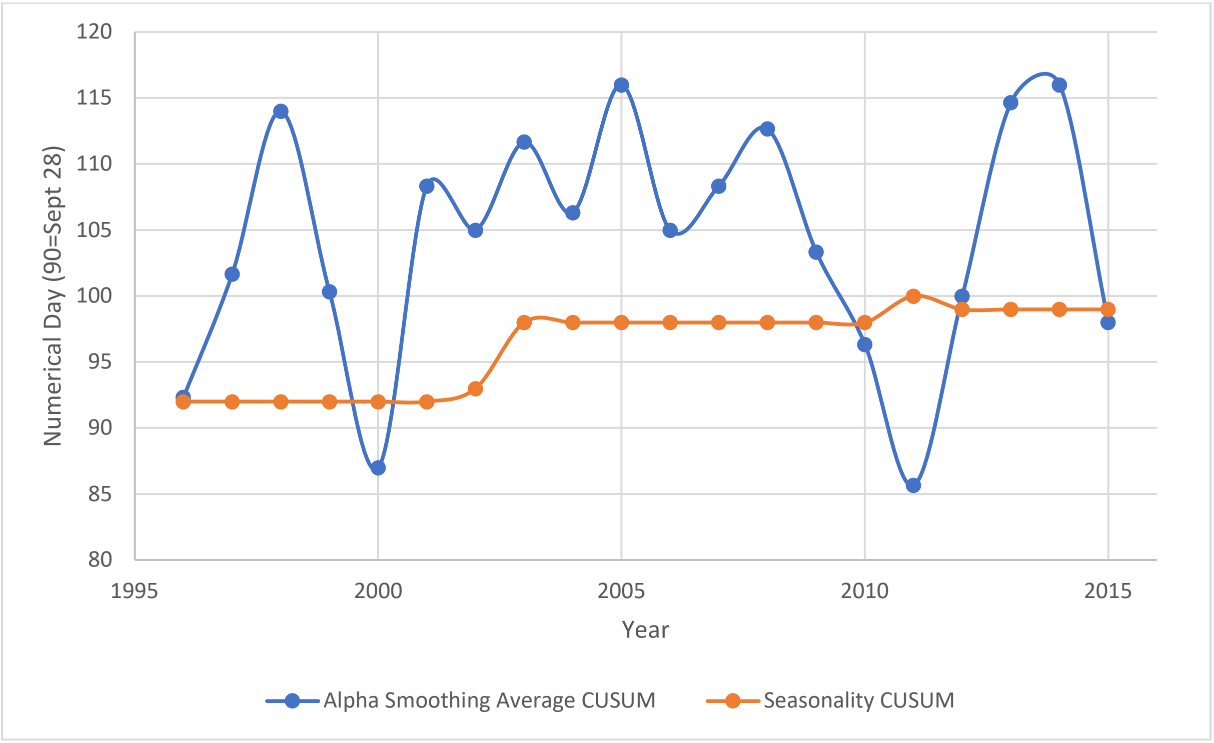
On an individual year basis, a simple exponential smoothing model (beta=False, gamma=False) was implemented, using the Holt-Winters model with alpha values of 0.1, 0.5, and 0.9. Ts objects were created for each year and run through the model. The output "level" values from the fitted model were then utilized in a descending CUSUM algorithm. As above, the mean for the CUSUM algorithm was calculated using the first 90 data points. Again, T and C values were iteratively adjusted to obtain non-reversing descending change detection across all years (for each alpha smoothing value). Values obtained were as follows:

alpha	T	C
0.1	8.0	3.0
0.5	13.0	2.25
0.9	14.0	8.0

For each value of alpha, the change detection was noted and summarized in an Excel table. The noted date of change was recorded for each year and then averaged to obtain an estimated end date of summer (day values are numerical, where July 1 = 1 and Oct 31 = 123).

Year	alpha=0.1	alpha=0.5	alpha=0.9	avg
1996	93	92	92	92
1997	108	89	108	102
1998	114	114	114	114
1999	98	98	105	100
2000	92	69	100	87
2001	108	108	109	108
2002	107	101	107	105
2003	120	95	120	112
2004	107	105	107	106
2005	116	116	116	116
2006	105	105	105	105
2007	105	104	116	108
2008	112	111	115	113
2009	106	97	107	103
2010	98	95	96	96
2011	95	69	93	86
2012	99	101	100	100
2013	115	112	117	115
2014	116	n/a	n/a	116
2015	95	88	111	98

Graphically, the average results, and the season factor results, are shown in the plot below.



Simple exponential smoothing (only alpha) on a per year basis provided generally similar results, which when averaged, did not seem to suggest any clear trend towards unofficial summer ending later in the calendar year. However, as indicated by the plot, the seasonal component does seem to suggest some modest movement (several days) to later in the calendar year, which may be indicative of a later unofficial end of summer.

```
library(stringr)
library(ggplot2)

temp_data<-read.csv("temps.csv",header=TRUE)
head(temp_data)
headers<-colnames(temp_data)

##############################
#Background reference information
#https://www.geeksforgeeks.org/how-to-check-if-characters-are-present-in-a-string-in-r/
#https://www.statology.org/str_replace-in-r/
#https://www.statology.org/r-combine-two-columns-into-one/
##############################

headers<-str_replace(headers,"X","")
headers<-headers[- 1]
headers<-headers[- 1]
df_all<-data.frame(matrix(nrow=0,ncol=2,dimnames=list(NULL,c("DD-MMM-YYYY","Temperature"))))
df_list<-list()

for (each in headers)
{
  temp_name<-paste0("df_",each)
  orig_name<-paste0("X",each)
  matrix_year<-matrix(data=each,nrow=nrow(temp_data),ncol=1)

  temp_df<-data.frame(paste0(temp_data$'DAY',"-",matrix_year,""),temp_data[orig_name])
  names(temp_df)<-c("DDMMMYYYY","Temperature")
  assign(temp_name,temp_df)
  df_all<-rbind(df_all,eval(parse(text=temp_name)))
  df_list<-append(df_list,temp_name)
}

df_all_ts<-ts(df_all["Temperature"],start=0,deltat=1/123)
df_model<-HoltWinters(df_all_ts,seasonal="mult")
df_model
#df_model$fitted
```

```
#plot(df_model)
write.csv(df_model$fitted,"df_seasonality.csv")
df_seasonality<-read.csv("df_seasonality.csv",header=TRUE)

seasonal_years<-c("X1997","X1998","X1999","X2000","X2001","X2002","X2003","X2004","X2005",
                 "X2006","X2007","X2008","X2009","X2010","X2011","X2012","X2013","X2014","X2015")

for (i in 1:19)
{
  x1<-1+123*(i-1)
  x2<-123*i
  #print(df_seasonality[x1:x2,5])

  temp_name<-paste0("df_seasonal_",seasonal_years[i])
  temp_df<-data.frame(df_seasonality[x1:x2,1],df_seasonality[x1:x2,5])
  names(temp_df)<-c("Day","Seasonality")
  assign(temp_name,temp_df)
}

df_fitted_list_season<-list()
mean_fitted_list_season<-list()

row_total<-123

j<-1
St<-matrix(data=0,nrow=row_total,ncol=length(seasonal_years))

for (year in (seasonal_years))
{
  temp_name<-paste0("df_seasonal_",year)
  df_temp<-eval(parse(text=temp_name))
  mean_df_temp<-mean(as.numeric(unlist(df_temp["Seasonality"][1:90,1])))
  assign(temp_name,df_temp)
  df_fitted_list_season<-append(df_fitted_list_season,temp_name)
  mean_fitted_list_season<-append(mean_fitted_list_season,mean_df_temp)

  t=0.12
  C=0.05

  for (i in 1:nrow(df_temp[1]))
  {
    if (i==1)
    {
      St_init<-max(0,0+(mean_df_temp-df_temp[i,2]-C))
    }
    if (i>1)
    {
      St_init<-max(0,St[i-1,j]+(mean_df_temp-df_temp[i,2]-C))
    }

    #Pre-scrub outputs

    if(St_init>=t)
    {
      St[i,j]<-St_init
    }
    if(St_init<t)
    {
      St[i,j]<-0.0
    }

    i<-i+1
  }
  j<-j+1
}

St_df_season<-data.frame(temp_data["DAY"][1:nrow(temp_data),1],St)
names(St_df_season)<-c("DAY",seasonal_years)
write.csv(St_df_season,"St_df_season.csv")


##################################################
##################################################
```

```
alpha_list<-list()
beta_list<-list()
gamma_list<-list()
SSE_list<-list()
year_list<-list()
alpha<-c(0.1,0.5,0.9)

csv_list<-list()

for (each in df_list)
{
  for (a in alpha)
  {
    year_list<-append(year_list,sub("df_","",each))
    temp_name<-eval(parse(text=each))
    df_ts1_temp<-temp_name["Temperature"]
    df_ts2_temp<-ts(df_ts1_temp,start=0,deltat=1/123)
    df_ts3_temp<-HoltWinters(df_ts2_temp,alpha=a,beta=FALSE,gamma=FALSE)
    assign(paste0(each,"_fitted"),df_ts3_temp)
    write.csv(df_ts3_temp$fitted,paste0(each,"_",a,"_fitted.csv"))
    csv_list<-append(csv_list,paste0(each,"_",a,"_fitted.csv"))
    alpha_list<-append(alpha_list,df_ts3_temp$alpha)
    beta_list<-append(beta_list,df_ts3_temp$beta)
    gamma_list<-append(gamma_list,df_ts3_temp$gamma)
    SSE_list<-append(SSE_list,df_ts3_temp$SSE)
    plot(df_ts3_temp)
    title(sub=paste0(each,"_",a))
  }
}

df_abg<-data.frame(unlist(year_list),unlist(alpha_list),unlist(beta_list),unlist(gamma_list),unlist(SSE_list))
names(df_abg)<-c("Year","Alpha","Beta","Gamma","SSE")
df_abg

##################################################
##################################################
```

```
csv_list_1<-list()
csv_list_2<-list()
csv_list_3<-list()

for (csv in csv_list)
{
  if(str_detect(csv,paste0("_",toString(alpha[1]),"_")))
  {
    csv_list_1<-append(csv_list_1,csv)
  }

  if(str_detect(csv,paste0("_",toString(alpha[2]),"_")))
  {
    csv_list_2<-append(csv_list_2,csv)
  }

  if(str_detect(csv,paste0("_",toString(alpha[3]),"_")))
  {
    csv_list_3<-append(csv_list_3,csv)
  }
}

##################################################
#alpha=0.1

df_fitted_list_1<-list()
mean_fitted_list_1<-list()

row_total<-nrow(df_ts3_temp$fitted)

j<-1
St<-matrix(data=0,nrow=row_total,ncol=length(csv_list_1))

for (csv in csv_list_1)
{
  temp_name<-sub("_fitted.csv","",csv)
  df_temp<-read.csv(csv,header=TRUE)
  mean_df_temp<-mean(as.numeric(unlist(df_temp["level"][1:90,1])))
  assign(temp_name,df_temp)
  df_fitted_list_1<-append(df_fitted_list_1,temp_name)
  mean_fitted_list_1<-append(mean_fitted_list_1,mean_df_temp)

  t=8.0
  C=3.0

  for (i in 1:nrow(df_temp[1]))
  {
   if (i==1)
   {
    St_init<-max(0,0+(mean_df_temp-df_temp[i,2]-C))
   }
   if (i>1)
   {
    St_init<-max(0,St[i-1,j]+(mean_df_temp-df_temp[i,2]-C))
   }

   #Pre-scrub outputs

   if(St_init>=t)
   {
    St[i,j]<-St_init
   }
   if(St_init<t)
   {
    St[i,j]<-0.0
   }

   i<-i+1
  }
  j<-j+1
}

St_df_1<-data.frame(temp_data["DAY"][2:nrow(temp_data),1],St)
names(St_df_1)<-c("DAY",sub(".csv","",csv_list_1))
```

```
write.csv(St_df_1,"St_df_1.csv")

###################################################
#alpha=0.5

df_fitted_list_2<-list()
mean_fitted_list_2<-list()

j<-1
St<-matrix(data=0,nrow=row_total,ncol=length(csv_list_2))

for (csv in csv_list_2)
{
 temp_name<-sub("_fitted.csv","",csv)
 df_temp<-read.csv(csv,header=TRUE)
 mean_df_temp<-mean(as.numeric(unlist(df_temp["level"][1:90,1])))
 assign(temp_name,df_temp)
 df_fitted_list_2<-append(df_fitted_list_2,temp_name)
 mean_fitted_list_2<-append(mean_fitted_list_2,mean_df_temp)

 t=13.0
 C=2.25

 for (i in 1:nrow(df_temp[1]))
 {
  if (i==1)
  {
   St_init<-max(0,0+(mean_df_temp-df_temp[i,2]-C))
  }
  if (i>1)
  {
   St_init<-max(0,St[i-1,j]+(mean_df_temp-df_temp[i,2]-C))
  }

  #Pre-scrub outputs

  if(St_init>=t)
```

```
    {
     St[i,j]<-St_init
    }
    if(St_init<t)
    {
     St[i,j]<-0.0
    }

    i<-i+1
  }
  j<-j+1
}

St_df_2<-data.frame(temp_data["DAY"][2:nrow(temp_data),1],St)
names(St_df_2)<-c("DAY",sub(".csv","",csv_list_2))
write.csv(St_df_2,"St_df_2.csv")

##################################################
#alpha=0.9

df_fitted_list_3<-list()
mean_fitted_list_3<-list()

j<-1
St<-matrix(data=0,nrow=row_total,ncol=length(csv_list_3))

for (csv in csv_list_3)
{
  temp_name<-sub("_fitted.csv","",csv)
  df_temp<-read.csv(csv,header=TRUE)
  mean_df_temp<-mean(as.numeric(unlist(df_temp["level"][1:90,1])))
  assign(temp_name,df_temp)
  df_fitted_list_3<-append(df_fitted_list_3,temp_name)
  mean_fitted_list_3<-append(mean_fitted_list_3,mean_df_temp)

  t=14.0
  C=8.0
```

```
 for (i in 1:nrow(df_temp[1]))
 {
  if (i==1)
  {
   St_init<-max(0,0+(mean_df_temp-df_temp[i,2]-C))
  }
  if (i>1)
  {
   St_init<-max(0,St[i-1,j]+(mean_df_temp-df_temp[i,2]-C))
  }

  #Pre-scrub outputs

  if(St_init>=t)
  {
   St[i,j]<-St_init
  }
  if(St_init<t)
  {
   St[i,j]<-0.0
  }

  i<-i+1
 }
 j<-j+1
}

St_df_3<-data.frame(temp_data["DAY"][2:nrow(temp_data),1],St)
names(St_df_3)<-c("DAY",sub(".csv","",csv_list_3))
write.csv(St_df_3,"St_df_3.csv")

##################################################
##################################################
#Plotting

df_all_fitted<-data.frame(df_1996_0.1["X"],
df_1996_0.1["level"],
df_1997_0.1["level"],
df_1998_0.1["level"],
df_1999_0.1["level"],
df_2000_0.1["level"],
df_2001_0.1["level"],
df_2002_0.1["level"],
df_2003_0.1["level"],
df_2004_0.1["level"],
df_2005_0.1["level"],
df_2006_0.1["level"],
df_2007_0.1["level"],
df_2008_0.1["level"],
df_2009_0.1["level"],
df_2010_0.1["level"],
df_2011_0.1["level"],
df_2012_0.1["level"],
df_2013_0.1["level"],
df_2014_0.1["level"],
df_2015_0.1["level"],
df_1996_0.5["level"],
df_1997_0.5["level"],
df_1998_0.5["level"],
df_1999_0.5["level"],
df_2000_0.5["level"],
df_2001_0.5["level"],
df_2002_0.5["level"],
df_2003_0.5["level"],
df_2004_0.5["level"],
df_2005_0.5["level"],
df_2006_0.5["level"],
df_2007_0.5["level"],
df_2008_0.5["level"],
df_2009_0.5["level"],
df_2010_0.5["level"],
df_2011_0.5["level"],
df_2012_0.5["level"],
                    df_2013_0.5["level"],
                    df_2014_0.5["level"],
                    df_2015_0.5["level"],
                    df_1996_0.9["level"],
                    df_1997_0.9["level"],
                    df_1998_0.9["level"],
                    df_1999_0.9["level"],
                    df_2000_0.9["level"],
                    df_2001_0.9["level"],
                    df_2002_0.9["level"],
                    df_2003_0.9["level"],
                    df_2004_0.9["level"],
                    df_2005_0.9["level"],
                    df_2006_0.9["level"],
                    df_2007_0.9["level"],
                    df_2008_0.9["level"],
                    df_2009_0.9["level"],
                    df_2010_0.9["level"],
                    df_2011_0.9["level"],
                    df_2012_0.9["level"],
                    df_2013_0.9["level"],
                    df_2014_0.9["level"],
                    df_2015_0.9["level"])

names(df_all_fitted)<-c("X",df_fitted_list_1,df_fitted_list_2,df_fitted_list_3)
year<-list(c(df_fitted_list_1,df_fitted_list_2,df_fitted_list_3))
mean_fitted_list<-list()
mean_fitted_list<-append(mean_fitted_list,mean_fitted_list_1)
mean_fitted_list<-append(mean_fitted_list,mean_fitted_list_2)
mean_fitted_list<-append(mean_fitted_list,mean_fitted_list_3)

plt<-ggplot(df_all_fitted,aes(X,group=1))

i<-1
for (each in unlist(year))
{
  print(plt+geom_line(aes_string(y=each),group=1,colour="red")+
        geom_hline(yintercept=unlist(mean_fitted_list[i]))+
        #geom_vline(xintercept=93)+
        labs(x="Day",y="Temperature",title=paste0(each," - Day vs Temperature")))
  i<-i+1
}

##################################################
##################################################
#Plotting 2

df_all_seasonal<-data.frame(df_seasonal_X1997["Day"],
                  df_seasonal_X1997["Seasonality"],
                  df_seasonal_X1998["Seasonality"],
                  df_seasonal_X1999["Seasonality"],
                  df_seasonal_X2000["Seasonality"],
                  df_seasonal_X2001["Seasonality"],
                  df_seasonal_X2002["Seasonality"],
                  df_seasonal_X2003["Seasonality"],
                  df_seasonal_X2004["Seasonality"],
                  df_seasonal_X2005["Seasonality"],
                  df_seasonal_X2006["Seasonality"],
                  df_seasonal_X2007["Seasonality"],
                  df_seasonal_X2008["Seasonality"],
                  df_seasonal_X2009["Seasonality"],
                  df_seasonal_X2010["Seasonality"],
                  df_seasonal_X2011["Seasonality"],
                  df_seasonal_X2012["Seasonality"],
                  df_seasonal_X2013["Seasonality"],
                  df_seasonal_X2014["Seasonality"],
                  df_seasonal_X2015["Seasonality"])

names(df_all_seasonal)<-c("Day",seasonal_years)

plt<-ggplot(df_all_seasonal,aes(Day,group=1))

i<-1
for (each in seasonal_years)
```

```
{
  print(plt+geom_line(aes_string(y=each),group=1,colour="red")+
          geom_hline(yintercept=unlist(mean_fitted_list_season[i]))+
          #geom_vline(xintercept=93)+
          labs(x="Day",y="Seasonality",title=paste0(each," - Seasonality vs Temperature")))
  i<-i+1
}
```