# 2021_KlamathMain_Linear

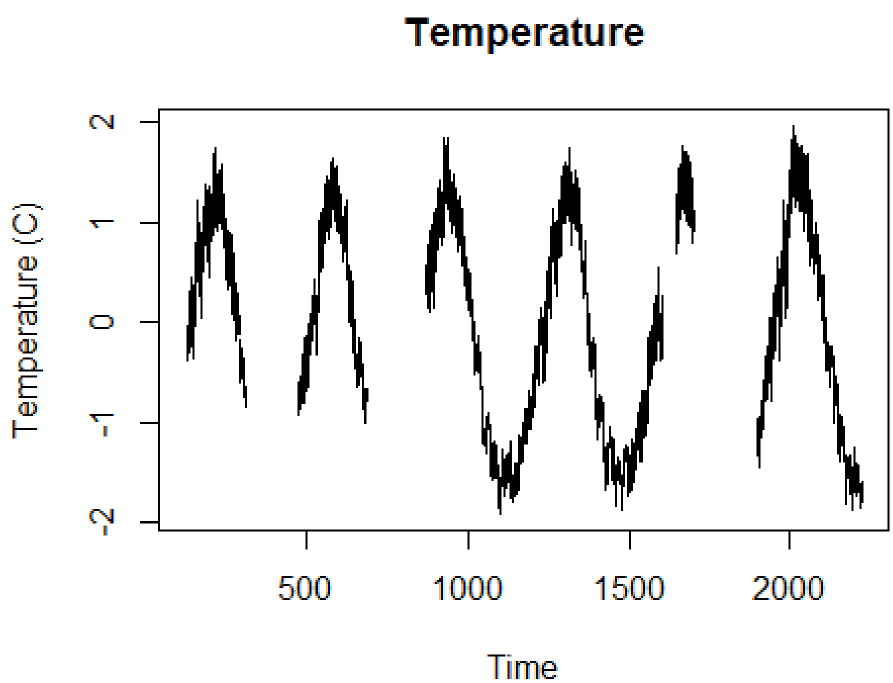
#Read in Data: Seiad Valley

```
##Reading in a 6 year 15 or 30 min temperature time series dataset from Klama
th River at Seiad Valley called KSV(Klamath Seiad Valley). Data collected by
Karuk Tribe
KSV <- read.csv("SeiadValley_KlamathMain_AllData.csv")
KSV$date <- lubridate::mdy_hm(KSV$Date_Time)#convert dates to POSIXct format

#Trim the dataset
KSV <- KSV[c(2585:117486),] #Most missing data is between 2015-Feb 2016, so r
emoving the first ~ year of data to make dataset less sketchy

#Check for missing data
missing_data <- KSV[!complete.cases(KSV),]
missing_data
#Need a complete dataset with no missing data. Since there is missing data in
this dataset, we will need to interpolate

#Bin data by hour and average temperature recordings to the hourly level
KSV$hour <- lubridate::floor_date(KSV$date, unit="hour") #Before we interpola
te, let's bin by hour
KSV_hourly <- KSV %>% #Summarize recordings to the hourly level (we have a mi
x of 30 min and 15 min readings)
  group_by(hour) %>%
  summarize(mean_temp=mean(Temp))

head(KSV_hourly) #check the dataset start date, use for "hour" sequence
tail(KSV_hourly) #check the dataset end date, use for "hour" sequence

#Create hourly sequence to ensure all missing data is accounted for
hour <- seq(mdy_h('5/2/2016 15'),mdy_h('2/1/2022 13'),by = "hour") #Create an
object that goes hour by hour for the entire time series to ensure that ALL m
issing data is accounted for
hour <- as.data.frame(hour) #convert "hour" to data frame
KSV_hourly <- left_join(hour, KSV_hourly) #left join hour and dataset
```

```
## Joining, by = "hour"
```

```
#Convert NaNs to NAs
KSV_hourly$mean_temp[KSV_hourly$mean_temp == "NaN"] <- NA

#Double check missing data
missing_data <- KSV_hourly[!complete.cases(KSV_hourly),]
missing_data #Now we are sure that all the missing hour time steps are includ
ed.
```

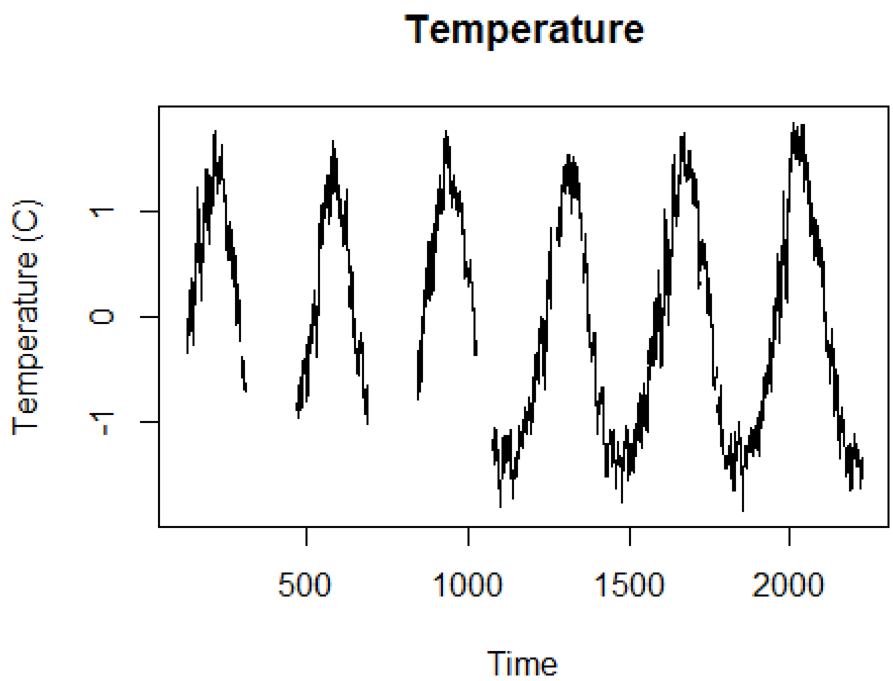
```
#z score to control for outliers
KSV_hourly$zTemp <- zscore(KSV_hourly$mean_temp)

#Convert to time series
KSV_ts <- ts(KSV_hourly$zTemp, start = c(123, 15), frequency = 24) # This tim
e series starts on 2 May 2016 at 2 am, so it starts on day 123 (leap year) at
hour 15 and the frequency is 24 (24 hours per day)
#^^^This is very confusing and I still don't fully understand how to convert
data to time series so may want to ask Albert for clarification.
ts.plot(KSV_ts,main="Temperature",ylab = "Temperature (C)", xlab = "Time")
```




#Read in Data: Orleans

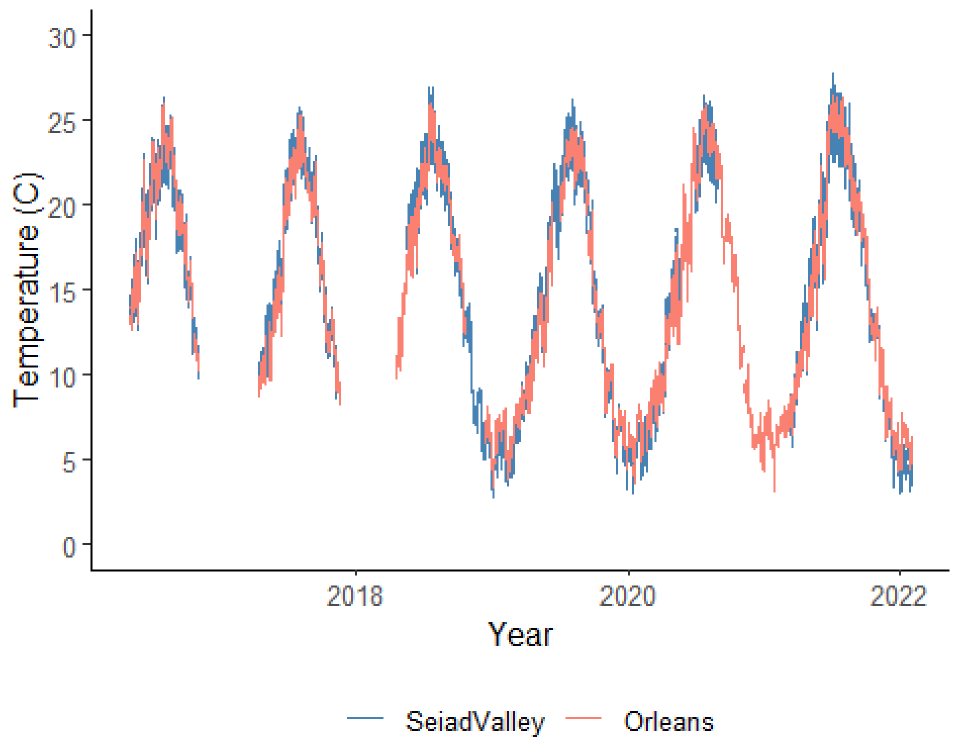
```
##Reading in a 6 year 15 or 30 min temperature time series dataset from Klama
th River at Seiad Valley called KSV(Klamath Seiad Valley). Data collected by
Karuk Tribe
KO <- read.csv("Orleans_KlamathMain_AllData.csv")
KO$date <- lubridate::mdy_hm(KO$Date_Time)#convert dates to POSIXct format

#Trim the dataset
KO <- KO[c(4190:135357),] #Removing the first ~ year of data to align with Se
iad Valley dataset

#Check for missing data
```

```
missing_data <- KO[!complete.cases(KO),]
missing_data
#Need a complete dataset with no missing data. Since there is missing data in
this dataset, we will need to interpolate

#Bin data by hour and average temperature recordings to the hourly level
KO$hour <- lubridate::floor_date(KO$date, unit="hour") #Before we interpolate
, let's bin by hour
KO_hourly <- KO %>% #Summarize recordings to the hourly level (we have a mix
of 30 min and 15 min readings)
  group_by(hour) %>%
  summarize(mean_temp=mean(Temp))

head(KO_hourly) #check the dataset start date, use for "hour" sequence
tail(KO_hourly) #check the dataset end date, use for "hour" sequence

#Create hourly sequence to ensure all missing data is accounted for
hour <- seq(mdy_h('5/2/2016 15'),mdy_h('2/1/2022 13'),by = "hour") #Create an
object that goes hour by hour for the entire time series to ensure that ALL m
issing data is accounted for
hour <- as.data.frame(hour) #convert "hour" to data frame
KO_hourly <- left_join(hour, KO_hourly) #left join hour and dataset

## Joining, by = "hour"

#Convert NaNs to NAs
KO_hourly$mean_temp[KO_hourly$mean_temp == "NaN"] <- NA

#Double check missing data
missing_data <- KO_hourly[!complete.cases(KO_hourly),]
missing_data #Now we are sure that all the missing hour time steps are includ
ed.

#z score to control for outliers
KO_hourly$zTemp <- zscore(KO_hourly$mean_temp)

#Convert to time series
KO_ts <- ts(KO_hourly$zTemp, start = c(123, 15), frequency = 24) # This time
series starts on 2 May 2016 at 2 am, so it starts on day 123 (leap year) at h
our 15 and the frequency is 24 (24 hours per day)
#^^^This is very confusing and I still don't fully understand how to convert
data to time series so may want to ask Albert for clarification.
ts.plot(KO_ts,main="Temperature",ylab = "Temperature (C)", xlab = "Time")
```



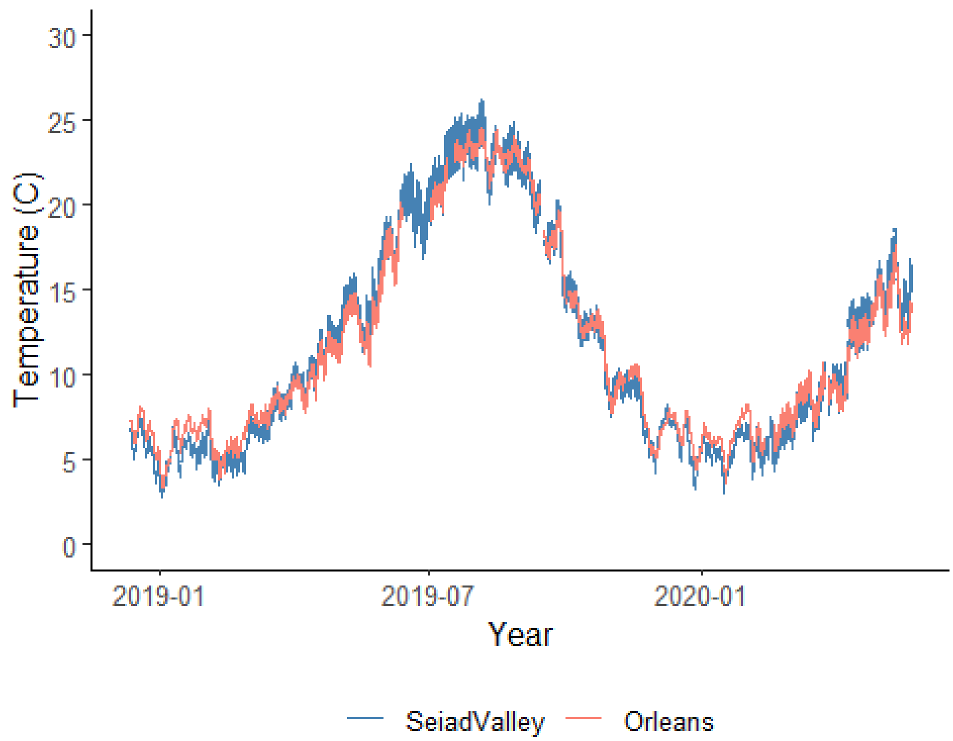

#Compare Orleans and Seiad Valley datasets

```
#Tightly correlated-- 0.99
cor(KSV_hourly$mean_temp, KO_hourly$mean_temp, use = "na.or.complete", method
= "pearson")

## [1] 0.9918316

#plot
ggplot()+
  geom_line(data = KSV_hourly, aes(x = hour, y = mean_temp, color = "SeiadVal
ley"))+
  geom_line(data = KO_hourly, aes(x = hour, y = mean_temp, color = "Orleans")
)+
  labs(x = "Year",
       y = "Temperature (C)")+
  theme_classic()+
  theme(text=element_text(size=12), legend.position = "bottom")+
  scale_colour_manual("", values = c("SeiadValley"="steelblue", "Orleans"="sa
lmon")) +
  scale_y_continuous("Temperature (C)", limits = c(0,30), breaks = 5*0:30)
```




#Trim Overlapping part of datasets

```
#Start overlap: 2018-12-12 10:00:00 (from KO dataset, 22892)
#End overlap: 2020-05-08 10:00:00 (from KSV dataset, 35515)

KO_hourly_cut <- KO_hourly[c(22892:35516),]
KSV_hourly_cut <- KSV_hourly[c(22892:35516),]

ggplot()+
  geom_line(data = KSV_hourly_cut, aes(x = hour, y = mean_temp, color = "Seia
dValley"))+
  geom_line(data = KO_hourly_cut, aes(x = hour, y = mean_temp, color = "Orlea
ns"))+
  labs(x = "Year",
       y = "Temperature (C)")+
  theme_classic()+
  theme(text=element_text(size=12), legend.position = "bottom")+
  scale_colour_manual("", values = c("SeiadValley"="steelblue", "Orleans"="sa
lmon")) +
  scale_y_continuous("Temperature (C)", limits = c(0,30), breaks = 5*0:30)
```




#Check correlation between overlapping data points

```
cor(KSV_hourly_cut$mean_temp, KO_hourly_cut$mean_temp, use = "na.or.complete"
, method = "pearson")

## [1] 0.9921006

#correlation coefficient is 0.9921006
```

#Create a linear model

```
#convert dataset to time series to ensure equal steps
KSV_cut_ts <- ts(KSV_hourly_cut$mean_temp, start = c(346, 10), frequency = 24
)
KO_cut_ts <- ts(KO_hourly_cut$mean_temp, start = c(346, 10), frequency = 24)

fit_lm <- lm(KSV_cut_ts~KO_cut_ts, na.action=na.exclude)  # fit with na.exclu
de
fit_lm#So the equation y = mx+b is... KSV = 1.0617(KO) - 0.7997

##
## Call:
## lm(formula = KSV_cut_ts ~ KO_cut_ts, na.action = na.exclude)
##
## Coefficients:
## (Intercept)    KO_cut_ts
##     -0.7997       1.0617
```

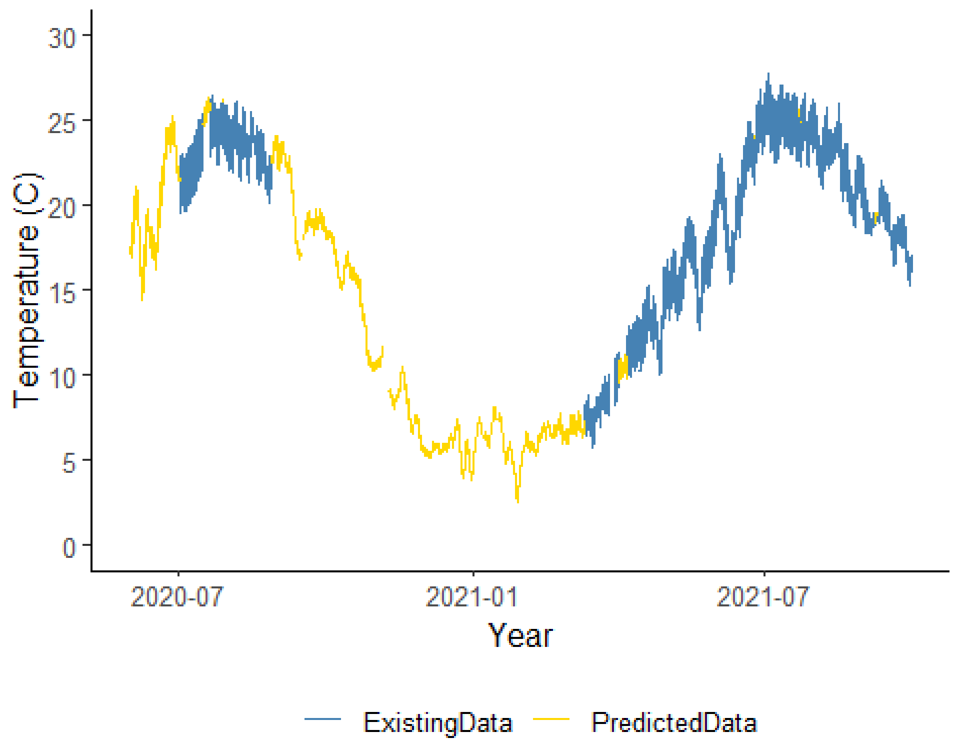
#Predict missing KSV values

```
#Let's chop off everything prior to June 1 2020 bc we don't care about that,
and chop off everything after 1 October 2021 bc we also don't care about that
KSV_hourly_2020 <- KSV_hourly[c(35770:47457),]
KO_hourly_2020 <- KO_hourly[c(35770:47457),]
KO_hourly_2020_ts <- ts(KO_hourly_2020$mean_temp, start = c(152, 0), frequenc
y = 24)#need it in ts format for predict function

#Need to name the "newdata" variable that we are plugging into the predict fu
nction the SAME NAME as the variable we used in the fit_lm linear equation ab
ove. Not sure why but this works.
KO_cut_ts <- KO_hourly_2020_ts

#Run the predict model, newdata being the values of KO that align with the mi
ssing values of KSV, but named same as linear model
KSV_predict <- predict(fit_lm, newdata = KO_cut_ts, )
KSV_predict <- as.data.frame(KSV_predict)
str(KO_hourly_2020)#checking # of results

## 'data.frame':    11688 obs. of  3 variables:
##  $ hour     : POSIXct, format: "2020-06-01 00:00:00" "2020-06-01 01:00:00"
...
##  $ mean_temp: num  17.2 17.2 17.1 17 16.9 ...
##  $ zTemp    : num  0.384 0.373 0.36 0.346 0.338 ...

#left join the datasets
KSV_KO_2020 <- left_join(KO_hourly_2020,KSV_hourly_2020,by="hour")
#Add the values from predict model to the joined dataset
KSV_KO_2020 <- cbind(KSV_KO_2020,KSV_predict$KSV_predict)

#Plug in the predicted values to the missing KSV temps
KSV_KO_2020$mean_temp.y <- ifelse(is.na(KSV_KO_2020$mean_temp.y), KSV_KO_2020
$KSV_predict, KSV_KO_2020$mean_temp.y)

#Make a separate predicted vector for plotting purposes
KSV_KO_2020$predicted <- (ifelse(is.na(KSV_KO_2020$zTemp.y), KSV_KO_2020$KSV_
predict, NA))#We already plugged in missing mean_temps for KSV, so use the zT
emp variable which is still empty for predicted values. This is just for grap
hing.
```

#Plot missing KSV values

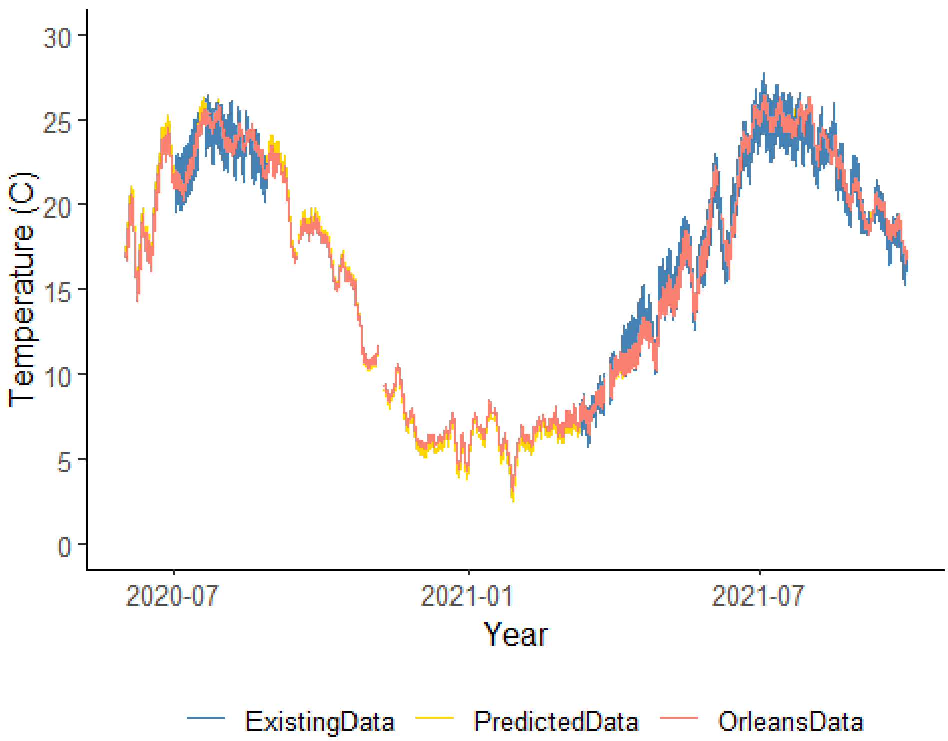
```
ggplot()+
  geom_line(data = KSV_KO_2020, aes(x = hour, y = mean_temp.y, color = "Exist
ingData"))+
  geom_line(data = KSV_KO_2020, aes(x = hour, y = predicted, color = "Predict
edData"))+
  labs(x = "Year",
       y = "Temperature (C)")+
  theme_classic()+
```

```
  theme(text=element_text(size=12), legend.position = "bottom")+
  scale_colour_manual("", values = c("ExistingData"="steelblue", "PredictedDa
ta"="gold")) +
  scale_y_continuous("Temperature (C)", limits = c(0,30), breaks = 5*0:30)

## Warning: Removed 418 row(s) containing missing values (geom_path).
```




#Plot predicted values with Orleans data

```
ggplot()+
  geom_line(data = KSV_KO_2020, aes(x = hour, y = mean_temp.y, color = "Exist
ingData"))+
  geom_line(data = KSV_KO_2020, aes(x = hour, y = predicted, color = "Predict
edData"))+
  geom_line(data = KSV_KO_2020, aes(x = hour, y = mean_temp.x, color = "Orlea
nsData"))+
  labs(x = "Year",
       y = "Temperature (C)")+
  theme_classic()+
  theme(text=element_text(size=12), legend.position = "bottom")+
  scale_colour_manual("", values = c("ExistingData"="steelblue", "PredictedDa
ta"="gold","OrleansData"="salmon")) +
  scale_y_continuous("Temperature (C)", limits = c(0,30), breaks = 5*0:30)

## Warning: Removed 418 row(s) containing missing values (geom_path).
```



30
25
20
15
10
5
0
Temperature (C)
2020-07
2021-01
2021-07
Year
ExistingData
PredictedData
OrleansData