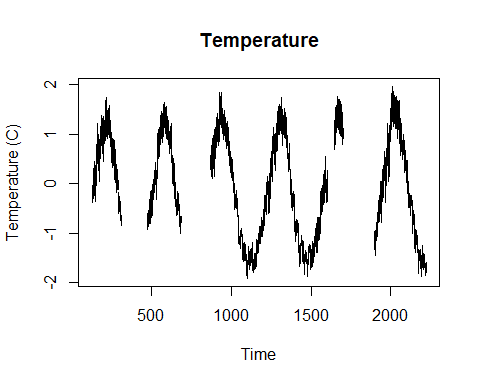
2021\_KlamathMain\_Linear

#Read in Data: Seiad Valley

##Reading in a 6 year 15 or 30 min temperature time series dataset from Klamath 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 removing 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 interpolate, let's bin by hour  
KSV\_hourly <- KSV %>% #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(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 missing 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"

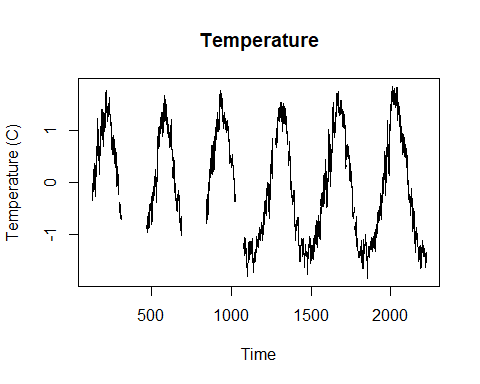
#Convert NaNs to NAs  
KSV\_hourly$mean\_temp[KSV\_hourly$mean\_temp == "NaN"] <- NA  
  
#Double check missing data  
missing\_data <- KSV\_hourly[!complete.cases(KSV\_hourly),]   
missing\_data #Now we are sure that all the missing hour time steps are included.  
  
#z score to control for outliers  
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 time 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 Klamath 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 Seiad 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 missing 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 included.  
  
#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 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(KO\_ts,main="Temperature",ylab = "Temperature (C)", xlab = "Time")

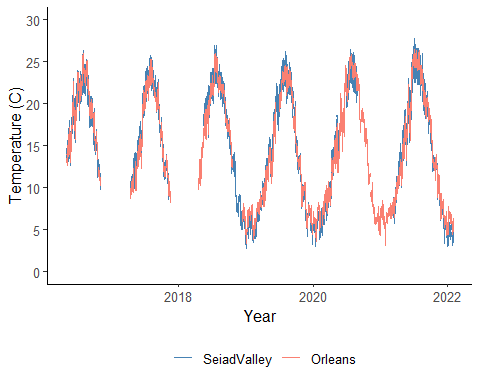


#Compare Orleans and Seiad Valley datasets

#Tightly correlated-- 0.99  
cor(KSV\_hourly$mean\_temp, KO\_hourly$mean\_temp, use = "na.or.complete", method = "pearson")

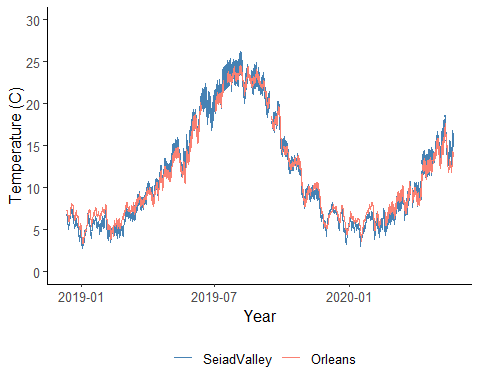
## [1] 0.9918316

#plot   
ggplot()+  
 geom\_line(data = KSV\_hourly, aes(x = hour, y = mean\_temp, color = "SeiadValley"))+  
 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"="salmon")) +  
 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 = "SeiadValley"))+  
 geom\_line(data = KO\_hourly\_cut, 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"="salmon")) +  
 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.exclude  
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), frequency = 24)#need it in ts format for predict function  
  
#Need to name the "newdata" variable that we are plugging into the predict function the SAME NAME as the variable we used in the fit\_lm linear equation above. 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 missing 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

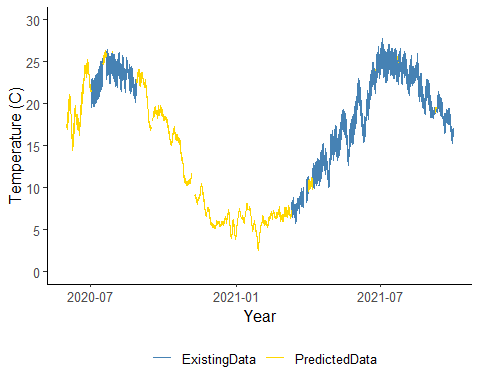
## 'data.frame': 11688 obs. of 3 variables:  
## $ hour : POSIXct, format: "2020-06-01 00:00:00" "2020-06-01 01:00:00" ...  
## $ mean\_temp: num 17.2 17.2 17.1 17 16.9 ...  
## $ zTemp : num 0.384 0.373 0.36 0.346 0.338 ...

#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 zTemp variable which is still empty for predicted values. This is just for graphing.

#Plot missing KSV values

ggplot()+  
 geom\_line(data = KSV\_KO\_2020, aes(x = hour, y = mean\_temp.y, color = "ExistingData"))+  
 geom\_line(data = KSV\_KO\_2020, aes(x = hour, y = predicted, color = "PredictedData"))+  
 labs(x = "Year",  
 y = "Temperature (C)")+  
 theme\_classic()+  
 theme(text=element\_text(size=12), legend.position = "bottom")+  
 scale\_colour\_manual("", values = c("ExistingData"="steelblue", "PredictedData"="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 = "ExistingData"))+  
 geom\_line(data = KSV\_KO\_2020, aes(x = hour, y = predicted, color = "PredictedData"))+  
 geom\_line(data = KSV\_KO\_2020, aes(x = hour, y = mean\_temp.x, color = "OrleansData"))+  
 labs(x = "Year",  
 y = "Temperature (C)")+  
 theme\_classic()+  
 theme(text=element\_text(size=12), legend.position = "bottom")+  
 scale\_colour\_manual("", values = c("ExistingData"="steelblue", "PredictedData"="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).

