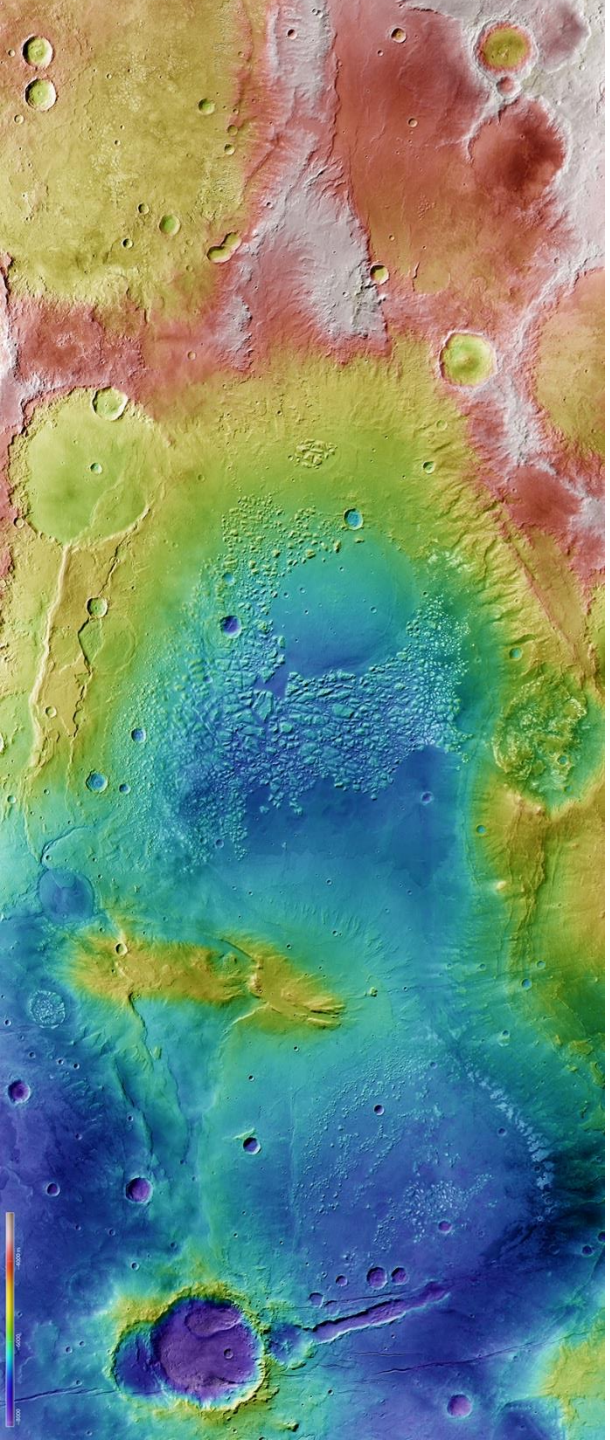
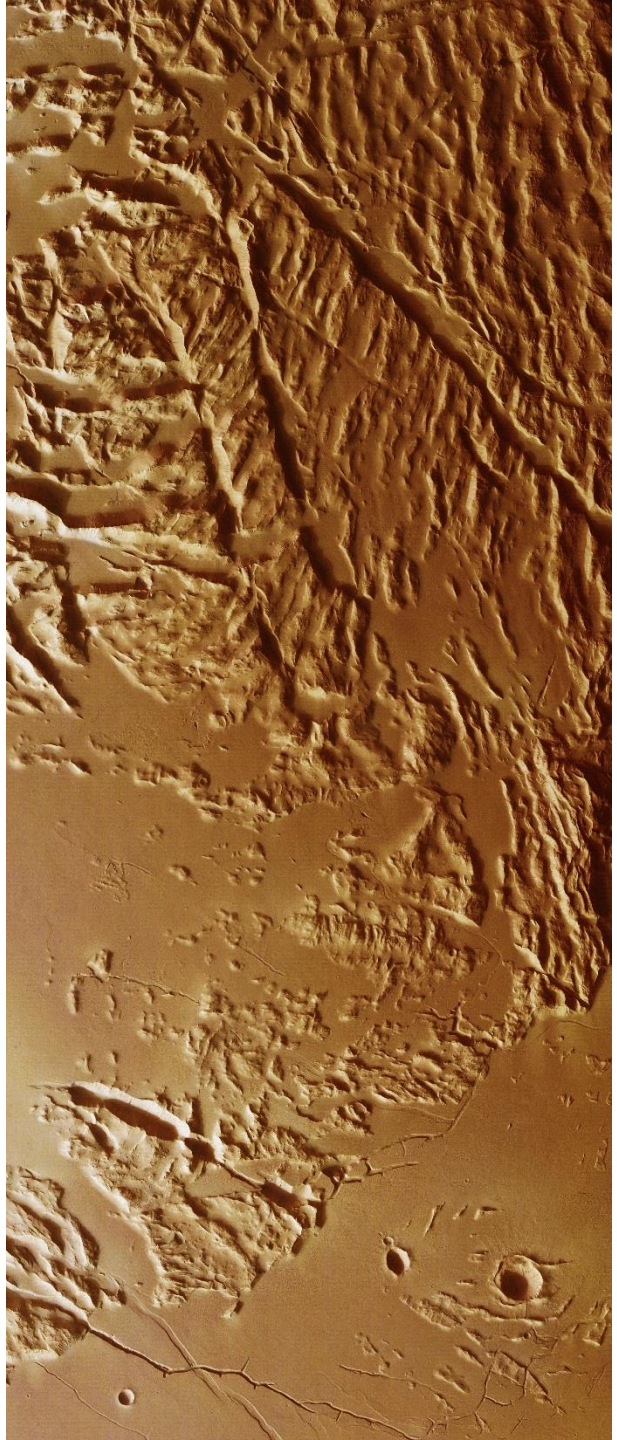
Mars Express Power Challenge

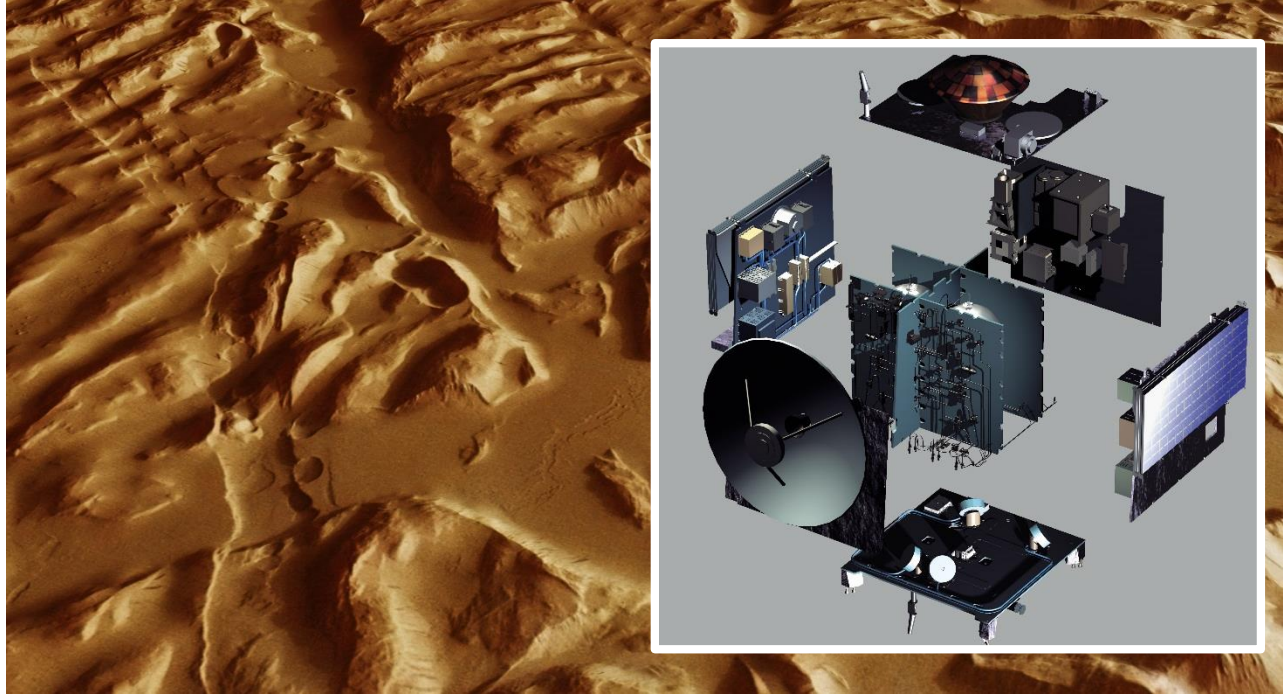
CC71Q - Introducción a la Minería de Datos

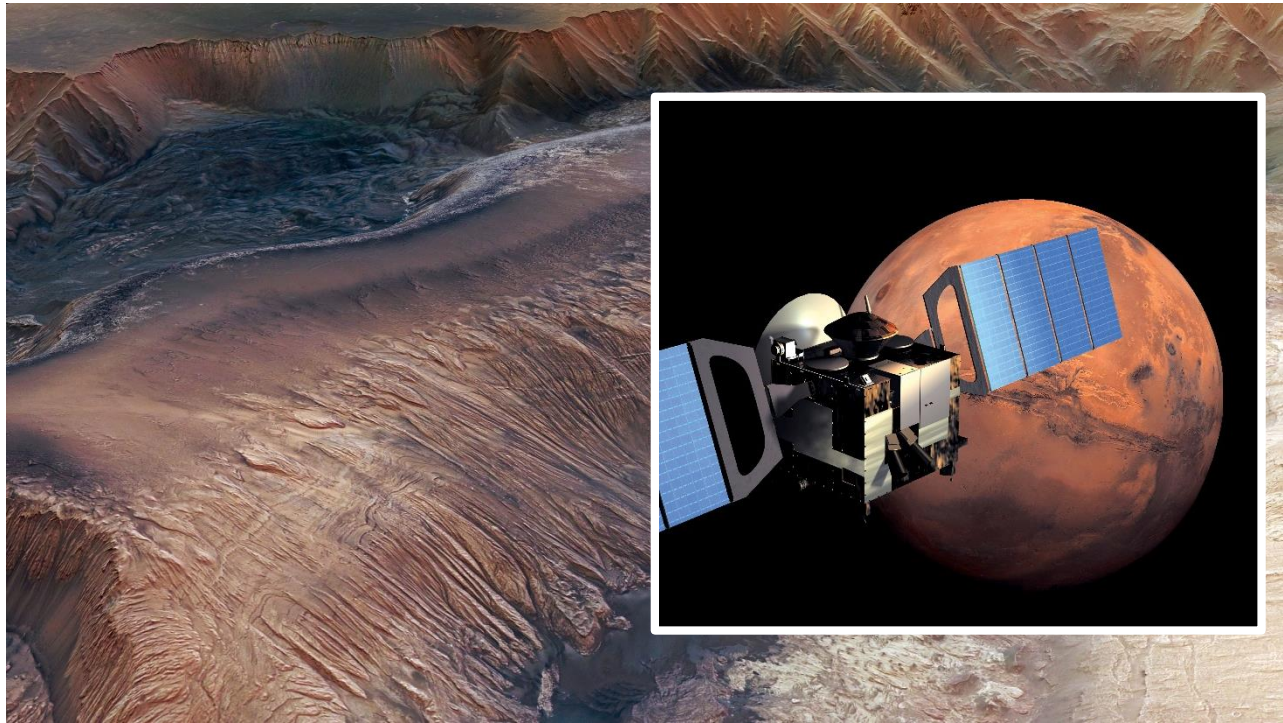
Gabriel De La Parra

30.Mayo.2016





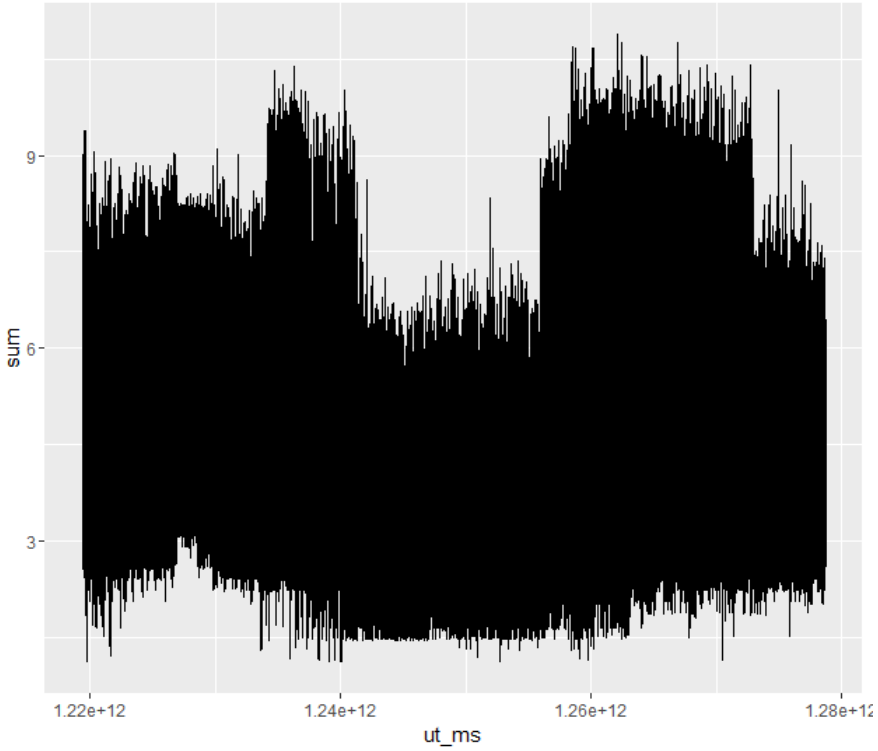




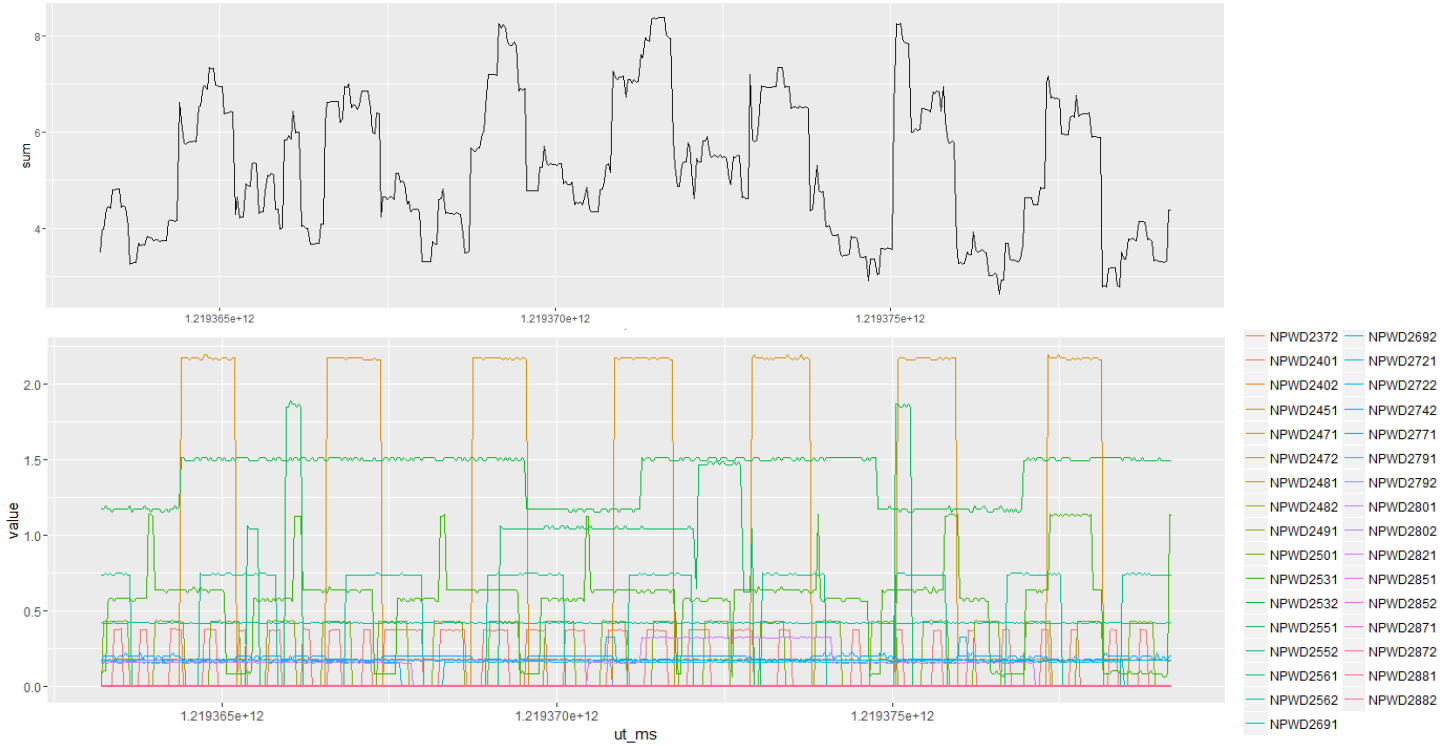
Mars Express Power Challenge

Problema: Predicción del consumo energético del satélite

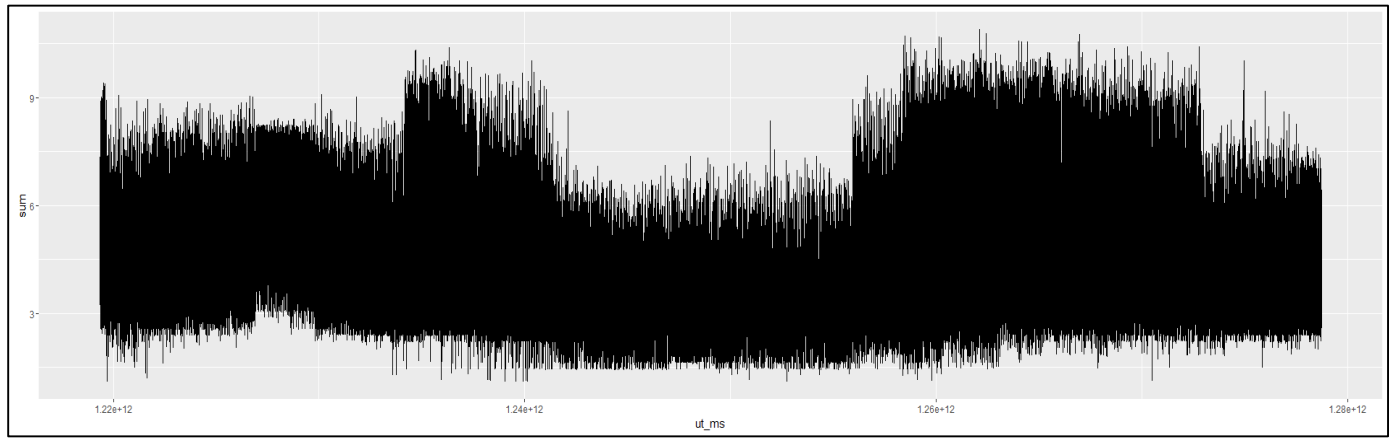
$E_{Ciencia} = E_{Solar} - E_{Nave} - E_{Climatización}$

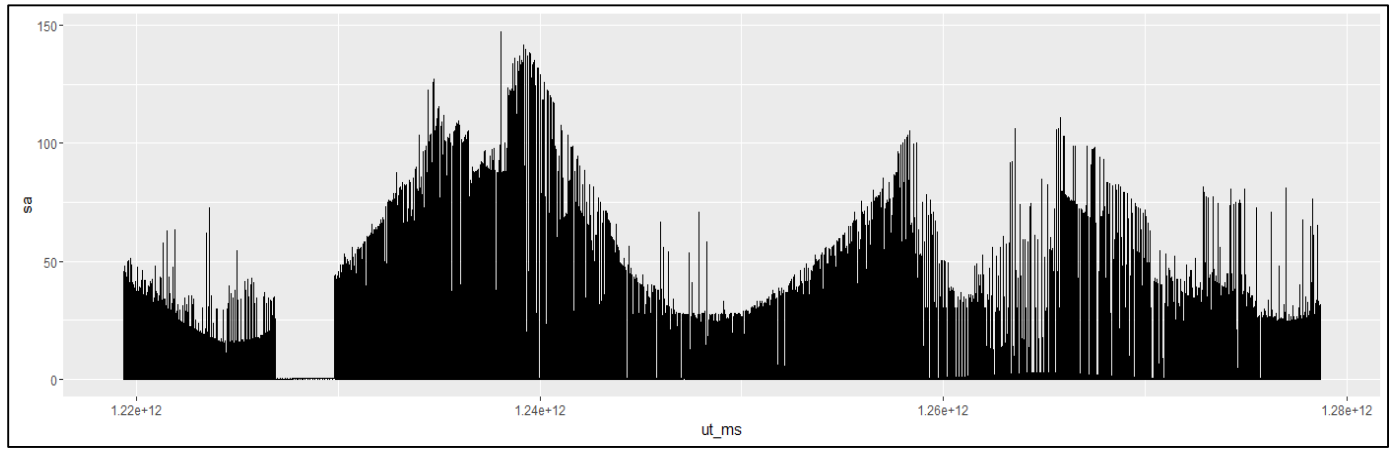


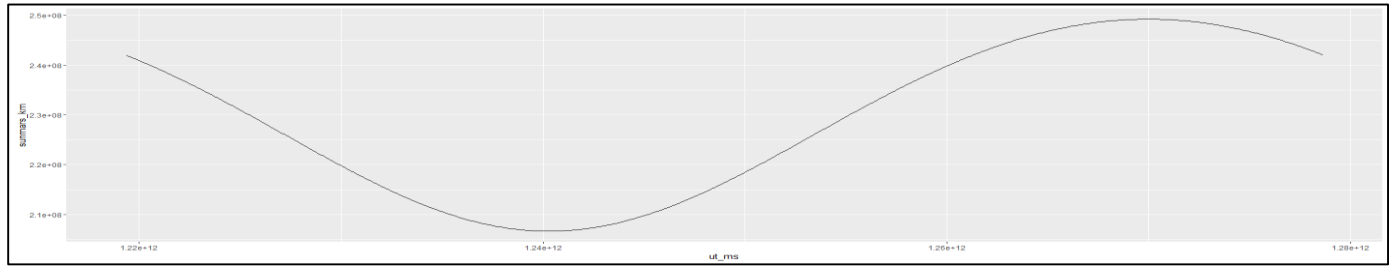
Consumo: 1 año, Suma circuitos



Consumo [1:500]







Pre-procesamiento

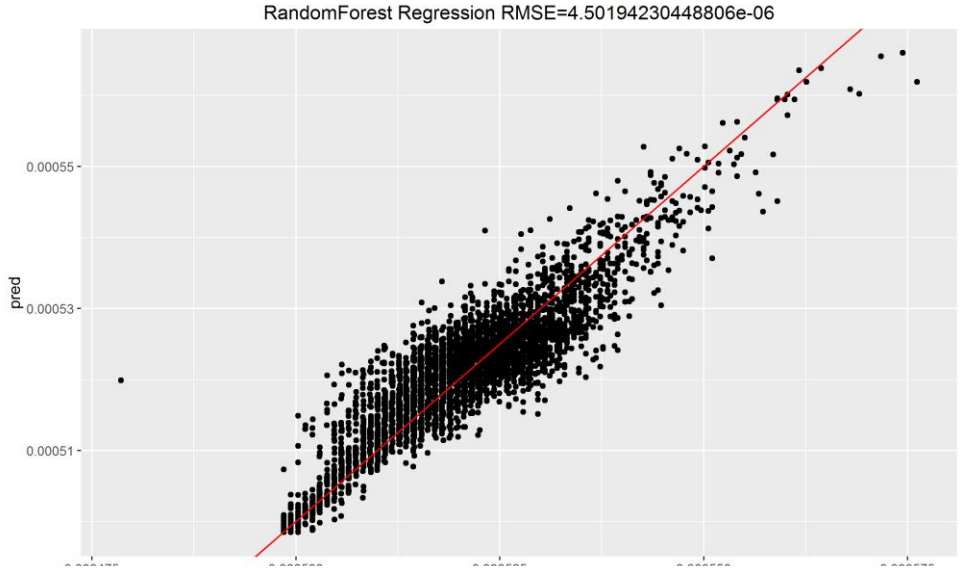
- Ajuste: Escala temporal
- Agrupamiento: Promedio por hora
- Match: Time Scale
- Join *
- Interpolación de valores faltantes
- Split: Train, Test

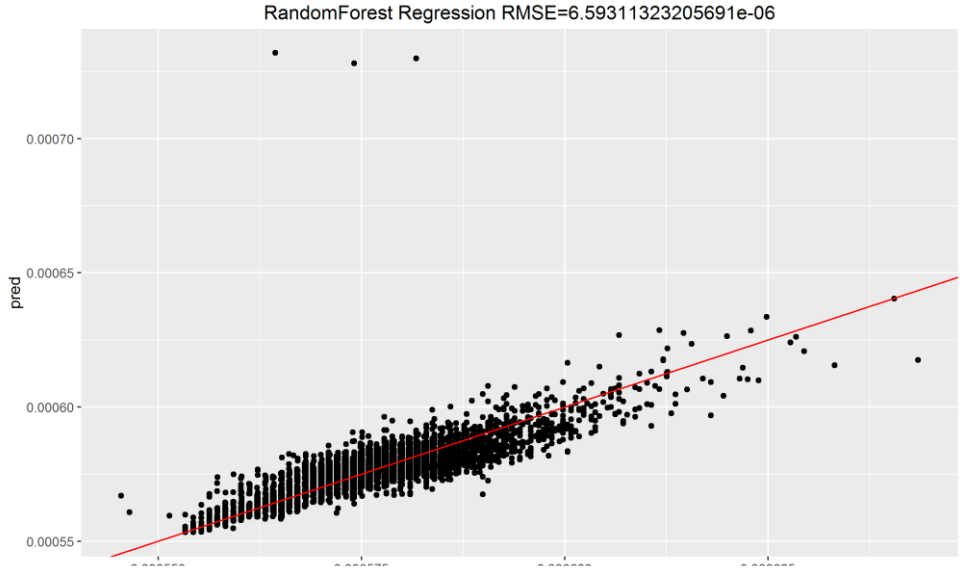
Procesamiento

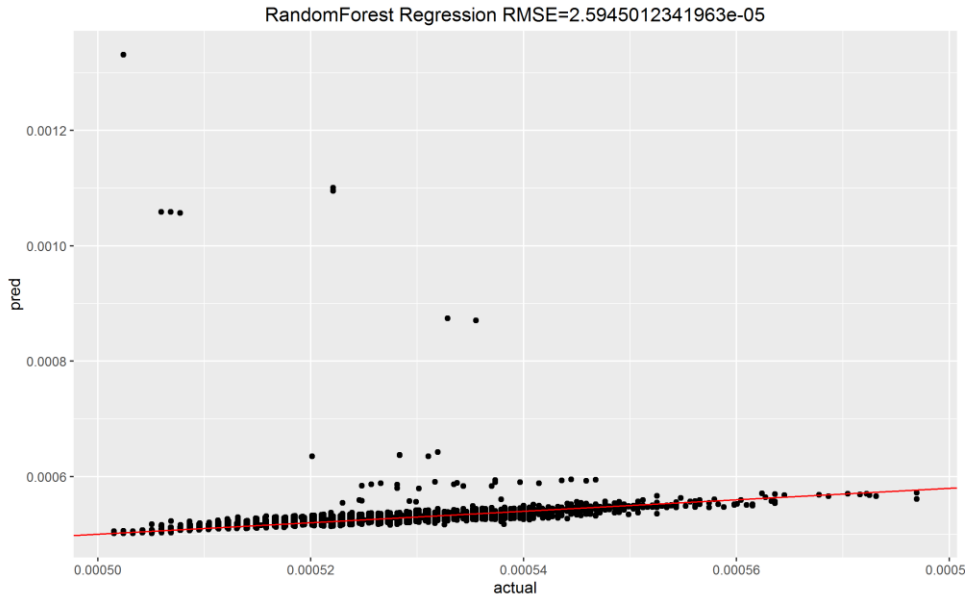
- Regresión: Random Forest
- Evaluación: Root Mean Square

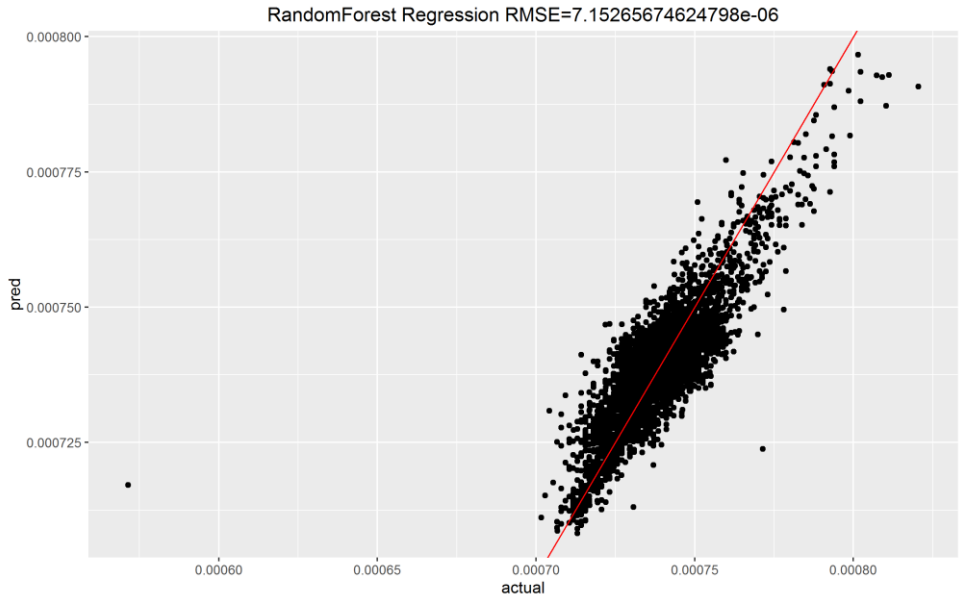
Mars Express Power Challenge

Resultados positivos



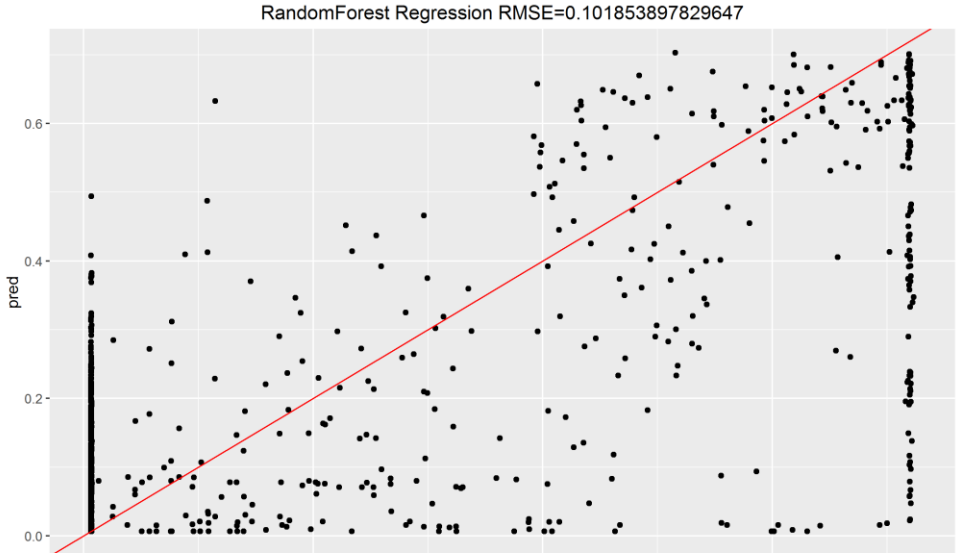


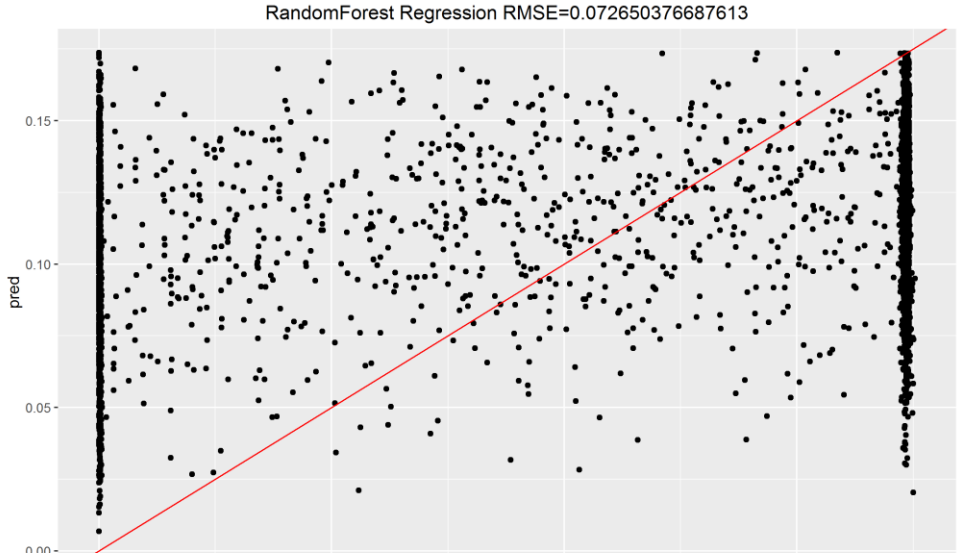


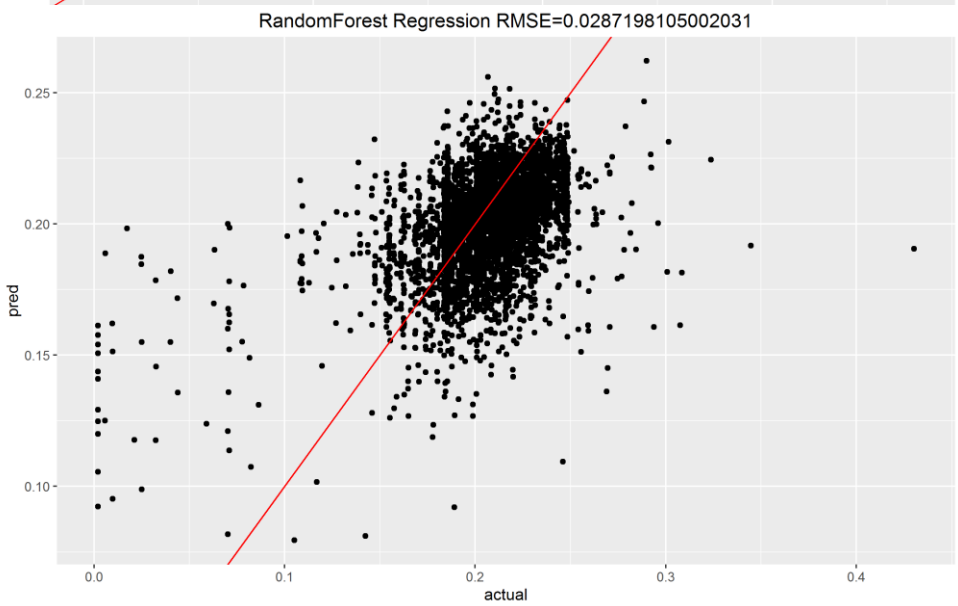


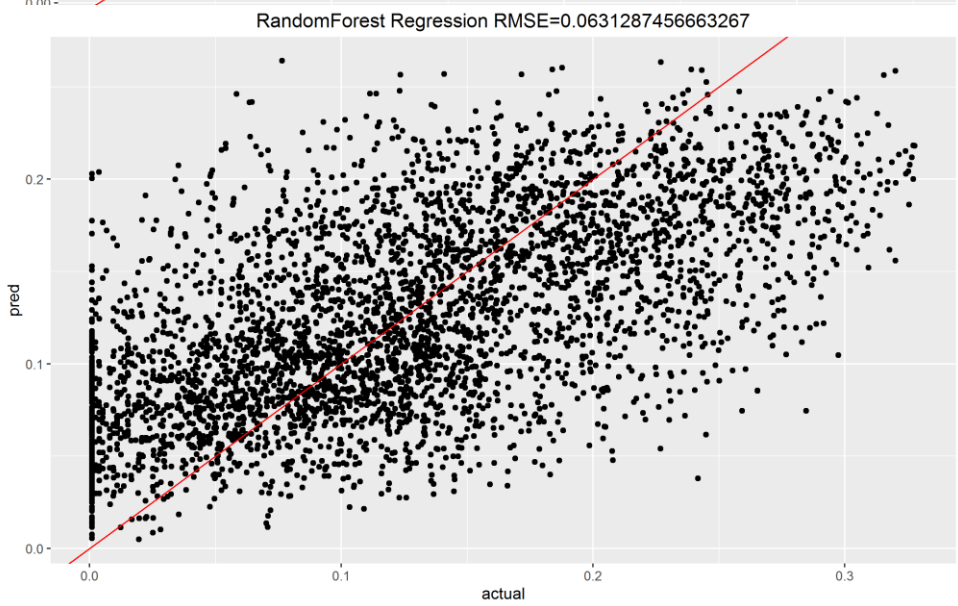
Mars Express Power Challenge

Resultados negativos





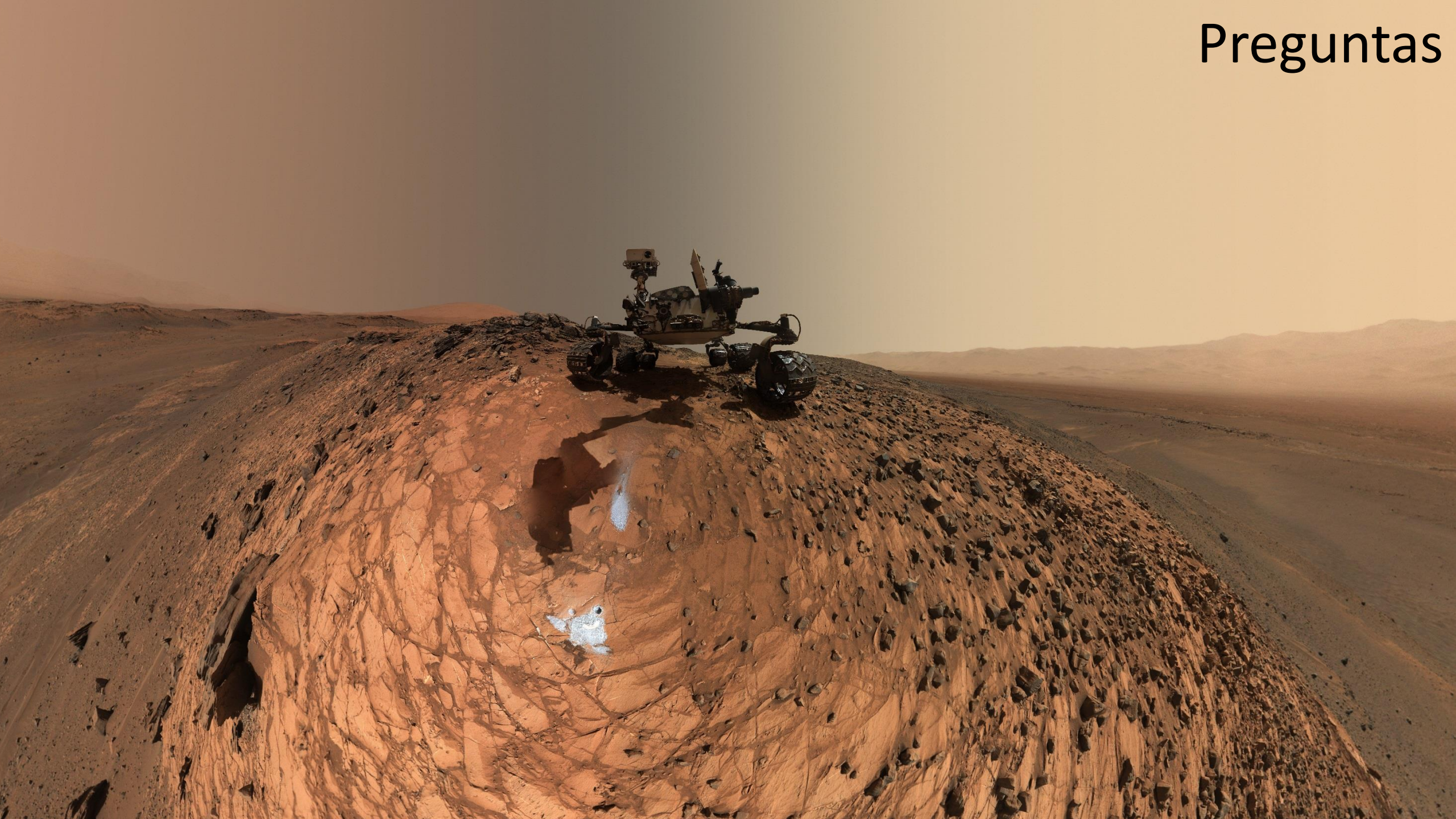




Mars Express Power Challenge
Conclusiones

- Error total: 0.04979785 (5%)
- (Pre)Procesamiento = +Tiempo
- Recursos computacionales
- Generación ~ Consumo
- Predicción macro: Ok
- Predicción micro: Falta procesamiento: DMOP, EVTF, FTL

Preguntas



```r
#Escala Tiempo:
power1DT <- power1
power1DT$ut_ms <- as.POSIXct((((power1['ut_ms'])/1000)[,]), origin="1970-01-01")

#Agrupamiento: Promedio por hora:
power1DT$ut_ms <- cut(power1DT$ut_ms, breaks="hour")
power1DTHourMean <- power1DT %>% group_by(ut_ms) %>% summarise_each(funs(mean))

#Match: TimeScale
power1DTHourMeanMS <- power1DTHourMean$ut_ms
for (i in 1:nrow(saaf1DTHourMean)) {
  saaf1DTHourMean$ut_ms[i] <- power1DTHourMeanMS[findInterval(saaf1DTHourMean$ut_ms[i],power1DTHourMeanMS)]
}

#Merge:
power1DTHourMean<-merge(x=power1DTHourMean, y=saaf1DTHourMean, by="ut_ms", all.x=TRUE)
power1DTHourMean<-merge(x=power1DTHourMean, y=ltdata1DTHourMean, by="ut_ms", all.x=TRUE)

#Interpolacion:
power1DTHourMean$sunmars_km <- na.spline(power1DTHourMean[,grep("sunmars_km",
colnames(power1DTHourMean))],na.rm = FALSE)
power1DTHourMean$earthmars_km <- na.spline(power1DTHourMean[,grep("earthmars_km",
colnames(power1DTHourMean))],na.rm = FALSE)
```

```r
#Regresión
rmseSum <- 0
for(i in 1:33){
  predictField <- i #Campo a predecir
  predictCols <- colnames(power1DT[,-1]) #Columnas en juego:
  #Set de entrenamiento y pruebas
  train <- power1DTHourMean[1:12000,-1]
  test <- power1DTHourMean[12001:16000,-1]
  colName <- predictCols[predictField]
  #Entrenamiento:
  rf <- randomForest(as.formula(paste(colName," ~ .")) ,data=train, ntree=10)
  #Prueba:
  predicted <- predict(rf, test)
  predCol <- test[,c(colName)]
  #Medición error:
  r2 <- RMSE(predCol, predicted)
    #Graficar Predicción vs. Referencia:
  p <- ggplot(aes(x=actual, y=pred), data=data.frame(actual=predCol, pred=predict(rf, test)))
  p <- p + geom_point() +  geom_abline(color="red") + ggtitle(paste("RandomForest Regression RMSE=", r2, sep=""))
  rmseSum <- rmseSum + r2 #Acumulación Error
  ggsave(paste("Predict",i,".png"), p) #Guardar Imagen
}
errorTotal<-rmseSum/33
```