data_challenge

Don Johnson

6/26/2018

Intro

The devices report data from 5 measurements: rpm, voltage, current, motor temperature, and inlet temperature.

Noise: There are many outliers due to noise. Some of these are bunched.

Data is written at irregular intervals. The time difference between data points is a normal distribution with mean 10 and deviance 2. The shortest intervals are about 3.5 minutes and the longest intervals are about 16.5 minutes.

Time to failure	hours	days
min	238.1	10
max	26663.7	1111
avg	8251.2	344

Data Clean-up

Steps for Data Clean-up (found in noiseclean.R): * Convert timestamp to POSIX time object * Determine the normal range of the measurement using the 1st and 99th quantile * Increase range with a 30% buffer on each side to capture rare spikes * Replace outliers with a rolling median * New variable: "TTF" Time to Failure in hours * Remove Seasonal component to data. - create sin and cos with period of 365 days - fit linear model to a measurement with sin/cos - take residuals of measurement (repeat for all 5) * new Boolean variables: will it fail in 3 hours/days/weeks?

(as a next step, express time to failure as an exponential such as, expTTF = exp(-TTF/72), and choose hours (e.g., 72) to obtain best fit)

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
```

```
##     date
```

```
setwd("/Users/dojo/data/tagup/data-science-challenge/data/train/")

source("./noiseclean.R")
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

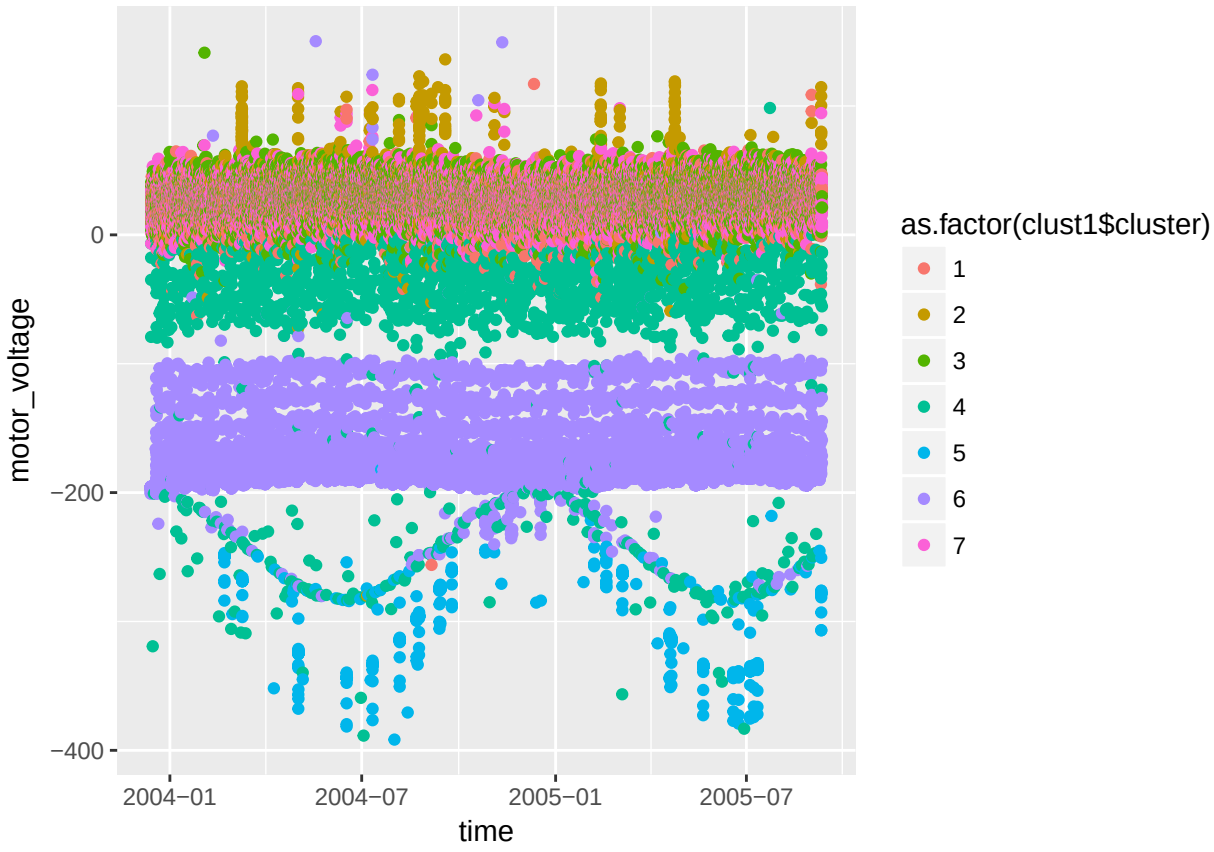
```
myfiles <- grep("_rms", dir(), value = TRUE)
unitDF <- lapply(myfiles, clean_noise)

ttf <- lapply(unitDF, function(x) head(x$TTF, 1))
ttf <- unlist(ttf)
summary(ttf)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   238.1  1697.2  6910.9  8251.2 12464.3 26663.7
```

Identify states via clustering

I will cluster one unit's data and use those centers to create clusters for the rest of the data. I'm not sure if this is a reliable way to obtain consistent clustering

```
measurements <- c("rpm", "motor_voltage", "motor_current",
               "motor_temp", "inlet_temp")
clust1 <- kmeans(unitDF[[1]][,measurements],7,
                 algorithm = "Lloyd", iter.max = 100)  # 7 states?
centers <- clust1$centers
ggplot(data=unitDF[[1]], aes(x=time, y=motor_voltage,
         color=as.factor(clust1$cluster))) + geom_point()
```



Looks okay. Repeat clustering for all datasets.

```
unitDF <- lapply(unitDF, function(x) {
    x$cluster <- as.factor(kmeans(x[,measurements],
                                  centers = centers, algorithm = "Lloyd",
                                  iter.max = 100)$cluster)
    return(x)
    })
```

```
## Warning: empty cluster: try a better set of initial centers

## Warning: empty cluster: try a better set of initial centers

## Warning: empty cluster: try a better set of initial centers

## Warning: empty cluster: try a better set of initial centers
```

Next steps: * find ‘states’ associated with end of device lifetime * tokenize state-chain to find state transitions associated with end of lifetime