2016 OSIsoft TechCon

Use Data Science
for Machine Learning
and Predictions Based
on your PI System Data

OSIsoft, LLC
777 Davis St., Suite 250
San Leandro, CA 94577 USA
Tel: (01) 510-297-5800
Web: http://www.osisoft.com

Published: April 11, 2016

Use Data Science for Machine Learning and Predictions based on PI System Data
Hands-on Lab – OSIsoft TechCon 2016

Lead: Gopal GopalKrishnan, P.E., Solution Architect

Lead: Curt Hertler, Solution Architect

Table of Contents

Learning Objectives, Problem Statement and Dataset

In this Lab, we will walk through a statistical approach to predict equipment failure. We will use sensor data taken from 100 engines prior to engine failure. We will use the R scripting language to develop a predictive modelling equation that can be run in real time to predict engine failure before it actually happens.

In a deployment with about 100 engines which are similar, sensor data such as rpm, burner fuel/air ratio, pressure at fan inlet, and twenty other measurements plus settings for each engine – for a total of about 2000 tags – are available. On average, an engine fails after 206 cycles, but it varies widely - from about 130 to 360 cycles.

Using an open source tool such as R for machine learning, you will create a multivariate model to predict engine failures within approximately a 15 cycle window before they fail. The lab will walk through the end-to-end data science process – preparing the dataset, visually exploring it, partitioning the data for training and testing, validating the models using previously unseen data, and finally deploying the model with AF asset analytics for predictive maintenance.

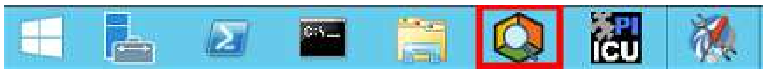
The lab consists of three parts:

- **Part I** - Extract data from the PI System for a set of Engines' operation parameters into a text file using PI Integrator for Business Analytics (BA)
- **Part II** - Load the exported dataset from PI System into R, and utilize the functionality in R to perform machine learning on the dataset for prediction of engine failures.
- **Part III** - Convert the output prediction logic from R into an equivalent equation in PI Analytics for continuous execution and prediction.

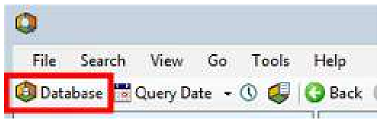
The PI System Infrastructure and the Dataset

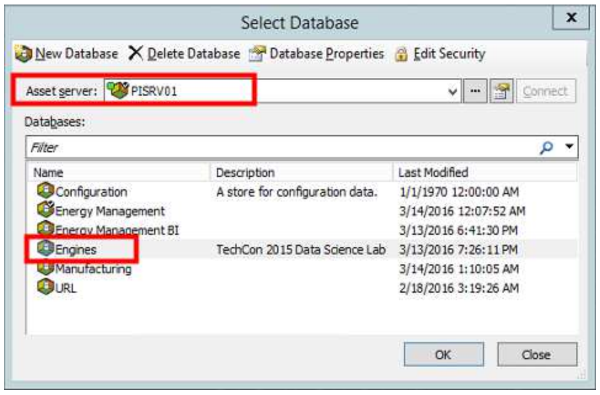
Let's start by looking at the AF model that represents the set of engines and take a look at the dataset for these set of engines.

Open PI System Explorer from the Taskbar.



In PI System Explorer, ensure that you are connected to the AF Server PISRV01 and looking at **Engines** database. Select "Database" from the top toolbar and select "Engines" from the ***Select Database*** dialog.





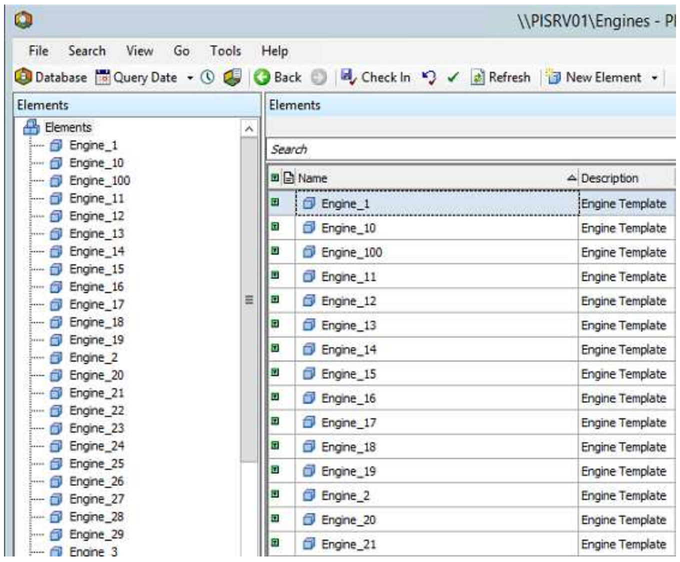
Asset server: PISRV01

Databases:

Filter

Name	Description	Last Modified
Configuration	A store for configuration data.	1/1/1970 12:00:00 AM
Energy Management		3/14/2016 12:07:52 AM
Energy Management BI		3/13/2016 6:41:30 PM
Engines	TechCon 2015 Data Science Lab	3/13/2016 7:26:11 PM
Manufacturing		3/14/2016 1:10:05 AM
URL		2/18/2016 3:19:26 AM

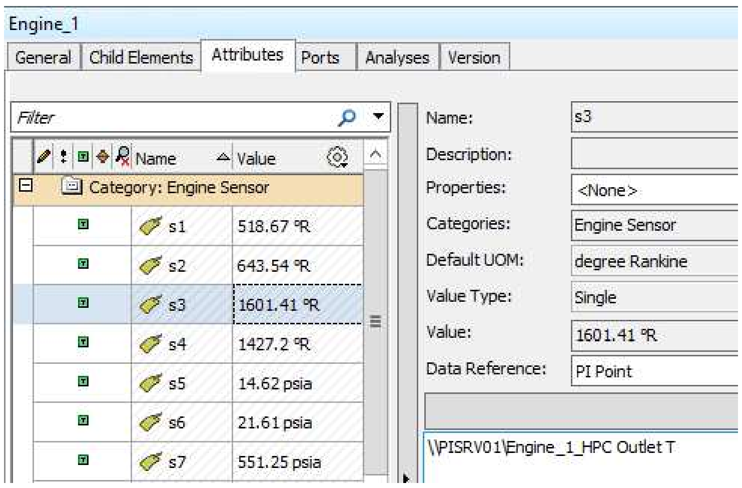
The AF Database is quite simple, you should see a list of 100 elements at the root level of the database, each representing an engine:



Elements
- Engine_1
- Engine_10
- Engine_100
- Engine_11
- Engine_12
- Engine_13
- Engine_14
- Engine_15
- Engine_16
- Engine_17
- Engine_18
- Engine_19
- Engine_2
- Engine_20
- Engine_21
- Engine_22
- Engine_23
- Engine_24
- Engine_25
- Engine_26
- Engine_27
- Engine_28
- Engine_29
- Engine_3

Name	Description
Engine_1	Engine Template
Engine_10	Engine Template
Engine_100	Engine Template
Engine_11	Engine Template
Engine_12	Engine Template
Engine_13	Engine Template
Engine_14	Engine Template
Engine_15	Engine Template
Engine_16	Engine Template
Engine_17	Engine Template
Engine_18	Engine Template
Engine_19	Engine Template
Engine_2	Engine Template
Engine_20	Engine Template
Engine_21	Engine Template

Select an Engine element and look under the ***Attributes*** tab to see the set of attributes defined for the 21 sensors and 3 settings. In this example, we have kept the attribute names short, i.e. “s3” for “sensor 3”. This makes the R script easier to manage. As you can see, these sensors map to the temperatures, pressures, rotation speeds, etc. measured for each engine.



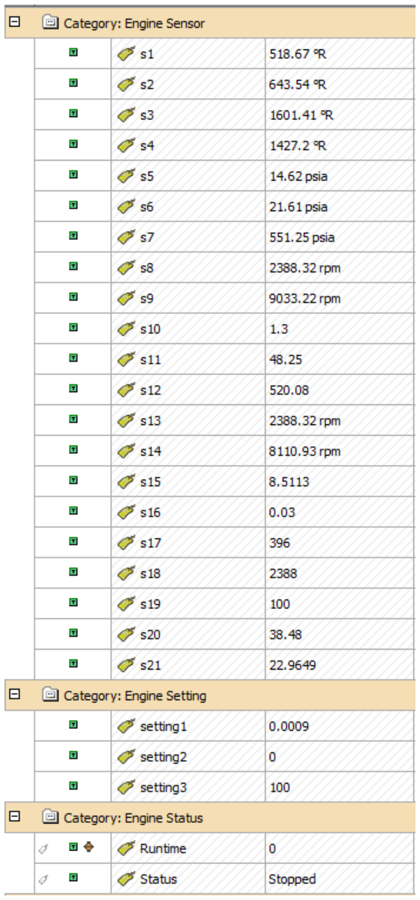
Engine_1

General | Child Elements | Attributes | Ports | Analyses | Version

Name	Value
Category: Engine Sensor	
s1	518.67 °R
s2	643.54 °R
s3	1601.41 °R
s4	1427.2 °R
s5	14.62 psia
s6	21.61 psia
s7	551.25 psia

Name:	s3
Description:	
Properties:	<None>
Categories:	Engine Sensor
Default UOM:	degree Rankine
Value Type:	Single
Value:	1601.41 °R
Data Reference:	PI Point

\\PISRV01\Engine_1_HPC Outlet T

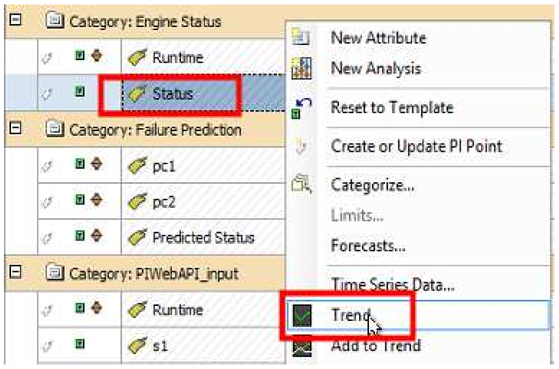


Category: Engine Sensor	
s1	518.67 °R
s2	643.54 °R
s3	1601.41 °R
s4	1427.2 °R
s5	14.62 psia
s6	21.61 psia
s7	551.25 psia
s8	2388.32 rpm
s9	9033.22 rpm
s10	1.3
s11	48.25
s12	520.08
s13	2388.32 rpm
s14	8110.93 rpm
s15	8.5113
s16	0.03
s17	396
s18	2388
s19	100
s20	38.48
s21	22.9649
Category: Engine Setting	
setting1	0.0009
setting2	0
setting3	100
Category: Engine Status	
Runtime	0
Status	Stopped

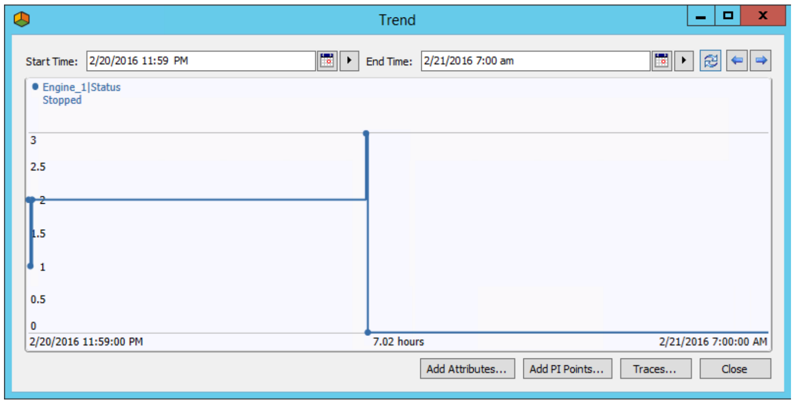
The **Runtime** attribute totalizes the minutes each engine has been running. The **Status** attribute indicates the operating status of the engine - Started, Running, Failed, and Stopped.

The attributes shown under the “Failure Prediction” category will be used in Part III of this lab. They will contain the results of the predictive model we will implement.

Now, let's look at the dataset we will be using for Part I. Right-click on the **Status** attribute and **Trend** the attribute



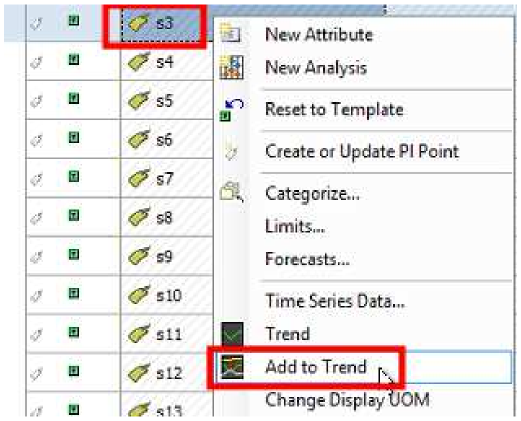
Change the trend time range to start from **20-Feb-2016 11:59:00 pm** to **21-Feb-2016 07:00 am** and click the Refresh button. Although the runtime for each engine is different, this time range is long enough to examine every dataset.



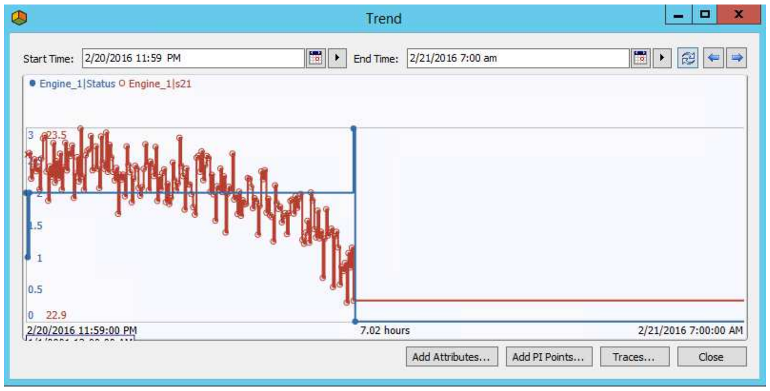
We have backfilled each engine's dataset starting at 21-Feb-2016 12:00:00 am.

We will be extracting this data for all 100 engines in the next section using the **PI Integrator for Business Analytics**.

For Engine_1, we see the engine's operating states for the period, the spike at 21-Feb-2016 3:11 am indicates when the engine failed. To overlay other sensor readings onto the same trend, right-click on one of the sensor attributes and choose **Add to Trend**.



Feel free to explore the AF model and the data further. Next step will be to prepare and export the data for all 100 engines to a text file.



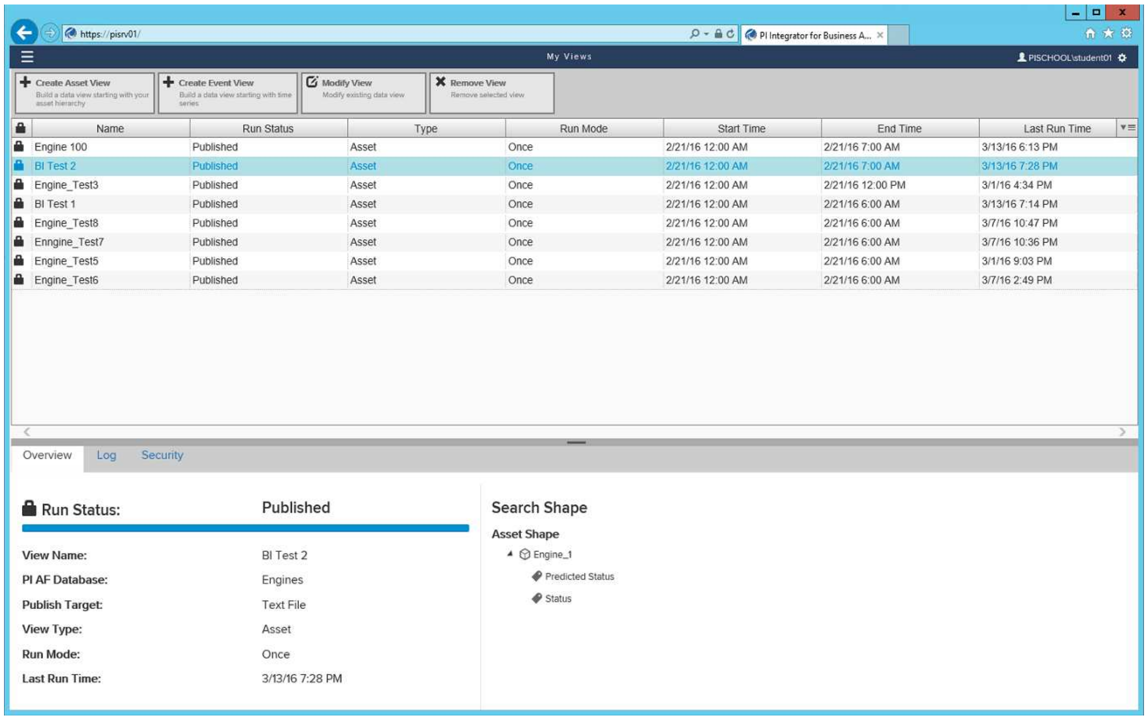
Part I – Extracting PI System Data using the PI Integrator for Business Analytics

The PI Integrator for BA is a tool developed by OSIsoft to support the integration of real-time operations data with various environments including:

- Self-service BI Analytic tools like Microsoft Power BI, Tableau or Tibco Spotfire
- Data Warehouse databases like Teradata, Oracle or Microsoft SQL Server
- Big Data platforms like SAP HANA and Hadoop
- Statistical analytics tools like SAS and R

In this lab, we will be working with R for predictive analytics. Hence we will use the PI Integrator for BA to organize the dataset and publish data into a text file format that can be consumed by R.

Access the PI Integrator for BA by opening Internet Explorer. The default web site for the PI Integrator is https://pisrv01/ . You should see the main page of the integrator, showing a list of previously configured views:

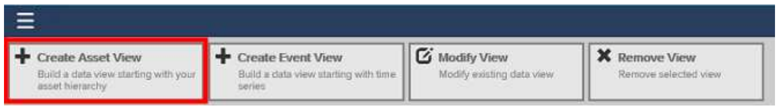

https://pisrv01/
PI Integrator for Business A...
My Views
PISCHOOL\student01
Create Asset View
Build a data view starting with your asset hierarchy
Create Event View
Build a data view starting with time series
Modify View
Modify existing data view
Remove View
Remove selected view
Overview
Log
Security

Name	Run Status	Type	Run Mode	Start Time	End Time	Last Run Time
Engine 100	Published	Asset	Once	2/21/16 12:00 AM	2/21/16 7:00 AM	3/13/16 6:13 PM
BI Test 2	Published	Asset	Once	2/21/16 12:00 AM	2/21/16 7:00 AM	3/13/16 7:28 PM
Engine_Test3	Published	Asset	Once	2/21/16 12:00 AM	2/21/16 12:00 PM	3/1/16 4:34 PM
BI Test 1	Published	Asset	Once	2/21/16 12:00 AM	2/21/16 6:00 AM	3/13/16 7:14 PM
Engine_Test8	Published	Asset	Once	2/21/16 12:00 AM	2/21/16 6:00 AM	3/7/16 10:47 PM
Enngine_Test7	Published	Asset	Once	2/21/16 12:00 AM	2/21/16 6:00 AM	3/7/16 10:36 PM
Engine_Test5	Published	Asset	Once	2/21/16 12:00 AM	2/21/16 6:00 AM	3/1/16 9:03 PM
Engine_Test6	Published	Asset	Once	2/21/16 12:00 AM	2/21/16 6:00 AM	3/7/16 2:49 PM

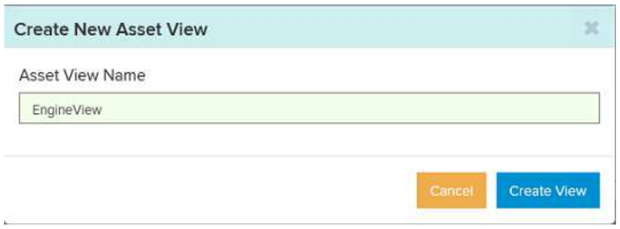
Run Status:	Published
View Name:	BI Test 2
PI AF Database:	Engines
Publish Target:	Text File
View Type:	Asset
Run Mode:	Once
Last Run Time:	3/13/16 7:28 PM

Search Shape
Asset Shape
- Engine_1
  - Predicted Status
  - Status

To get started, select “Create Asset View” from the PI Integrator’s top menu.

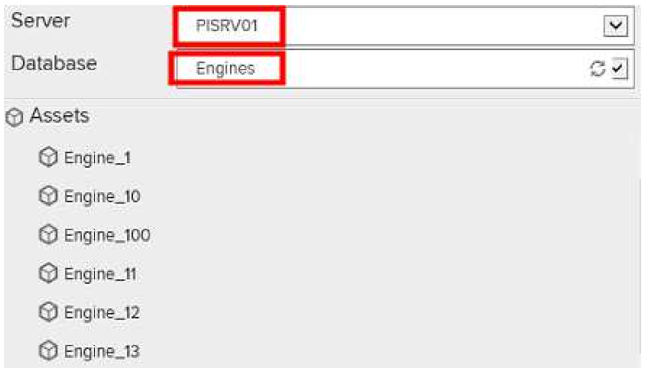

Create Asset View
Build a data view starting with your asset hierarchy
Create Event View
Build a data view starting with time series
Modify View
Modify existing data view
Remove View
Remove selected view

Next, you will be prompted to give your Asset View a name. Enter a name and click “Create View”.


Create New Asset View

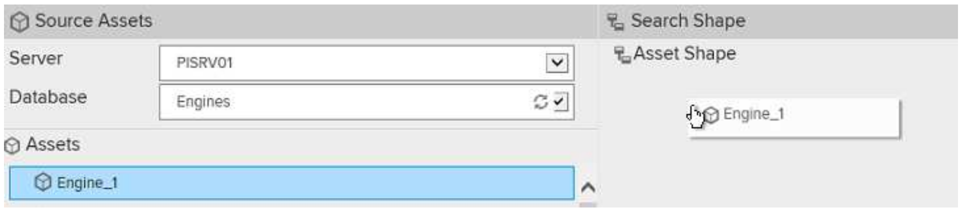
Asset View Name: EngineView

The first step in using the PI Integrator for BA is to select the data from an AF model. Select the AF Server “PISRV01” and the AF Database “Engines”. You should see the engine elements appear in the Assets hierarchy below.


Server: PISRV01

Database: Engines

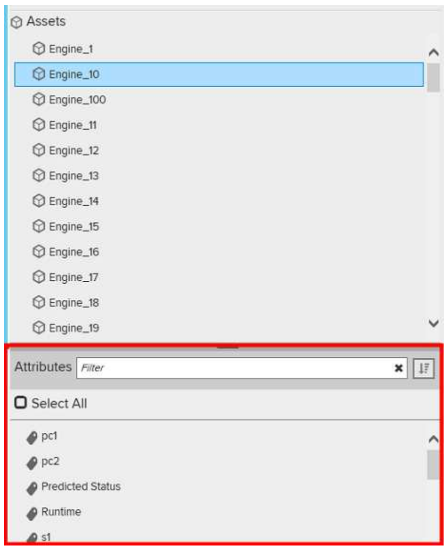
Select and drag one of the engine elements into the middle section under “Asset Shape”.



Server: PISRV01

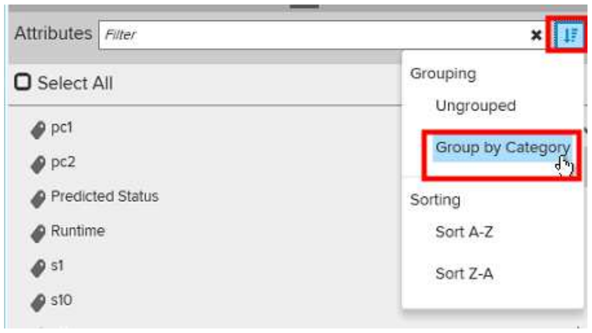
Database: Engines

When elements are selected, the pane below shows the list of the element’s attributes. These can be selected ad dragged in to the “Asset Shape” area either individually or as groups.



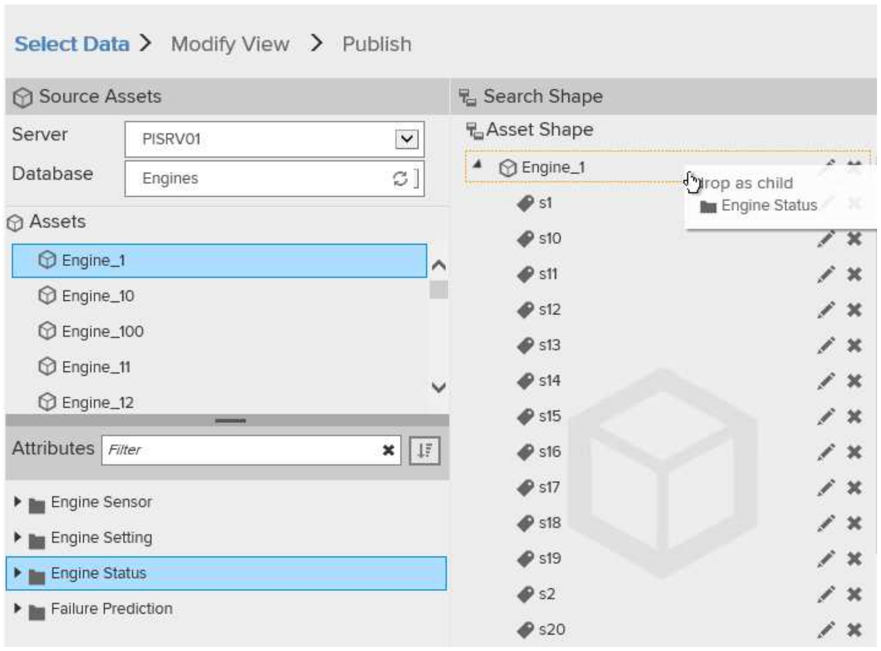
Next, we want to select all the sensor attributes as well as the settings, runtime and status attributes and include them as part of the view.

We can drag and drop all the attributes, but to simplify the process, let's group the attributes by category.



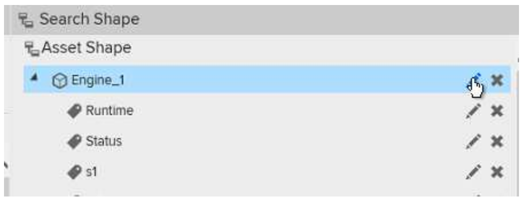
Select and drag the "Engine Sensor" category into the "Asset Shape" area and drop it **as a child** to the "Engine_1" element. To do this you must position your cursor directly over the engine element shown in the "Asset Shape" region.

Repeat this for the "Engine Setting", and "Engine Status" categories.

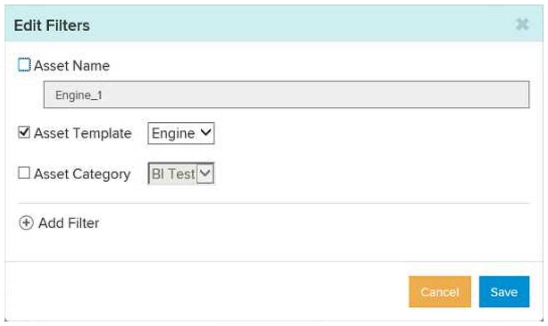


After selecting the required attributes, we need to modify the asset shape to include all other engines that are based on the same template.

Click on the edit (pencil icon) on the right of the selected engine. This brings up a dialog to edit the asset selection filter shown below.

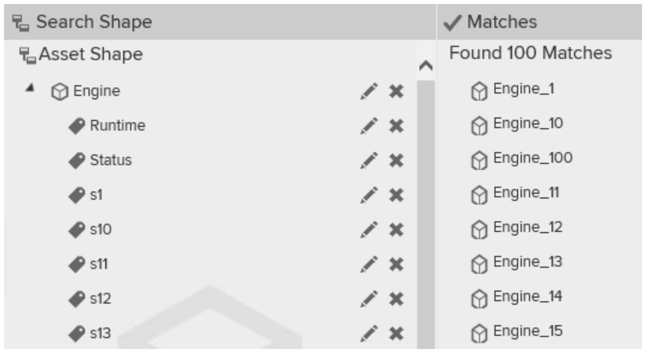


Uncheck the filter on “Asset Name” and check the filter on “Asset Template”. This will expand our selection to include all 100 engines which have been derived from the “Engine” template in AF database.

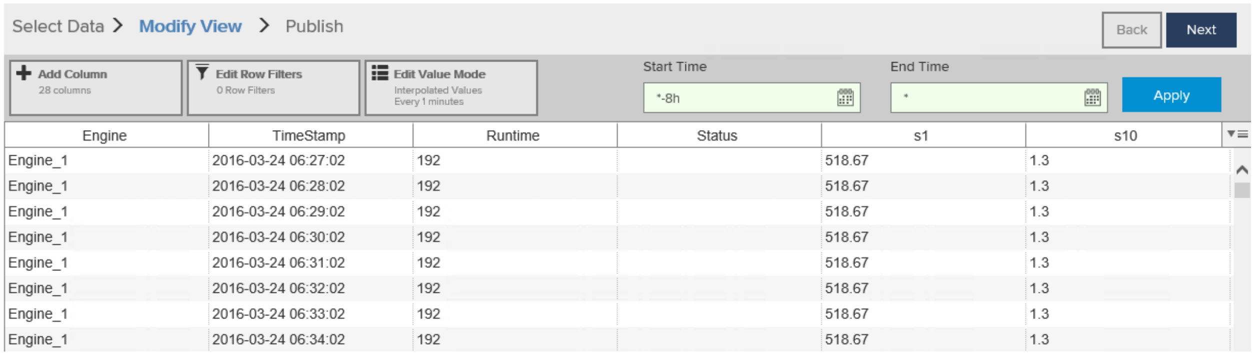


Click on **Save** button to confirm the changes. And you should see that 100 elements derived from the engines template have been found.

Click **Next** at the top right-hand side of the page to move to the “Modify View” page (shown below).

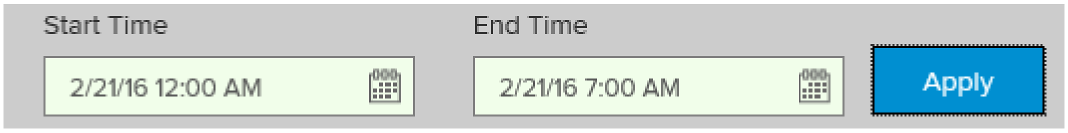


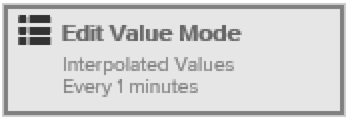
In the Modify View step, you specify the shape of the dataset to extract. Shaping involves configuring the time range and interval, aggregations, and filters you want applied when publishing the dataset.



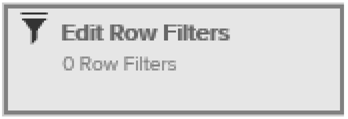
Engine	TimeStamp	Runtime	Status	s1	s10
Engine_1	2016-03-24 06:27:02	192		518.67	1.3
Engine_1	2016-03-24 06:28:02	192		518.67	1.3
Engine_1	2016-03-24 06:29:02	192		518.67	1.3
Engine_1	2016-03-24 06:30:02	192		518.67	1.3
Engine_1	2016-03-24 06:31:02	192		518.67	1.3
Engine_1	2016-03-24 06:32:02	192		518.67	1.3
Engine_1	2016-03-24 06:33:02	192		518.67	1.3
Engine_1	2016-03-24 06:34:02	192		518.67	1.3

As mentioned in previous section, the running data for the engines are backfilled for the time period between 21-Feb-2016 12:00:00 am to 21-Feb-2016 07:00:00 am. Hence let’s change the start and end time of the view and click on **Apply** to update the view.



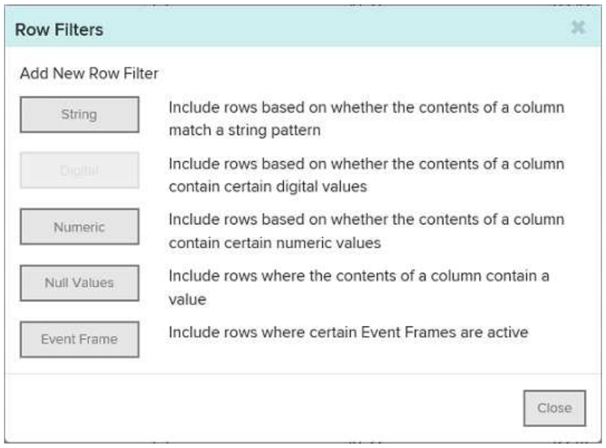

Edit Value Mode
Interpolated Values
Every 1 minutes

Using the “Edit Value Mode” dialog, we have the option to change how often we sample the data from PI System. The default selection is to interpolate values every minute. The data are backfilled at 1 minute interval, so we can keep the default setting.


Edit Row Filters
0 Row Filters

As we have seen in the previous section, engines are not running for the entire period from midnight to 7am. For this analysis, we are only interested in the sensor data for the time period when the engines are running and status is not “Stopped”. Hence we will filter and exclude data for periods when engine status is “Stopped”. Click on “Edit Row Filters” to launch the dialog show below.

Using this dialog, select **String** to add a new string filter and filter based on the Status attribute’s value.

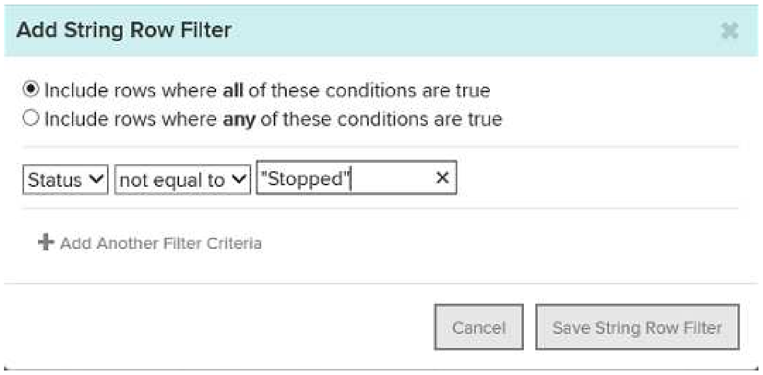

Row Filters

Add New Row Filter

String	Include rows based on whether the contents of a column match a string pattern
Digital	Include rows based on whether the contents of a column contain certain digital values
Numeric	Include rows based on whether the contents of a column contain certain numeric values
Null Values	Include rows where the contents of a column contain a value
Event Frame	Include rows where certain Event Frames are active

Close

Specify that for the “Status” column, any record containing the “Stopped” state (double quotes are required) should not be published. **Save String Row Filter** and **Close** the “Row Filters” dialog.


Add String Row Filter

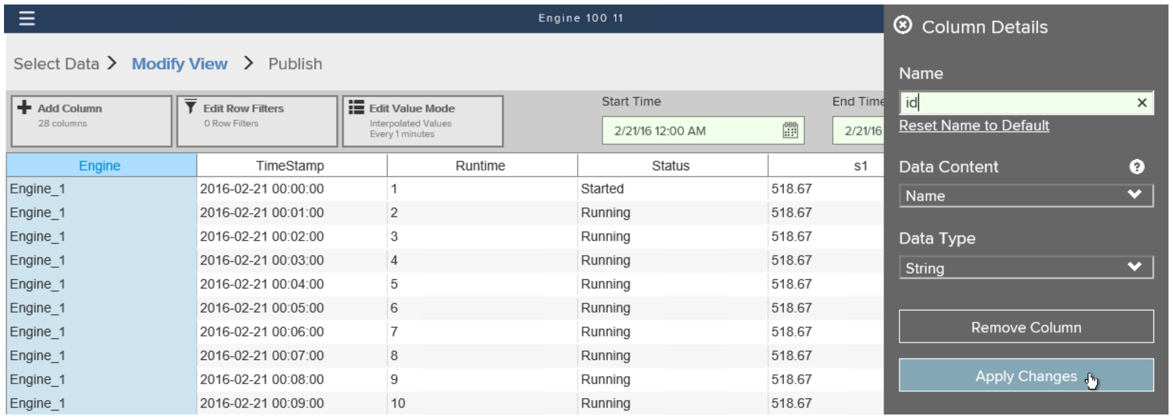
(●) Include rows where **all** of these conditions are true
( ) Include rows where **any** of these conditions are true

Status | not equal to | "Stopped"

+ Add Another Filter Criteria

Cancel | Save String Row Filter

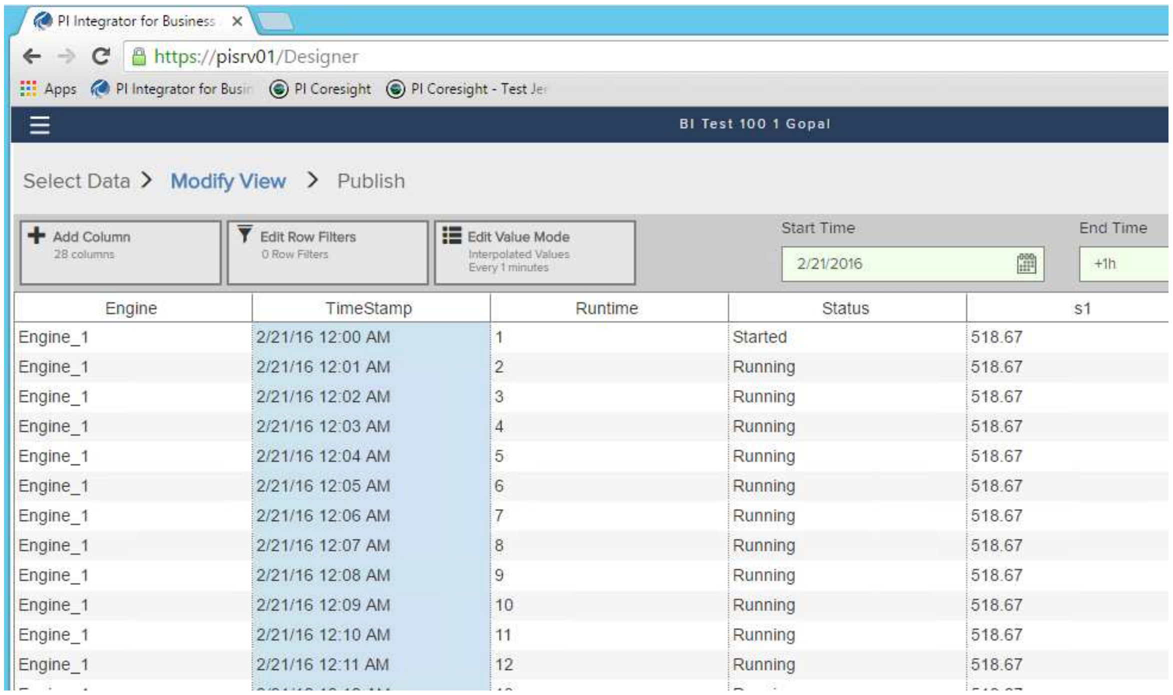
The last shaping step involves renaming and removing columns not required by the R script being used to analyze this data. First, select the “Engine” column. The “Column Details” panel will show up on the right-hand side of the page. Change the “Name” field to “id” and **Apply Changes**.



Engine	TimeStamp	Runtime	Status	s1
Engine_1	2016-02-21 00:00:00	1	Started	518.67
Engine_1	2016-02-21 00:01:00	2	Running	518.67
Engine_1	2016-02-21 00:02:00	3	Running	518.67
Engine_1	2016-02-21 00:03:00	4	Running	518.67
Engine_1	2016-02-21 00:04:00	5	Running	518.67
Engine_1	2016-02-21 00:05:00	6	Running	518.67
Engine_1	2016-02-21 00:06:00	7	Running	518.67
Engine_1	2016-02-21 00:07:00	8	Running	518.67
Engine_1	2016-02-21 00:08:00	9	Running	518.67
Engine_1	2016-02-21 00:09:00	10	Running	518.67

Repeat this step by renaming the “Runtime” column to “cycle” (small letter “c”, R is case sensitive).

Next, select the “TimeStamp” column. In the Column Detail pane, select **Remove Column**. Also remove the “Status” column. We don’t need them for our analysis. **Note**: Although we are filtering the data based on the “Status” column (not equal to “Stopped”), we do not need it in the published dataset.



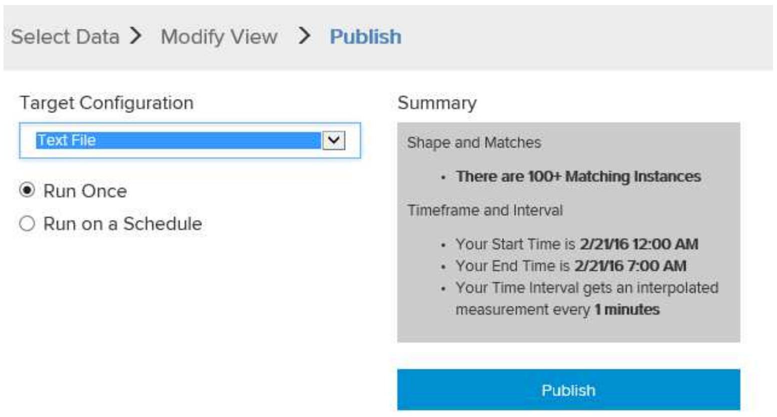
Engine	TimeStamp	Runtime	Status	s1
Engine_1	2/21/16 12:00 AM	1	Started	518.67
Engine_1	2/21/16 12:01 AM	2	Running	518.67
Engine_1	2/21/16 12:02 AM	3	Running	518.67
Engine_1	2/21/16 12:03 AM	4	Running	518.67
Engine_1	2/21/16 12:04 AM	5	Running	518.67
Engine_1	2/21/16 12:05 AM	6	Running	518.67
Engine_1	2/21/16 12:06 AM	7	Running	518.67
Engine_1	2/21/16 12:07 AM	8	Running	518.67
Engine_1	2/21/16 12:08 AM	9	Running	518.67
Engine_1	2/21/16 12:09 AM	10	Running	518.67
Engine_1	2/21/16 12:10 AM	11	Running	518.67
Engine_1	2/21/16 12:11 AM	12	Running	518.67

We are now ready to publish our dataset. Select **Next** in the upper right-hand corner of the page to move to the “Publish” page of the PI Integrator for BA.

For **Target Configuration**, click on the drop down and select **Text File**. We are also going to perform a one-time publish of this view. Hence you can select **Run Once**.

Click on **Publish** to get the PI Integrator to start publishing the dataset to a file.

You will have to **Confirm** that this is Ok.



Target Configuration

Text File

● Run Once
○ Run on a Schedule

Summary

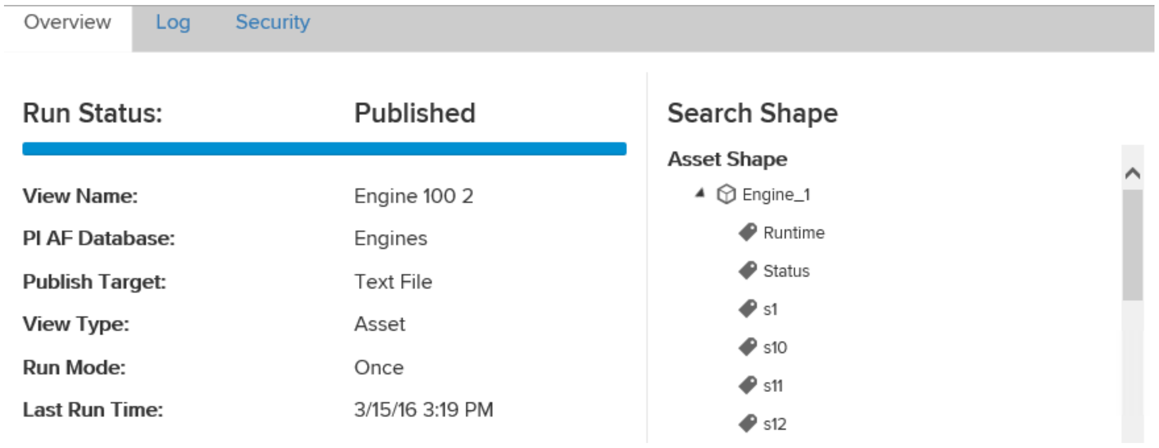
Shape and Matches

- **There are 100+ Matching Instances**

Timeframe and Interval

- Your Start Time is **2/21/16 12:00 AM**
- Your End Time is **2/21/16 7:00 AM**
- Your Time Interval gets an interpolated measurement every **1 minutes**

Then you should be directed back to the page showing the existing list of views. The bottom of the page shows a run status of the publish action.



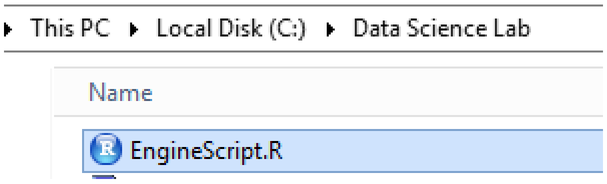
Run Status:	Published
View Name:	Engine 100 2
PI AF Database:	Engines
Publish Target:	Text File
View Type:	Asset
Run Mode:	Once
Last Run Time:	3/15/16 3:19 PM

Once the publication is finished, open the “Data Science Lab” (C:\Data Science Lab) folder on the desktop. You will see the text file you have just published. It will be named with your Asset View name followed by a time stamp.

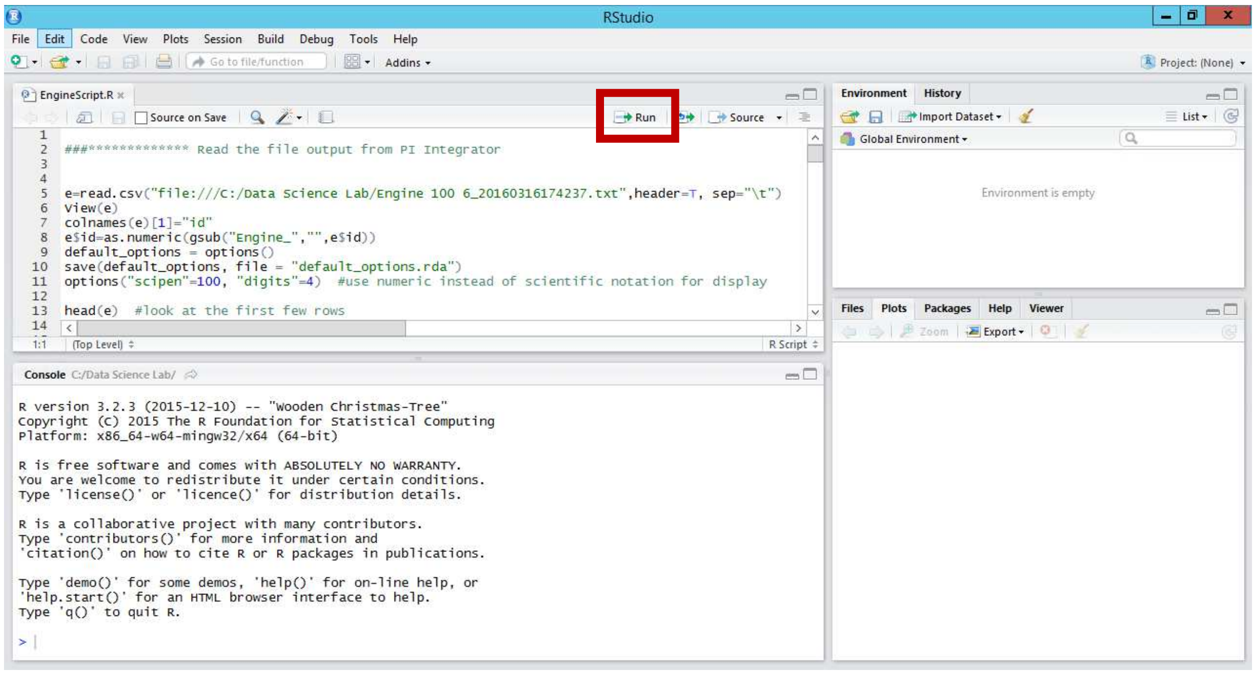
We will use this file as input to the R script analysis in the next section of the lab.

Part II – Predict Equipment Failure Using Statistical Analysis

From the Data Science Lab folder, select EngineScript.R and double-click to open it in R Studio.



The screen below shows the R Studio user interface.

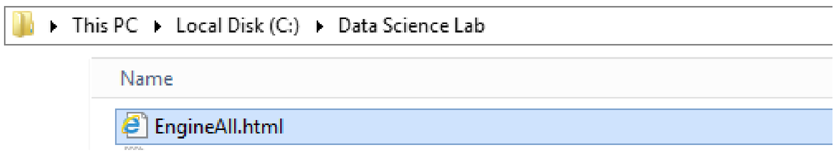


```
###************** Read the file output from PI Integrator


e=read.csv("file:///C:/Data Science Lab/Engine 100 6_20160316174237.txt",header=T, sep="\t")
View(e)
colnames(e)[1]="id"
e$id=as.numeric(gsub("Engine_","",e$id))
default_options = options()
save(default_options, file = "default_options.rda")
options("scipen"=100, "digits"=4)  #use numeric instead of scientific notation for display

head(e)  #look at the first few rows
```

```
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

The following sections show the output when you step through the script line by line via <<Run>>.

The following pages have been extracted from the script document EngineAll.html (see Data Science Lab folder for the latest revision).



Read the PI Integrator file output

```
e = read.csv("file:///C:/Users/gopal/Documents/UC16/Engine 100 6_20160316174237.txt",
    header = T, sep = "\t")
colnames(e)[1] = "id"
e$id = as.numeric(gsub("Engine_", "", e$id))

# options('scipen'=100, 'digits'=4) #use numeric instead of scientific notation
# for display

head(e)  #look at the first few rows
```

```
##   id cycle     s1 s10   s11    s12     s13     s14    s15  s16 s17  s18
## 1  1     1 518.67 1.3 47.47 521.66 2388.02 8138.62 8.4195 0.03 392 2388
## 2  1     2 518.67 1.3 47.49 522.28 2388.07 8131.49 8.4318 0.03 392 2388
## 3  1     3 518.67 1.3 47.27 522.42 2388.03 8133.23 8.4178 0.03 390 2388
## 4  1     4 518.67 1.3 47.13 522.86 2388.08 8133.83 8.3682 0.03 392 2388
## 5  1     5 518.67 1.3 47.28 522.19 2388.04 8133.80 8.4294 0.03 393 2388
## 6  1     6 518.67 1.3 47.16 521.68 2388.03 8132.85 8.4108 0.03 391 2388
##   s19     s2   s20     s21      s3      s4    s5    s6     s7      s8
## 1 100 641.82 39.06 23.4190 1589.70 1400.60 14.62 21.61 554.36 2388.06
## 2 100 642.15 39.00 23.4236 1591.82 1403.14 14.62 21.61 553.75 2388.04
## 3 100 642.35 38.95 23.3442 1587.99 1404.20 14.62 21.61 554.26 2388.08
## 4 100 642.35 38.88 23.3739 1582.79 1401.87 14.62 21.61 554.45 2388.11
## 5 100 642.37 38.90 23.4044 1582.85 1406.22 14.62 21.61 554.00 2388.06
## 6 100 642.10 38.98 23.3669 1584.47 1398.37 14.62 21.61 554.67 2388.02
##        s9 setting1 setting2 setting3
## 1 9046.19  -0.0007   -4e-04      100
## 2 9044.07   0.0019   -3e-04      100
## 3 9052.94  -0.0043    3e-04      100
## 4 9049.48   0.0007    0e+00      100
## 5 9055.15  -0.0019   -2e-04      100
## 6 9049.68  -0.0043   -1e-04      100
```

```
View(e)  #look at the full dataset in a grid format
```

Do some statistics and data munging

```
summary(e)  # stat summary - quartiles, mean etc.
```

```
##        id             cycle             s1              s10     
##  Min.   :  1.00   Min.   :  1.0   Min.   :518.7   Min.   :1.3  
##  1st Qu.: 26.00   1st Qu.: 52.0   1st Qu.:518.7   1st Qu.:1.3  
##  Median : 52.00   Median :104.0   Median :518.7   Median :1.3  
##  Mean   : 51.51   Mean   :108.8   Mean   :518.7   Mean   :1.3  
##  3rd Qu.: 77.00   3rd Qu.:156.0   3rd Qu.:518.7   3rd Qu.:1.3  
##  Max.   :100.00   Max.   :362.0   Max.   :518.7   Max.   :1.3  
##       s11             s12             s13             s14      
##  Min.   :46.85   Min.   :518.7   Min.   :2388   Min.   :8100  
##  1st Qu.:47.35   1st Qu.:521.0   1st Qu.:2388   1st Qu.:8133  
##  Median :47.51   Median :521.5   Median :2388   Median :8141  
##  Mean   :47.54   Mean   :521.4   Mean   :2388   Mean   :8144  
##  3rd Qu.:47.70   3rd Qu.:522.0   3rd Qu.:2388   3rd Qu.:8148  
##  Max.   :48.53   Max.   :523.4   Max.   :2389   Max.   :8294  
##       s15             s16            s17             s18      
##  Min.   :8.325   Min.   :0.03   Min.   :388.0   Min.   :2388  
##  1st Qu.:8.415   1st Qu.:0.03   1st Qu.:392.0   1st Qu.:2388  
##  Median :8.439   Median :0.03   Median :393.0   Median :2388  
##  Mean   :8.442   Mean   :0.03   Mean   :393.2   Mean   :2388  
##  3rd Qu.:8.466   3rd Qu.:0.03   3rd Qu.:394.0   3rd Qu.:2388  
##  Max.   :8.585   Max.   :0.03   Max.   :400.0   Max.   :2388  
##       s19           s2             s20             s21       
##  Min.   :100   Min.   :641.2   Min.   :38.14   Min.   :22.89  
##  1st Qu.:100   1st Qu.:642.3   1st Qu.:38.70   1st Qu.:23.22  
##  Median :100   Median :642.6   Median :38.83   Median :23.30  
##  Mean   :100   Mean   :642.7   Mean   :38.82   Mean   :23.29  
##  3rd Qu.:100   3rd Qu.:643.0   3rd Qu.:38.95   3rd Qu.:23.37  
##  Max.   :100   Max.   :644.5   Max.   :39.43   Max.   :23.62  
```

```
##        s3             s4             s5              s6       
##  Min.   :1571   Min.   :1382   Min.   :14.62   Min.   :21.60  
##  1st Qu.:1586   1st Qu.:1402   1st Qu.:14.62   1st Qu.:21.61  
##  Median :1590   Median :1408   Median :14.62   Median :21.61  
##  Mean   :1591   Mean   :1409   Mean   :14.62   Mean   :21.61  
##  3rd Qu.:1594   3rd Qu.:1415   3rd Qu.:14.62   3rd Qu.:21.61  
##  Max.   :1617   Max.   :1441   Max.   :14.62   Max.   :21.61  
##        s7              s8             s9          setting1        
##  Min.   :549.9   Min.   :2388   Min.   :9022   Min.   :-8.70e-03  
##  1st Qu.:552.8   1st Qu.:2388   1st Qu.:9053   1st Qu.:-1.50e-03  
##  Median :553.4   Median :2388   Median :9061   Median : 0.00e+00  
##  Mean   :553.4   Mean   :2388   Mean   :9065   Mean   :-8.87e-06  
##  3rd Qu.:554.0   3rd Qu.:2388   3rd Qu.:9069   3rd Qu.: 1.50e-03  
##  Max.   :556.1   Max.   :2389   Max.   :9245   Max.   : 8.70e-03  
##     setting2              setting3  
##  Min.   :-6.000e-04   Min.   :100  
##  1st Qu.:-2.000e-04   1st Qu.:100  
##  Median : 0.000e+00   Median :100  
##  Mean   : 2.351e-06   Mean   :100  
##  3rd Qu.: 3.000e-04   3rd Qu.:100  
##  Max.   : 6.000e-04   Max.   :100  
```

```
sapply(e, sd)  #sdev for each variable
```

```
##           id        cycle           s1          s10          s11
## 2.922763e+01 6.888099e+01 0.000000e+00 0.000000e+00 2.670874e-01
##          s12          s13          s14          s15          s16
## 7.375534e-01 7.191892e-02 1.907618e+01 3.750504e-02 0.000000e+00
##          s17          s18          s19           s2          s20
## 1.548763e+00 0.000000e+00 0.000000e+00 5.000533e-01 1.807464e-01
##          s21           s3           s4           s5           s6
## 1.082509e-01 6.131150e+00 9.000605e+00 0.000000e+00 1.388985e-03
##           s7           s8           s9     setting1     setting2
## 8.850923e-01 7.098548e-02 2.208288e+01 2.187313e-03 2.930621e-04
##     setting3
## 0.000000e+00
```

```
e = subset(e, select = -c(setting3, s1, s5, s10, s16, s18, s19))  #based on sdev, delete columns that are not chan
ging
```

Plotting engine failure distribution

```
e.fail = tapply(e$cycle, e$id, max)  #get cycle number when engine failed
e.fail  #display the list of engines and when they failed
```

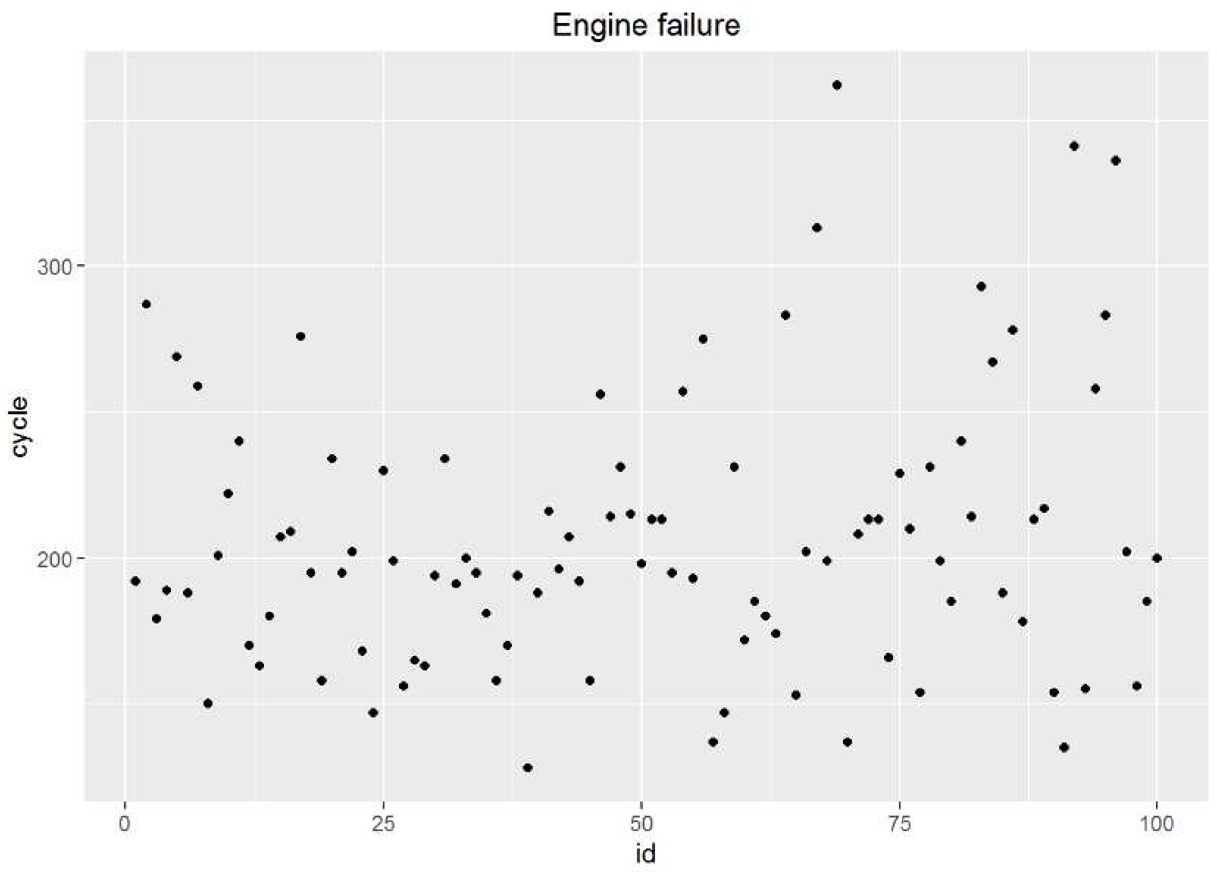
```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## 192 287 179 189 269 188 259 150 201 222 240 170 163 180 207 209 276 195
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
## 158 234 195 202 168 147 230 199 156 165 163 194 234 191 200 195 181 158
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
## 170 194 128 188 216 196 207 192 158 256 214 231 215 198 213 213 195 257
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
## 193 275 137 147 231 172 185 180 174 283 153 202 313 199 362 137 208 213
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
## 213 166 229 210 154 231 199 185 240 214 293 267 188 278 178 213 217 154
##  91  92  93  94  95  96  97  98  99 100
## 135 341 155 258 283 336 202 156 185 200
```

```
library(ggplot2)
qplot(y = e.fail, x = 1:length(e.fail), main = "Engine failure", ylab = "cycle",
    xlab = "id")
```

```
summary(e.fail)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   128.0   177.0   199.0   206.3   229.2   362.0
```

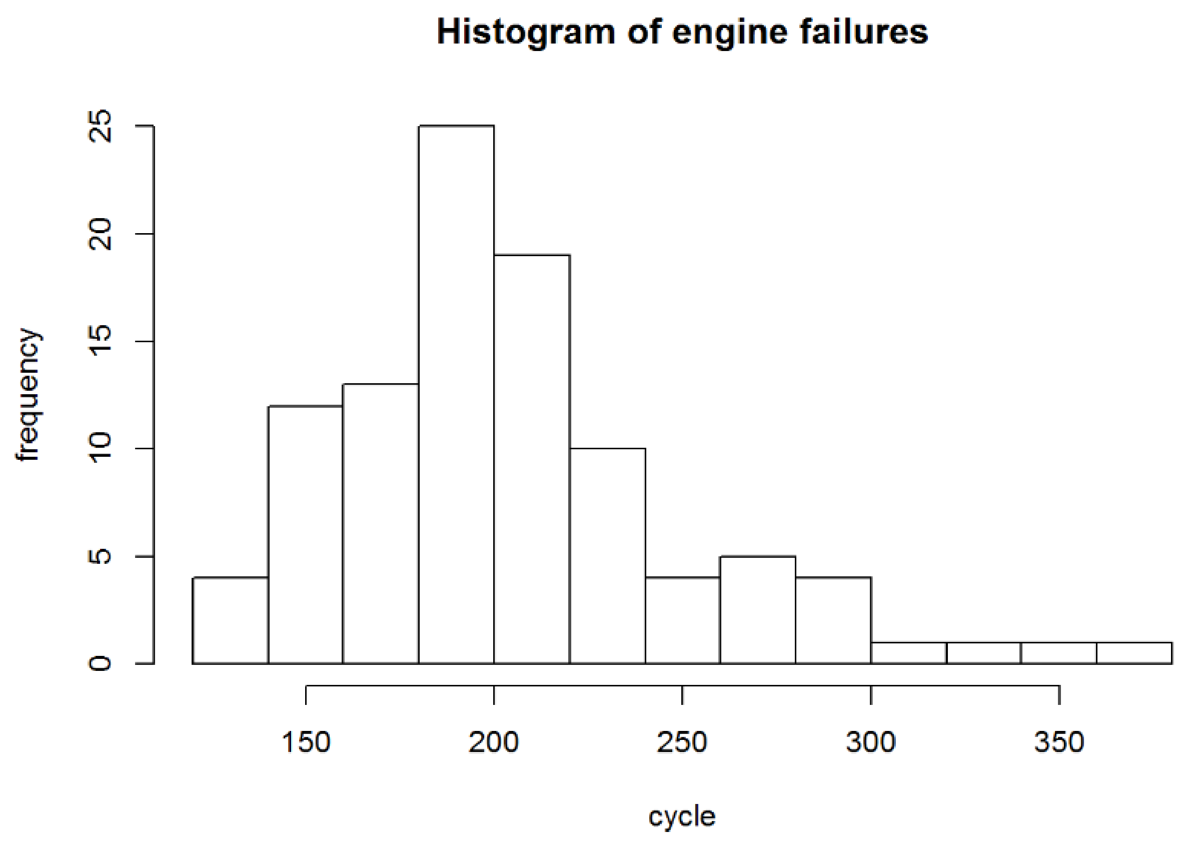
```
hist(e.fail, 10, ylab = "frequency", xlab = "cycle", main = "Histogram of engine failures")  #histogram, 10 bucket
s
```



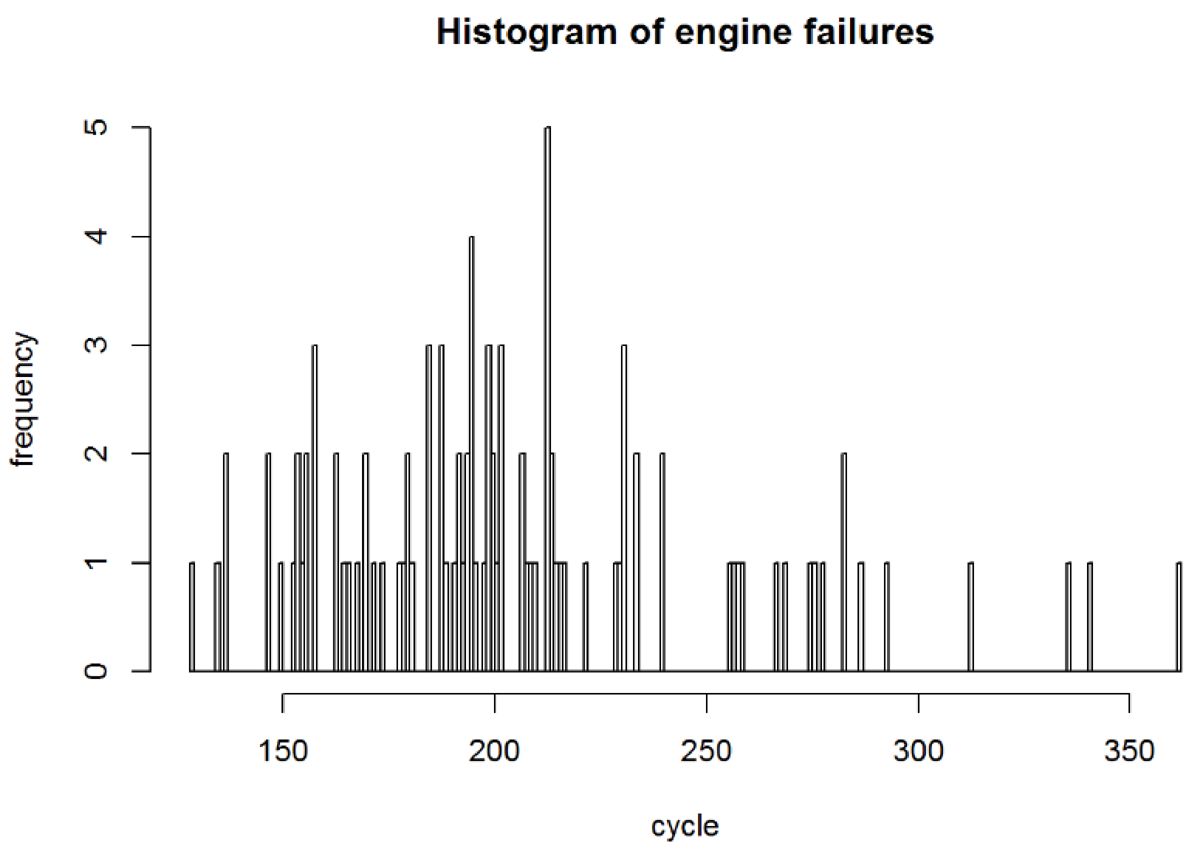
```
summary(e.fail)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   128.0   177.0   199.0   206.3   229.2   362.0
```

```
hist(e.fail, 10, ylab = "frequency", xlab = "cycle", main = "Histogram of engine failures")  #histogram, 10 bucket
s
```



```
hist(e.fail, 200, ylab = "frequency", xlab = "cycle", main = "Histogram of engine failures")  #histogram, 200 buck
ets
```



```
e$rul = e.fail[e$id] - e$cycle  #rul remaining useful life; add the column to dataset

# Look at the Grid to confirm
```

Closer look at Engine 1 - is there a correlation among the variables?

```
e1 = subset(e, id == 1)  #let's take a closer look at engine1 - subset data for e1
e2 = subset(e, id == 2)  #and for engine2

e1.obs = subset(e1, select = -c(id, cycle, rul))  #separate the observations for e1
head(e1.obs)  #look at the first few values and confirm the numbers look OK
```

```
##     s11    s12     s13     s14    s15 s17     s2   s20     s21      s3
## 1 47.47 521.66 2388.02 8138.62 8.4195 392 641.82 39.06 23.4190 1589.70
## 2 47.49 522.28 2388.07 8131.49 8.4318 392 642.15 39.00 23.4236 1591.82
## 3 47.27 522.42 2388.03 8133.23 8.4178 390 642.35 38.95 23.3442 1587.99
## 4 47.13 522.86 2388.08 8133.83 8.3682 392 642.35 38.88 23.3739 1582.79
## 5 47.28 522.19 2388.04 8133.80 8.4294 393 642.37 38.90 23.4044 1582.85
## 6 47.16 521.68 2388.03 8132.85 8.4108 391 642.10 38.98 23.3669 1584.47
##        s4    s6     s7      s8      s9 setting1 setting2
## 1 1400.60 21.61 554.36 2388.06 9046.19  -0.0007   -4e-04
## 2 1403.14 21.61 553.75 2388.04 9044.07   0.0019   -3e-04
## 3 1404.20 21.61 554.26 2388.08 9052.94  -0.0043    3e-04
## 4 1401.87 21.61 554.45 2388.11 9049.48   0.0007    0e+00
## 5 1406.22 21.61 554.00 2388.06 9055.15  -0.0019   -2e-04
## 6 1398.37 21.61 554.67 2388.02 9049.68  -0.0043   -1e-04
```

```
sapply(e1, sd)  #sdev for variables for engine1; note s6 sdev is zero
```

```
##           id        cycle          s11          s12          s13
## 0.000000e+00 5.556978e+01 2.683460e-01 7.491763e-01 7.696013e-02
##          s14          s15          s17           s2          s20
## 5.563129e+00 3.412461e-02 1.475661e+00 4.867952e-01 1.669984e-01
##          s21           s3           s4           s6           s7
## 1.051013e-01 5.759776e+00 8.565610e+00 0.000000e+00 9.104140e-01
##           s8           s9     setting1     setting2          rul
## 7.091352e-02 4.911853e+00 1.953446e-03 2.822825e-04 5.556978e+01
```

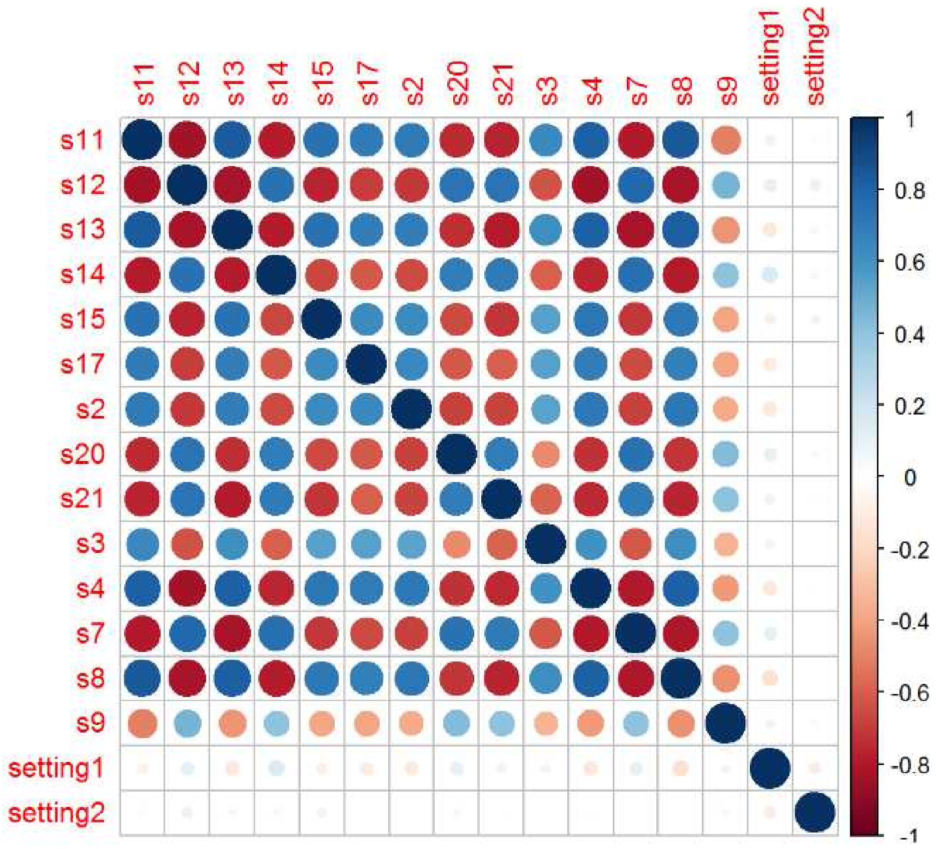
```
e1.obs$s6 = NULL  #remove s6 since sdev=0

e1.obs.cor = cor(e1.obs)  #get correlation among variables
e1.obs.cor  #look at the raw numbers
```

```
##                  s11         s12         s13         s14         s15
## s11       1.00000000 -0.83567272  0.83070910 -0.78847549  0.74882569
## s12      -0.83567272  1.00000000 -0.82538617  0.74761844 -0.76968432
## s13       0.83070910 -0.82538617  1.00000000 -0.78084011  0.74742069
## s14      -0.78847549  0.74761844 -0.78084011  1.00000000 -0.66288631
## s15       0.74882569 -0.76968432  0.74742069 -0.66288631  1.00000000
## s17       0.70648126 -0.69510642  0.69167504 -0.61909229  0.63848957
## s2        0.70892236 -0.70945643  0.69312704 -0.65032042  0.63735833
## s20      -0.74584382  0.73152497 -0.73192400  0.69703426 -0.65279475
## s21      -0.76705481  0.73379881 -0.78624008  0.70816583 -0.71270523
## s3        0.64583709 -0.63068412  0.61343906 -0.59324696  0.54474263
## s4        0.81734355 -0.83804063  0.81893189 -0.75720976  0.72788595
## s7       -0.79253149  0.78669863 -0.82903694  0.75923573 -0.70501690
## s8        0.84908656 -0.82704263  0.82595027 -0.78904065  0.71671389
## s9       -0.50641451  0.46689444 -0.44611064  0.40002014 -0.39006634
## setting1 -0.07484391  0.10118350 -0.12999547  0.15461020 -0.08398133
## setting2 -0.02963625  0.08383545 -0.04934473 -0.04908604  0.06076079
##                  s17           s2         s20         s21           s3
## s11       0.706481260  0.708922359 -0.74584382 -0.76705481  0.645837085
## s12      -0.695106422 -0.709456432  0.73152497  0.73379881 -0.630684121
## s13       0.691675035  0.693127036 -0.73192400 -0.78624008  0.613439058
## s14      -0.619092292 -0.650320420  0.69703426  0.70816583 -0.593246956
## s15       0.638489572  0.637358325 -0.65279475 -0.71270523  0.544742629
## s17       1.000000000  0.648226680 -0.61821418 -0.59733207  0.546325528
## s2        0.648226680  1.000000000 -0.68868414 -0.67493600  0.533992780
## s20      -0.618214183 -0.688684144  1.00000000  0.69601165 -0.470541419
## s21      -0.597332070 -0.674935999  0.69601165  1.00000000 -0.583545957
## s3        0.546325528  0.533992780 -0.47054142 -0.58354596  1.000000000
## s4        0.691185916  0.715269184 -0.72241582 -0.74452806  0.600407683
## s7       -0.655866038 -0.689220441  0.74473482  0.70911279 -0.612002343
## s8        0.683807028  0.725279146 -0.71594555 -0.76094272  0.618465028
## s9       -0.392893764 -0.378997689  0.43390382  0.40927155 -0.342261063
## setting1 -0.108964405 -0.117885277  0.10266482  0.07221062 -0.064242585
## setting2 -0.003063662 -0.002959971  0.04270554 -0.01268866  0.008844666
```

```
##                  s4           s7           s8          s9    setting1
## s11       0.81734355 -0.792531494  0.849086563 -0.50641451 -0.07484391
## s12      -0.83804063  0.786698631 -0.827042625  0.46689444  0.10118350
## s13       0.81893189 -0.829036942  0.825950267 -0.44611064 -0.12999547
## s14      -0.75720976  0.759235726 -0.789040652  0.40002014  0.15461020
## s15       0.72788595 -0.705016899  0.716713894 -0.39006634 -0.08398133
## s17       0.69118592 -0.655866038  0.683807028 -0.39289376 -0.10896441
## s2        0.71526918 -0.689220441  0.725279146 -0.37899769 -0.11788528
## s20      -0.72241582  0.744734821 -0.715945551  0.43390382  0.10266482
## s21      -0.74452806  0.709112791 -0.760942723  0.40927155  0.07221062
## s3        0.60040768 -0.612002343  0.618465028 -0.34226106 -0.06424258
## s4        1.00000000 -0.799178266  0.814944317 -0.42570953 -0.12023640
## s7       -0.79917827  1.000000000 -0.809018046  0.40213333  0.10421542
## s8        0.81494432 -0.809018046  1.000000000 -0.45893545 -0.15834455
## s9       -0.42570953  0.402133334 -0.458935453  1.00000000  0.06295341
## setting1 -0.12023640  0.104215415 -0.158344549  0.06295341  1.00000000
## setting2 -0.01500111  0.006118427 -0.007956821  0.03300548 -0.09819150
```

```
##              setting2
## s11      -0.029636254
## s12       0.083835448
## s13      -0.049344730
## s14      -0.049086045
## s15       0.060760795
## s17      -0.003063662
## s2       -0.002959971
## s20       0.042705544
## s21      -0.012688660
## s3        0.008844666
## s4       -0.015001112
## s7        0.006118427
## s8       -0.007956821
## s9        0.033005477
## setting1 -0.098191499
## setting2  1.000000000
```

```
library(corrplot)  #load library
par.defaults = par(no.readonly = TRUE)
save(par.defaults, file = "R.default.par.RData")  #save plot defaults for later use

corrplot(e1.obs.cor)  #plot correlation
```



Use odd numbered engines for training

```
e.obs = subset(e, select = -c(cycle, id, rul))  #obs for all engines
head(e.obs)  #look at the first few values and confirm the numbers look OK
```

```
##     s11    s12     s13     s14    s15 s17     s2   s20     s21      s3
## 1 47.47 521.66 2388.02 8138.62 8.4195 392 641.82 39.06 23.4190 1589.70
## 2 47.49 522.28 2388.07 8131.49 8.4318 392 642.15 39.00 23.4236 1591.82
## 3 47.27 522.42 2388.03 8133.23 8.4178 390 642.35 38.95 23.3442 1587.99
## 4 47.13 522.86 2388.08 8133.83 8.3682 392 642.35 38.88 23.3739 1582.79
## 5 47.28 522.19 2388.04 8133.80 8.4294 393 642.37 38.90 23.4044 1582.85
## 6 47.16 521.68 2388.03 8132.85 8.4108 391 642.10 38.98 23.3669 1584.47
##        s4    s6     s7      s8      s9 setting1 setting2
## 1 1400.60 21.61 554.36 2388.06 9046.19  -0.0007   -4e-04
## 2 1403.14 21.61 553.75 2388.04 9044.07   0.0019   -3e-04
## 3 1404.20 21.61 554.26 2388.08 9052.94  -0.0043    3e-04
## 4 1401.87 21.61 554.45 2388.11 9049.48   0.0007    0e+00
## 5 1406.22 21.61 554.00 2388.06 9055.15  -0.0019   -2e-04
## 6 1398.37 21.61 554.67 2388.02 9049.68  -0.0043   -1e-04
```

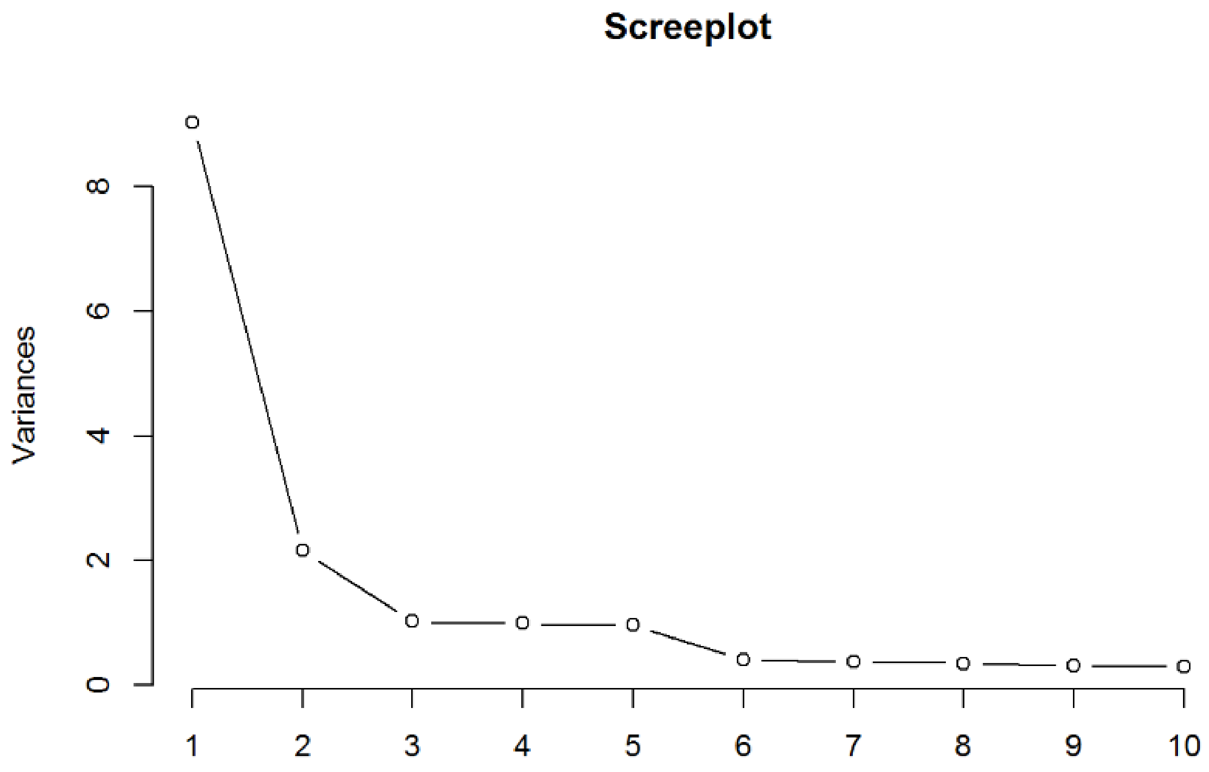
```
e.odd.obs = subset(e.obs, e$id%%2 == 1)  #odd engines - observations only
```

Extract principal components

```
e.odd.pca = prcomp(e.odd.obs, scale = T, center = T)  #fit principal components (PC), use only odd engine data

summary(e.odd.pca)  #look at PC contributions
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     3.0043 1.4670 1.01012 0.99519 0.97771 0.63974
## Proportion of Variance 0.5309 0.1266 0.06002 0.05826 0.05623 0.02407
## Cumulative Proportion  0.5309 0.6575 0.71754 0.77579 0.83203 0.85610
##                           PC7     PC8     PC9    PC10    PC11   PC12
## Standard deviation     0.6032 0.58852 0.55019 0.53828 0.49954 0.4479
## Proportion of Variance 0.0214 0.02037 0.01781 0.01704 0.01468 0.0118
## Cumulative Proportion  0.8775 0.89787 0.91568 0.93272 0.94740 0.9592
##                           PC13    PC14    PC15    PC16    PC17
## Standard deviation     0.43739 0.41394 0.40052 0.37833 0.16538
## Proportion of Variance 0.01125 0.01008 0.00944 0.00842 0.00161
## Cumulative Proportion  0.97046 0.98054 0.98997 0.99839 1.00000
```

```
load("R.default.par.RData")
par(par.defaults)
screeplot(e.odd.pca, type = "line", main = "Screeplot")  #screeplot
```



```
e.pr = predict(e.odd.pca, e.obs)  #predict for all engines - but using model fit based on only odd engine data
head(e.pr)  #look at the first few rows - these are in terms of PCs
```

```
##         PC1        PC2        PC3        PC4         PC5         PC6
## 1 -2.579784 -0.5305803  1.1514098 -0.3097415 -0.81013076  0.82916426
## 2 -1.982479 -0.8503658  0.1721622 -1.0657629 -0.84199682  0.96303049
## 3 -2.609334 -0.6269731  0.3714194  2.0887107  0.55204118  0.54551121
## 4 -3.009002 -0.9115345 -0.2830690 -0.1297345 -0.32977079 -0.39202173
## 5 -2.063739 -0.4313629  0.9711654  0.4216735 -0.38192589 -0.72831083
## 6 -3.283768 -0.6027247  1.3815984  1.4374694 -0.09538299  0.06675428
##            PC7         PC8         PC9        PC10        PC11       PC12
## 1 -0.609300635 -0.35200453  0.04239202 -0.69886185  0.23533245 -0.1536808
## 2 -0.292776817 -0.04814031 -0.34816417 -0.65306492 -0.02198571  0.1862107
## 3  1.184176200 -0.32701796  0.31599377 -0.09438622 -0.13612781 -0.3168786
## 4 -0.000860599  0.10032179 -0.09194942  1.15685502  0.34151314 -0.8165153
## 5 -0.341011329 -0.02885155 -0.51346334 -0.17619308 -0.14935820 -0.1008669
## 6  0.339944963 -0.34242499  0.29115428 -0.03869910 -0.27262292 -0.2042652
##          PC13       PC14        PC15        PC16         PC17
## 1  0.04401746 -0.6946445 -0.45792303 -0.37182178 -0.336241717
## 2  0.09372164  0.2179582  0.36080959 -0.69208000 -0.155761131
## 3  0.10050245  0.6096518 -0.38194821 -0.09050710  0.059800048
## 4 -0.54913207  1.1921734 -0.12805689  0.12100721 -0.067595725
## 5  0.38146247  0.4616110 -0.14741048  0.05076316  0.101819839
## 6  0.14767365 -0.6128534 -0.02089315  0.58368569 -0.003589362
```

```
# get pc1 equation
pc1eq = ""
for (j in 1:17) {
    pc1eq = cat(sep = "", pc1eq, "+(", "'", names(e.odd.pca$center[j]), "'", "-(",
        e.odd.pca$center[j], ")", ")/", e.odd.pca$scale[j], "*", e.odd.pca$rotation[j,
            1])
}
```

```
## +('s11'-(47.51488))/0.2701003*0.3090913+('s12'-(521.4901))/0.7517117*-0.3049236+('s13'-(2388.09))/0.07484883*0.
2845465+('s14'-(8143.502))/19.7965*0.04163657+('s15'-(8.438634))/0.03782789*0.2868222+('s17'-(393.0714))/1.561964*
0.2685557+('s2'-(642.638))/0.5043607*0.2734667+('s20'-(38.83337))/0.1812555*-0.2819219+('s21'-(23.29963))/0.108387
2*-0.2834525+('s3'-(1590.048))/6.186916*0.2604444+('s4'-(1408.104))/9.077463*0.3006121+('s6'-(21.60976))/0.0015392
59*0.06360376+('s7'-(553.4522))/0.8983562*-0.2995252+('s8'-(2388.091))/0.07388822*0.2847322+('s9'-(9064.651))/22.7
2082*0.08204075+('setting1'-(-3.554925e-05))/0.002184843*0.003580013+('setting2'-(5.022518e-06))/0.0002931999*0.00
3136759
```

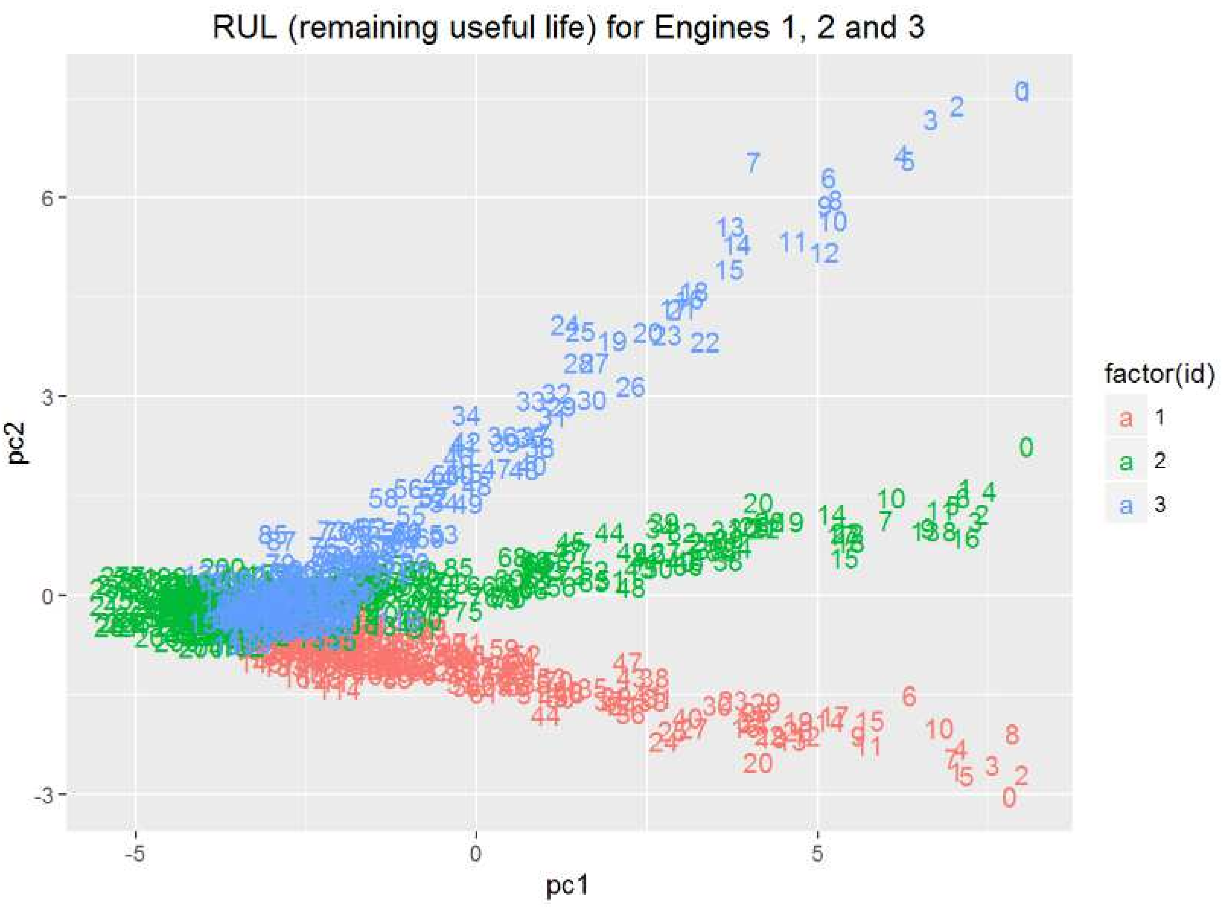
```
# get pc2 equation
pc2eq = ""
for (j in 1:17) {
    pc2eq = cat(sep = "", pc2eq, "+(", "'", names(e.odd.pca$center[j]), "'", "-(",
        e.odd.pca$center[j], ")", ")/", e.odd.pca$scale[j], "*", e.odd.pca$rotation[j,
            2])
}
```

```
## +('s11'-(47.51488))/0.2701003*-0.005176845+('s12'-(521.4901))/0.7517117*0.05515457+('s13'-(2388.09))/0.07484883
*-0.2340729+('s14'-(8143.502))/19.7965*0.6669612+('s15'-(8.438634))/0.03782789*0.02955775+('s17'-(393.0714))/1.561
964*0.0884907+('s2'-(642.638))/0.5043607*0.02942649+('s20'-(38.83337))/0.1812555*-0.02673732+('s21'-(23.29963))/0.
1083872*-0.03537639+('s3'-(1590.048))/6.186916*0.08201158+('s4'-(1408.104))/9.077463*0.02025907+('s6'-(21.60976))/
0.001539259*-0.04061592+('s7'-(553.4522))/0.8983562*0.04585588+('s8'-(2388.091))/0.07388822*-0.2283776+('s9'-(906
4.651))/22.72082*0.6501037+('setting1'-(-3.554925e-05))/0.002184843*-0.004670446+('setting2'-(5.022518e-06))/0.000
2931999*-0.00809504
```

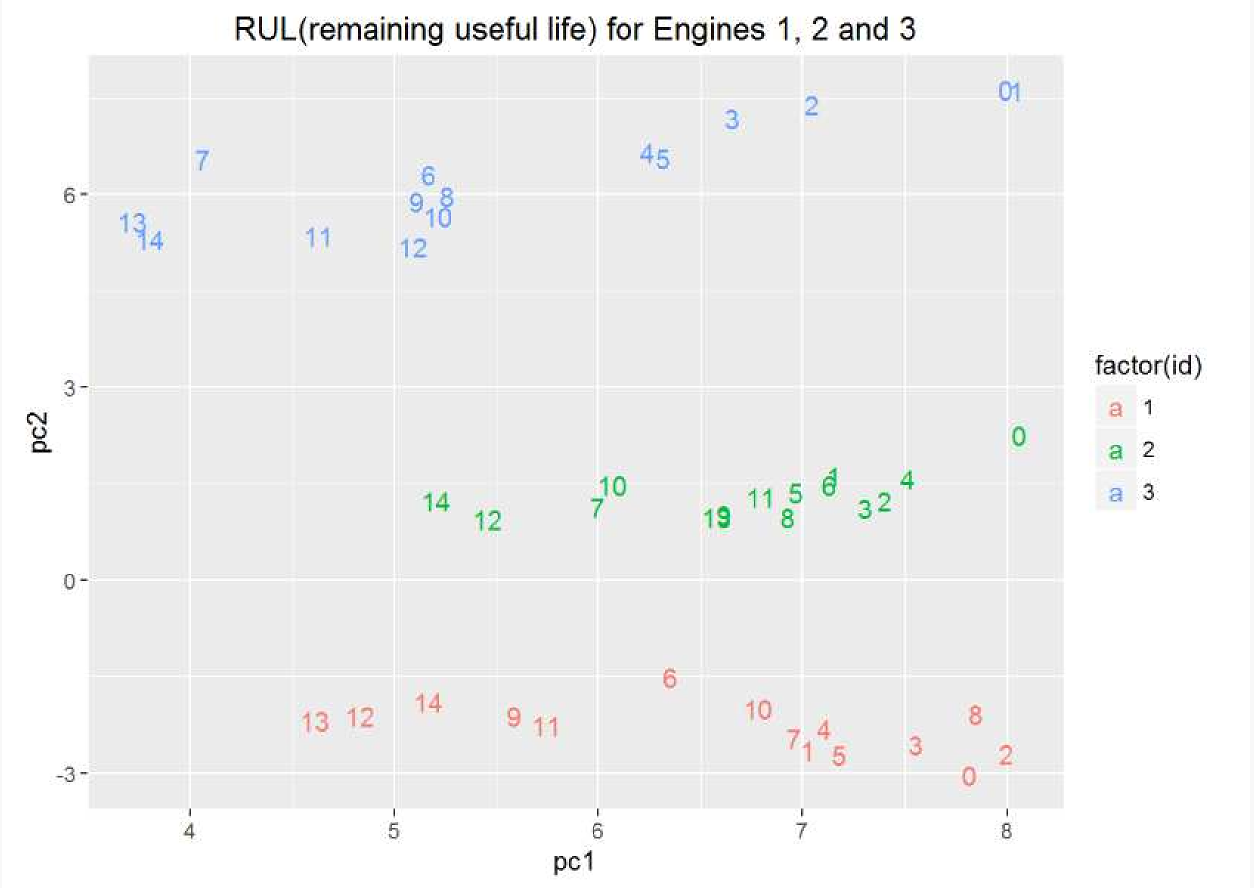
```
e$pc1 = e.pr[, 1]  #pc1 column added to dataset
e$pc2 = e.pr[, 2]  #pc2 column added to dataset
```

Plot engine cycle in terms of PC1 and PC2

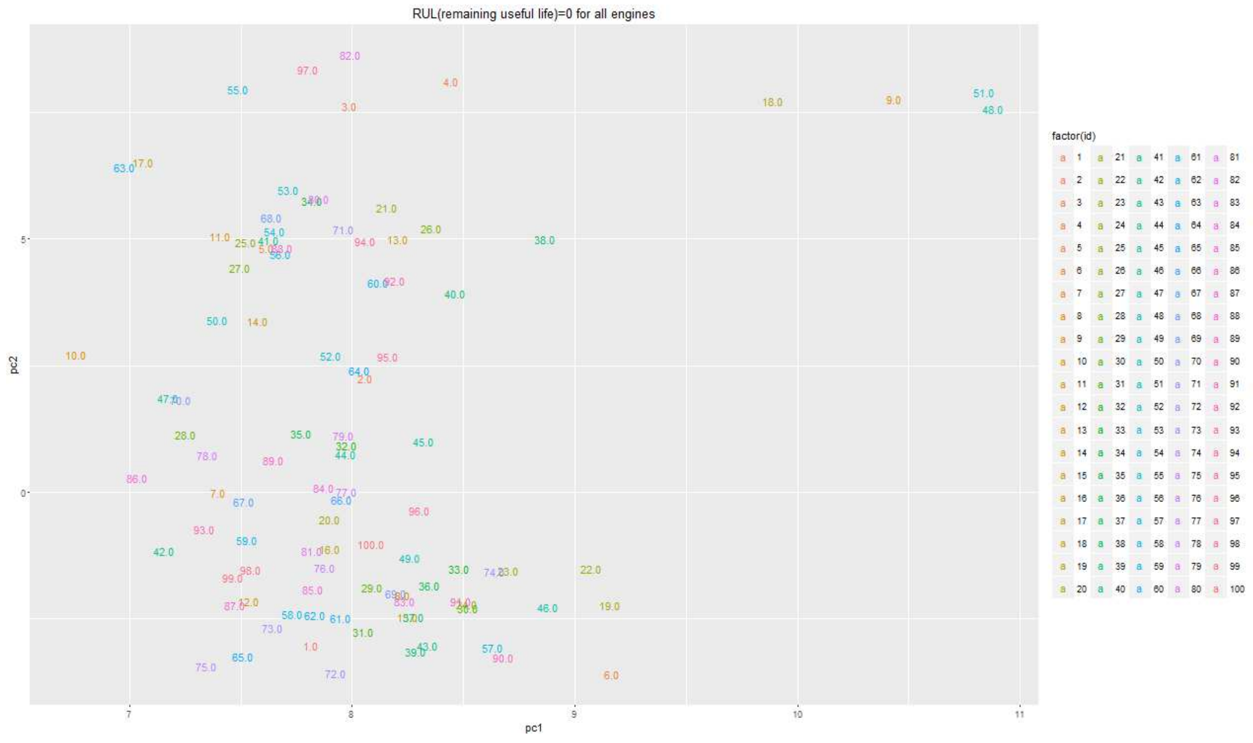
```
ggplot(subset(e, e$id < 4)) + aes(x = pc1, y = pc2, label = rul, color = factor(id)) +
    geom_text() + ggtitle("RUL (remaining useful life) for Engines 1, 2 and 3")  #plot first 3 engines using PC1 a
nd PC2
```



```
ggplot(subset(e, e$id < 4 & e$rul < 15)) + aes(x = pc1, y = pc2, label = rul, color = factor(id)) +
    geom_text() + ggtitle("RUL(remaining useful life) for Engines 1, 2 and 3")  #zoom in where rul<15
```



```
e$idrul = paste(sep = ".", e$id, e$rul)
library(ggrepel)
ggplot(subset(e, e$rul == 0)) + aes(x = pc1, y = pc2, label = idrul, color = factor(id)) +
    geom_text() + ggtitle("RUL(remaining useful life)=0 for all engines")
```

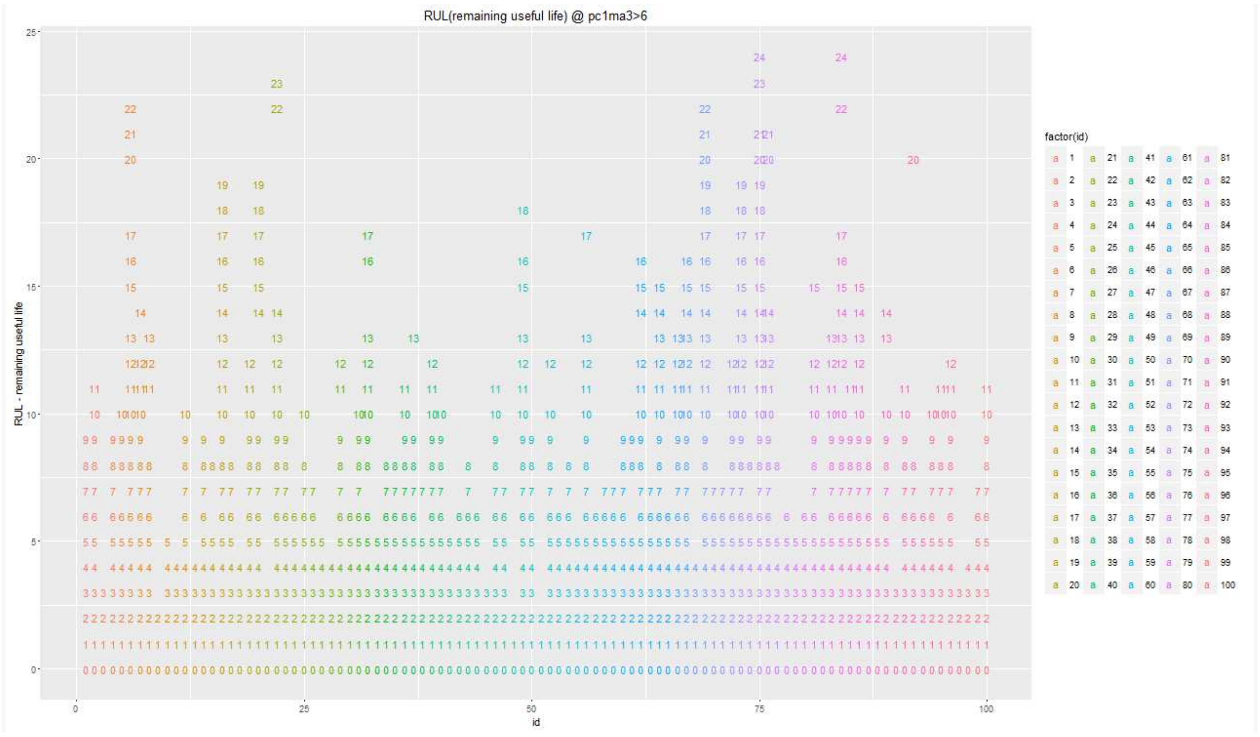


```
library(zoo)
```

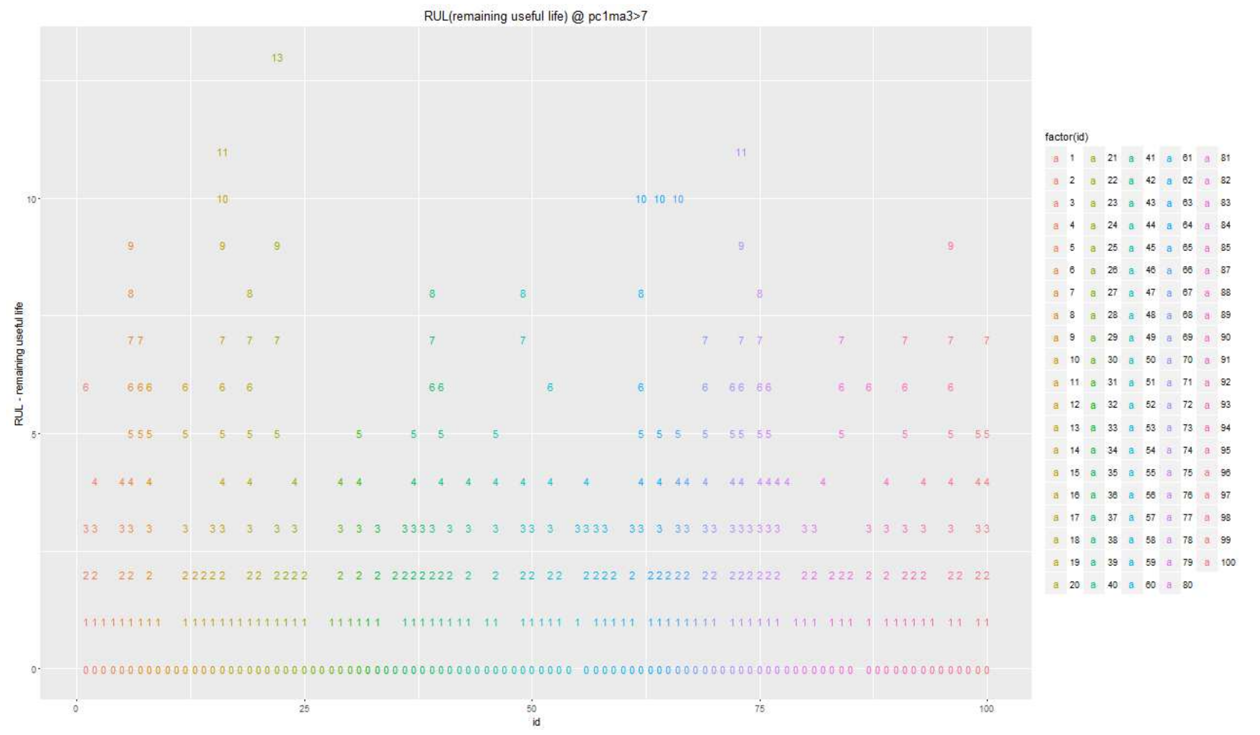
```
##
## Attaching package: 'zoo'
```

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

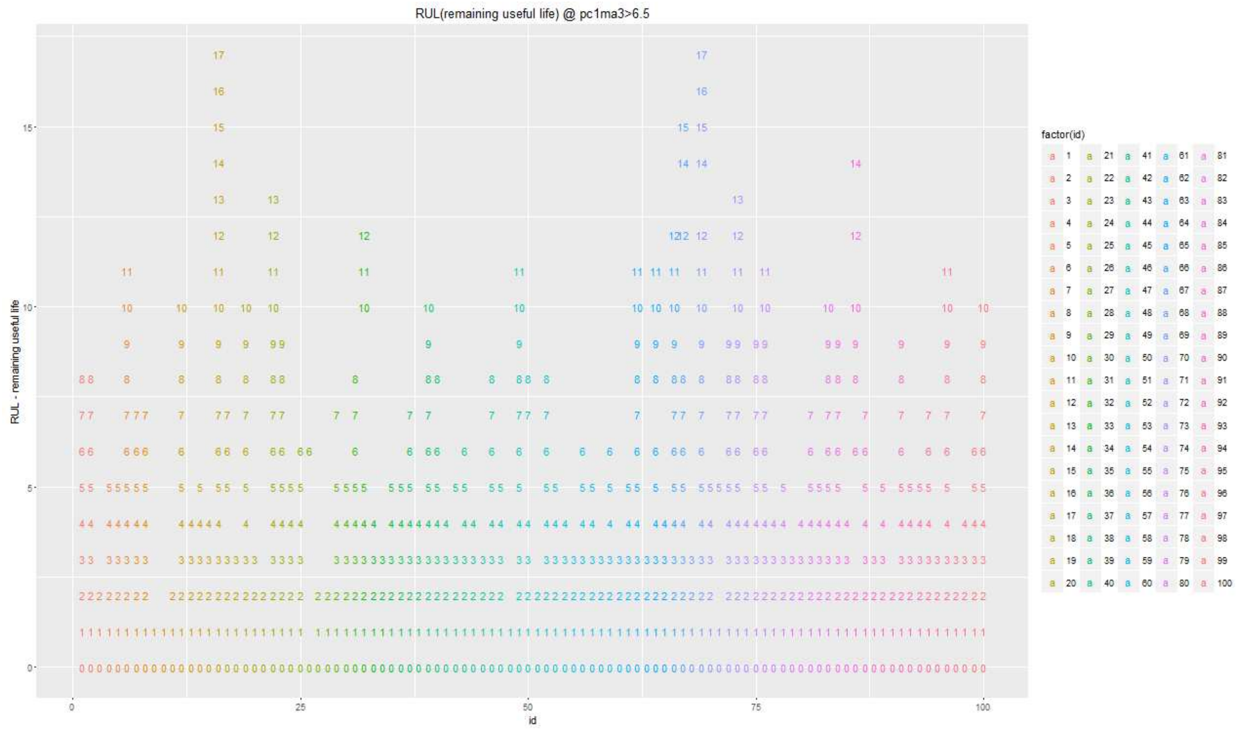
```
e$pc1ma3 = ave(e$pc1, e$id, FUN = function(x) rollmean(x, k = 3, na.pad = T, align = "right"))  #use a 3 moving av
erage instead of single value

ggplot(subset(e, e$pc1ma3 > 6)) + aes(x = id, y = rul, label = rul, color = factor(id)) +
    geom_text() + ylab("RUL - remaining useful life")
```



```
ggplot(subset(e, e$pc1ma3 > 7)) + aes(x = id, y = rul, label = rul, color = factor(id)) +
    geom_text() + ylab("RUL - remaining useful life")
```



```
ggplot(subset(e, e$pc1ma3 > 6.5)) + aes(x = id, y = rul, label = rul, color = factor(id)) +
    geom_text() + ylab("RUL - remaining useful life")
```



```
tapply(subset(e, e$pc1ma3 > 6.5)$rul, subset(e, e$pc1ma3 > 6.5)$id, max)
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   8   8   2   5   5  11   7   7   1   1   2  10   4   5   4  17   7   3
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  10   3   2  13   9   5   6   6   2   2   7   5   8  12   4   3   5   5
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   7   4  10   8   4   5   6   4   3   8   5   1  11   8   2   8   5   4
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   3   6   5   3   6   3   5  11   3  11   4  12  15   3  17   5   5   9
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  13   4   9  11   4   5   3   4   7   5  10   9   4  14   7   3   5   2
##  91  92  93  94  95  96  97  98  99 100
##   9   5   5   7   3  11   3   4   6  10
```

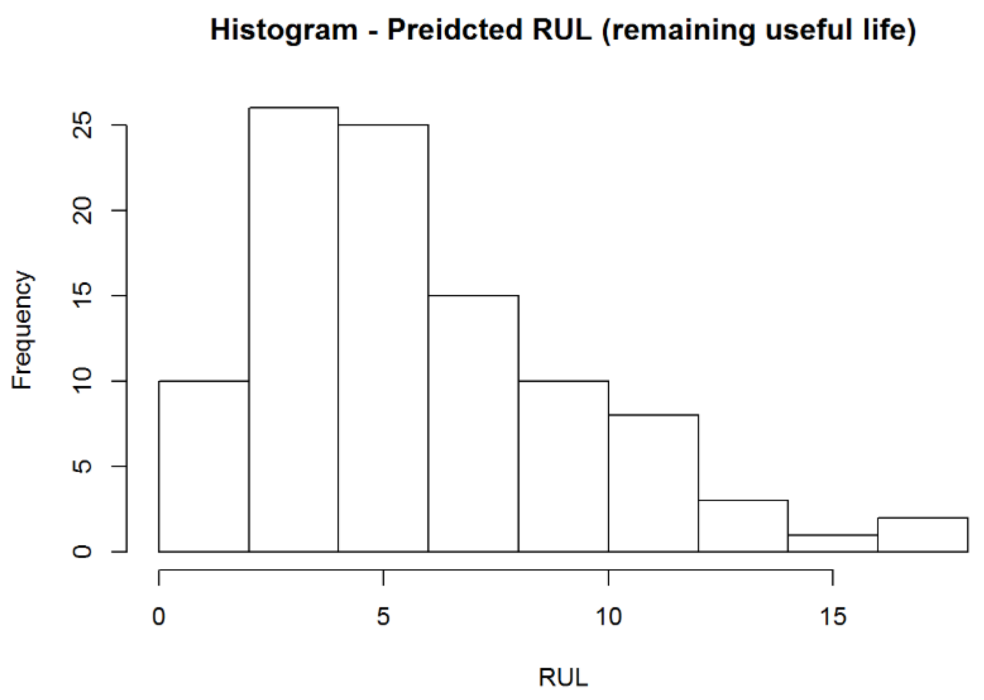
```
summary(tapply(subset(e, e$pc1ma3 > 6.5)$rul, subset(e, e$pc1ma3 > 6.5)$id, max))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00    4.00    5.00    6.25    8.00   17.00
```

```
sd(tapply(subset(e, e$pc1ma3 > 6.5)$rul, subset(e, e$pc1ma3 > 6.5)$id, max))
```

```
## [1] 3.534462
```

```
hist(tapply(subset(e, e$pc1ma3 > 6.5)$rul, subset(e, e$pc1ma3 > 6.5)$id, max), main = "Histogram - Preidcted RUL
(remaining useful life)",
    xlab = "RUL")
```



Predictive Equation

```
# if pc1ma3>6.5, predict FAILING i.e. will fail in the next 1 to 15 cycles; but a
# few may last longer
```

```
pc1eq = ""  #get pc1 equation
for (j in 1:17) {
    pc1eq = cat(sep = "", pc1eq, "+(", "'", names(e.odd.pca$center[j]), "'", "-(",
        e.odd.pca$center[j], ")", ")/", e.odd.pca$scale[j], "*", e.odd.pca$rotation[j,
            1])
}
```

```
## +('s11'-(47.51488))/0.2701003*0.3090913+('s12'-(521.4901))/0.7517117*-0.3049236+('s13'-(2388.09))/0.07484883*0.
2845465+('s14'-(8143.502))/19.7965*0.04163657+('s15'-(8.438634))/0.03782789*0.2868222+('s17'-(393.0714))/1.561964*
0.2685557+('s2'-(642.638))/0.5043607*0.2734667+('s20'-(38.83337))/0.1812555*-0.2819219+('s21'-(23.29963))/0.108387
2*-0.2834525+('s3'-(1590.048))/6.186916*0.2604444+('s4'-(1408.104))/9.077463*0.3006121+('s6'-(21.60976))/0.0015392
59*0.06360376+('s7'-(553.4522))/0.8983562*-0.2995252+('s8'-(2388.091))/0.07388822*0.2847322+('s9'-(9064.651))/22.7
2082*0.08204075+('setting1'-(-3.554925e-05))/0.002184843*0.003580013+('setting2'-(5.022518e-06))/0.0002931999*0.00
3136759
```

```
pc2eq = ""  #get pc1 equation
for (j in 1:17) {
    pc2eq = cat(sep = "", pc2eq, "+(", "'", names(e.odd.pca$center[j]), "'", "-(",
        e.odd.pca$center[j], ")", ")/", e.odd.pca$scale[j], "*", e.odd.pca$rotation[j,
            2])
}
```

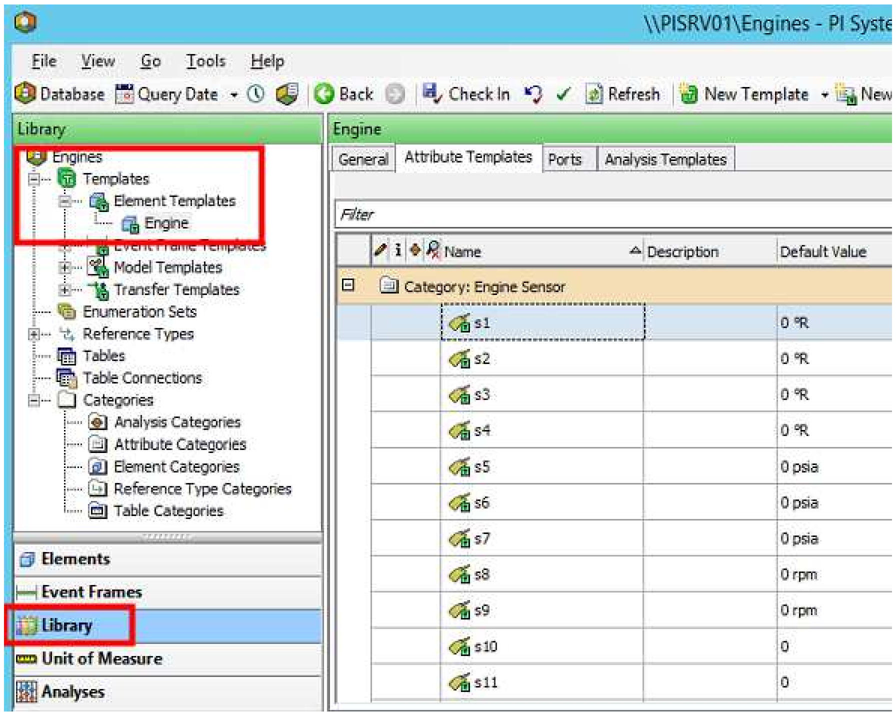
```
## +('s11'-(47.51488))/0.2701003*-0.005176845+('s12'-(521.4901))/0.7517117*0.05515457+('s13'-(2388.09))/0.07484883
*-0.2340729+('s14'-(8143.502))/19.7965*0.6669612+('s15'-(8.438634))/0.03782789*0.02955775+('s17'-(393.0714))/1.561
964*0.0884907+('s2'-(642.638))/0.5043607*0.02942649+('s20'-(38.83337))/0.1812555*-0.02673732+('s21'-(23.29963))/0.
1083872*-0.03537639+('s3'-(1590.048))/6.186916*0.08201158+('s4'-(1408.104))/9.077463*0.02025907+('s6'-(21.60976))/
0.001539259*-0.04061592+('s7'-(553.4522))/0.8983562*0.04585588+('s8'-(2388.091))/0.07388822*-0.2283776+('s9'-(906
4.651))/22.72082*0.6501037+('setting1'-(-3.554925e-05))/0.002184843*-0.004670446+('setting2'-(5.022518e-06))/0.000
2931999*-0.00809504
```

Part III – Ready the Prediction for Deployment using PI Analytics

With the model trained in R, we have developed the equations that will allow us to predict the failures before they occur based on the engine sensor and settings data. These can be expressed as arithmetic equations and configured into the engine template in AF. We can then run these calculations in AF using PI Analytics. By doing so, we are able to run the model in real-time, using any newly observed sensor and settings values to warn us when a failure is anticipated and prevent unexpected shutdowns.

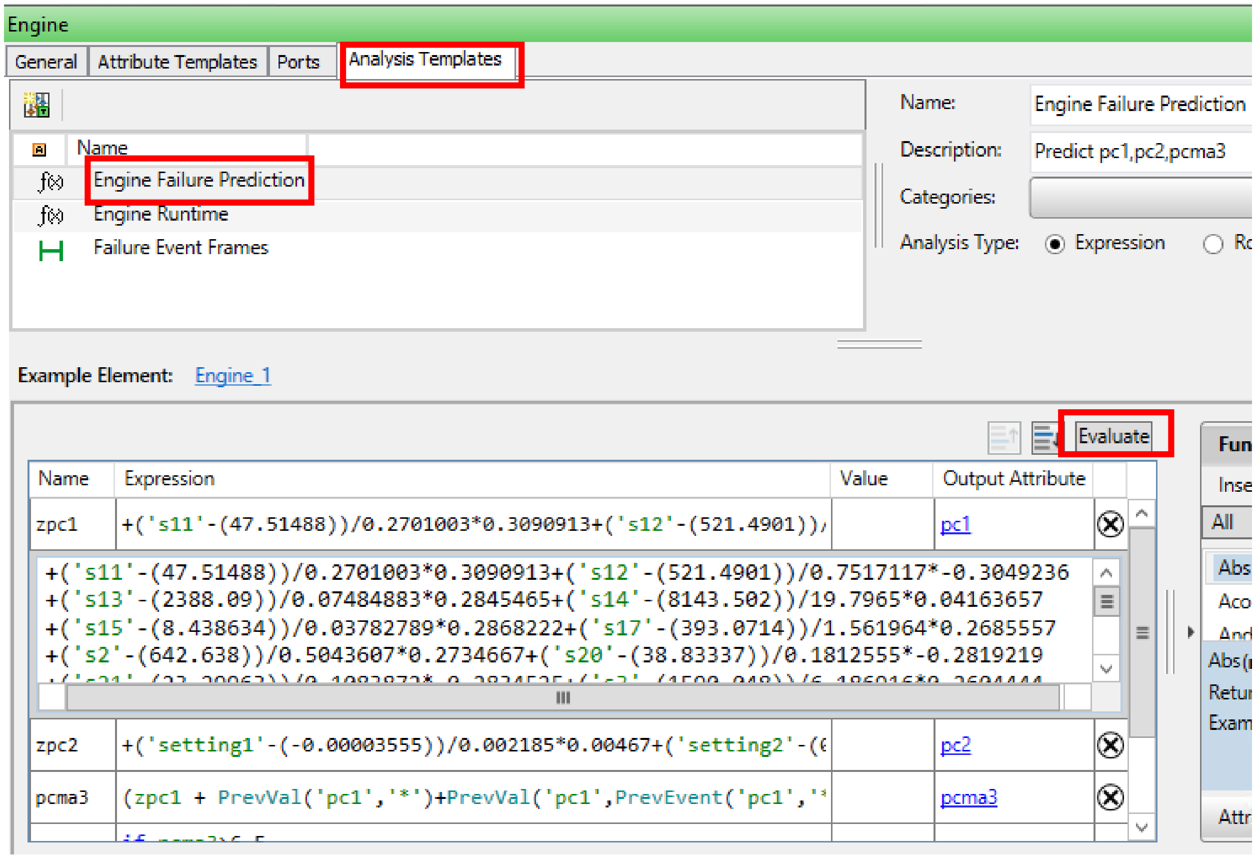
In this part of the lab, we will be testing our predictive equations against the data we have already loaded in our PI system. Remember, the analysis we did using R only involved 50 of the 100 engines in our sample dataset. We have still not seen how well things will work with the remaining 50 engines. To perform this test, we will use the backfill feature of PI Analytics to see if our equations will accurately predict an engine failure before it actually occurs.

The predictive analytic equations developed in Part II of this lab have already been configured in the Engines AF model. To see them, go into the "Library" section and select the "Engine" template.



Name	Description	Default Value
Category: Engine Sensor		
s1		0 °R
s2		0 °R
s3		0 °R
s4		0 °R
s5		0 psia
s6		0 psia
s7		0 psia
s8		0 rpm
s9		0 rpm
s10		0
s11		0

In the “Analysis Template” tab, and you will see that three analyses have been added to this template. Selecting the “Engine Failure Prediction” analysis will take you to the screen shown below. (Selecting each step of the analysis calculation will expand its view showing more detail.



Engine

General | Attribute Templates | Ports | Analysis Templates

Name
Engine Failure Prediction
Engine Runtime
Failure Event Frames

Name: Engine Failure Prediction
Description: Predict pc1,pc2,pcma3
Categories:
Analysis Type: [✓ Expression] Ro

Example Element: Engine_1

Evaluate

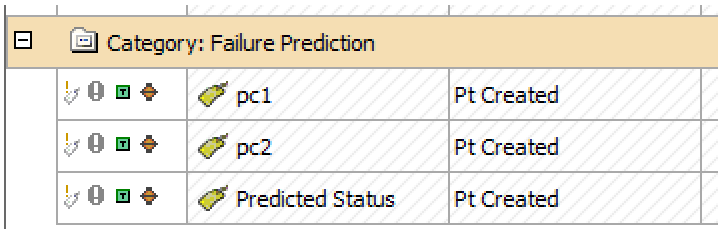
Name	Expression	Value	Output Attribute
zpc1	+('s11'-(47.51488))/0.2701003*0.3090913+('s12'-(521.4901))/		pc1
zpc2	+('setting1'-(-0.00003555))/0.002185*0.00467+('setting2'-(		pc2
pcma3	(zpc1 + PrevVal('pc1','*')+PrevVal('pc1',PrevEvent('pc1','		pcma3

```
+('s11'-(47.51488))/0.2701003*0.3090913+('s12'-(521.4901))/0.7517117*-0.3049236
+('s13'-(2388.09))/0.07484883*0.2845465+('s14'-(8143.502))/19.7965*0.04163657
+('s15'-(8.438634))/0.03782789*0.2868222+('s17'-(393.0714))/1.561964*0.2685557
+('s2'-(642.638))/0.5043607*0.2734667+('s20'-(38.83337))/0.1812555*-0.2819219
```

The failure prediction has three steps;

1. Calculate of principal component 1, zpc1, using the equation derived from the R script in Part II of the lab.
2. Calculate pcma3 - the three minute moving average of principal component 1,.
3. If the moving average is above the threshold value of 6.5, set the predicted status tag to a digital state value of “Will Fail”. Otherwise, set the predicted status value to “Ok”.

In the “Library” section, we are viewing the Engine template. The “Example Element” has been set to “Engine_1”. Clicking on the Evaluate button will perform the analytic using data from the Engine_1 element. If you would like to evaluate the predictive analytic for other elements, navigate back to the “Elements” section of PI System Explorer and choose the “Analysis” tab from any engine element.

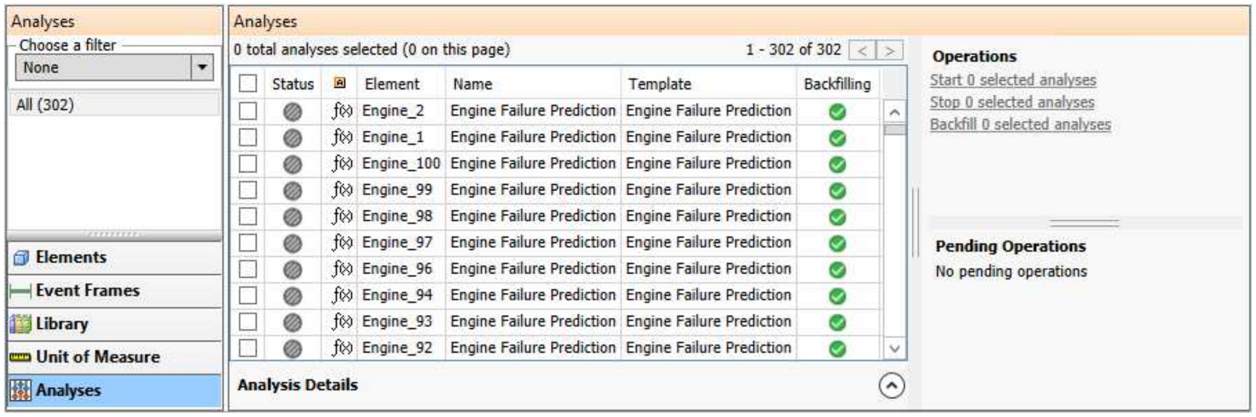


Category: Failure Prediction

pc1	Pt Created
pc2	Pt Created
Predicted Status	Pt Created

Now that we have reviewed the predictive calculation in PI Analytics and have evaluated it, we can now backfill the calculation for all 100 engines to see how well it works. Again, the data for 50 of these engines was not included in our R analysis, so we need to verify that our methodology will work.

It would be tedious to manage the analytics for 100 engines one-by-one so we will use the features of the “Analyses” section of PI System Explorer to make this much easier. Select **Analyses** from the menu in the bottom left-hand corner of the PI System Explorer to get to the screen below.



Analyses

0 total analyses selected (0 on this page) 1 - 302 of 302

☐	Status	Element	Name	Template	Backfilling
☐	●	Engine_2	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_1	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_100	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_99	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_98	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_97	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_96	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_94	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_93	Engine Failure Prediction	Engine Failure Prediction	✓
☐	●	Engine_92	Engine Failure Prediction	Engine Failure Prediction	✓

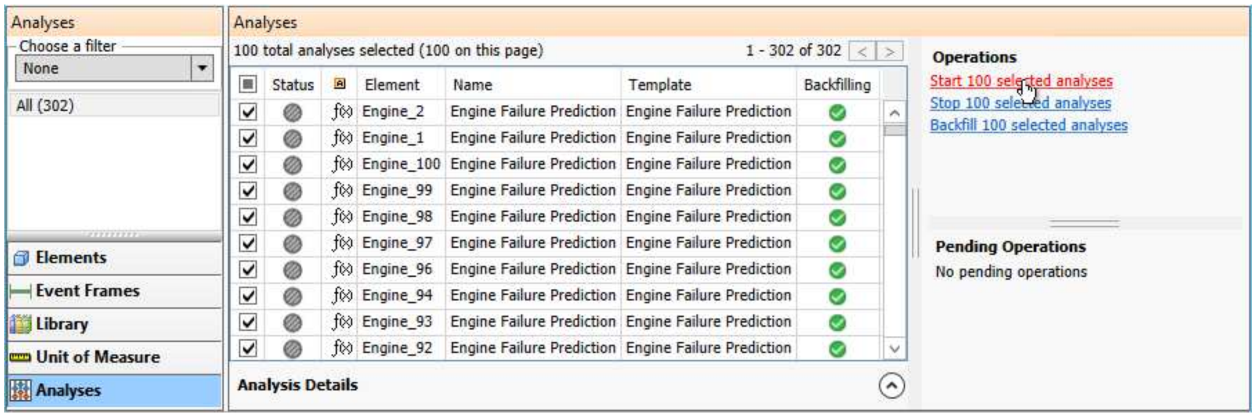
Analysis Details

Operations
Start 0 selected analyses
Stop 0 selected analyses
Backfill 0 selected analyses

Pending Operations
No pending operations

The Analysis view contains a complete tabular list of all the analytics configured for the Engines AF model. From here, we can start, stop, or backfill selected groups of analytics.

To start the Engine Failure Prediction, select the checkbox to the left of the “Status” column – this will select all the 100 engines. Then click the “Start” link on the right. The PI Analytics service will Start them all and you will see a green icon for each analytic in the “Status” column.



Analyses

100 total analyses selected (100 on this page) 1 - 302 of 302

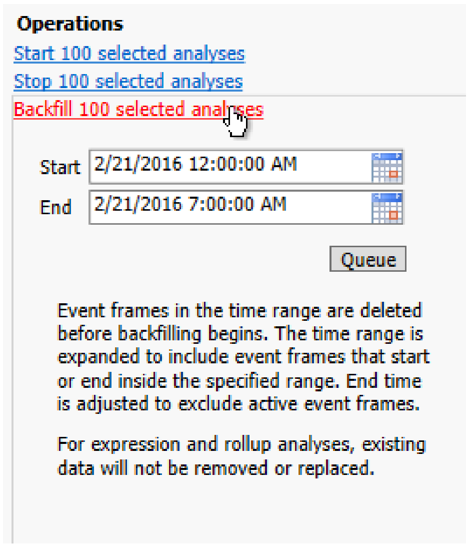
☐	Status	Element	Name	Template	Backfilling
✓	●	Engine_2	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_1	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_100	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_99	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_98	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_97	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_96	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_94	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_93	Engine Failure Prediction	Engine Failure Prediction	✓
✓	●	Engine_92	Engine Failure Prediction	Engine Failure Prediction	✓

Analysis Details

Operations
Start 100 selected analyses
Stop 100 selected analyses
Backfill 100 selected analyses

Pending Operations
No pending operations

Once the analytics are all running, click the "Backfill" link to open the time range dialog. Set the tie range to 21-Feb-2016 12:00 am to 21-Feb-2016 7:00 am. Click **Queue** to begin the backfilling process. You will see progress bars appear for each analytic as the PI Analytics Service performs the calculation over the prescribed time range.


Operations

Start 100 selected analyses

Stop 100 selected analyses

Backfill 100 selected analyses

Start 2/21/2016 12:00:00 AM

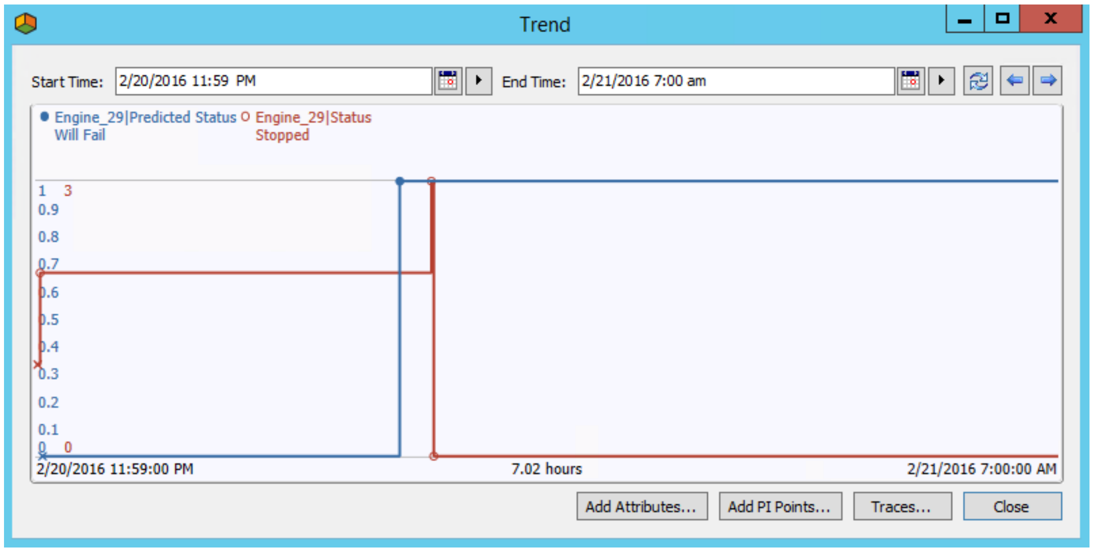
End 2/21/2016 7:00:00 AM

Queue

Event frames in the time range are deleted before backfilling begins. The time range is expanded to include event frames that start or end inside the specified range. End time is adjusted to exclude active event frames.

For expression and rollup analyses, existing data will not be removed or replaced.

Now, you can return to the "Elements" section of PI System Explorer to look at the results. Use the Explorer's trending tool to examine the data. Select any engine and from its "Attributes" tab right-click on the "Predicted Status" and **Trend** it. Close the trend window and right-click on the "Status" attribute and **Add to Trend** it. For Engine 29 you will see the trend below. Taking a closer look, you can see that our prediction comes a few minutes before the engine actually fails. It works!


Start Time: 2/20/2016 11:59 PM

End Time: 2/21/2016 7:00 am

● Engine_29|Predicted Status Will Fail

○ Engine_29|Status Stopped

You can confirm it by checking a few other engines in the above manner.

Appendix A: Using Power BI to Evaluate the Predictive Model

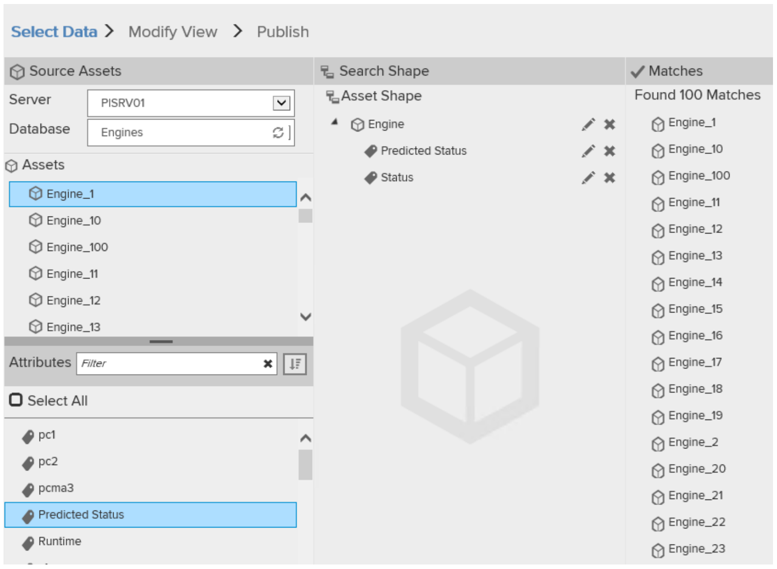
After we have configured and backfilled the engine failure prediction using PI Analytics, we would like to see if our predictive model will be effective in warning us of potential engine failure. We would like to know how far in advance the predictive model first warns us of an eventual failure. And we would like to do this for all 100 engines in an easy way.

To accomplish this, we will use the PI Integrator for BA to create another text file containing the "Status" and "Predicted Status" attributes for all 100 engines over the entire time range. The text file we create can be imported and quickly analyzed in a tool such as Microsoft Power BI Desktop.

Return to Internet Explorer and access the PI Integrator for BA User Interface. Create a new Asset View and give it a meaningful name like, "Failure Prediction Test". Select data from the "Engines" AF Model using the steps listed below,

- Select "Engines" from the **Database** dropdown.
- Select and drag element "Engine_1" to the "Asset Shape" area.
- Select and drag attribute "Predicted Status" and drop it **as a child** to the "Engine_1" element in the "Shape Asset" area (position the cursor on top of "Engine_1" and observe the tooltip).
- Repeat the step above for attribute "Status".
- Click the edit pencil next to element "Engine_1" in the "Asset Shape" area. Uncheck the filter on "Asset Name" and check the filter on "Asset Template". This will expand our selection to include all 100 engines which have been derived from the "Engine" template in our AF database.

Once you complete these steps, the PI Integrator for BA User Interface should look like the one shown here.



Server: PISRV01
Database: Engines

Assets: Engine_1, Engine_10, Engine_100, Engine_11, Engine_12, Engine_13

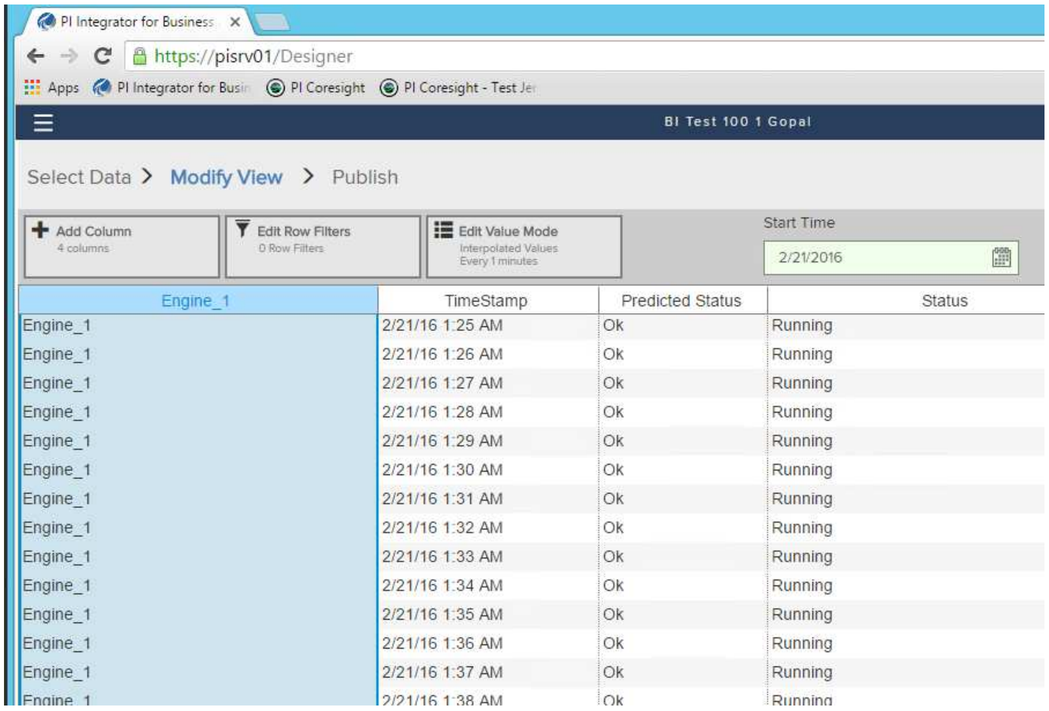
Attributes: pc1, pc2, pcma3, Predicted Status, Runtime

Asset Shape:
- Engine
  - Predicted Status
  - Status

Found 100 Matches: Engine_1, Engine_10, Engine_100, Engine_11, Engine_12, Engine_13, Engine_14, Engine_15, Engine_16, Engine_17, Engine_18, Engine_19, Engine_2, Engine_20, Engine_21, Engine_22, Engine_23

With the data selection step complete, click **Next** in the upper right-hand corner of the page to move to the view modification step. This time, we're not going to remove the engine "Stopped" events, instead we will publish all the data for the entire time range. We'll let Power BI Desktop provide the data filtering functionality.

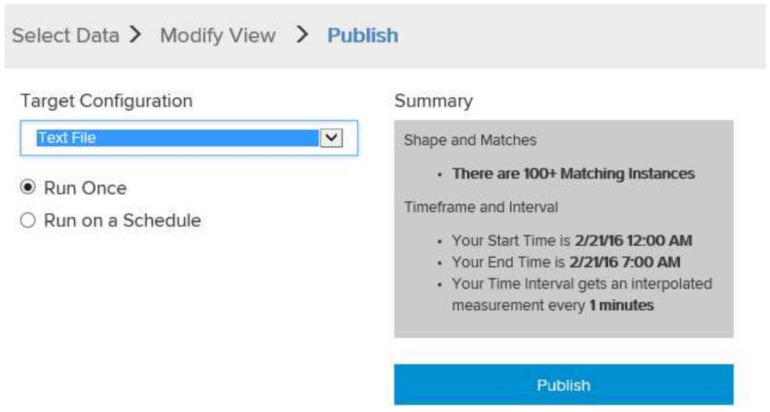
On the "Modify View" page, set the time period for the publication to be between 2/21/16 12:00 AM to 2/21/16 7:00:00 AM and click **Apply**. You should see the results shown below.



Engine_1	TimeStamp	Predicted Status	Status
Engine_1	2/21/16 1:25 AM	Ok	Running
Engine_1	2/21/16 1:26 AM	Ok	Running
Engine_1	2/21/16 1:27 AM	Ok	Running
Engine_1	2/21/16 1:28 AM	Ok	Running
Engine_1	2/21/16 1:29 AM	Ok	Running
Engine_1	2/21/16 1:30 AM	Ok	Running
Engine_1	2/21/16 1:31 AM	Ok	Running
Engine_1	2/21/16 1:32 AM	Ok	Running
Engine_1	2/21/16 1:33 AM	Ok	Running
Engine_1	2/21/16 1:34 AM	Ok	Running
Engine_1	2/21/16 1:35 AM	Ok	Running
Engine_1	2/21/16 1:36 AM	Ok	Running
Engine_1	2/21/16 1:37 AM	Ok	Running
Engine_1	2/21/16 1:38 AM	Ok	Running

With the data selection step complete, click **Next** in the upper right-hand corner of the page to move to the publication step. For **Target Configuration**, click on the drop down and select **Text File**.

Since we only need to perform a one-time publish of this view, select **Run Once**.



Target Configuration: Text File

[✓ Run Once] Run on a Schedule

Summary

Shape and Matches

- There are 100+ Matching Instances

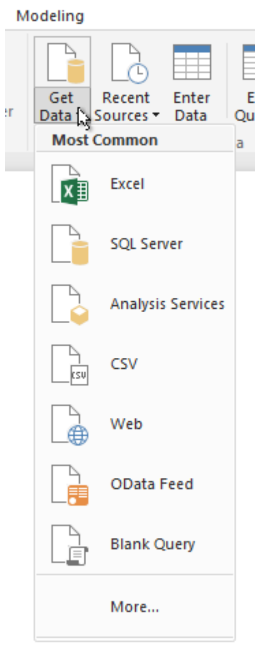
Timeframe and Interval

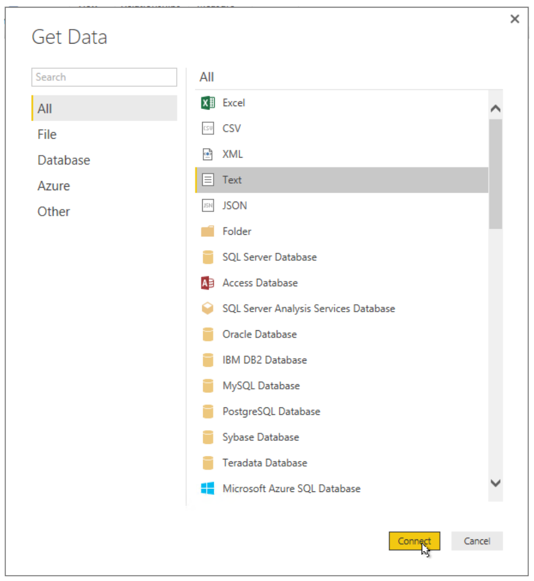
- Your Start Time is 2/21/16 12:00 AM
- Your End Time is 2/21/16 7:00 AM
- Your Time Interval gets an interpolated measurement every 1 minutes

Click on **Publish** to get the PI Integrator to start publishing the dataset to a file. You will have to **Confirm** that this is Ok. Once the publication is complete, you will find the text file has been created in "C:\Data Science Lab" folder.

For the analysis, we will be using Microsoft Power BI Desktop. To open it, click Power BI Desktop

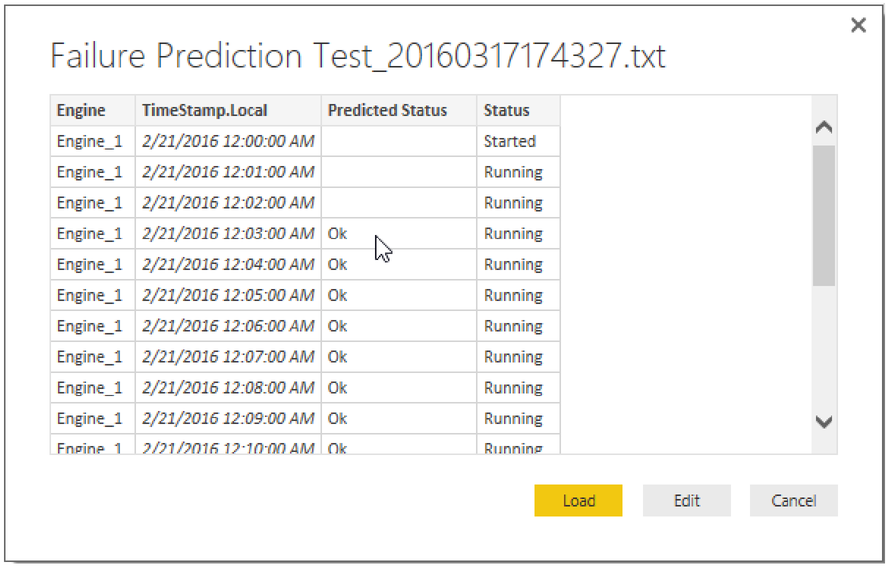
The first step is to import the text file we just created into Power BI Desktop. From the top menu ribbon, click on "Get Data" to expose the dropdown shown at right below. Select "More…" at the bottom of the list. This will open the "Get Data" dialog where you can chose "Text" as you data source. Click **Connect** at the bottom to open the File Open window. Navigate to the "C:\Data Science Lab" folder and select the file you published.





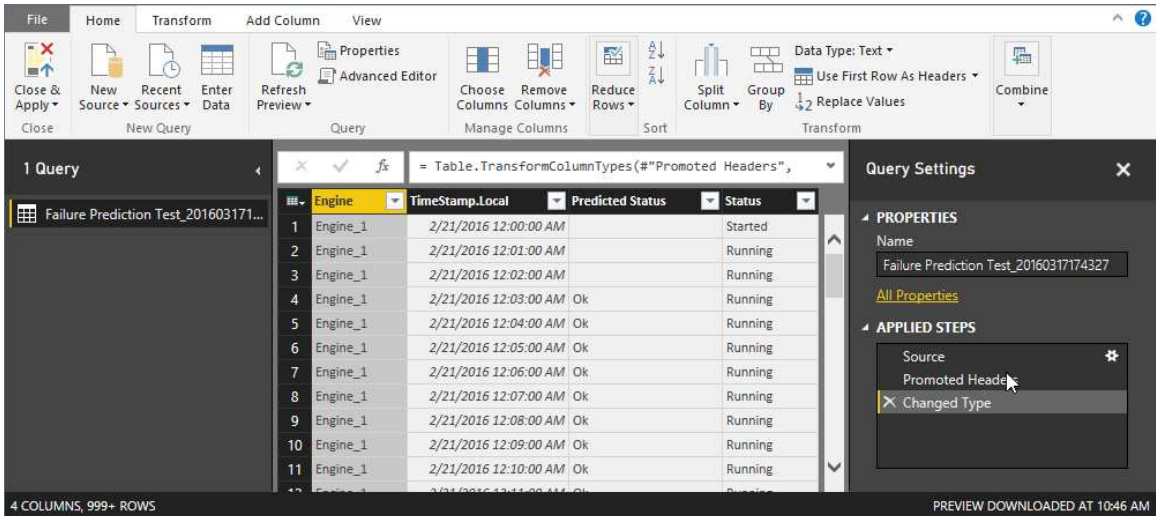
Once you open the text file, Power BI will present the data preview window.

We would like to remove the blank values shown under "Predicted Status" and also the timestamp column. To edit the dataset before we import it we'll click on **Edit.** **(**Clicking **Load** will bring the data directly in Power BI Desktop.)


Failure Prediction Test_20160317174327.txt

Engine	TimeStamp.Local	Predicted Status	Status
Engine_1	*2/21/2016 12:00:00 AM*		Started
Engine_1	*2/21/2016 12:01:00 AM*		Running
Engine_1	*2/21/2016 12:02:00 AM*		Running
Engine_1	*2/21/2016 12:03:00 AM*	Ok	Running
Engine_1	*2/21/2016 12:04:00 AM*	Ok	Running
Engine_1	*2/21/2016 12:05:00 AM*	Ok	Running
Engine_1	*2/21/2016 12:06:00 AM*	Ok	Running
Engine_1	*2/21/2016 12:07:00 AM*	Ok	Running
Engine_1	*2/21/2016 12:08:00 AM*	Ok	Running
Engine_1	*2/21/2016 12:09:00 AM*	Ok	Running
Engine_1	*2/21/2016 12:10:00 AM*	Ok	Running

Once we click **Edit**, the Power BI Desktop Query Editor window is shown below.



```
= Table.TransformColumnTypes(#"Promoted Headers",
```

Name: Failure Prediction Test_20160317174327

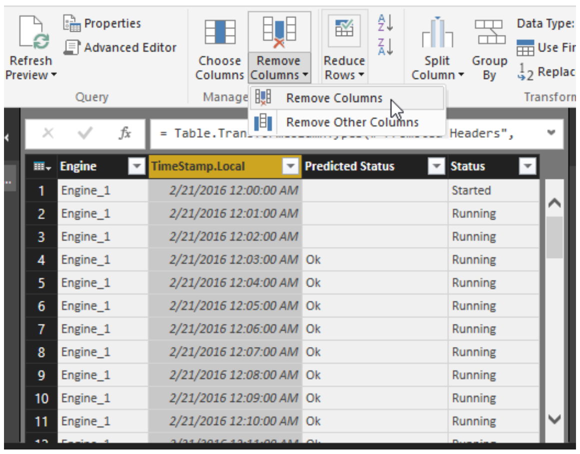
	Engine	TimeStamp.Local	Predicted Status	Status
1	Engine_1	*2/21/2016 12:00:00 AM*		Started
2	Engine_1	*2/21/2016 12:01:00 AM*		Running
3	Engine_1	*2/21/2016 12:02:00 AM*		Running
4	Engine_1	*2/21/2016 12:03:00 AM*	Ok	Running
5	Engine_1	*2/21/2016 12:04:00 AM*	Ok	Running
6	Engine_1	*2/21/2016 12:05:00 AM*	Ok	Running
7	Engine_1	*2/21/2016 12:06:00 AM*	Ok	Running
8	Engine_1	*2/21/2016 12:07:00 AM*	Ok	Running
9	Engine_1	*2/21/2016 12:08:00 AM*	Ok	Running
10	Engine_1	*2/21/2016 12:09:00 AM*	Ok	Running
11	Engine_1	*2/21/2016 12:10:00 AM*	Ok	Running

4 COLUMNS, 999+ ROWS

PREVIEW DOWNLOADED AT 10:46 AM

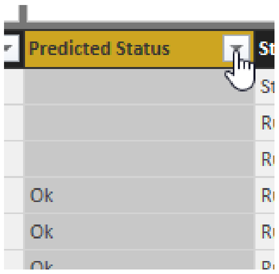
Select the timestamp column and click the “Remove Columns” item in the menu ribbon. Choose “Remove Columns” to take this colum out of our dataset.

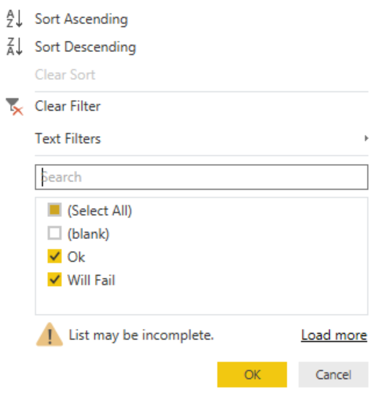
Note: This action does not affect the text file, it only instructs Power BI to not import this column from the text file.



	Engine	TimeStamp.Local	Predicted Status	Status
1	Engine_1	2/21/2016 12:00:00 AM		Started
2	Engine_1	2/21/2016 12:01:00 AM		Running
3	Engine_1	2/21/2016 12:02:00 AM		Running
4	Engine_1	2/21/2016 12:03:00 AM	Ok	Running
5	Engine_1	2/21/2016 12:04:00 AM	Ok	Running
6	Engine_1	2/21/2016 12:05:00 AM	Ok	Running
7	Engine_1	2/21/2016 12:06:00 AM	Ok	Running
8	Engine_1	2/21/2016 12:07:00 AM	Ok	Running
9	Engine_1	2/21/2016 12:08:00 AM	Ok	Running
10	Engine_1	2/21/2016 12:09:00 AM	Ok	Running
11	Engine_1	2/21/2016 12:10:00 AM	Ok	Running

Select the “Predicted Status” column and click on the down arrow to expose the filter list shown below. Uncheck the “(blank)” item to remove rows containing blank values of the Predicted Status.

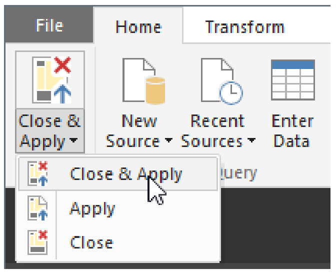




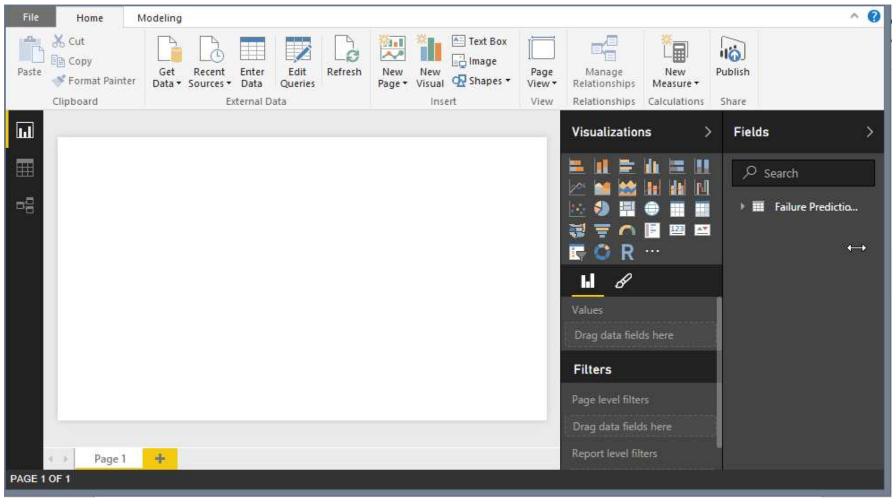
Note: We are getting two blank values for the Predicted Status because our calculation fails for the first two minutes of the backfill time range since we are using a three minute moving average and there are missing values until we get to the third minute of the calculation.

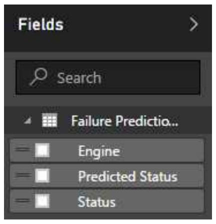
Referring back the screenshot at the top of this page, you will see that as you perform the steps above, they are recorded by the Query Editor in the “Applied Steps” window.

We are now ready to import the edited dataset into Power BI Desktop. Select “Close & Apply” from the menu ribbon and select “Close & Apply”. Power BI will now import the dataset from the text file, applying the filtering steps you have specified.



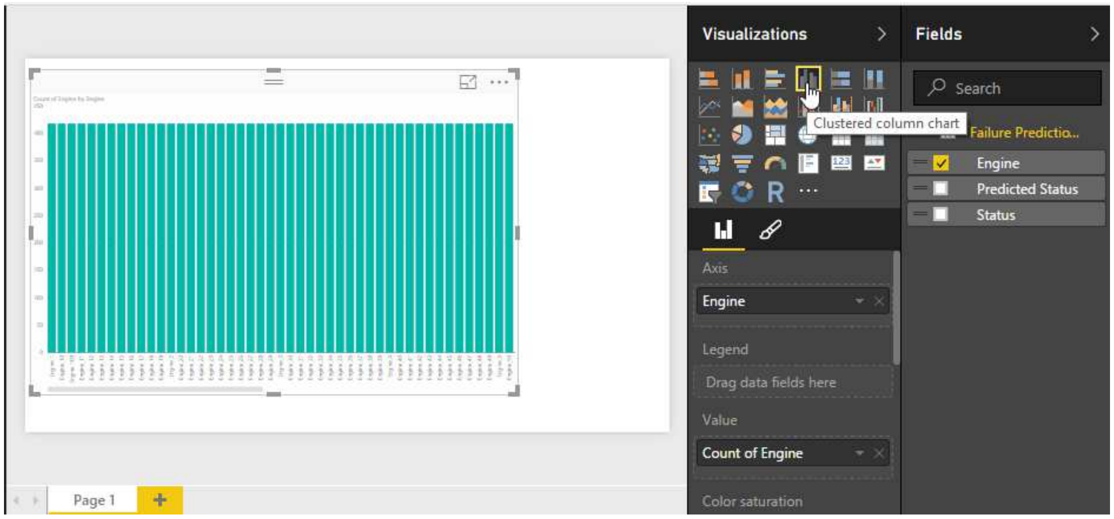
Once the data has been loaded, you will be returned to the Power BI Desktop reporting canvas shown below.





With the data loaded, we're just a few simple steps to finishing our analysis. In the "Fields" area at the right, expand the table to see the columns in you dataset.

From the "Visualization" area, select the **Clustered Column Chart**. To configure the chart, select, drag, and drop the "Engine" column into the "Axis" field. Repeat this by placing the "Engine" column in the "Values" field. You will notice that since this column contains text, the engine number, Power BI converts this to a record count.



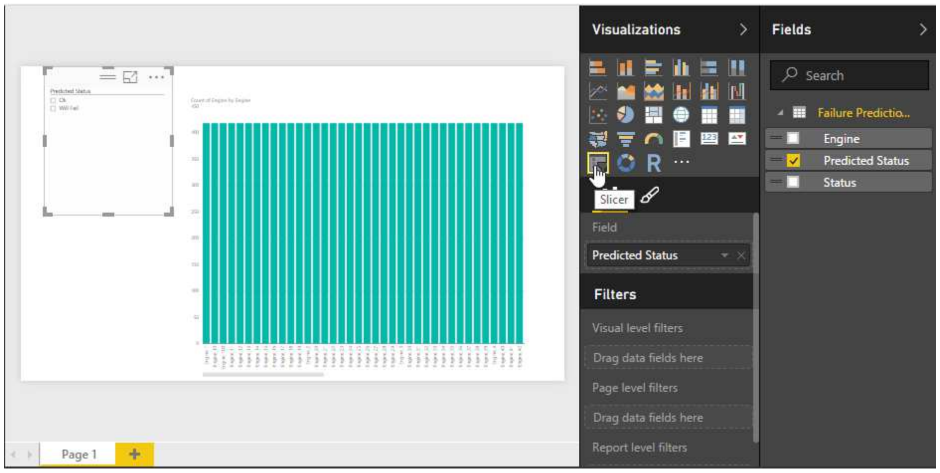
Taking a look at things, we see that we have a bar chart that shows the number of records for each engine in our dataset. As expected, since we did not filter the publication to exclude times when the engine was "Stopped", there are the same number for each engine. Not too interesting.

However, recognizing that each record represents one minute of operation, we can use some slicers, based on the "Status" and "Predicted Status" columns to finish the analysis.

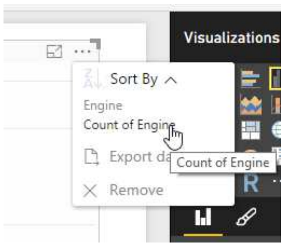
What we are looking for is the amount of time (number of records) where the Predicted Status has a value of "Will Fail" and the "Status" is "Running. Select the bar chart object handles and make some room in the reporting canvas for a couple of slicers.

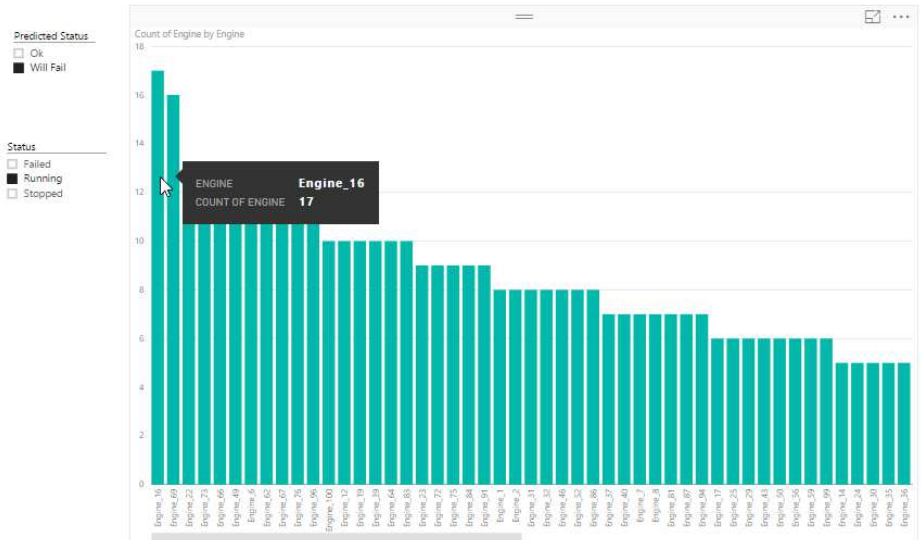
Select the "Predicted Status" column and drag it onto a free space on the canvas. While it's still selected, click on the **Slicer** to convert the list to a slicer visual object.

Create a second slicer for the "Status" column.



Rearranging the size of things and filtering the record counts (minutes) for each engine gives the analysis shown below. In the upper right-hand corner of the bar chart, you can sort the data by "Count of Engine" to make things easier to assess.





You can conclude that for Engine_16, we predicted that it will fail 17 minutes before it actually fails. And, similarly for the other engines.

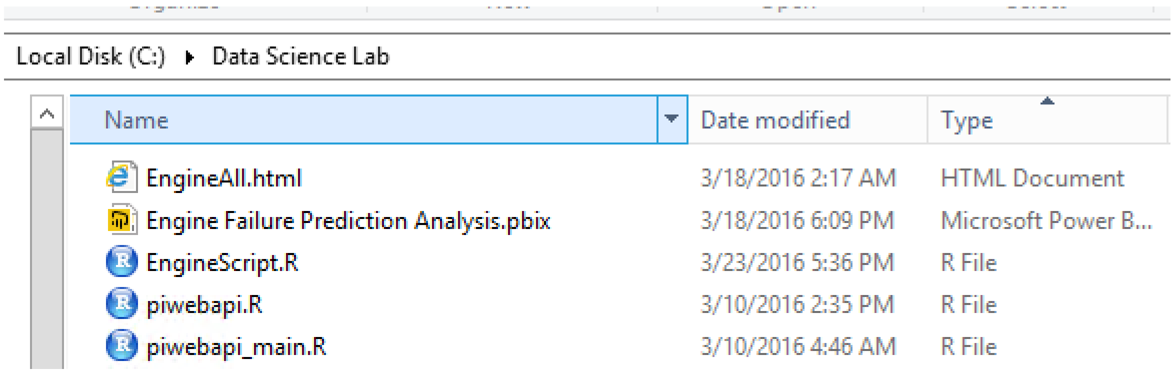
Also, since the bar chart is in descending sort order, we can also conclude that 17 minutes (or cycles) is the maximum.

In the R session (Part II), we had seen similar results.

Appendix B: Using PI WebAPI with R

If you prefer, you can stay completely within the R environment – and read/write PI System data using PI WebAPI, available as part of the PI Developer technologies https://livelibrary.osisoft.com/LiveLibrary/web/ui.xql?action=html&resource=publist_home.html&filter=developer

As such, all the three parts, namely, *Part I Extracting PI System data*, *Part II Predict Equipment Failure Using Statistical Analysis*, and *Part III Ready the Prediction for Deployment* can all be done entirely using R and PI WebAPI.



Local Disk (C:) ▸ Data Science Lab

Name	Date modified	Type
EngineAll.html	3/18/2016 2:17 AM	HTML Document
Engine Failure Prediction Analysis.pbix	3/18/2016 6:09 PM	Microsoft Power B...
EngineScript.R	3/23/2016 5:36 PM	R File
piwebapi.R	3/10/2016 2:35 PM	R File
piwebapi_main.R	3/10/2016 4:46 AM	R File

For more details, please refer to the R code samples in the files with "piwebapi" prefix in their names.

The PI WebAPI approach is particularly useful when the predictive machine learning models are complex – such as those based on decision trees, neural net etc. and hence cannot be deployed using PI AF Analysis.

Appendix C: Follow-up questions from Part II

Follow up questions

How will you improve the prediction? What other criteria besides “threshold for PC1 – three moving average” can you use?

Is the selection for the training data set i.e. engines with odd ids the best technique or can you suggest a better criteria?

In this lab, we use only PC1; how will you incorporate PC2 - will it improve your prediction?

Which variables have the most influence on PC1 (and PC2) - and in what proportion?

What other statistical or machine learning approach can you use to predict engine failure?

Why does the PCA approach work so well in this use case?

Reference Materials

The data used for the lab has been adapted from:

A. Saxena, K. Goebel, D. Simon, and N. Eklund, “Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation”, in the Proceedings of the Ist International Conference on Prognostics and Health Management (PHM08), Denver CO, Oct 2008.