---

title: "B-CDL-PCA-Builder-v01a"

author: 'Author: To\_Be\_Added'

date: "`r format(Sys.time(), '%d&period; %B %Y')`"

output:

html\_document: default

pdf\_document: default

word\_document: default

subtitle: "CDL Industrial Plant Assessment - Part B"

---

<style type="text/css">

body{ /\* Normal \*/

font-size: 12px;

}

td { /\* Table \*/

font-size: 12px;

}

h1.title {

font-size: 38px;

color: DarkRed;

}

h1 { /\* Header 1 \*/

font-size: 28px;

color: DarkBlue;

}

h2 { /\* Header 2 \*/

font-size: 22px;

color: DarkBlue;

}

h3 { /\* Header 3 \*/

font-size: 18px;

font-family: "Times New Roman", Times, serif;

color: DarkBlue;

}

code.r{ /\* Code block \*/

font-size: 12px;

}

pre { /\* Code block - determines code spacing between lines \*/

font-size: 12px;

}

</style>

# Introduction

The \*\*Industrial Plant Assessment\*\* (IPA) has been developed for the continuous processes such as refineries or chemical plants.

It consist of the following phases

A- Selecting and collecting the Plant Analog Inputs (AI), Analog Outputs (AO) Data

B- Partial Component Analysis of AI, AO and Both

C- Oscillation Analysis

D- Controller Performance Analysis

The following folders are used:

N | Folder Name | Description

---|:--------------------------------|:----------------------

1 | IPA-Industrial-Plant-Assessment | Top folder

It contains

N | Folder Name | Description

---|:--------------------------------|:----------------------

1 | IPA-Tools | IPA Tools such as XLSM and RMD

2 | Project... | Project generated by IPA tools

The \*\*IPA-Tools\*\* folder contains

N | Folder Name | Description

---|:--------------------------|:----------------------

1 | A-Work-Process | Step by step documentation and presentations

2 | B-Programs | R And Excel programs

3 | C-References | Academic References

3 | D-Sample-Data | Sample Data to check the program

The data and results of the analysis are stored in \*\*Project Folder\*\*:

N | Folder Name | Description

---|:-----------------------------------|:----------------------

1 | Proj-CDL-T123-Case-A-PC-2023-10-16 | Sample project folder

The project folder is divided into the following sub-folders:

N | Folder Name | Description

---|:---------------------------|:----------------------

1 | A-Raw-Data | Data collected from historian

2 | B-Consolidate-Data | Raw data consolidated into one file

3 | C-PCA-Analysis | Results of PCA Analysis

4 | D-Oscillation-Analysis | Results of Oscillation Analysis

5 | E-Controller-Data | Controllers data

6 | F-Controller-Performance | Controllers Performance Analysis

7 | R-Reports | Report of Analysis Analysis

## Steps of the work process

The following steps are execute:

### Select Tags

Use the documentation and select the Analog-Inputs (AI) and Analog-Outputs (AO)

N | Step | Description

---|:----------------------------------------|:----------------------

1 | Select Tags AI and AO | Get list of tags

2 | Check Historian | Check if Tags are stored very minutes

### Collect Raw Data and consolidate Data

N | Step | Description

---|:----------------------------------------|:----------------------

1 | Collect minute data (3 to 7 days) | Normally collect 100 tags store in A-Raw-Data folder

2 | cleanup raw data | Review and cleanup data

3 | Consolidate raw data | Consolidate all Raw data in one CVS file

### Run PCA-Analysis

N | Step | Description

---|:----------------------------------------|:----------------------

1 | Start RStudio | Platform to run RMD Programs

2 | Open B-CDL-PCA-Builder-v01c.Rmd | Open it from b-Program folder

3 | Click the Knit buttons | generate a HTML report

### Run Oscillation-Analysis

N | Step | Description

---|:-----------------------------------------------|:----------------------

1 | Start RStudio | Platform to run RMD Programs

2 | Open C-CDL-Oscillation-Builder-2023-07-03a.Rmd | Open it from b-Program folder

3 | Click the Knit buttons | generate a HTML report

### Run Control-Performance-Analysis

N | Step | Description

---|:----------------------------------------------|:----------------------

1 | Start RStudio | Platform to run RMD Programs

2 | Open D-CDL-Control-Analysis-2023-07-03a.Rmd | Open it from b-Program folder

3 | Click the Knit buttons | generate a HTML report

<!-- CDL-IPA-B-Part-01 -->

<!-- https://www.dataquest.io/blog/r-markdown-guide-cheatsheet/ -->

<!-- https://www.xquartz.org/ -->

# Import and Setup R

## Imported Setup Data

The external data needed is imported from file \*\*CDM\_Setup.CSV\*\* in the program folder \*\*B-Programs\*\*.

```{r 1.1 setup echo , echo = FALSE}

# +-------------------------------------------------------------------------------------------

# | rm(list = setdiff(ls(), lsf.str())) will remove all variables

# +-------------------------------------------------------------------------------------------

rm(list = setdiff(ls(), lsf.str()))

# +-------------------------------------------------------------------------------------------

# | Set echo\_flag to TRUE if you want to see the code when running KNIT

# +-------------------------------------------------------------------------------------------

echo\_flag = TRUE

warning\_flag = FALSE

```

```{r 1.2 definition , warning= warning\_flag, echo = echo\_flag}

# Chunk output can be customized with knitr options, arguments set in the {} of a chunk header.

# we use five arguments:

#

# include = FALSE

# prevents code and results from appearing in the finished file.

# R Markdown still runs the code in the chunk, and the results can be used by other chunks.

#

# echo = FALSE

# prevents code, but not the results from appearing in the finished file.

# This is a useful way to embed figures.

# message = FALSE

# prevents messages that are generated by code from appearing in the finished file.

# warning = FALSE

# prevents warnings that are generated by code from appearing in the finished.

# fig.cap = "..."

# adds a caption to graphical results.

```

```{r 1.3 - setup, warning= warning\_flag, echo = echo\_flag}

# +-------------------------------------------------------------------------------------------

# | the knitr::opts\_chunk$set(echo = FALSE) in a chunk at the beginning of your document

# | is the same of having {r echo = FALSE} for all chunks.

# +-------------------------------------------------------------------------------------------

knitr::opts\_chunk$set(echo = FALSE)

# +-------------------------------------------------------------------------------------------

# | date\_format <-"%Y-%m-%d %H:%M"

# | where

# | %Y Year

# | %m Month

# | %d Day

# | %H Hour

# | %M Minute

# +-------------------------------------------------------------------------------------------

Sys.setenv(TZ="UTC")

date\_format <-"%Y-%m-%d %H:%M"

# +-------------------------------------------------------------------------------------------

# |start\_time <- Sys.time()

# |start\_time is set to current time like "2023-06-29 17:12:44 EDT"

# +-------------------------------------------------------------------------------------------

start\_time <- Sys.time()

# +-------------------------------------------------------------------------------------------

# |Assign cdl\_engine\_path to the program folder name

# +-------------------------------------------------------------------------------------------

cdl\_engine\_path <- dirname(rstudioapi::getSourceEditorContext()$path)

# +-------------------------------------------------------------------------------------------

# | List all files in the cdl\_engine\_path

# +-------------------------------------------------------------------------------------------

dir(cdl\_engine\_path)

# +-------------------------------------------------------------------------------------------

# | Read CDL\_Setup.CSV file located in the cdl\_engine\_path folder

# | read.csv function that reads a CSV file

# | paste0 function that add string together

# | / path split character for R (Windows is \ )

# | header TRUE means file first row is a header

# +-------------------------------------------------------------------------------------------

sysinf <- Sys.info()

os <- sysinf['sysname']

if (os =="Darwin") {

RCmdData <- read.csv (paste0(cdl\_engine\_path,"/","Setup\_MAC.csv"),header=TRUE)

} else {

RCmdData <- read.csv (paste0(cdl\_engine\_path,"/","Setup\_PC.csv"),header=TRUE)

}

# +-------------------------------------------------------------------------------------------

# | Assign CSV file data to R variables

# | as.vector(RCmdData[row,column])

# |

# +-------------------------------------------------------------------------------------------

proj\_folder <- as.vector(RCmdData[1,3])

company\_name <- as.vector(RCmdData[2,3])

case\_name <- as.vector(RCmdData[3,3])

first\_col\_AI <- as.vector(RCmdData[4,3])

last\_col\_AI <- as.vector(RCmdData[5,3])

first\_col\_AO <- as.vector(RCmdData[6,3])

last\_col\_AO <- as.vector(RCmdData[7,3])

first\_row <- as.vector(RCmdData[8,3])

last\_row <- as.vector(RCmdData[9,3])

n\_acv <- as.vector(RCmdData[10,3])

n\_cluster <- as.vector(RCmdData[11,3])

install\_flag <- as.vector(RCmdData[12,3])

echo\_flag <- as.vector(RCmdData[13,3])

info\_flag <- as.vector(RCmdData[14,3])

data\_file <- as.vector(RCmdData[15,3])

glos\_file <- as.vector(RCmdData[16,3])

info\_file <- as.vector(RCmdData[17,3])

perl <- as.vector(RCmdData[18,3])

# +-------------------------------------------------------------------------------------------

# | The folder name in Windows is separated by character "\" in R it should be changed to "/"

# |

# +-------------------------------------------------------------------------------------------

proj\_folder <- gsub('\\\\', '/', proj\_folder)

```

The following setup data is imported

N | What | value

---|:--------------------|:-----------------

1 | Project Folder | `r proj\_folder`

2 | Company Name | `r company\_name`

3 | Case Name | `r case\_name`

4 | First Column of AI | `r first\_col\_AI`

5 | Last Column of AI | `r last\_col\_AI`

6 | First Column of AO | `r first\_col\_AO`

7 | Last Column of AO | `r last\_col\_AO`

8 | N Auto-Correlation | `r n\_acv`

9 | N of Cluster | `r n\_cluster`

10 | Lib Install Flag | `r install\_flag`

11 | Echo Flag | `r echo\_flag`

12 | Info Flag | `r info\_flag`

13 | Data File | `r data\_file`

14 | Glos File | `r glos\_file`

15 | Info File | `r info\_file`

16 | Perl Folder | `r perl`

## Install Liberaries

The following data has been set by the setup file

N | what | Value

---|:---------------------------|:----------

1 | install\_flag | `r install\_flag`

The following Libraries are installed

N | what | Value

---|:---------------------------|:----------

1 | corrr | Correlations in R

2 | dplyr | A Grammar of Data Manipulation

3 | dygraphs | Interface to 'Dygraphs' Interactive Time Series Charting Library

4 | gdata | Various R Programming Tools for Data Manipulation

5 | ggplot2 | Create Elegant Data Visualisations Using the Grammar of Graphics

6 | gtools | Various R Programming Tools

7 | knitr | A General-Purpose Package for Dynamic Report Generation in R

8 | lares | Analytics & Machine Learning Sidekick

9 | lubridate | Make Dealing with Dates a Little Easier

10 | markdown | Render Markdown with 'commonmark'

11 | mclust | Gaussian Mixture Modelling for Model-Based Clustering, Classification

12 | onion | Octonions and Quaternions

13 | PerformanceAalytics") |Econometric Tools for Performance and Risk Analysis

14 | pracma | Practical Numerical Math Functions

15 | quantmod | Quantitative Financial Modelling Framework

16 | rgl | 3D Visualization Using OpenGL

17 | rmarkdown | Dynamic Documents for R

18 | stringr | Simple, Consistent Wrappers for Common String Operations

19 | tibble | Simple Data Frames

20 | tidyr | Tidy Messy Data

21 | xts | eXtensible Time Series

22 | zoo | S3 Infrastructure for Regular and Irregular Time Series (Z's O

```{r 1.4 - Install Libraries, warning= FALSE, echo = FALSE,include = FALSE}

# +---------------------------------------------------------------------------

# |

# | Installing required packages

# | install.packages("package name")

# |

# +----------------------------------------------------------------------------

# The install\_flag must be set to TRUE in the CDL\_Setup.CSV

if (install\_flag == TRUE ) {

install.packages("corrr") # Correlations in R

install.packages("dplyr") # A Grammar of Data Manipulation

install.packages("dygraphs") # Interface to 'Dygraphs' Interactive Time Series Charting Library

install.packages("gdata") # Various R Programming Tools for Data Manipulation

install.packages("ggplot2") # Create Elegant Data Visualisations Using the Grammar of Graphics

install.packages("gtools") # Various R Programming Tools

install.packages("knitr") # A General-Purpose Package for Dynamic Report Generation in R

install.packages("lares") # Analytics & Machine Learning Sidekick

install.packages("lubridate") # Make Dealing with Dates a Little Easier

install.packages("markdown") # Render Markdown with 'commonmark'

install.packages("mclust") # Gaussian Mixture Modelling for Model-Based Clustering, Classification

install.packages("onion") # Octonions and Quaternions

install.packages("PerformanceAnalytics") # Econometric Tools for Performance and Risk Analysis

install.packages("pracma") # Practical Numerical Math Functions

install.packages("quantmod") # Quantitative Financial Modelling Framework

install.packages("rgl") # 3D Visualization Using OpenGL

install.packages("rmarkdown") # Dynamic Documents for R

install.packages("stringr") # Simple, Consistent Wrappers for Common String Operations

install.packages("tibble") # Simple Data Frames

install.packages("tidyr") # Tidy Messy Data

install.packages("xts") # eXtensible Time Series

install.packages("zoo") # S3 Infrastructure for Regular and Irregular Time Series (Z's O

#

# https://www.rdocumentation.org/packages/anomalyDetection/versions/1.0

#

install.packages("AnomalyDetection") # Anomaly Detection

}

```

```{r 1.5 - Load Liberaries, warning= FALSE, echo = FALSE,include = FALSE}

# +---------------------------------------------------------------------------

# |

# | Load all needed libraries

# |

# +---------------------------------------------------------------------------

library(corrr)

library(dplyr)

library(dygraphs)

library(gdata)

library(ggplot2)

library(gtools)

library(knitr)

library(lares)

library(lubridate)

library(markdown)

library(mclust)

library(onion)

library(PerformanceAnalytics)

library(pracma)

library(quantmod)

library(rgl)

library(rmarkdown)

library(stringr)

library(tibble)

library(tidyr)

library(xts)

library(zoo)

# +---------------------------------------------------------------------------

# |

# | List all loaded Libraries if info\_flag is set to TRUE in the CDL\_Setup.csv

# |

# +---------------------------------------------------------------------------

if (info\_flag) {

my\_packages <- library()$results[,c(1,3)]

nrow(my\_packages)

View(my\_packages)

}

```

<!-- CDL-IPA-B-Part-02 -->

# Read Consolidated Data

Read the data from the data file from the \*\*B-Consolidate-Data\*\* folder

The data is store as

Proj... >> \*\*B-Consolidate-Data\*\*

There are three files:

\* Data,CSV

\* Glos.CS

\* Info.CSV

## Data.csv

The data is stored as:

Time Stamp | AI-Tag1 | AI-Tag2 | ... | AI-TagN | AO-Tag1 | AO-Tag2 | ... | AO-TagM |

:---------------|:--------|:--------|:----|:--------|:--------|:--------|:----|:--------|

2020-03-21 00:01|FT.1101 |FT1102 | ... |TT-1699 | CV-1101 | CV-1102 | ... |CV-1617 |

\* AI Columns: start at 2 to 283

\* AO Columns: start at 284 to 333

\* number of Rows: 2881

The data is store it in the following

\* all\_data: contains all data

\* AI\_data : sub-set of all\_data colum 1--283

\* AO\_data : sub-set of all\_data columns 1 and 284 to 333

## Glos.csv

The data is stored as:

ID | TAGNAME | Description | EU

:---|:---------|:-----------------------------|:---------------

1 | FT-1101 | FT-1101 Description | GPM

```{r 2.1-read-data, warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | Read data

# | proj\_folder == defined in CDL\_Setup.CSV

# | B-Consolidate-Data == folder name of consolidated data

# | data\_file == file name of consolidated data

# +------------------------------------------------------------------------

csv\_file = paste(proj\_folder,"/","B-Consolidate-Data/",data\_file,sep="")

# +------------------------------------------------------------------------

# | store all data in the "all\_data"

# +------------------------------------------------------------------------

all\_data <<- read.csv(csv\_file,sep=",")

# +------------------------------------------------------------------------

# | Replace the first column name to "TimeStamp"

# +------------------------------------------------------------------------

colnames(all\_data)[1] <- "TimeStamp"

# +------------------------------------------------------------------------

# | Assign all\_n\_rows to number of rows in all\_data

# | Assign all\_n\_cols to number of columns in all\_data

# +------------------------------------------------------------------------

all\_n\_rows <<- nrow(all\_data)

all\_n\_cols <<- ncol(all\_data)

# +------------------------------------------------------------------------

# | extract AI data

# +------------------------------------------------------------------------

AI\_data <- all\_data[,c(1,first\_col\_AI:last\_col\_AI)]

AI\_n\_rows <<- nrow(AI\_data)

AI\_n\_cols <<- ncol(AI\_data)

# +------------------------------------------------------------------------

# | extract AO data

# +------------------------------------------------------------------------

AO\_data <- all\_data[,c(1,first\_col\_AO:last\_col\_AO)]

AO\_n\_rows <<- nrow(AO\_data)

AO\_n\_cols <<- ncol(AO\_data)

```

The data marker are:

What | value

:-----------------------------------|:-----------------

All Data Number of Rows | `r all\_n\_rows`

All Data Number of Columns | `r all\_n\_cols`

First Row (from Setup) | `r first\_row`

Last Row (from Setup) | `r last\_row`

AI Data Number of Rows | `r AI\_n\_rows`

AI Data Number of Columns | `r AI\_n\_cols`

AO Data Number of Rows | `r AO\_n\_rows`

AO Data Number of Columns | `r AO\_n\_cols`

## Define Result Folder

The following folders or files are at the top folder

```{r 2.2-create sub-folder,warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | define a subfolder name

# +------------------------------------------------------------------------

subfolder <- paste("/C-PCA-Analysis","/",case\_name, "-Analysis","/", sep = "")

# +------------------------------------------------------------------------

# | Create sub folder

# +------------------------------------------------------------------------

dir.create(file.path(proj\_folder, subfolder), showWarnings = FALSE)

# +------------------------------------------------------------------------

# | list files in case there is any

# +------------------------------------------------------------------------

list.files(path = ".")

# +------------------------------------------------------------------------

# | delete all existing files using unlink function

# +------------------------------------------------------------------------

unlink(paste(proj\_folder,subfolder,"/\*",sep=""),recursive = TRUE)

```

```{r 2.3 -write files,warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | save input data for future use

# | tempfile == folder + subfolder + filename

# | write.csv write the data to tempfile with row.name set to TRUE

# +------------------------------------------------------------------------

tempfile = paste(proj\_folder,subfolder,"01-input-data.csv",sep="")

write.csv(all\_data, tempfile, row.names=T)

tempfile = paste(proj\_folder,subfolder,"02-input-data-summary.csv",sep="")

write.csv(summary(all\_data), tempfile, row.names=T)

```

The following files are stored in 'r subfolder`

What | value

:-------------------------|:-----------------

Project Folder | `r proj\_folder`

sub Folder | `r subfolder`

all data | 01-input-data.csv

all data Summary | 02-input-data-summary.csv

<!-- CDL-IPA-B-Part-03 -->

# Principal Component Analysis (PCA)

## Introduction

Principal Component Analysis (PCA) is a widely used technique in statistics and data analysis for dimensional reduction and feature extraction. It aims to transform a high-dimensional data set into a lower-dimensional space while retaining the most important information.

The main idea behind PCA is to find a new set of variables, called principal components, that are linear combinations of the original variables. These principal components are ordered in such a way that the first component captures the maximum amount of variance in the data, the second component captures the second most variance, and so on. In other words, PCA seeks to represent the data in a way that the most significant patterns and structures are emphasized in the first few components.

The process of performing PCA involves the following steps:

1. Standardization: If the variables in the dataset have different scales, it is common to standardize them (subtract the mean and divide by the standard deviation) to ensure that each variable contributes equally to the analysis.

2. Covariance matrix computation: The covariance matrix is calculated based on the standardized variables. It shows how each pair of variables varies together.

3. Eigenvector and eigenvalue calculation: The eigenvectors and eigenvalues of the covariance matrix are computed. The eigenvectors represent the directions (principal components) in which the data vary the most, and the eigenvalues represent the amount of variance explained by each eigenvector.

4. Selection of principal components: The eigenvectors are sorted in descending order of their corresponding eigenvalues. The top-k eigenvectors are chosen to form the principal components, where k is the desired dimensionality of the reduced dataset.

5. Projection: The original high-dimensional data is projected onto the selected principal components to obtain the lower-dimensional representation.

PCA has various applications, including data visualization, noise reduction, feature selection, and data compression. By reducing the dimensionality of the data, PCA can simplify complex datasets, remove redundant or irrelevant information, and facilitate subsequent analysis or visualization.

It's important to note that PCA assumes a linear relationship between variables and may not be suitable for capturing non-linear patterns in the data. In such cases, nonlinear dimensionality reduction techniques like manifold learning methods or kernel PCA can be more appropriate.

The data is consist of Analog Input (AI) and Analog Output (AO).

The PCA Analysis is done for

\* Analog Inputs (AI)

\* Analog Outputs (AO)

\* Both AI and AO

# Analyze AI Data

In this section, PCA analysis is applied to all AI tags.

## AI - Cross Correlation Analysis

The industrial data sometimes contains similar tags. the purpose of this analysis is to show which tags are correlated with another tags.

```{r 3.1.a - cor , warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | corr\_cross Cross correlation function

# | max\_pvalue display only significant correlations (at 5% level)

# | top display top 20 couples of variables (by correlation coefficient)

# +------------------------------------------------------------------------

cc <-corr\_cross(AI\_data, # name of data set

plot = TRUE,

max\_pvalue = 0.05, # display only significant correlations (at 5% level)

top = 5 # display top 5 couples of variables (by correlation coefficient)

)

cc

tags\_top <- append(cc$data$key , cc$data$mix)

tags\_top

```

```{r 3.1.b -Build PC2, out.height="400px", warning= warning\_flag, echo = echo\_flag}

#

# Plot PC2

#

df\_dyg = data.frame(AI\_data[,tags\_top])

chart.Correlation(df\_dyg[,1:10], histogram=TRUE, pch=19)

```

```{r 3.1.c -Build PC2, out.height="400px", warning= warning\_flag, echo = echo\_flag}

df\_dyg$TimeStamp <- AI\_data$TimeStamp

df\_dyg$TimeStamp <- strptime(df\_dyg$TimeStamp, date\_format,tz="GMT")

xts\_df\_df\_dyg <- xts(df\_dyg[,1:10] , df\_dyg$TimeStamp)

dygraph(xts\_df\_df\_dyg, main = "AI-Top 10 ", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## AI - Remove columns with Standard Diviation of zero

The columns with Standard Deviation of ZERO are removed. These are either have a fixed value or zero.

```{r 3.2 -build-PCA-Part-A , warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | Do the following:

# | 1- Assign run\_data <- AI\_data

# | 2- loop in columns 2 to ncols (which is AI\_n\_cols)

# | 3- remove columns with standard divation of zero

# +------------------------------------------------------------------------

run\_data <- AI\_data

ncols <- AI\_n\_cols

ncol\_deleted <- 0

firstrow <- first\_row

nrows <-last\_row

for ( icol in 2:ncols ) {

if(sd(run\_data[,icol]) == 0.0 ){

run\_data[icol] <- NULL

ncol\_deleted <- ncol\_deleted + 1

}

}

```

## AI- Scale consolidated Data

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

x\_scaled <- (x - mean(x)) / sd(x)

Once the standardization is done, all the variables will be transformed to the same scale.

```{r 3.3 -build-PCA-Part-A, warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | Do the following:

# | 1- ncols <- number of columns in run\_data using ncol function

# | 2- use scale function and scale run\_data to sdata

# | 3- write the sdata

# +------------------------------------------------------------------------

ncols = ncol(run\_data)

sdata <- scale(run\_data[firstrow:nrows,2:ncols])

tempfile = paste(proj\_folder,subfolder,"03-01-AI-sdata.csv",sep="")

write.csv(sdata, tempfile, row.names=T)

tempfile = paste(proj\_folder,subfolder,"04-01-AI-sdata-summary.csv",sep="")

write.csv(summary(sdata), tempfile, row.names=T)

```

The following files are stored in 'r subfolder`

What | value

:-------------------------|:-----------------

Project Folder | `r proj\_folder`

sub Folder | `r subfolder`

number of columns deleted | 'r ncol\_deleted`

AI data | 03-AI-sdata.csv

AI data Summary | 04-AI-sdata-summary.csv

## Generate PCA from AI Scaled Data

The calculation is done by a singular value decomposition of the centered and scaled data matrix, not by using eigen on the covariance matrix.

This is generally the preferred method for numerical accuracy. The print method for these objects prints the results in a nice format and the plot method produces a scree plot.

The purpose of PCA is to transpose the AI\_Data

Variable | Value

:--------------------------|:------------------------

AI\_Data Number of Columns | `r ncols`

AI\_Data Number of rows | `r nrows`

to

Variable | Value

:--------------------------|:------------------------

PCA Number of Columns | 3

PCA Number of rows | `r nrows`

```{r 3.4-build-PCA-Part-B ,warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | Do the following:

# | 1- transpose sdata to pca using prcomp function

# | 2- use scale function and scale run\_data to sdata

# | 3- write the sdata

# +------------------------------------------------------------------------

# Performs a principal components analysis on the given data matrix and returns the results

pca <- prcomp(sdata)

scores <- pca$x

x <- summary(pca)

vars <- x$sdev^2

vars <- vars/sum(vars)

tempfile = paste(proj\_folder,subfolder,"05-01-AI-PCA.csv",sep="")

write.csv(rbind("Standard deviation" = x$sdev, "Proportion of Variance" = vars,

"Cumulative Proportion" = cumsum(vars)), tempfile)

```

The following files are stored in 'r subfolder`

What | value

:-------------------------|:-----------------

Project Folder | `r proj\_folder`

number of columns deleted | 'r ncol\_deleted`

sub Folder | `r subfolder`

all data | 05-01-AI-PCA.csv

```{r 3.5-build-PCA-Part-B , warning= warning\_flag, echo = echo\_flag}

kaiser<- pca$sdev^2

var<- (pca$sdev^2 / sum(pca$sdev^2 ))

par(mfrow=c(1,1))

par(fig = c(0.0, 1.0, 0.0, 1.0))

#

# do it twice

#

for (iplot in 1:2)

{

if (iplot == 1 ) {

jpeg(paste(proj\_folder,subfolder,"A-AI-Cumulative-Sum.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

}

par(mar = c(5,5,2,5))

plot(cumsum(var[1:20])\*100,type="b",col="red3",ylab="CumSum",xlab="AI PCA sdev^2/Sum(sdev^2)")

abline(h=80,lty = 2,col="red3")

par(new = T)

plot(kaiser,axes=F,type="b", xlab=NA, ylab=NA)

axis(side = 4)

mtext(side = 4, line = 3, 'Kaiser StDev^2')

abline(h=1,lty = 2,col="black")

legend("bottomright",

legend=c("Cumulative-Sum > 80", "Kaiser(Stdev^2) > 1"),

lty=c(0,0), pch=c(16, 16), col=c("red3", "black"))

if (iplot == 1 ) {

dev.off()

}

}

# build fit

#

par(mfrow=c(1,1))

fit <- Mclust(pca$x[,1:3], G=n\_cluster)

```

## AI-3D Plot of PC1, PC2 and PC3

If we look at PCA more formally, it turns out that the PCA is based on a decomposition of the data matrix X into two matrices V and U:

```{r 3.6 - include image , warning= warning\_flag, echo = echo\_flag}

knitr::include\_graphics("./images/hl\_pca\_matmult.png")

```

The two matrices V and U are orthogonal. The matrix V is usually called the loadings matrix, and the matrix U is called the scores matrix. The loadings can be understood as the weights for each original variable when calculating the principal component. The matrix U contains the original data in a rotated coordinate system.

The PC1, PC2 and PC3 are represented in a dynamic 3D plot.

```{r 3.7 ploty, warning= warning\_flag, echo = echo\_flag}

if (os =="Darwin") {

library(plotly)

df\_pca <- data.frame(pca$x[,1:3])

fig <- plot\_ly(df\_pca, x=df\_pca$PC1,y=df\_pca$PC2,z=df\_pca$PC3, col = fit$classification, colors = c('#BF382A', '#0C4B8E'))

fig <- fig %>% add\_markers()

fig <- fig %>% layout(scene = list(xaxis = list(title = 'Weight'),

yaxis = list(title = 'Gross horsepower'),

zaxis = list(title = '1/4 mile time')))

fig

}

```

```{r 3.7-build-PCA-Part-B , warning= warning\_flag, echo = echo\_flag}

if (os !="Darwin") {

par3d(windowRect = c(100, 100, 1024, 1024))

plot3d(pca$x[,1],pca$x[,2],pca$x[,3], col = fit$classification,size=1,type='s',

xlab = 'PC1',

ylab = 'PC2',

zlab = 'PC3',

main = 'Principle Component Analysis-AI Data',

sub = 'Copyright (C) 2023 CTRL Designer LLC CtrlDesigner.Com ',

colkey = list(length = 0.5, width = 0.5, cex.clab = 0.75) )

tempfile = paste(proj\_folder,subfolder,"B-01-AI-PCA-3D-Overview.png",sep="")

rgl.snapshot(tempfile,fmt="png",top=TRUE)

knitr::include\_graphics(tempfile)

}

```

Please review the 3D plot using the following URL

`r tempfile`

## AI Plot PC1, PC2 and PC3 Pairs

The diagonal shows the names of the three numeric variables of our example data.

The other cells of the plot matrix show a scatterplot (i.e. correlation plot) of each variable combination of our data frame. The middle plot in the first row illustrates the correlation between x1 & x2; The right plot in the first row illustrates the correlation between x1 & x3; The left plot in the second row illustrates the correlation between x1 & x2 once more and so on.

The PC1, PC2 and PC3 are represented in a dynamic 3D plot.

What | value

:---------------------------|:-----------------

Folder | `r subfolder`

PC1, PC2 and PC3 Pairs plot | C-pairs.png

```{r 3.8-build-PCA-Part-C ,warning= warning\_flag, echo = echo\_flag}

#

# [1] Build CumSum Plot C

#

for (iplot in 1:2) {

if (iplot == 1) {

jpeg(paste(proj\_folder,subfolder,"C-AI-pairs.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

}

pairs(pca$x[,1:3],col=fit$classification)

if (iplot == 1) {

dev.off()

}

}

```

## Build AI-T2

The PC1, PC2 and PC3 are saved

What | value

:---------------------------|:-----------------

Folder | `r subfolder`

PC1, PC2 and PC3 Pairs plot | 09-AI-PCA.csv

```{r 3.9-Build T2, warning= warning\_flag, echo = echo\_flag}

#

# Build CumSum Plot C

#

pc1<-pca$x[,1]

pc2<-pca$x[,2]

pc3<-pca$x[,3]

t2 <<- sqrt(pc1^2+pc2^2+pc3^2)

tempfile = paste(proj\_folder,subfolder,"09-01-AI-PCA.csv",sep="")

write.csv(pca$x[,1:5], tempfile, row.names=F)

df\_pc <- NULL

df\_pc <- data.frame(pca$x[,1:10])

df\_pc$TimeStamp <- run\_data$TimeStamp

df\_pc$T2 <- t2

tempfile = paste(proj\_folder,subfolder,"10-01-AI-PCA-T2.csv",sep="")

write.csv(df\_pc, tempfile, row.names=F)

df\_T2summary <- NULL

df\_T2summary$TimeStamp <- run\_data$TimeStamp

df\_T2summary$AIT2 <- t2

```

## Plot AI-T2

The PC1, PC2 and PC3 are saved

What | value

:-------------------------|:-----------------

Folder | `r subfolder`

D-T2-PCs.png | Plot of T2

```{r 3.10-Build T2 , warning= warning\_flag, echo = echo\_flag}

par(mfrow = c(1,1))

jpeg(paste(proj\_folder,subfolder,"D-AI-T2-PCs.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

#

# Plot T2

#

par(fig = c(0.0, 0.7, 0.60, 1.0))

plot(t2,xlab="",xaxt="n", col = fit$classification, ylab=" Hotelling's T2")

par(fig = c(0.55, 1.0, 0.60, 1.0), new=TRUE)

p3d(pc1,pc2,pc3,d0=0.5,h=1.0)

#

# plot PC1 and PC2 vs PC1

#

par(fig = c(0.0, 0.7, 0.4, 0.8), new=TRUE)

plot(pc1,xlab="",xaxt="n",col = fit$classification)

par(fig = c(0.6, 1.0, 0.4, 0.8), new=TRUE)

plot(pc2,pc1,xlab="",ylab="", yaxt="n", col = fit$classification)

#

# plot PC2 and PC3 vs PC2

#

par(fig = c(0.0, 0.7, 0.2, 0.6), new=TRUE)

plot(pc2,xlab="",xaxt="n",col = fit$classification)

par(fig = c(0.6, 1.0, 0.2, 0.6), new=TRUE)

plot(pc3,pc2 ,xlab="",ylab="", yaxt="n",col = fit$classification)

#

# plot PC3 and PC3 vs PC1

#

par(fig = c(0.0, 0.7, 0.0, 0.4), new=TRUE)

plot(pc3,col = fit$classification)

par(fig = c(0.6, 1.0, 0.0, 0.4), new=TRUE)

plot(pc3,pc1,ylab="", yaxt="n",col = fit$classification)

dev.off()

```

## Plot AI-T2 Dynamic

```{r 3.11 -Build T2, out.height="400px", warning= warning\_flag, echo = echo\_flag}

par(mfrow = c(4,1))

#

# Plot T2

#

df\_t2 = data.frame(t2)

df\_t2$TimeStamp <- run\_data$TimeStamp

df\_t2$TimeStamp <- strptime(df\_t2$TimeStamp, date\_format,tz="GMT")

xts\_df\_t2 <- xts(df\_t2[,1] , df\_t2$TimeStamp)

dygraph(xts\_df\_t2, main = "AI-T2 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## Plot AI-PC1 Dynamic

```{r 3.12 -Build PC1, out.height="400px", warning= warning\_flag, echo = echo\_flag}

#

# Plot PC1

#

df\_pc1 = data.frame(pc1)

df\_pc1$TimeStamp <- run\_data$TimeStamp

df\_pc1$TimeStamp <- strptime(df\_pc1$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc1 <- xts(df\_pc1[,1] , df\_pc1$TimeStamp)

dygraph(xts\_df\_pc1, main = "AI-PC1 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## Plot AI-PC2 Dynamic

```{r 3.12 -Build PC2, out.height="400px", warning= warning\_flag, echo = echo\_flag}

#

# Plot PC2

#

df\_pc2 = data.frame(pc2)

df\_pc2$TimeStamp <- run\_data$TimeStamp

df\_pc2$TimeStamp <- strptime(df\_pc2$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc2 <- xts(df\_pc2[,1] , df\_pc1$TimeStamp)

dygraph(xts\_df\_pc2, main = "AI-PC2 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## Plot AI-PC3 Dynamic

```{r 3.13-Build PC3, out.height="400px",warning= warning\_flag, echo = echo\_flag}

#

# Plot PC3

#

df\_pc3 = data.frame(pc3)

df\_pc3$TimeStamp <- run\_data$TimeStamp

df\_pc3$TimeStamp <- strptime(df\_pc3$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc3 <- xts(df\_pc3[,1] , df\_pc3$TimeStamp)

dygraph(xts\_df\_pc3, main = "AI-PC3 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## Analyzing AI PCA

```{r 3.14-build-PCA-Part-D,width=11.0,height=8.0,warning= warning\_flag, echo = echo\_flag}

rotation<- pca$rotation

absrotation <- abs(pca$rotation)

#dividing each column by sum

load\_data <- sweep(absrotation,2,colSums(absrotation),"/")

tempfile = paste(proj\_folder,subfolder,"06-01-AI-load.csv",sep="")

write.csv(load\_data, tempfile, row.names=T)

tempfile = paste(proj\_folder,subfolder,"07-01-AI-load-summary.csv",sep="")

write.csv(summary(load\_data), tempfile, row.names=T)

for (i in 1:3)

{

#

# extract the load 1 , Load 2 and Load 3

#

ord<- order(load\_data[,i],decreasing = T)

sloading <- load\_data[ord,]\*100

if (i == 1 ) {

myrow\_1 <- row.names(sloading)

myrow\_1[1:16]

sortdata <- run\_data[,myrow\_1]

sortdata$TimeStamp <- run\_data$TimeStamp

sortdata$TimeStamp <- strptime(sortdata$TimeStamp, date\_format,tz="GMT")

}

if (i == 2 ) {myrow\_2 <- row.names(sloading)}

if (i == 3 ) {myrow\_3 <- row.names(sloading)}

for (iplot in 1:1) {

tempfile = paste(proj\_folder,subfolder,"08-01-AI-PCA",i,"-load-sorted.csv",sep="")

write.csv(sloading, tempfile, row.names=T)

par(mfrow = c(1,1))

par(fig = c(0.1, 1.0, 0.0, 1.0))

bplt <-barplot(sloading[1:25,i], las=2, horiz=T, col = "blue", xlab=paste0("Sorted Loading for [",i,"]"))

text(x= sloading[1:25,i]+0.1, y= bplt, labels=as.character(round(sloading[1:25,i],digits=2)), xpd=TRUE)

par(mar=c(5, 4, 4, 4) + 0.1)

}

}

```

The PC1, PC2 and PC3 are saved

What | value

:-------------------------|:-----------------

Folder | `r subfolder`

Load Data | 06-AI-load.csv

Load data summary | 07-AI-load-summary.csv

## AI-Main Tag Contributer

Based on the analysis above the following 10 tags contributes to the variations.

```{r 3.15,warning= warning\_flag, echo = echo\_flag}

i <- 1

j <- 5

xts\_sortdata <- xts(sortdata[,c(i:j)] , sortdata$TimeStamp)

myrow\_1[i]

dygraph(xts\_sortdata, main = paste0("AI Tag ",myrow\_1[i]), ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "always") %>%

dyRangeSelector()

```

```{r 3.16, warning= warning\_flag, echo = echo\_flag}

i <- 1

xts\_sortdata <- xts(sortdata[,c(i:i)] , sortdata$TimeStamp)

myrow\_1[i]

dygraph(xts\_sortdata, main = paste0("Tag ",myrow\_1[i]), ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "always") %>%

dyRangeSelector()

```

<!-- CDL-IPA-B-Part-04 -->

# Analyze AO Data

In this section, PCA analysis is applied to all AO tags.

## AO - Cross Correlation Analysis

The industrial data sometimes contains similar tags. the purpose of this analysis is to show which tags are correlated with another tags.

```{r 4.1.a - cor , warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | corr\_cross Cross correlation function

# | max\_pvalue display only significant correlations (at 5% level)

# | top display top 20 couples of variables (by correlation coefficient)

# +------------------------------------------------------------------------

cc <-corr\_cross(AO\_data, # name of data set

plot = TRUE,

max\_pvalue = 0.05, # display only significant correlations (at 5% level)

top = 5 # display top 5 couples of variables (by correlation coefficient)

)

cc

tags\_top <- append(cc$data$key , cc$data$mix)

tags\_top

```

```{r 4.1.b -Build PC2, out.height="400px", warning= warning\_flag, echo = echo\_flag}

#

# Plot PC2

#

df\_dyg = data.frame(AO\_data[,tags\_top])

chart.Correlation(df\_dyg[,1:10], histogram=TRUE, pch=19)

```

```{r 4.1.c -Build PC2, out.height="400px", warning= warning\_flag, echo = echo\_flag}

df\_dyg$TimeStamp <- AO\_data$TimeStamp

df\_dyg$TimeStamp <- strptime(df\_dyg$TimeStamp, date\_format,tz="GMT")

xts\_df\_df\_dyg <- xts(df\_dyg[,1:10] , df\_dyg$TimeStamp)

dygraph(xts\_df\_df\_dyg, main = "AO-Top 10 ", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## AO - Remove columns with Standard Diviation of zero

The columns with Standard Deviation of ZERO are removed. These are either have a fixed value or zero.

```{r 4.2 -build-PCA-Part-A , warning=warning\_flag, echo = echo\_flag}

#

# +------------------------------------------------------------------------

# | generate PCA for AO Data

# +------------------------------------------------------------------------

#

run\_data <- AO\_data

ncols = AO\_n\_cols

ncol\_deleted <- 0

firstrow <- first\_row

nrows <-last\_row

for ( icol in 2:ncols ) {

if(sd(run\_data[,icol]) == 0.0 ){

run\_data[icol] <- NULL

ncol\_deleted <- ncol\_deleted + 1

}

}

```

## AO- Scale consolidated Data

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

x\_scaled <- (x - mean(x)) / sd(x)

Once the standardization is done, all the variables will be transformed to the same scale.

```{r 4.3 -build-PCA-Part-A , warning=warning\_flag, echo = echo\_flag}

ncols = ncol(run\_data)

sdata <- scale(run\_data[firstrow:nrows,2:ncols])

tempfile = paste(proj\_folder,subfolder,"03-02-AO-sdata.csv",sep="")

write.csv(sdata, tempfile, row.names=T)

tempfile = paste(proj\_folder,subfolder,"04-02-AO-sdata-summary.csv",sep="")

write.csv(summary(sdata), tempfile, row.names=T)

```

The following files are stored in 'r subfolder`

What | value

:-------------------------|:-----------------

Project Folder | `r proj\_folder`

sub Folder | `r subfolder`

number of columns deleted | 'r ncol\_deleted`

AO data | 03-AO-sdata.csv

AO data Summary | 04-AO-sdata-summary.csv

## Generate PCA from AO Scaled Data

The calculation is done by a singular value decomposition of the centered and scaled data matrix, not by using eigen on the covariance matrix.

This is generally the preferred method for numerical accuracy. The print method for these objects prints the results in a nice format and the plot method produces a scree plot.

```{r 4.4-build-PCA-Part-B , warning=warning\_flag, echo = echo\_flag}

#

# [1] apply PCA

#

pca <- prcomp(sdata)

#summary(pca)

scores <- pca$x

x <- summary(pca)

vars <- x$sdev^2

vars <- vars/sum(vars)

tempfile = paste(proj\_folder,subfolder,"05-02-AO-PCA.csv",sep="")

write.csv(rbind("Standard deviation" = x$sdev, "Proportion of Variance" = vars,

"Cumulative Proportion" = cumsum(vars)), tempfile)

```

The following files are stored in 'r subfolder`

What | value

:-------------------------|:-----------------

Project Folder | `r proj\_folder`

number of columns deleted | 'r ncol\_deleted`

sub Folder | `r subfolder`

all data | 05-AO-PCA.csv

```{r 4.5-build-PCA-Part-B , warning=warning\_flag, echo = echo\_flag}

#

# Build CumSum Plot A

#

kaiser<- pca$sdev^2

var<- (pca$sdev^2 / sum(pca$sdev^2 ))

par(mfrow=c(1,1))

par(fig = c(0.0, 1.0, 0.0, 1.0))

#

# do it twice

#

for (iplot in 1:2)

{

if (iplot == 1 ) {

jpeg(paste(proj\_folder,subfolder,"A-AO-Cumulative-Sum.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

}

par(mar = c(5,5,2,5))

plot(cumsum(var[1:20])\*100,type="b",col="red3",ylab="CumSum",xlab="AO PCA sdev^2/Sum(sdev^2)")

abline(h=80,lty = 2,col="red3")

par(new = T)

plot(kaiser,axes=F,type="b", xlab=NA, ylab=NA)

axis(side = 4)

mtext(side = 4, line = 3, 'Kaiser StDev^2')

abline(h=1,lty = 2,col="black")

legend("bottomright",

legend=c("Cumulative-Sum > 80", "Kaiser(Stdev^2) > 1"),

lty=c(0,0), pch=c(16, 16), col=c("red3", "black"))

if (iplot == 1 ) {

dev.off()

}

}

# build fit

#

par(mfrow=c(1,1))

fit <- Mclust(pca$x[,1:3], G=n\_cluster)

```

## AO-3D Plot of PC1, PC2 and PC3

If we look at PCA more formally, it turns out that the PCA is based on a decomposition of the data matrix X into two matrices V and U:

```{r 4.6 - include image , warning=warning\_flag, echo = echo\_flag }

dir()

knitr::include\_graphics("./images/hl\_pca\_matmult.png")

```

The two matrices V and U are orthogonal. The matrix V is usually called the loadings matrix, and the matrix U is called the scores matrix. The loadings can be understood as the weights for each original variable when calculating the principal component. The matrix U contains the original data in a rotated coordinate system.

The PC1, PC2 and PC3 are represented in a dynamic 3D plot.

```{r 4.7 ploty, warning= warning\_flag, echo = echo\_flag}

if (os =="Darwin") {

library(plotly)

df\_pca <- data.frame(pca$x[,1:3])

fig <- plot\_ly(df\_pca, x=df\_pca$PC1,y=df\_pca$PC2,z=df\_pca$PC3, col = fit$classification, colors = c('#BF382A', '#0C4B8E'))

fig <- fig %>% add\_markers()

fig <- fig %>% layout(scene = list(xaxis = list(title = 'Weight'),

yaxis = list(title = 'Gross horsepower'),

zaxis = list(title = '1/4 mile time')))

fig

}

```

```{r 4.7-build-PCA-Part-B , warning=warning\_flag, echo = echo\_flag }

if (os !="Darwin") {

par3d(windowRect = c(100, 100, 1024, 1024))

plot3d(pca$x[,1],pca$x[,2],pca$x[,3], col = fit$classification,size=1,type='s',

xlab = 'PC1',

ylab = 'PC2',

zlab = 'PC3',

main = 'Principle Component Analysis',

sub = 'Copyright (C) 2023 CTRL Designer LLC CtrlDesigner.Com ',

colkey = list(length = 0.5, width = 0.5, cex.clab = 0.75) )

tempfile = paste(proj\_folder,subfolder,"B-02-AO-PCA-3D-Overview.png",sep="")

rgl.snapshot(tempfile,fmt="png",top=TRUE)

knitr::include\_graphics(tempfile)

}

```

Please review the 3D plot using the following URL

`r tempfile`

## AO Plot PC1, PC2 and PC3 Pairs

The diagonal shows the names of the three numeric variables of our example data.

The other cells of the plot matrix show a scatterplot (i.e. correlation plot) of each variable combination of our data frame. The middle plot in the first row illustrates the correlation between x1 & x2; The right plot in the first row illustrates the correlation between x1 & x3; The left plot in the second row illustrates the correlation between x1 & x2 once more and so on.

The PC1, PC2 and PC3 are represented in a dynamic 3D plot.

What | value

:---------------------------|:-----------------

Folder | `r subfolder`

PC1, PC2 and PC3 Pairs plot | C-pairs.png

```{r 4.8-build-PCA-Part-C , warning=warning\_flag, echo = echo\_flag }

#

# [1] Build CumSum Plot C

#

for (iplot in 1:2) {

if (iplot == 1) {

jpeg(paste(proj\_folder,subfolder,"C-AO-pairs.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

}

pairs(pca$x[,1:3],col=fit$classification)

if (iplot == 1) {

dev.off()

}

}

```

## Build AO-T2

The PC1, PC2 and PC3 are saved

What | value

:---------------------------|:-----------------

Folder | `r subfolder`

PC1, PC2 and PC3 Pairs plot | 09-AO-PCA.csv

```{r 4.9-Build T2, warning=warning\_flag, echo = echo\_flag}

#

# Build CumSum Plot C

#

pc1<-pca$x[,1]

pc2<-pca$x[,2]

pc3<-pca$x[,3]

t2 <<- sqrt(pc1^2+pc2^2+pc3^2)

tempfile = paste(proj\_folder,subfolder,"09-02-AO-PCA.csv",sep="")

write.csv(pca$x[,1:5], tempfile, row.names=F)

df\_pc <- NULL

df\_pc <- data.frame(pca$x[,1:10])

df\_pc$TimeStamp <- run\_data$TimeStamp

df\_pc$T2 <- t2

tempfile = paste(proj\_folder,subfolder,"10-02-AO-PCA-T2.csv",sep="")

write.csv(df\_pc, tempfile, row.names=F)

df\_T2summary$AOT2 <- t2

```

## Plot AO-T2

The PC1, PC2 and PC3 are saved

What | value

:-------------------------|:-----------------

Folder | `r subfolder`

D-T2-PCs.png | Plot of T2

```{r 4.10-Build T2 , warning=warning\_flag, echo = echo\_flag}

par(mfrow = c(1,1))

jpeg(paste(proj\_folder,subfolder,"D-AO-T2-PCs.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

#

# Plot T2

#

par(fig = c(0.0, 0.7, 0.60, 1.0))

plot(t2,xlab="",xaxt="n", col = fit$classification, ylab=" Hotelling's T2")

par(fig = c(0.55, 1.0, 0.60, 1.0), new=TRUE)

p3d(pc1,pc2,pc3,d0=0.5,h=1.0)

#

# plot PC1 and PC2 vs PC1

#

par(fig = c(0.0, 0.7, 0.4, 0.8), new=TRUE)

plot(pc1,xlab="",xaxt="n",col = fit$classification)

par(fig = c(0.6, 1.0, 0.4, 0.8), new=TRUE)

plot(pc2,pc1,xlab="",ylab="", yaxt="n", col = fit$classification)

#

# plot PC2 and PC3 vs PC2

#

par(fig = c(0.0, 0.7, 0.2, 0.6), new=TRUE)

plot(pc2,xlab="",xaxt="n",col = fit$classification)

par(fig = c(0.6, 1.0, 0.2, 0.6), new=TRUE)

plot(pc3,pc2 ,xlab="",ylab="", yaxt="n",col = fit$classification)

#

# plot PC3 and PC3 vs PC1

#

par(fig = c(0.0, 0.7, 0.0, 0.4), new=TRUE)

plot(pc3,col = fit$classification)

par(fig = c(0.6, 1.0, 0.0, 0.4), new=TRUE)

plot(pc3,pc1,ylab="", yaxt="n",col = fit$classification)

dev.off()

```

##Plot AO-T2 Dynamic

```{r 4.11 -Build T2, out.height="400px", warning=warning\_flag, echo = echo\_flag}

par(mfrow = c(4,1))

#

# Plot T2

#

df\_t2 = data.frame(t2)

df\_t2$TimeStamp <- run\_data$TimeStamp

df\_t2$TimeStamp <- strptime(df\_t2$TimeStamp, date\_format,tz="GMT")

xts\_df\_t2 <- xts(df\_t2[,1] , df\_t2$TimeStamp)

dygraph(xts\_df\_t2, main = "AO-T2 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

##Plot AO-PC1 Dynamic

```{r 4.12 -Build PC1, out.height="400px", warning=warning\_flag, echo = echo\_flag}

#

# Plot PC1

#

df\_pc1 = data.frame(pc1)

df\_pc1$TimeStamp <- run\_data$TimeStamp

df\_pc1$TimeStamp <- strptime(df\_pc1$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc1 <- xts(df\_pc1[,1] , df\_pc1$TimeStamp)

dygraph(xts\_df\_pc1, main = "AO-PC1 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

##Plot AO-PC2 Dynamic

```{r 4.12 -Build PC2, out.height="400px", warning=warning\_flag, echo = echo\_flag}

#

# Plot PC2

#

df\_pc2 = data.frame(pc2)

df\_pc2$TimeStamp <- run\_data$TimeStamp

df\_pc2$TimeStamp <- strptime(df\_pc2$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc2 <- xts(df\_pc2[,1] , df\_pc1$TimeStamp)

dygraph(xts\_df\_pc2, main = "AO-PC2 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## Plot AO-PC3 Dynamic

```{r 4.13-Build PC3, out.height="400px", warning=warning\_flag, echo = echo\_flag}

#

# Plot PC1

#

df\_pc3 = data.frame(pc3)

df\_pc3$TimeStamp <- run\_data$TimeStamp

df\_pc3$TimeStamp <- strptime(df\_pc3$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc3 <- xts(df\_pc3[,1] , df\_pc3$TimeStamp)

dygraph(xts\_df\_pc3, main = "AO-PC3 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## Analyzing AO=PCA

```{r 4.14-build-PCA-Part-D,width=11.0,height=8.0, warning=warning\_flag, echo = echo\_flag}

#

# [3] Loading

#

rotation<- pca$rotation

absrotation <- abs(pca$rotation)

#dividing each column by sum

load\_data <- sweep(absrotation,2,colSums(absrotation),"/")

tempfile = paste(proj\_folder,subfolder,"06-02-AO-load.csv",sep="")

write.csv(load\_data, tempfile, row.names=T)

tempfile = paste(proj\_folder,subfolder,"07-02-AO-load-summary.csv",sep="")

write.csv(summary(load\_data), tempfile, row.names=T)

for (i in 1:3)

{

#

# extract the load 1 , Load 2 and Load 3

#

ord<- order(load\_data[,i],decreasing = T)

sloading <- load\_data[ord,]\*100

if (i == 1 ) {

myrow\_1 <- row.names(sloading)

myrow\_1[1:16]

sortdata <- run\_data[,myrow\_1]

sortdata$TimeStamp <- run\_data$TimeStamp

sortdata$TimeStamp <- strptime(sortdata$TimeStamp, date\_format,tz="GMT")

}

if (i == 2 ) {myrow\_2 <- row.names(sloading)}

if (i == 3 ) {myrow\_3 <- row.names(sloading)}

for (iplot in 1:1) {

tempfile = paste(proj\_folder,subfolder,"08-02-AO-PCA",i,"-load-sorted.csv",sep="")

write.csv(sloading, tempfile, row.names=T)

par(mfrow = c(1,1))

par(fig = c(0.1, 1.0, 0.0, 1.0))

bplt <-barplot(sloading[1:25,i], las=2, horiz=T, col = "blue", xlab=paste0("Sorted Loading for [",i,"]"))

text(x= sloading[1:25,i]+0.1, y= bplt, labels=as.character(round(sloading[1:25,i],digits=2)), xpd=TRUE)

par(mar=c(5, 4, 4, 4) + 0.1)

}

}

```

The PC1, PC2 and PC3 are saved

What | value

:-------------------------|:-----------------

Folder | `r subfolder`

Load Data | 06-AO-load.csv

Load data summary | 07-AO-load-summary.csv

## AO-Main Tag Contributer

Based on the analysis above the following 10 tags contributes to the variations.

```{r 4.15, warning=warning\_flag, echo = echo\_flag}

i <- 1

j <- 5

xts\_sortdata <- xts(sortdata[,c(i:j)] , sortdata$TimeStamp)

myrow\_1[i]

dygraph(xts\_sortdata, main = paste0("AO Tag ",myrow\_1[i]), ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "always") %>%

dyRangeSelector()

```

```{r 4.16, warning=warning\_flag, echo = echo\_flag}

i <- 1

xts\_sortdata <- xts(sortdata[,c(i:i)] , sortdata$TimeStamp)

myrow\_1[i]

dygraph(xts\_sortdata, main = paste0("Tag ",myrow\_1[i]), ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "always") %>%

dyRangeSelector()

```

```{r 4.17, warning=warning\_flag, echo = echo\_flag}

df\_pc3 = data.frame(df\_T2summary)

df\_pc3$TimeStamp <- df\_T2summary$TimeStamp

df\_pc3$TimeStamp <- strptime(df\_pc3$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc3 <- xts(df\_pc3[,2:3] , df\_pc3$TimeStamp)

dygraph(xts\_df\_pc3, main = "AI AO T2 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

```{r 4.18 , warning=warning\_flag, echo = echo\_flag}

df\_pc3 = data.frame(df\_T2summary)

df\_pc3$TimeStamp <- df\_T2summary$TimeStamp

df\_pc3$TimeStamp <- strptime(df\_pc3$TimeStamp, date\_format,tz="GMT")

sdata <- scale(df\_pc3[firstrow:nrows,2:3])

xts\_df\_pc3 <- xts(sdata[,1:2] , df\_pc3$TimeStamp)

dygraph(xts\_df\_pc3, main = "AI AO T2 Scaled Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

```{r 4.18 fit , warning=warning\_flag, echo = echo\_flag}

plot(df\_T2summary$AOT2, df\_T2summary$AIT2, pch=12, xlab='T2-AO', ylab='T2-AI')

fit1 <- lm(df\_T2summary$AIT2 ~df\_T2summary$AOT2, data=df\_T2summary)

summary(fit1)

```

<!-- CDL-IPA-B-Part-05 -->

# Analyze AI-AO Data

In this section, PCA analysis is applied to all AI-AO tags.

## AI-AO - Cross Correlation Analysis

The industrial data sometimes contains similar tags. the purpose of this analysis is to show which tags are correlated with another tags.

```{r 5.1.a - cor , warning= warning\_flag, echo = echo\_flag}

# +------------------------------------------------------------------------

# | corr\_cross Cross correlation function

# | max\_pvalue display only significant correlations (at 5% level)

# | top display top 20 couples of variables (by correlation coefficient)

# +------------------------------------------------------------------------

cc <-corr\_cross(all\_data, # name of data set

plot = TRUE,

max\_pvalue = 0.05, # display only significant correlations (at 5% level)

top = 5 # display top 5 couples of variables (by correlation coefficient)

)

cc

tags\_top <- append(cc$data$key , cc$data$mix)

tags\_top

```

```{r 5.1.b -Build PC2, out.height="400px", warning= warning\_flag, echo = echo\_flag}

#

# Plot PC2

#

df\_dyg = data.frame(all\_data[,tags\_top])

chart.Correlation(df\_dyg[,1:10], histogram=TRUE, pch=19)

```

```{r 5.1.c -Build PC2, out.height="400px", warning= warning\_flag, echo = echo\_flag}

df\_dyg$TimeStamp <- all\_data$TimeStamp

df\_dyg$TimeStamp <- strptime(df\_dyg$TimeStamp, date\_format,tz="GMT")

xts\_df\_df\_dyg <- xts(df\_dyg[,1:10] , df\_dyg$TimeStamp)

dygraph(xts\_df\_df\_dyg, main = "AI-AO-Top 5 ", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## AI-AO - Remove columns with Standard Diviation of zero

The columns with Standard Deviation of ZERO are removed. These are either have a fixed value or zero.

```{r 5.2 -build-PCA-Part-A , warning=warning\_flag, echo = echo\_flag}

#

# +------------------------------------------------------------------------

# | generate PCA for AI-AO Data

# +------------------------------------------------------------------------

#

run\_data <- all\_data

ncols = all\_n\_cols

ncol\_deleted <- 0

firstrow <- first\_row

nrows <-last\_row

for ( icol in 2:ncols ) {

if(sd(run\_data[,icol]) == 0.0 ){

run\_data[icol] <- NULL

ncol\_deleted <- ncol\_deleted + 1

}

}

```

## AI-AO- Scale consolidated Data

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

x\_scaled <- (x - mean(x)) / sd(x)

Once the standardization is done, all the variables will be transformed to the same scale.

```{r 5.3 -build-PCA-Part-A , warning=warning\_flag, echo = echo\_flag}

ncols = ncol(run\_data)

sdata <- scale(run\_data[firstrow:nrows,2:ncols])

tempfile = paste(proj\_folder,subfolder,"03-03-AI-AO-sdata.csv",sep="")

write.csv(sdata, tempfile, row.names=T)

tempfile = paste(proj\_folder,subfolder,"04-03-AI-AO-sdata-summary.csv",sep="")

write.csv(summary(sdata), tempfile, row.names=T)

```

The following files are stored in 'r subfolder`

What | value

:-------------------------|:-----------------

Project Folder | `r proj\_folder`

sub Folder | `r subfolder`

number of columns deleted | 'r ncol\_deleted`

AI-AO data | 03-AI-AO-sdata.csv

AI-AO data Summary | 04-AI-AO-sdata-summary.csv

## Generate PCA from AI-AO Scaled Data

The calculation is done by a singular value decomposition of the centered and scaled data matrix, not by using eigen on the covariance matrix.

This is generally the preferred method for numerical accuracy. The print method for these objects prints the results in a nice format and the plot method produces a scree plot.

```{r 5.4-build-PCA-Part-B , warning=warning\_flag, echo = echo\_flag}

#

# [1] apply PCA

#

pca <- prcomp(sdata)

#summary(pca)

scores <- pca$x

x <- summary(pca)

vars <- x$sdev^2

vars <- vars/sum(vars)

tempfile = paste(proj\_folder,subfolder,"05-03-AI-AO-PCA.csv",sep="")

write.csv(rbind("Standard deviation" = x$sdev, "Proportion of Variance" = vars,

"Cumulative Proportion" = cumsum(vars)), tempfile)

```

The following files are stored in 'r subfolder`

What | value

:-------------------------|:-----------------

Project Folder | `r proj\_folder`

number of columns deleted | 'r ncol\_deleted`

sub Folder | `r subfolder`

all data | 05-AI-AO-PCA.csv

```{r 5.5-build-PCA-Part-B , warning=warning\_flag, echo = echo\_flag}

#

# Build CumSum Plot A

#

kaiser<- pca$sdev^2

var<- (pca$sdev^2 / sum(pca$sdev^2 ))

par(mfrow=c(1,1))

par(fig = c(0.0, 1.0, 0.0, 1.0))

#

# do it twice

#

for (iplot in 1:2)

{

if (iplot == 1 ) {

jpeg(paste(proj\_folder,subfolder,"A-AI-AO-Cumulative-Sum.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

}

par(mar = c(5,5,2,5))

plot(cumsum(var[1:20])\*100,type="b",col="red3",ylab="CumSum",xlab="AI-AO PCA sdev^2/Sum(sdev^2)")

abline(h=80,lty = 2,col="red3")

par(new = T)

plot(kaiser,axes=F,type="b", xlab=NA, ylab=NA)

axis(side = 4)

mtext(side = 4, line = 3, 'Kaiser StDev^2')

abline(h=1,lty = 2,col="black")

legend("bottomright",

legend=c("Cumulative-Sum > 80", "Kaiser(Stdev^2) > 1"),

lty=c(0,0), pch=c(16, 16), col=c("red3", "black"))

if (iplot == 1 ) {

dev.off()

}

}

# build fit

#

par(mfrow=c(1,1))

fit <- Mclust(pca$x[,1:3], G=n\_cluster)

```

## AI-AO-3D Plot of PC1, PC2 and PC3

If we look at PCA more formally, it turns out that the PCA is based on a decomposition of the data matrix X into two matrices V and U:

```{r 5.6 - include image , warning=warning\_flag, echo = echo\_flag }

knitr::include\_graphics("./images/hl\_pca\_matmult.png")

```

The two matrices V and U are orthogonal. The matrix V is usually called the loadings matrix, and the matrix U is called the scores matrix. The loadings can be understood as the weights for each original variable when calculating the principal component. The matrix U contains the original data in a rotated coordinate system.

The PC1, PC2 and PC3 are represented in a dynamic 3D plot.

```{r 5.7 ploty, warning= warning\_flag, echo = echo\_flag}

if (os =="Darwin") {

library(plotly)

df\_pca <- data.frame(pca$x[,1:3])

fig <- plot\_ly(df\_pca, x=df\_pca$PC1,y=df\_pca$PC2,z=df\_pca$PC3, col = fit$classification, colors = c('#BF382A', '#0C4B8E'))

fig <- fig %>% add\_markers()

fig <- fig %>% layout(scene = list(xaxis = list(title = 'Weight'),

yaxis = list(title = 'Gross horsepower'),

zaxis = list(title = '1/4 mile time')))

fig

}

```

```{r 5.7-build-PCA-Part-B , warning=warning\_flag, echo = echo\_flag }

if ((os !="Darwin") ) {

par3d(windowRect = c(100, 100, 1024, 1024))

plot3d(pca$x[,1],pca$x[,2],pca$x[,3], col = fit$classification,size=1,type='s',

xlab = 'PC1',

ylab = 'PC2',

zlab = 'PC3',

main = 'Principle Component Analysis',

sub = 'Copyright (C) 2023 CTRL Designer LLC CtrlDesigner.Com ',

colkey = list(length = 0.5, width = 0.5, cex.clab = 0.75) )

tempfile = paste(proj\_folder,subfolder,"B-03-AI-AO-PCA-3D-Overview.png",sep="")

rgl.snapshot(tempfile,fmt="png",top=TRUE)

knitr::include\_graphics(tempfile)

}

```

Please review the 3D plot using the following URL

`r tempfile`

## AI-AO Plot PC1, PC2 and PC3 Pairs

The diagonal shows the names of the three numeric variables of our example data.

The other cells of the plot matrix show a scatterplot (i.e. correlation plot) of each variable combination of our data frame. The middle plot in the first row illustrates the correlation between x1 & x2; The right plot in the first row illustrates the correlation between x1 & x3; The left plot in the second row illustrates the correlation between x1 & x2 once more and so on.

The PC1, PC2 and PC3 are represented in a dynamic 3D plot.

What | value

:---------------------------|:-----------------

Folder | `r subfolder`

PC1, PC2 and PC3 Pairs plot | C-pairs.png

```{r 5.8-build-PCA-Part-C , warning=warning\_flag, echo = echo\_flag }

#

# [1] Build CumSum Plot C

#

for (iplot in 1:2) {

if (iplot == 1) {

jpeg(paste(proj\_folder,subfolder,"C-AI-AO-pairs.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

}

pairs(pca$x[,1:3],col=fit$classification)

if (iplot == 1) {

dev.off()

}

}

```

## Build AI-AO-T2

The PC1, PC2 and PC3 are saved

What | value

:---------------------------|:-----------------

Folder | `r subfolder`

PC1, PC2 and PC3 Pairs plot | 09-AI-AO-PCA.csv

```{r 5.9-Build T2, warning=warning\_flag, echo = echo\_flag}

#

# Build CumSum Plot C

#

pc1<-pca$x[,1]

pc2<-pca$x[,2]

pc3<-pca$x[,3]

t2 <<- sqrt(pc1^2+pc2^2+pc3^2)

tempfile = paste(proj\_folder,subfolder,"09-03-AI-AO-PCA.csv",sep="")

write.csv(pca$x[,1:5], tempfile, row.names=F)

df\_pc <- NULL

df\_pc <- data.frame(pca$x[,1:10])

df\_pc$TimeStamp <- run\_data$TimeStamp

df\_pc$T2 <- t2

tempfile = paste(proj\_folder,subfolder,"10-03-AI-AO-PCA-T2.csv",sep="")

write.csv(df\_pc, tempfile, row.names=F)

df\_T2summary$AIAOT2 <- t2

```

## Plot AI-AO-T2

The PC1, PC2 and PC3 are saved

What | value

:-------------------------|:-----------------

Folder | `r subfolder`

D-T2-PCs.png | Plot of T2

```{r 5.10-Build T2 , warning=warning\_flag, echo = echo\_flag}

par(mfrow = c(1,1))

jpeg(paste(proj\_folder,subfolder,"D-AI-AO-T2-PCs.png",sep=""),width=11.0,height=8.0,units="in",res=1200)

#

# Plot T2

#

par(fig = c(0.0, 0.7, 0.60, 1.0))

plot(t2,xlab="",xaxt="n", col = fit$classification, ylab=" Hotelling's T2")

par(fig = c(0.55, 1.0, 0.60, 1.0), new=TRUE)

p3d(pc1,pc2,pc3,d0=0.5,h=1.0)

#

# plot PC1 and PC2 vs PC1

#

par(fig = c(0.0, 0.7, 0.4, 0.8), new=TRUE)

plot(pc1,xlab="",xaxt="n",col = fit$classification)

par(fig = c(0.6, 1.0, 0.4, 0.8), new=TRUE)

plot(pc2,pc1,xlab="",ylab="", yaxt="n", col = fit$classification)

#

# plot PC2 and PC3 vs PC2

#

par(fig = c(0.0, 0.7, 0.2, 0.6), new=TRUE)

plot(pc2,xlab="",xaxt="n",col = fit$classification)

par(fig = c(0.6, 1.0, 0.2, 0.6), new=TRUE)

plot(pc3,pc2 ,xlab="",ylab="", yaxt="n",col = fit$classification)

#

# plot PC3 and PC3 vs PC1

#

par(fig = c(0.0, 0.7, 0.0, 0.4), new=TRUE)

plot(pc3,col = fit$classification)

par(fig = c(0.6, 1.0, 0.0, 0.4), new=TRUE)

plot(pc3,pc1,ylab="", yaxt="n",col = fit$classification)

dev.off()

```

##Plot AI-AO-T2 Dynamic

```{r 5.11 -Build T2, out.height="400px", warning=warning\_flag, echo = echo\_flag}

par(mfrow = c(4,1))

#

# Plot T2

#

df\_t2 = data.frame(t2)

df\_t2$TimeStamp <- run\_data$TimeStamp

df\_t2$TimeStamp <- strptime(df\_t2$TimeStamp, date\_format,tz="GMT")

xts\_df\_t2 <- xts(df\_t2[,1] , df\_t2$TimeStamp)

dygraph(xts\_df\_t2, main = "AI-AO-T2 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

##Plot AI-AO-PC1 Dynamic

```{r 5.12 -Build PC1, out.height="400px", warning=warning\_flag, echo = echo\_flag}

#

# Plot PC1

#

df\_pc1 = data.frame(pc1)

df\_pc1$TimeStamp <- run\_data$TimeStamp

df\_pc1$TimeStamp <- strptime(df\_pc1$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc1 <- xts(df\_pc1[,1] , df\_pc1$TimeStamp)

dygraph(xts\_df\_pc1, main = "AI-AO-PC1 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

##Plot AI-AO-PC2 Dynamic

```{r 5.12 -Build PC2, out.height="400px", warning=warning\_flag, echo = echo\_flag}

#

# Plot PC2

#

df\_pc2 = data.frame(pc2)

df\_pc2$TimeStamp <- run\_data$TimeStamp

df\_pc2$TimeStamp <- strptime(df\_pc2$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc2 <- xts(df\_pc2[,1] , df\_pc1$TimeStamp)

dygraph(xts\_df\_pc2, main = "AI-AO-PC2 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

##Plot AI-AO-PC3 Dynamic

```{r 5.13-Build PC3, out.height="400px", warning=warning\_flag, echo = echo\_flag}

#

# Plot PC1

#

df\_pc3 = data.frame(pc3)

df\_pc3$TimeStamp <- run\_data$TimeStamp

df\_pc3$TimeStamp <- strptime(df\_pc3$TimeStamp, date\_format,tz="GMT")

xts\_df\_pc3 <- xts(df\_pc3[,1] , df\_pc3$TimeStamp)

dygraph(xts\_df\_pc3, main = "AI-AO-PC3 Data", ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "follow") %>%

dyRangeSelector(height = 20)

```

## Analyzing AI-AO=PCA

```{r 5.14-build-PCA-Part-D,width=11.0,height=8.0}

#

# [3] Loading

#

rotation<- pca$rotation

absrotation <- abs(pca$rotation)

#dividing each column by sum

load\_data <- sweep(absrotation,2,colSums(absrotation),"/")

tempfile = paste(proj\_folder,subfolder,"06-03-AI-AO-load.csv",sep="")

write.csv(load\_data, tempfile, row.names=T)

tempfile = paste(proj\_folder,subfolder,"07-03-AI-AO-load-summary.csv",sep="")

write.csv(summary(load\_data), tempfile, row.names=T)

for (i in 1:3)

{

#

# extract the load 1 , Load 2 and Load 3

#

ord<- order(load\_data[,i],decreasing = T)

sloading <- load\_data[ord,]\*100

if (i == 1 ) {

myrow\_1 <- row.names(sloading)

myrow\_1[1:16]

sortdata <- run\_data[,myrow\_1]

sortdata$TimeStamp <- run\_data$TimeStamp

sortdata$TimeStamp <- strptime(sortdata$TimeStamp, date\_format,tz="GMT")

}

if (i == 2 ) {myrow\_2 <- row.names(sloading)}

if (i == 3 ) {myrow\_3 <- row.names(sloading)}

for (iplot in 1:1) {

tempfile = paste(proj\_folder,subfolder,"08-03-AI-AO-PCA",i,"-load-sorted.csv",sep="")

write.csv(sloading, tempfile, row.names=T)

par(mfrow = c(1,1))

par(fig = c(0.1, 1.0, 0.0, 1.0))

bplt <-barplot(sloading[1:25,i], las=2, horiz=T, col = "blue", xlab=paste0("Sorted Loading for [",i,"]"))

text(x= sloading[1:25,i]+0.1, y= bplt, labels=as.character(round(sloading[1:25,i],digits=2)), xpd=TRUE)

par(mar=c(5, 4, 4, 4) + 0.1)

}

}

```

The PC1, PC2 and PC3 are saved

What | value

:-------------------------|:-----------------

Folder | `r subfolder`

Load Data | 06-AI-AO-load.csv

Load data summary | 07-AI-AO-load-summary.csv

## AI-AO-Main Tag Contributer

Based on the analysis above the following 10 tags contributes to the variations.

```{r 5.15}

i <- 1

j <- 5

xts\_sortdata <- xts(sortdata[,c(i:j)] , sortdata$TimeStamp)

myrow\_1[i]

dygraph(xts\_sortdata, main = paste0("AI-AO Tag ",myrow\_1[i]), ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "always") %>%

dyRangeSelector()

```

```{r 5.16}

i <- 1

xts\_sortdata <- xts(sortdata[,c(i:i)] , sortdata$TimeStamp)

myrow\_1[i]

dygraph(xts\_sortdata, main = paste0("Tag ",myrow\_1[i]), ylab = "Y") %>%

dyOptions(stepPlot = TRUE) %>%

dyLegend( show = "always") %>%

dyRangeSelector()

```

```{r 5.17- execution Time}

tempfile = paste0(proj\_folder,subfolder)

dir(tempfile)

end\_time <- Sys.time()

cat(format(Sys.time(),usetz = TRUE))

cat(paste("Program Execution Time :", format(end\_time-start\_time) ,sep=""), sep="\n")

```