An Introduction To Using NAPPA To Pre-Process NanoString nCounter data

Chris Harbron, Mark Wappett

March 3, 2015

1 Introduction

NAPPA is an algorithm for the pre-processing of mRNA data from the NanoString nCounter software, working from the output RCC files saved as a tab delimited text file and imported into R.

NAPPA has been optimised to generate high quality robust data for subsequent analyses by optimizing steps to reduce bias and variability and reducing arbitrary decisions. The NAPPA function also provides options for how each algorithmic step is performed.

2 Getting Started

We illustrate NAPPA with data from an experiment on 50 lung cancer samples as well as a dilution series experiment.

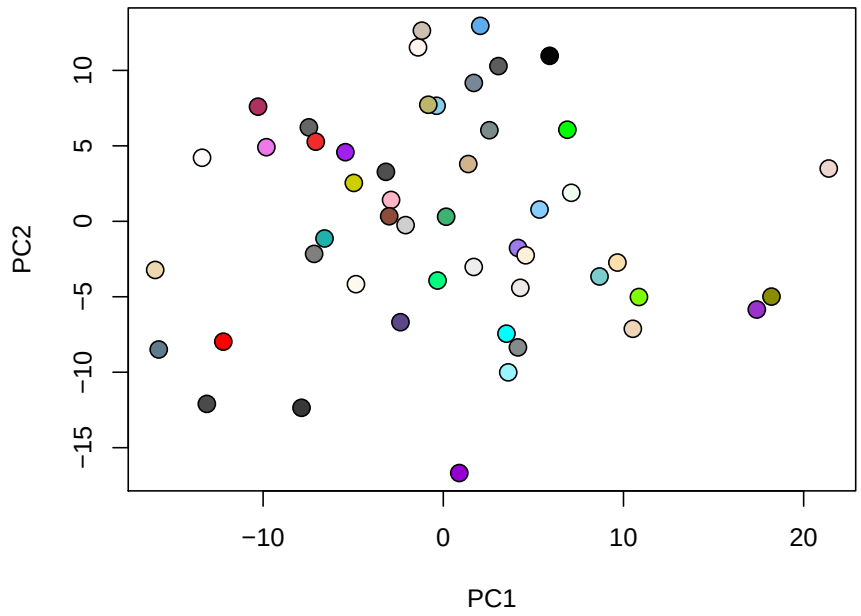
To generate basic output using the default options:

```
> require(NAPPA)
> data(NS.Lung)
> NAPPA.Lung.simple <- NAPPA(NS.Lung , tissueType = "tumour")
```

This generates a matrix of gene expression values which can then be used for downstream analysis. For example we may wish to perform a Principal Components Analysis on the data and look at the score plot.

```
> lungsamplecolours <- sample(colors(),ncol(NAPPA.Lung.simple))

> plot(prcomp(t(NAPPA.Lung.simple))$x[,1:2], pch=21 ,
+ bg=lungsamplecolours , cex=1.5)
```



We can request more detailed output using the output argument.

```
> NAPPA.Lung.detailed <- NAPPA(NS.Lung , tissueType = "tumour" ,
+  output=c("All","Steps"))
```

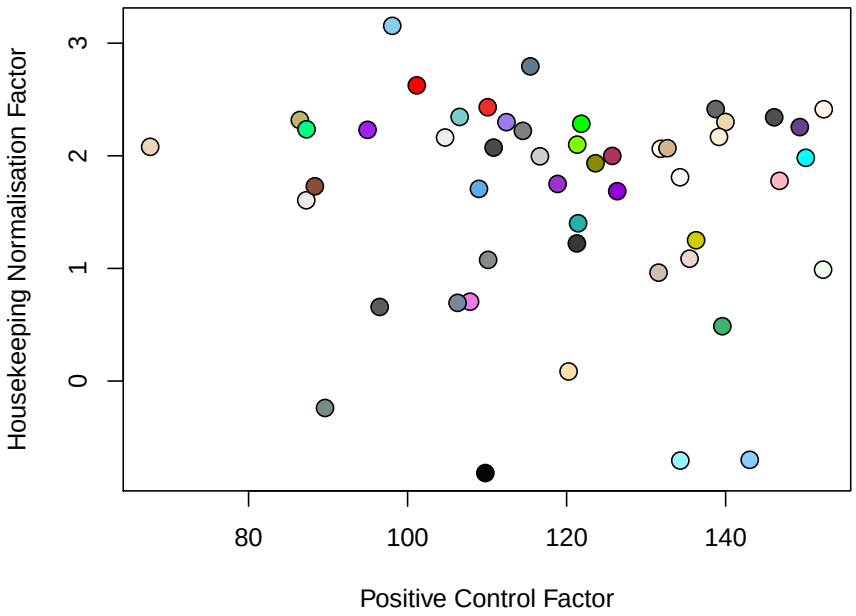
The output of NAPPA is now a list where the matrix of gene expression values is the first item of the list with name GeneExpression.

```
> identical( NAPPA.Lung.simple , NAPPA.Lung.detailed$GeneExpression )
```

```
[1] TRUE
```

The more detailed output can be used to generate diagnostics. For example if we plot the internal positive control factor against the external housekeeping normalisation factor we see that they are uncorrelated. This plot would also highlight any samples with low values in either control which may impact confidence in data from these samples.

```
> plot(NAPPA.Lung.detailed$PosFactor , NAPPA.Lung.detailed$HousekeepingFactor,
+ xlab="Positive Control Factor" , ylab="Housekeeping Normalisation Factor" ,
+ pch=21 , bg=lungsamplecolours  , cex=1.5)
```



3 Sequential Analyses

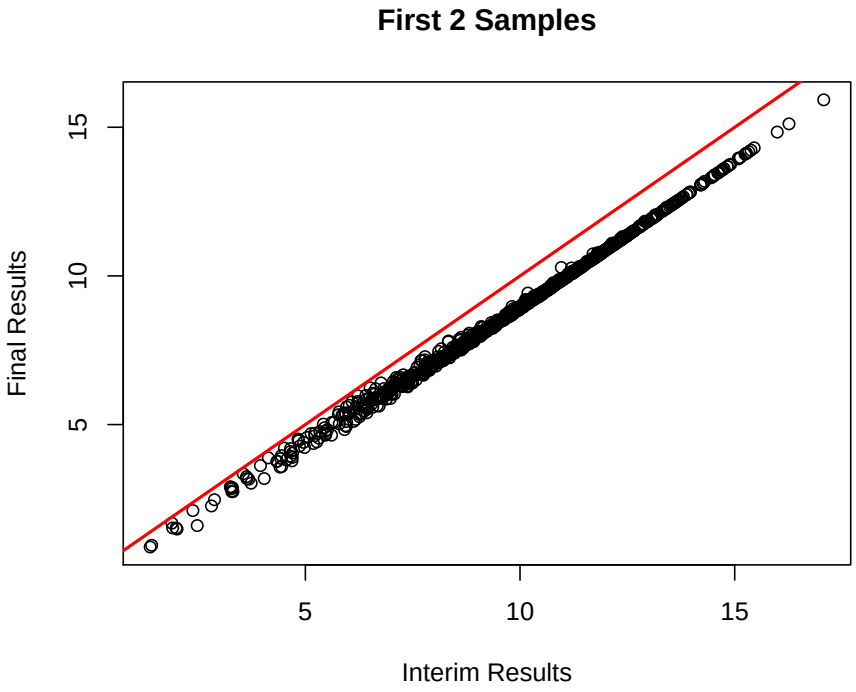
NAPPA also allows data to be run in batches. As an extreme example, suppose the first two samples had been run as an initial batch, followed by the remaining 48 samples in a subsequent batch.

If we were to run NAPPA on the first batch (the first two samples):

```
> NAPPA.Lung.batch1 <- NAPPA(NS.Lung[,1:5] , tissueType = "tumour" ,
+ output="All")
```

Then the results of these samples would change slightly when the second batch of data was included in the analysis due to the parameters relating to the centering and shrinkage of the housekeeping genes being updated with more data.

```
> plot(NAPPA.Lung.batch1$GeneExpression , NAPPA.Lung.simple[,1:2],
+ xlab="Interim Results" , ylab="Final Results" , main="First 2 Samples")
> abline(0 , 1 , col="RED" , lwd=2)
```



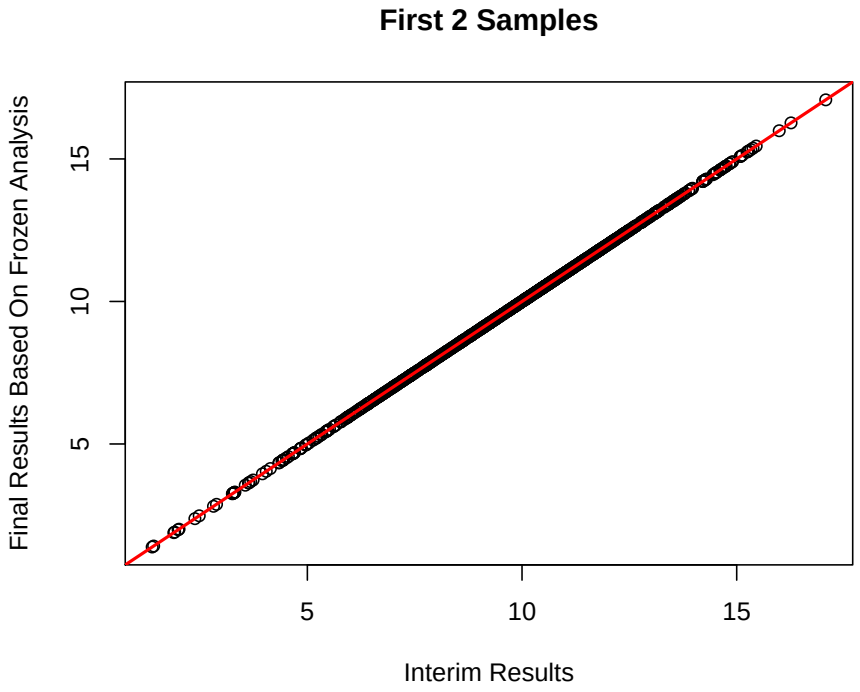

```
> identical( NAPPA.Lung.batch1$GeneExpression , NAPPA.Lung.simple[,1:2] )
```

```
[1] FALSE
```

Using the NReferenceSamples argument, NAPPA allows the final analysis to be based upon the parameters derived from the interim analysis, so that the interim data values remain unchanged in the final analysis.

```
> NAPPA.Lung.frozen1 <- NAPPA(NS.Lung , tissueType = "tumour" , NReferenceSamples = 2)
```

```
> plot(NAPPA.Lung.batch1$GeneExpression , NAPPA.Lung.frozen1[,1:2],
+ xlab="Interim Results", ylab="Final Results Based On Frozen Analysis",
+ main="First 2 Samples")
> abline(0 , 1 , col="RED" , lwd=2)
```



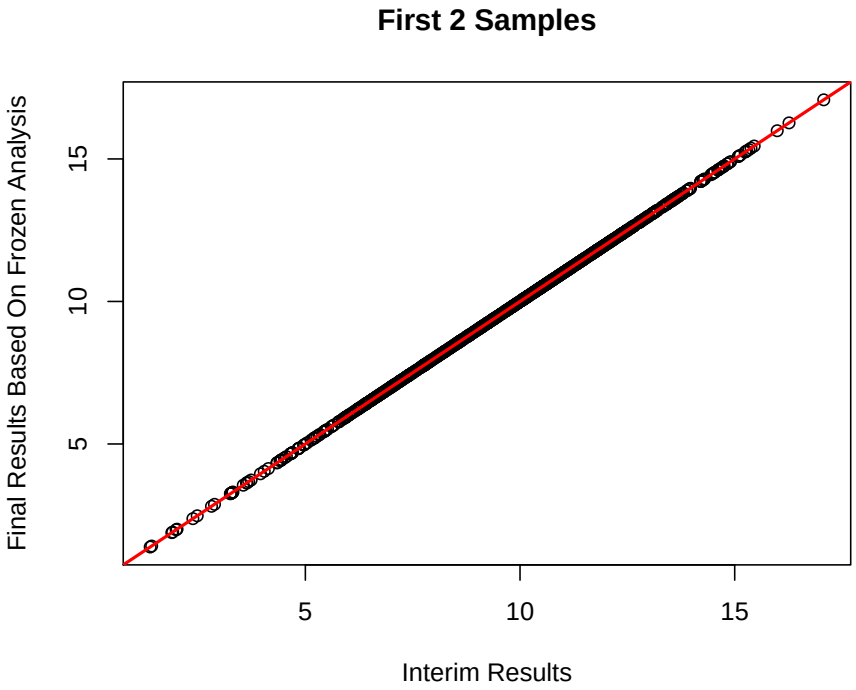

```
> identical( NAPPA.Lung.batch1$GeneExpression , NAPPA.Lung.frozen1[,1:2] )

[1] TRUE
```

The same effect can be achieved in a more advanced manner by directly passing the parameters betas and hknormfactor.mean from the interim analysis to the final analysis.

```
> NAPPA.Lung.frozen2 <- NAPPA(NS.Lung , tissueType = "tumour" ,
+  betas=NAPPA.Lung.batch1$Betas ,
+  hknormfactor.mean=NAPPA.Lung.batch1$HousekeepingFactor.Mean)

> plot(NAPPA.Lung.batch1$GeneExpression , NAPPA.Lung.frozen2[,1:2],
+ xlab="Interim Results" , ylab="Final Results Based On Frozen Analysis",
+ main="First 2 Samples")
> abline(0 , 1 , col="RED" , lwd=2)
```




```
> identical( NAPPA.Lung.batch1$GeneExpression , NAPPA.Lung.frozen2[,1:2] )

[1] TRUE
```

4 A Dilution Data Set

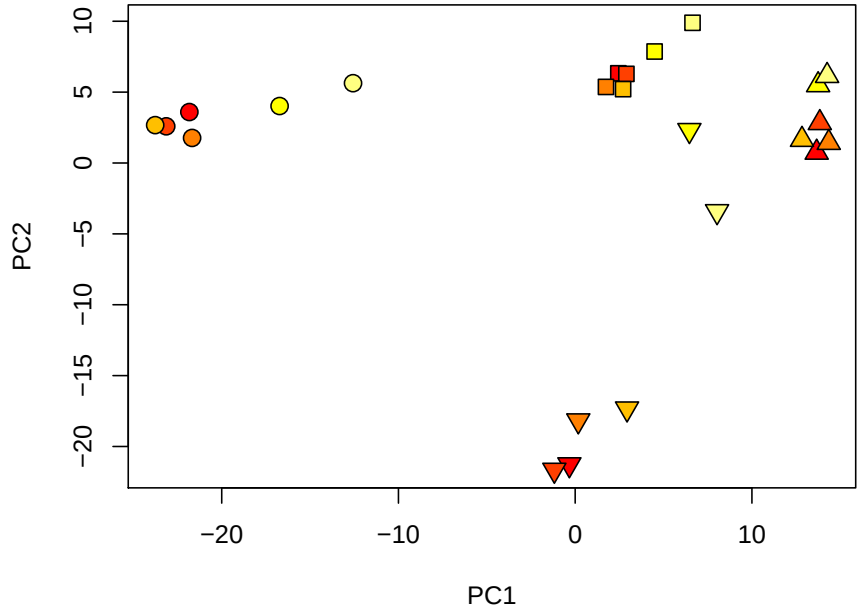
To explore the robustness of the NanoString platform to variability in the quantity of input mRNA, four of the lung tumour samples were run using a two-fold dilution series, at concentrations of 100ng, 50ng, 25ng, 12.5ng, 6.25ng + 3.125ng. In order to analyse this data, the slope parameters (betas) are taken from the larger dataset as these will be more representative of the overall behaviour of the genes rather than from this dataset where the raw count levels have been lowered through the study design.

```
> data(NS.Dilution)
> NAPPA.Dilution <- NAPPA(NS.Dilution , betas=NAPPA.Lung.detailed$Betas,
+ output="All")
```

A Score Plot of a Principal Components Analysis of these results with different samples being represented by different shapes and concentrations being represented by a colour gradient from red (100ng) to yellow (3.125ng) shows the 4 individual samples generally clustering together, with greatest deviations being seen in the lowest concentration samples, including two large outliers at the lowest concentrations for one of the samples.

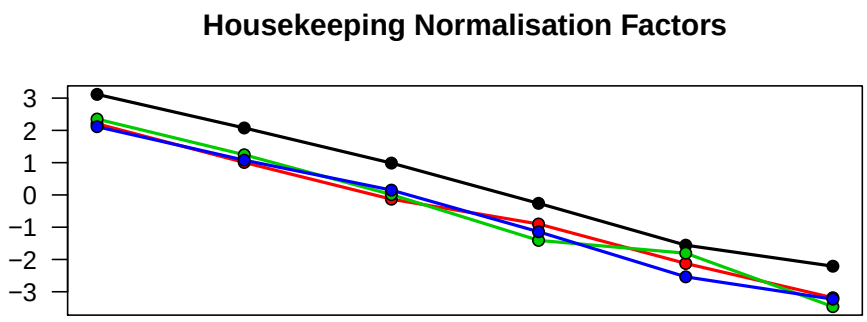
```
> dilutioncolours <- rep(heat.colors(6),4)
> dilutionshapes <- rep(c(21,22,24,25) , each=6)

> plot(prcomp(t(NAPPA.Dilution$GeneExpression))$x[,1:2],
+ pch=dilutionshapes , bg=dilutioncolours , cex=1.5)
```

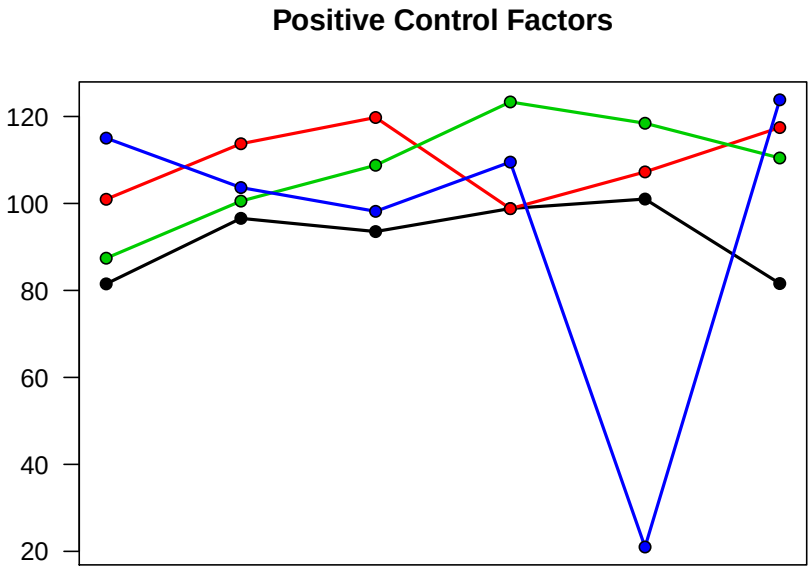


We can plot both the housekeeping normalisation and positive control factors against concentration for each sample. The housekeeping normalisation factor shows a strong relationship with concentration with increased variability at the lowest concentrations, whilst the internal positive control factor shows no relationship, but does highlight one of the outliers previously observed in the PCA, possibly suggesting a failed run.

```
> opar <- par(mfrow=c(2,1) , mar=c(1,3,4,1))
> plot(c(1,6) , range(NAPPA.Dilution$HousekeepingFactor) , type="n",
+ xlab="" , ylab="" , axes=F , main="Housekeeping Normalisation Factors")
> box()
> axis(2 , las=2)
> for(i in 1:4) {
+   lines(1:6 , NAPPA.Dilution$HousekeepingFactor[6*i + (-5:0)] , col=i , lwd=2)
+   points(1:6 , NAPPA.Dilution$HousekeepingFactor[6*i + (-5:0)] , bg=i , pch=21)
+ }
```




```
> plot(c(1,6) , range(NAPPA.Dilution$PosFactor) , type="n",
+ xlab="" , ylab="" , axes=F , main="Positive Control Factors")
> box()
> axis(2 , las=2)
> for(i in 1:4) {
+   lines(1:6 , NAPPA.Dilution$PosFactor[6*i + (-5:0)] , col=i , lwd=2)
+   points(1:6 , NAPPA.Dilution$PosFactor[6*i + (-5:0)] , bg=i , pch=21)
+ }
> par(opar)
```



5 Changing The Pre-Processing Algorithm

Each of NAPPAs steps contains options which can be used to change the algorithm being performed. So for example we may investigate the impact of the shrunken housekeeper normalisation rather than a simpler subtraction normalisation, using the hk.method argument.

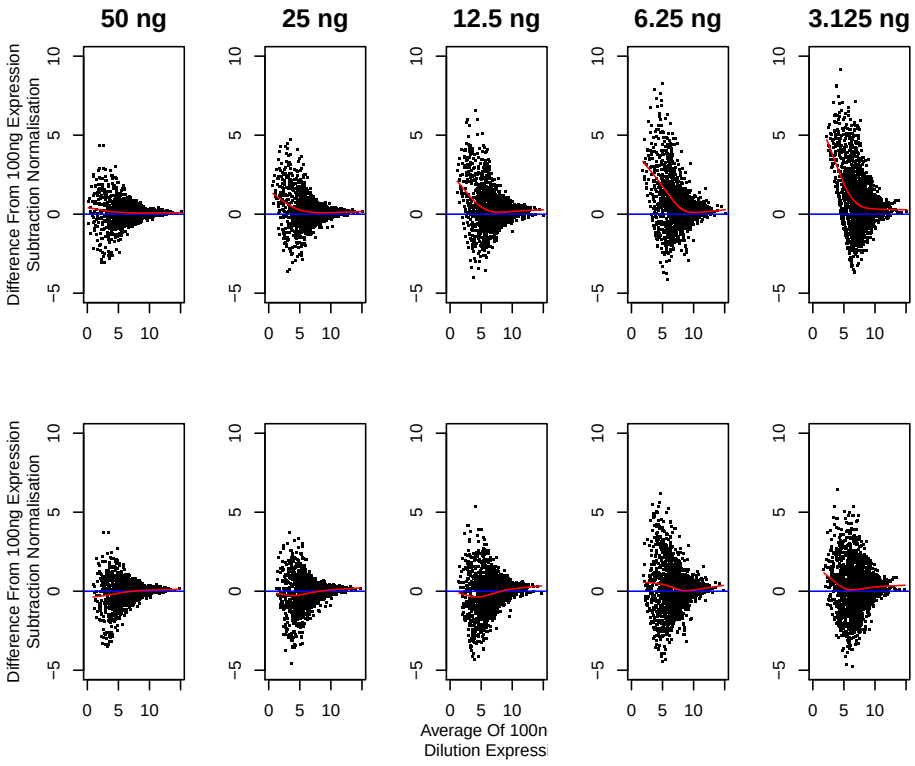
```
> NAPPA.Dilution.subtract <- NAPPA(NS.Dilution , hk.method="subtract",
+ output="All")
```

We can then visually observe the impact of this for the different concentrations using a series of Bland-Altman plots, plotting for each dilution the difference in expression to the sample run at 100ng against the average of the sample run diluted and at 100ng.

```
> blandaltmanaplot <- function(x,y,...) {
+   d <- x-y
+   a <- 0.5*(x+y)
+   plot( a , d , pch="." , ...)
+   abline(h=0 , col="BLUE")
+   lines(smooth.spline(y=d , x=a , df=4) , col="RED")
+ }

> opar <- par(mfcol=c(2,5) , mar=c(4,4,2,0) , oma=c(0,0,0.5,0.5))
> for(i in 1:5)
```

```
+ {
+ blandaltmanaplot(c(NAPPA.Dilution.subtract$GeneExpression[,i+c(1,7,13,19)]),
+ c(NAPPA.Dilution.subtract$GeneExpression[,c(1,7,13,19)]) , ylim=c(-5,10),
+ xlim=c(0,15) , ylab="" , xlab="")
+ title(paste(c(50,25,12.5,6.25,3.125)[i],"ng",collapse="") , line=1,
+ cex.main=1.5)
+ if(i==1) title(ylab="Difference From 100ng Expression\nSubtraction Normalisation" ,
+ line=2)
+
+ blandaltmanaplot( c(NAPPA.Dilution$GeneExpression[,i+c(1,7,13,19)]),
+ c(NAPPA.Dilution$GeneExpression[,c(1,7,13,19)]) , ylim=c(-5,10),
+ xlim=c(0,15) , ylab="" , xlab="")
+ if(i==1) title(ylab="Difference From 100ng Expression\nSubtraction Normalisation" ,
+ line=2)
+ if(i==3) title(xlab="Average Of 100ng &\nDilution Expression")
+ }
> par(opar)
```



These show a reduced bias, particularly with the results from lower expressing genes from low concentration samples with the shrunken housekeeping normalisation.

This also allows wider comparison with other NanoString pre-processing algorithms. If we load the dilution data as processed by NanoString's nsolver programme:

```
> data(nsolver.Dilution)
```

And we also process the data through NanoStringNorm using the settings used in NanoStringNorm's example documentation, with a little bit of data processing to get the outputs from all packages into the same format.

```
> require(NanoStringNorm)
> NS.Dilution.NSN <- cbind(NS.Dilution[-(1:21),1:3],
+ sapply(NS.Dilution[-(1:21),-(1:3)] , as.numeric))
> colnames(NS.Dilution.NSN)[1:3] <- c('Code.Class', 'Name', 'Accession')
> colnames(NS.Dilution.NSN)[4:27] <- colnames(nsolver.Dilution)
> NSN.Dilution <- NanoStringNorm(NS.Dilution.NSN , CodeCount = 'geo.mean',
+                            Background = 'mean',
+                            SampleContent = 'housekeeping.geo.mean',
+                            round.values = TRUE,
+                            take.log = TRUE)$normalized.data

##############################
### NanoStringNorm v1.1.17 ###
##############################

There are 24 samples and 461 Endogenous genes

CodeCount: The following samples have positive normalization factors outside the
         recommended range of (0.3 to 3).  Consider removing them.

    pos.norm.factor
4_5            4.74

Background: After correction 24 samples and 457
        Endogenous genes have less than 90% missing.

SampleContent: The following samples have sample/rna content greater than
        3 standard deviations from the mean.

    rna.zscore
1_1       3.27

log: Setting values less than 1 to 1 in order to calculate the log in positive space.

> NSN.Dilution <- NSN.Dilution[NSN.Dilution$Code.Class=="Endogenous",-(1:3)]
> all(rownames(NSN.Dilution) == rownames(NAPPA.Dilution$GeneExpression))

[1] TRUE

> nsolver.Dilution <- nsolver.Dilution[rownames(NAPPA.Dilution$GeneExpression),]
```

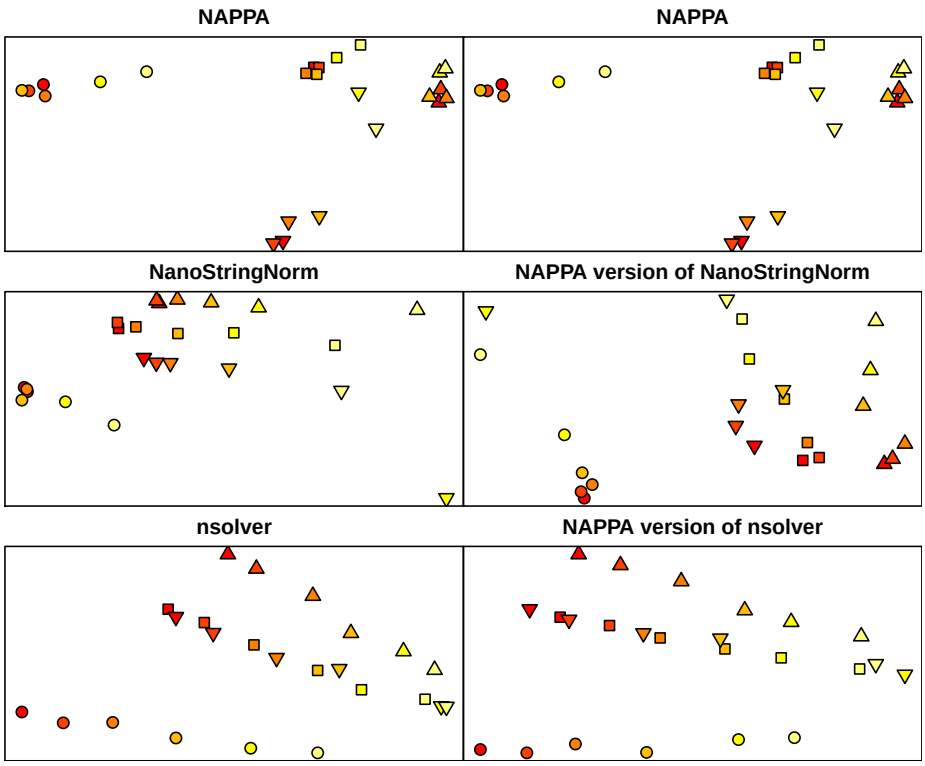
We can also specify arguments in NAPPA to run "nsolver-like" and "NanoStringNorm-like" analyses, although we were not able to reproduce these exactly.

```
> NAPPA.Dilution.NSN <-  NAPPA(NS.Dilution , scaleFOV=F,
+ background.method="subtract.global" , nposcontrols=6 ,
+ poscontrol.method="geometric.mean" , hk.method="subtract",
```

```
+ output="All")
> NAPPA.Dilution.NS <- NAPPA(NS.Dilution , scaleFOV=F,
+ background.method="none" , nposcontrols=6,
+ poscontrol.method="geometric.mean" , hk.method="subtract",
+ output="All")
```

Principal Components Analyses of each of these pre-processing methods give a visual indication of the variability of the different algorithms across the different dilutions.

```
> opar <- par(mfrow=c(3,2) , mar=c(0,0,2,0))
> plot(prcomp(t(NAPPA.Dilution$GeneExpression))$x[,1:2] , pch=dilutionshapes,
+ bg=dilutioncolours , cex=1.5 , axes=F , main="NAPPA")
> box()
> plot(prcomp(t(NAPPA.Dilution$GeneExpression))$x[,1:2] , pch=dilutionshapes,
+ bg=dilutioncolours , cex=1.5 , axes=F , main="NAPPA")
> box()
> plot(prcomp(t(NSN.Dilution))$x[,1:2] , pch=dilutionshapes , bg=dilutioncolours,
+ cex=1.5 , axes=F , main="NanoStringNorm")
> box()
> plot(prcomp(t(NAPPA.Dilution.NSN$GeneExpression))$x[,1:2] , pch=dilutionshapes,
+ bg=dilutioncolours , cex=1.5 , axes=F , main="NAPPA version of NanoStringNorm")
> box()
> plot(prcomp(t(log2(nsolver.Dilution)))$x[,1:2] , pch=dilutionshapes,
+ bg=dilutioncolours , cex=1.5 , axes=F , main="nsolver")
> box()
> plot(prcomp(t(NAPPA.Dilution.NS$GeneExpression))$x[,1:2] , pch=dilutionshapes,
+ bg=dilutioncolours , cex=1.5 , axes=F , main="NAPPA version of nsolver")
> box()
> par(opar)
```


NAPPA
NAPPA
NanoStringNorm
NAPPA version of NanoStringNorm
nsolver
NAPPA version of nsolver