Empirical Determination of Experimental Error Using the Twenty-Nine Duplicate Patches Within the SNAP IT8.7/4r

Ragy Isaac and Dennis Cheeseman

Keywords: characterization dataset, spectral reflectance, color reproduction, colorimetry, IT8.7/4r, principal component analysis, statistics, standard error, bootstrapping, R-programming language

Abstract

In this paper, a new approach to visualize and calculate the unavoidable error in the duplicated patches of the SNAP IT8.7/4r characterization datasets from eighty newspaper print sites is presented. This unavoidable error is due to the spectral difference between duplicated patches in the characterization datasets. The error is linearly transformed and reported as an eigenvector/eigenvalue approximating the total error. The analysis shows that the spectral error of the duplicated patches is not similar and a systematic error bias pattern is observed. The spectral error estimation procedure includes data manipulation, bootstrapping, and principal component analysis using the R-programming language. The R-script is included as well as a script to create artificial data as an effort to stimulate further research.

Introduction

The unavoidable error:

When an operation or experiment is repeated under what are, as nearly as possible, the same conditions, the observed results are never quite identical. The fluctuation that occurs from one repetition to another is referred to by a number of terms: noise, experimental variation, experimental error, or merely error. Noise obscures our vision, obfuscates the dynamics, and contaminates our data. This is a challenge experimenters face daily. In a statistical context, the word error is used in a technical and emotionally neutral sense. It refers to variation that is often unavoidable. We have to deal with this pesky, real-world problem of noise. Many sources contribute to experimental error such as errors of sampling, measurement, analysis, and production equipment in a less-than-ideal environment.

Many metrics have been used to evaluate the unavoidable spectral error such as the root mean square error (RMS), the weighted RMS, and the goodness of fit coefficient (GFC). Our approach, the standard error, uses statistical techniques and bootstrapping to provide the uncertainty in the data.

Advantages and disadvantages of physiological and physical error:

It has been customary to report the unavoidable error in the physiological tristimulus color space rather than to identify physical errors in the spectral data. Separate error term per wavelength or as a linear combination of the reflectance data, are not common, difficult to interpret, requires multivariate mathematical knowledge and does not possess a physical meaning.

However, spectral data has distinct advantages over tristimulus color spaces. Separate error terms can be a useful tool for color management and benchmarking profiles. Spectral match is independent of observers and illuminants. Also, spectral data is the only true physical descriptor of the colored object prior to its transformation to color stimulus function. Once light projected on the retina has been absorbed by the eye's three groups of cones, any knowledge of its spectral composition is lost. What remains are three levels of activity in the red, green, and blue cones.

What is so important about the unavoidable error?

An error term is valuable for calculating the confidence in the calculated averages of the characterization datasets. It is also a useful tool in production when determining what action is needed to achieve a color match.

Experimental Design

For the purpose of this research, the Specifications for Newsprint Advertising Production (SNAP) organization provided eighty characterization datasets from newspaper sites which met the SNAP recommendations. These datasets were measured with X-rite's SpectroLino spectrophotometer. The SNAP committee estimates the SNAP profile based on averages of thirty-six different wavelengths.

The standard IT8.7/4r test targets contain 1,617 patches, among which are twenty-nine duplicated patches. We extracted the spectral responses of these duplicated patches from the IT8.7/4r characterization datasets. These twenty-nine different duplicated patches are used as an estimator of a typical unavoidable experimental error, which can be generalized to all patches.

The retrospective data received from the SNAP organization does not follow a randomized paired design since the duplicated patches were not randomized for every site. The decision as to the placement of the first and the second duplicated

patch members on the IT8.7/4r was NOT determined randomly. However, the duplicated patches were run in pairs. Each newspaper simultaneously printed all the twenty-nine patch pairs. Each duplicate pair was printed at the same time by the same press crew, on the same press-couple and used the same press chemicals. One can safely assume that every site can be regarded as an independent sample.

Subtracting a given patch from its duplicate:

In this research, retrospective data were analyzed. Theoretically identical patches should produce identical spectral responses. In the absence of a bias, subtracting the spectral reflectance of a given patch from its duplicate renders its average to be almost zero, and its parent distribution to have a mean of zero. However, due to the lack of randomization, one might expect to observe a patch-location bias.

The duplicated pairs were differenced to obtain the spectral error. The differencing process of this retrospective paired design has the potential of removing site-to-site variation and provides clues about the noise. To evaluate the noise, the spectral responses for each patch were subtracted from the duplicate for cyan, magenta, yellow and black percentage coverage. In this study, we have eighty IT8.7/4r sets (one per print site), and every set contains twenty nine duplicate patches. Each duplicate patch has an associated thirty-six spectral responses. This provides 1,044 datasets; each dataset contains eighty cases.

The search for mu (μ):

One may suppose that this SNAP data is merely a sample from some large hypothetical, sought after dataset. This hypothetical dataset represents the population of all possible newspapers having an unknown multivariate population parameter μ. After observing the sample values, the sample mean $\bar{x}$ is calculated. This leads to a question of its accuracy as an estimate of the true population mean. If the parameter μ can be accurately calculated, it would be possible to create the optimum color profile.

The importance of $\bar{x}$ is that it is the estimator for the unknown parameter μ. Under repeated sampling, one can quantify the uncertainty in $\bar{x}$, to estimates μ, with the standard error. In turn, the standard error allows the calculation of the confidence interval which brackets the true population parameter if the same population is repeatedly sampled.

The validity of an accurate confidence interval depends on three basic assumptions:

1. $\bar{x}$ is unbiased or nearly unbiased for μ.

2. The standard error is a good estimate of the standard deviation of the sampling distribution of $\bar{x}$.

3. The sampling distribution of $\bar{x}$ is approximately normal.

Bootstrapping to calculate the unavoidable error:

This analysis operated on the reasonable assumption that every site can be regarded as an independent random sample. The SNAP characterization datasets constitutes a relatively large sample size from eighty newspapers satisfying the SNAP recommendation. Randomness eliminates bias, validating the first assumption. However, the differenced observations should not be assumed to have a mean of zero due to the lack of randomization for each site within the IT8.7/4r.

To fulfill the above second and the third assumptions, one may resort to an important mathematical procedure called bootstrapping. This procedure does not assume any knowledge of the form of the underlying parent distribution from which the sample arose. Traditional classical statistical parameter estimates are based on the normality assumption. Bootstrap deals with non-normality and is more accurate in practice than the classical methods.

Bootstrapping substitutes computers' raw computing power for rigorous theoretical analysis. It is a proxy/estimate for the sampling distribution of the characterization dataset error term. Generically, bootstrapping includes: re-sampling the dataset a specified number of times, calculating the mean from each sample, and finding the standard error of the mean. In this paper, the SNAP characterization datasets' duplicated differenced patches were re-sampling 1,000 times.

Results and Discussion

Bootstrapping was carried out for every wavelength independently to preserve the relationship between wavelengths. An example histogram is given below of what bootstrapping accomplishes rather than attempting to present all twenty-nine duplicate patches for all thirty-six frequencies.

The data were transformed into a normal distribution as can be seen from Shapiro-Wilk normality test. The p-value before bootstrapping is 0.000037 which is highly significant indicating that the data far from normality. After bootstrapping the p-value is 0.8296 assuring normality. This is valuable for estimating an accurate standard error.

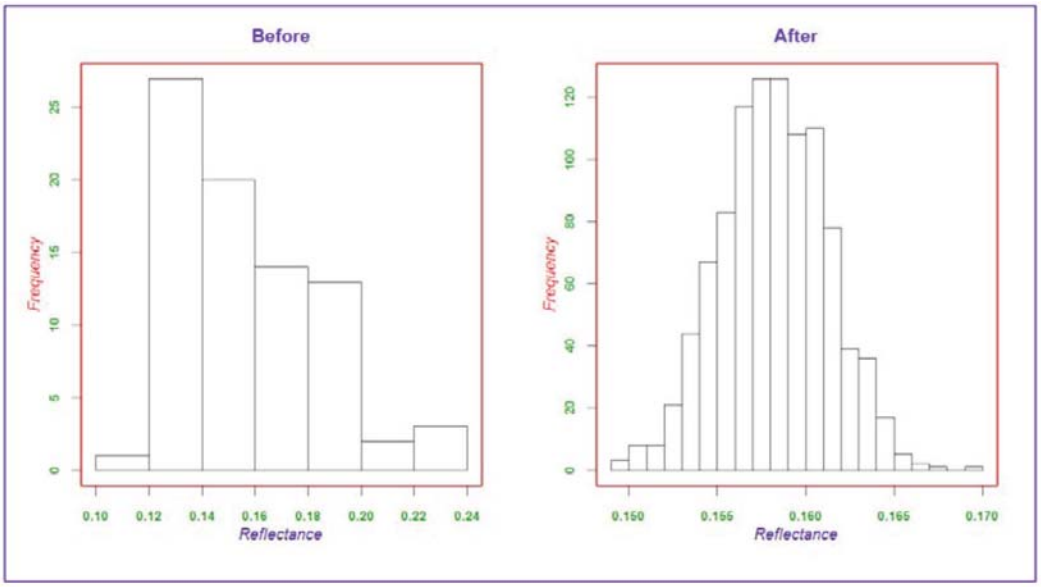

Figure 1: *10% Cyan Dot Before and After Bootstrapping*

Principal component analysis (PCA) Introduction:

It is a way of identifying and extracting patterns in correlated data. PCA is a powerful tool for analyzing and displaying patterns in high dimensional data where graphical representation is not an option. It is also valuable in reducing the number of data dimension without much loss of information.

PCA computes new uncorrelated variables called principal components (PCs) from correlated variables. The uncorrelated variables are obtained as linear combinations of the original variables. The first principal component will have the largest possible variance. The second component is computed under the constraint of being orthogonal to the first component, having the second largest variance. The other components are computed likewise, with systematically decreasing variances. Once the these PCs are made, one rotates the coordinates of all of the data points, called cases, relative to these perpendicular uncorrelated variables and then re-plot the data. By performing such a rotation, the new axes might have particular explanations. In this paper the latent PC1 axis indicates lightness, samples on the left have lower lightness and samples on the right have higher lightness. PC2 may explain the hues.

PC loadings, the eigenvectors, measure the importance of each variable (wavelength) in accounting for the variability in the PC. High correlation indicates that variables are associated with the maximum amount of variation in the dataset. Uncorrelated variables do not show a distinct pattern and do not contribute to the variation in the dataset and can be safely removed to simplify the data analyses.

PC scores, the eigenvalues, can be interpreted geometrically as the projections of the observations onto the principal components. The scores are the result of the matrix multiplication between the eigenvectors and the centered data matrix.

A SCREE plot is a plot of each component's variance against the corresponding PCs. It shows the eigenvalues decline as a function of the PCs. In correlated data, the first few PC's decline rate is fast, and then it levels off significantly signaling the maximum number of useful PCs to be extracted.

Principal component analysis (PCA) results

The R statistical programming language calculates the principal components analysis with two different commands: princomp, using the correlation matrix, and prcomp, which used the singular value decomposition. In this research, prcomp was used with the centered data matrix for the duplicated patches, their error terms, and the difference between their spectra.

PCA can model the spectra of a sample set with satisfactory accuracy. The first three principal components (PCs) appear to explain almost 99.8% of cumulative contribution of variance of spectral reflectance. According to the SCREE plot illustrated in Figure 2, color spectral reconstruction based on three sets of principal components will provide the best spectral data representation. For the purpose of brevity, we are only presenting two SCREE plots, one for a duplicate patch called, pcaA, and the other for the error difference between the duplicate patches, which is called pcaDiff. Arbitrarily, the duplicated patches were given the names 'A' and 'B'. This convention will be followed throughout this paper unless otherwise specified.

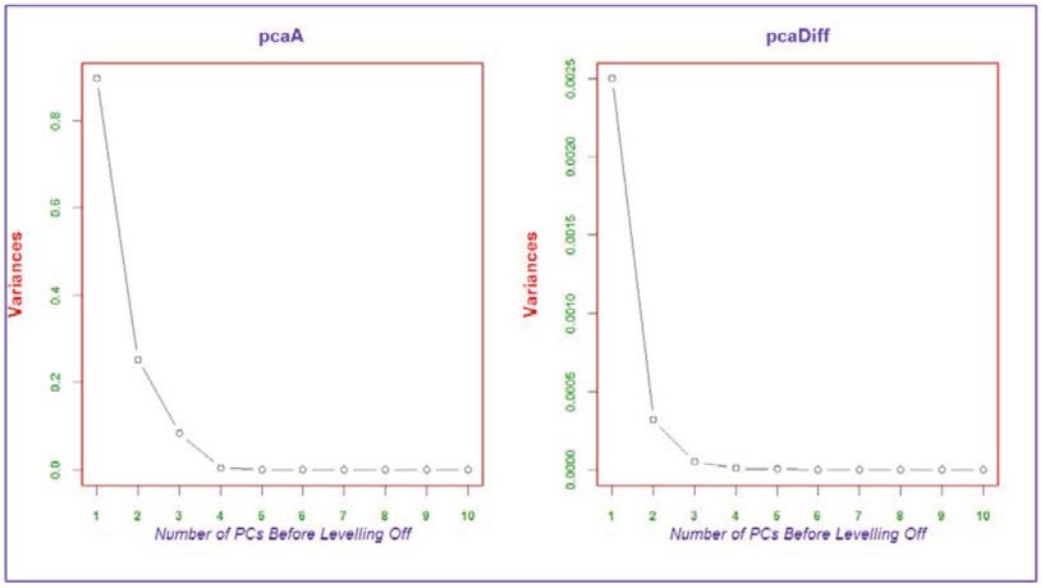

Figure 2: *SCREE Plot of a Duplicate Patch and the Error Difference between patches*

Synthesized, random data were also plotted to demonstrate the appearance of a SCREE plot for random uncorrelated data as illustrated in Figure 3. The graph below clearly shows that the variance, for the random data, does not level off. PCA calculation indicates that we need twenty-six principle components to describe 99.8% of the variance.

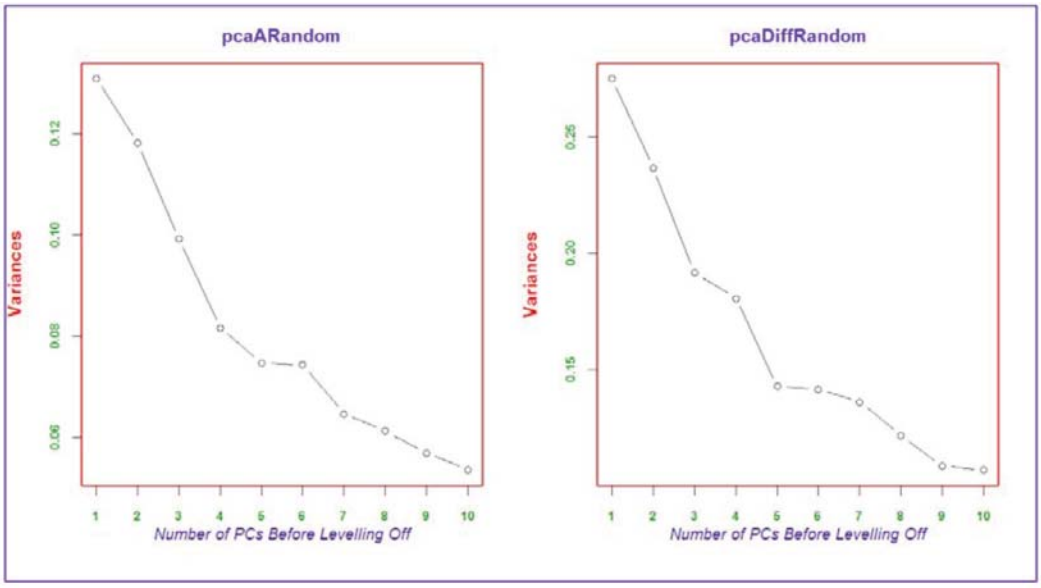

Figure 3: *SCREE Plot of synthesized Random Data*

It is also a good practice to visualize the principal components with a boxplot. Figure 4 is a boxplot for one set of the duplicated patches and the duplicated patches' error difference. The graph does not show any alarming outlier. It also shows the error compared to the actual spectra.

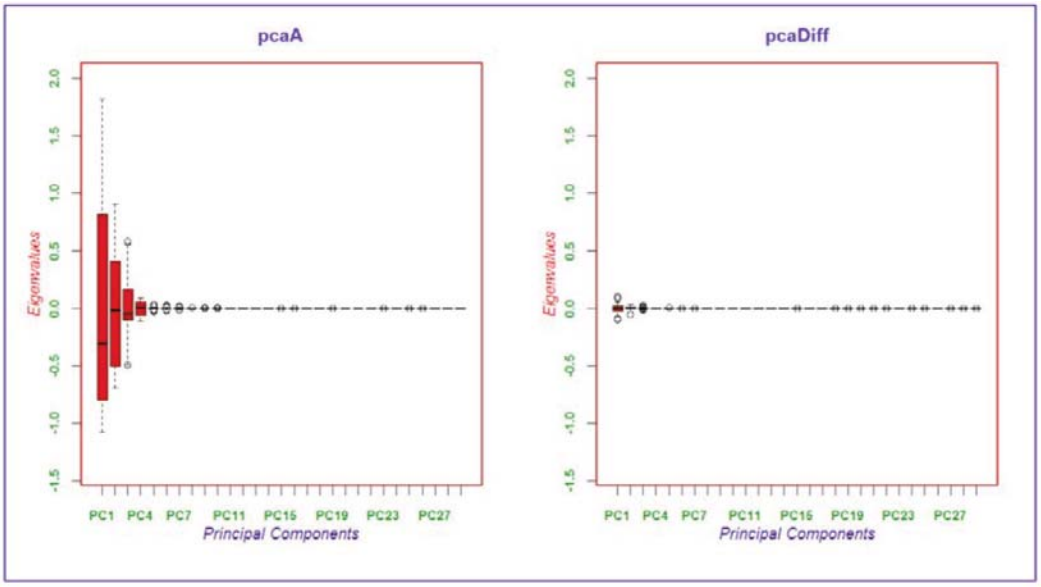

Figure 4: *Boxplot of Duplicate Patches and the Duplicated Patches' Error*

For comparison, the synthesized, random data are also plotted once again. The boxplots in Figure 5 shows multiple outliers and slow decay as compared to correlated data, which decays much faster. Also, it can be shown that differencing increased the variability.

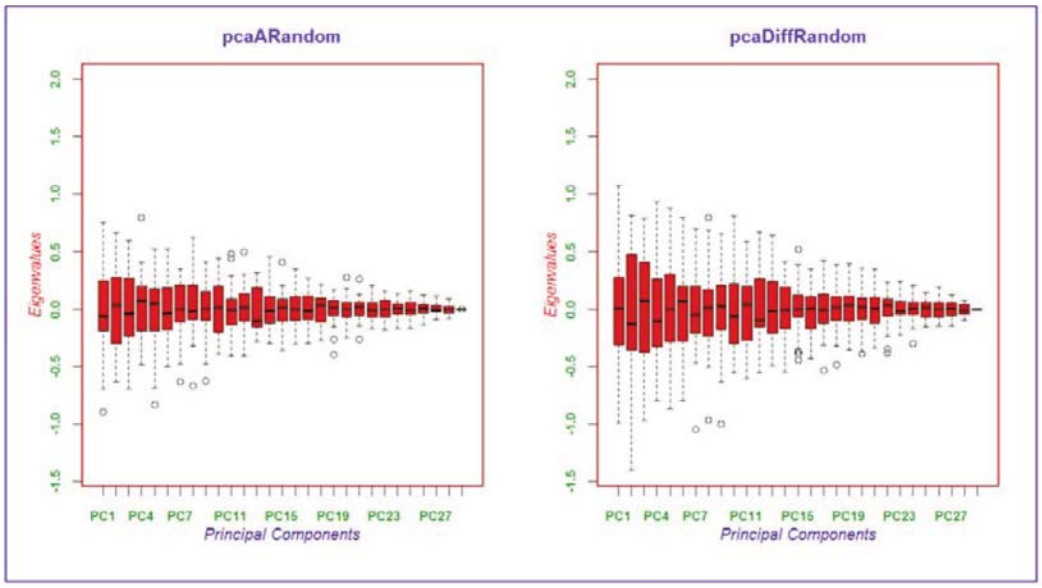

Figure 5: *Boxplot of Synthesized Uncorrelated data and their Difference*

Biplots are multivariate scatterplots. They represent both the samples and variables in the same graph. They show the score of each case (rotated patch reflectance) and the loading of each variable (wavelengths) on the first two principal components. Figure 6 illustrates PC1 on the x-axis trend together towards the negative side. In fact, all the eigenvectors are negative for the first principal component. Similarly, PC2 trends, both positive and negative are observed along the y-axis. The small angles between the variables, represented by arrows, are highly correlated.

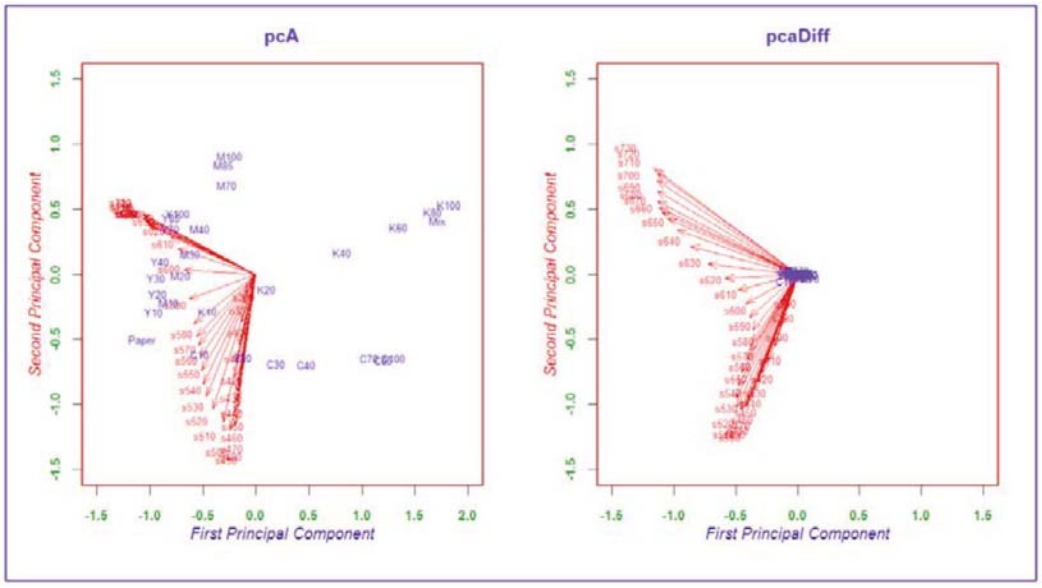

Figure 6: *Biplot of the First Two Principal Components*

Again, it is helpful to compare the above graph with a random synthesized data in Figure 7. One notices that the variables are split apart randomly due to the lack of correlation with no specific direction.

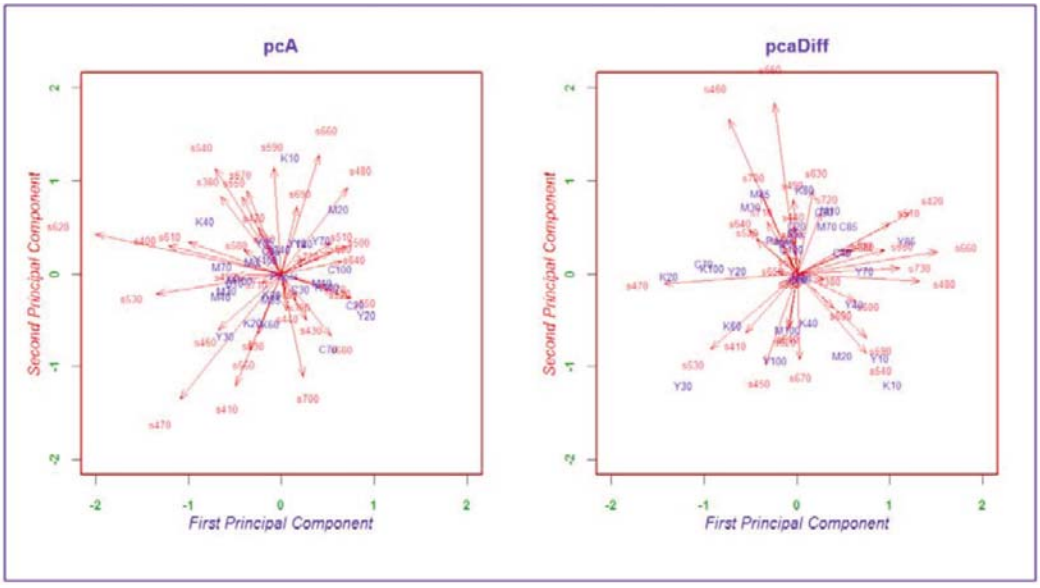

Figure 7: *Biplot of Synthesized Random Data for PC1 and PC2*

The eigenvector plot illustrated in Figure 8 appears to simulate the shape of the patches' composite spectra. This type of plot is very useful in designing and modifying test forms for profiling. Smooth curves indicate that the test form does not emphasize one tone at the cost of the other. Of course, the graph below only represents 29 patches. Other research conducted by the author but not represented in this paper, indicated that the 1,617 patches of the current IT8.7/4r need to be redesigned to ensure smooth transition between the various tones.

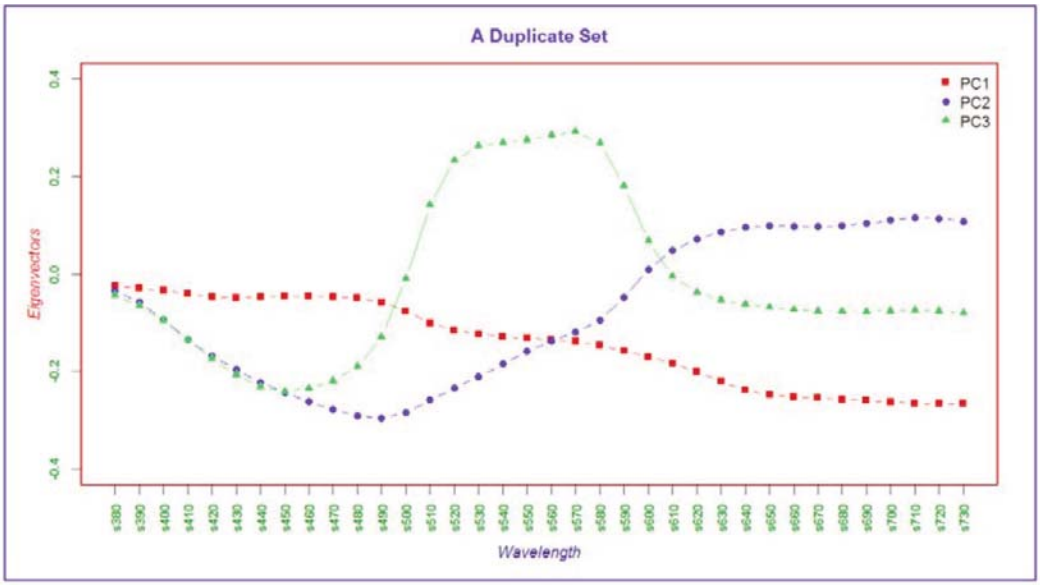

Figure 8: *Eigenvectors plot of a Duplicate Patch Set*

The eigenvalues can be thought of as the stretching of the eigenvectors. Figure 9 shows higher eigenvalue difference with magenta and yellow. The red and blue lines are not superimposed on each other for all patches.

More variability is observed with PC1. This is expected since PC1 accounts for the largest variance in the data. Comparing Figure 9 which plots PC1 and Figure 10 which plots PC2 show the contrast in variance.

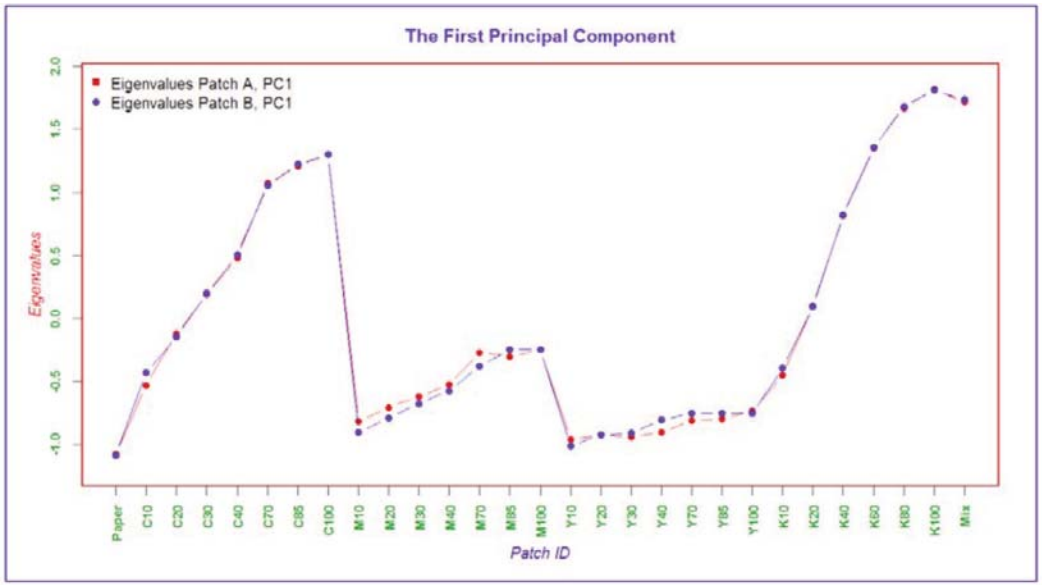

Figure 9: *Eigenvalues plot of PC1 for Both Duplicate Patches*

More variability is observed with PC1. This is expected since PC1 accounts for the largest variance in the data. Comparing Figure 9 which plots PC1 and Figure 10 which plots PC2 show the contrast in variance.

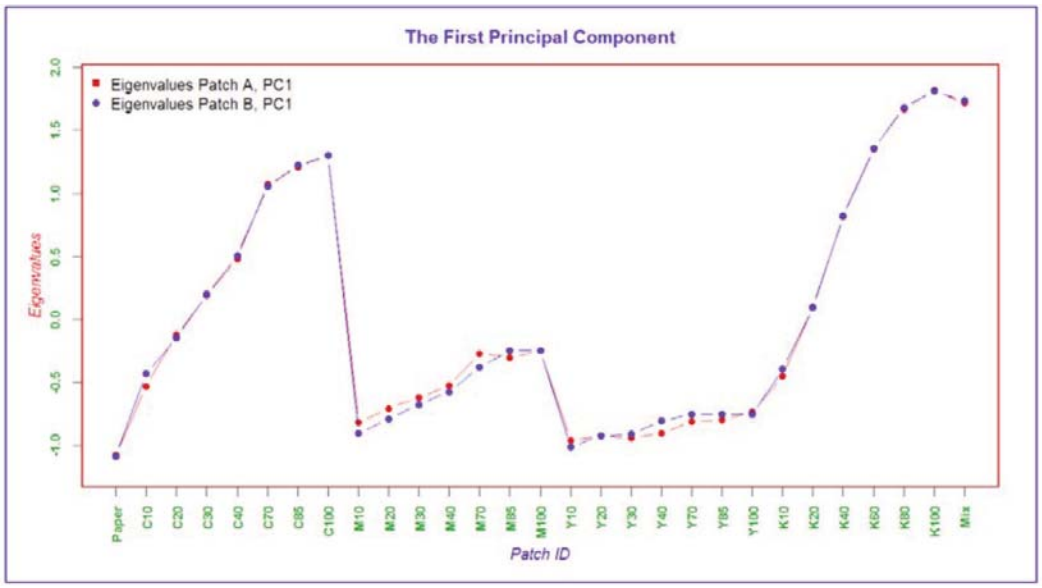

Figure 10: *Eigenvalues plot of PC2 for Both Duplicate Patches*

Next, the spectral error between the duplicated patches needs to be evaluated. The spectrum of each patch is subtracted from its duplicate patch. Figure 11 shows the boxplots of the spectral error as a function of wavelength. It is worth mentioning that subtracting a patch from its duplicate should result in a value of zero in the absence of bias. As can be seen below, there is a definitive error pattern that is believed to be a consequence of the lack of randomization of the IT8.7/4r. This lack of randomization is the reason why it was not possible to calculate an unbiased estimate for the standard error.

In the future, one would collect new data differently. It would be better to ensure that every one of the eighty newspaper sites would receive a new randomization of the IT8.7/4r target. This might provide new insights since each duplicate pair would be printed at the same time by the same press crew, on the same press-couple and use the same press chemicals.

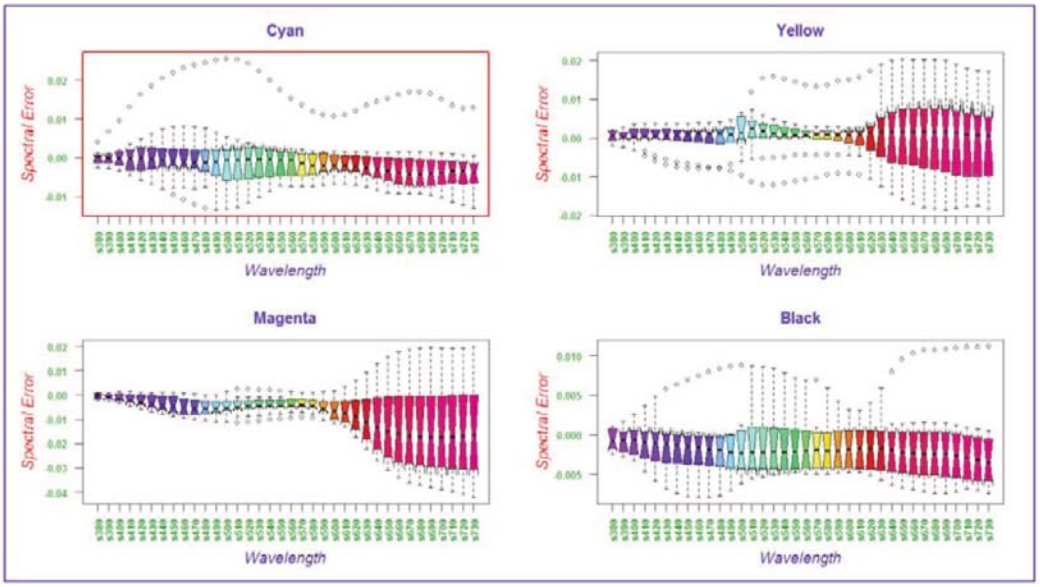

Figure 11: *Boxplot of Spectral Error between Duplicate Patches*

Finally we present the standard error for Patch 'A' for each cyan, magenta, yellow and black. As can be seen from Figures 12a though 12d, the standard errors are not similar. It depends on the patch color and the wavelength.

Figure 12a: *Plot of the Standard Error Per Patch for Duplicate Set A: Cyan Tone Ramp*

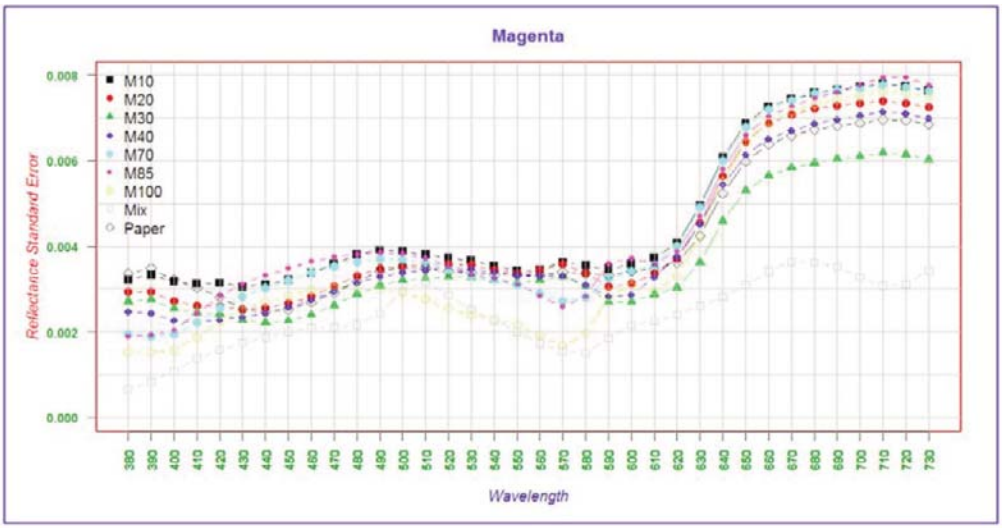

Figure 12b: *Plot of the Standard Error Per Patch for Duplicate Set A: Magenta Tone Ramp*

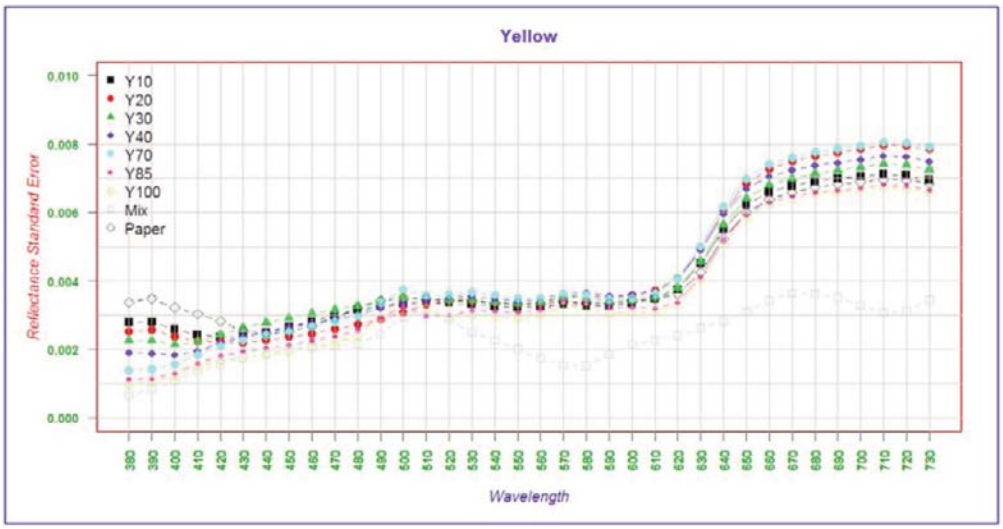

Figure 12c: *Plot of the Standard Error Per Patch for Duplicate Set A: YellowTone Ramp*

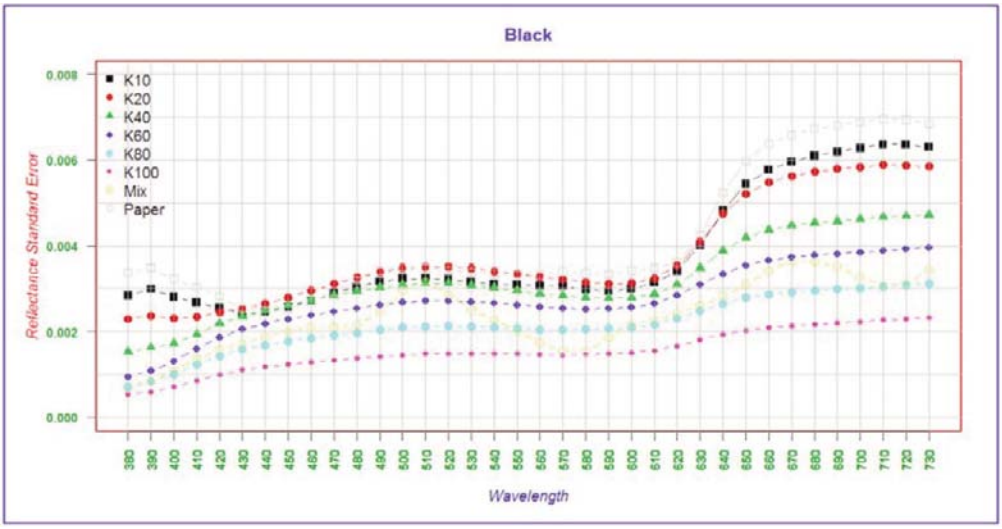

Figure 12d: *Plot of the Standard Error Per Patch for Duplicate Set A: Black Tone Ramp*

Conclusion

There is a clear bias pattern in the spectral errors; the mean-differences between identical patches are not zero. It is believed that this is the result of a location bias on the IT8.7/4r test form. The SNAP test target has been randomized once and placed on the SNAP website for download by the various newspapers printers. Multiple downloadable IT8.7/4r test forms should be posted on SNAP website. Plots of differenced identical patches should have no evidence of any pattern; it should resemble the stars dispersed in a clear night sky.

Current newspaper profile might be slightly skewed due to the lack of randomization. The test form was randomized once and printed by eighty different newspaper sites.

The standard error per wavelength varies as a function of wavelength and as a function of patch type. Further research is needed to discover the underlying relationship.

Using spectral data allowed the discovery of a process bias through the use of bootstrapping and principal component analysis.

Recommendation:

It is recommended that any standards organization or trade association follow a randomized pair design for their testing. Often this design can greatly increase precision by making comparisons within matched pairs of experimental material. Randomization can approximately validate the Student's t test derived on the assumption of random sampling from normal populations. Many fixed and unknown disturbances, such as press ghosting, starvation, rollers' settings …etc, affects the measured reflectance. It is questionable whether their joint effect could

be approximately represented unless some appropriate element of chance was specifically introduced into the experiment.

In order to avoid test form design-bias many randomized versions of the same color-profiling test form should be printed. This is critical for Standards' organizations who attempt to create a representative color profile. Many randomized versions should be posted for downloads on their websites. We even recommend a frequent periodical randomization.

The use of bootstrapping is strongly recommended. Bootstrapping is a powerful tool which is highly underutilized in our industry.

References:

- G. Wyszecki and W.S. Stiles, Color Science — Concepts and Methods, Quantitative Data and Formulae, 2nd ed., John Wiley & Sons, New York, 1982.
- CIE, 2004. CIE 20. 06. 2004. CIE Technical Report: Colorimetry, Third Edition, ISBN 3 901 906 33 9
- George E. P. Box, William G. Hunter, and J. Stuart Hunter, Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building (New York: John Wiley & Sons, Inc., 1978), 24
- Bradley Efron and Rob Tibshirani. An introduction to the bootstrap. CRC Press, Boca Raton, FL, 1993.
- R programming language, http://www.r-project.org/
- GNU EMACS, http://www.gnu.org/software/emacs/
- Technical discussion with John Seymour, QuadTech

A special thanks to those who edited this paper:

Paul Cousineau, Dow Jones
John Richards, Goss International
DR. Danny Rich, Sun Chemical
George Isaac
Sarah Isaac

R-Code and Example Computations

Data manipulations within the R-programming language:

This paper calculates the reflected frequency response error of the duplicated patches of the characterization datasets. In theory, identical patches produce identical spectral response. A computer program was created using the R-Programming language.

The SNAP characterization datasets were placed into a data matrix. The matrix has 129,360 rows with forty-two columns. The rows are the experimental units showing the spectral reflectance of eighty different newspaper sites. Columns one through

five and forty two describe the data and are titled: “patchId” “C” “M” “Y” “K” and “Customer” representing the patch number of the IT8.7/4r, the cyan, magenta, yellow, black and the newspaper sites respectively.

The R-program below recreates our results using any characterization datasets. The actual script follows our notes and is in small type. For this script to run correctly, the file names for the characterization datasets must be characterizationdata1.csv and characterizationdata2.csv

Extract, order, and pair identical duplicate patches with their color percentages:
Select the SNAP characterization datasets that meet the ISO-12647 standard.
Open each characterization dataset in Excel (use the delimited option). Reformat the datasets to create a standardized format with forty two columns. The column names are:
patchId, C, M, Y, K, s380, s390, s400, s410, s420, s430, s440, s450, s460, s470, s480, s490, 500, s510, s520, s530, s540, s550, s560, s570, s580, s590, s600, s610, s620, s630, s640, s650, s660, s670, s680, s690, s700, s710, s720, s730, Customer

In Excel combine all the SNAP characterization datasets into two files characterizationdata1.csv and characterizationdata2.csv
In Excel create a file, name iT8.csv. It includes 1,617 rows, one row per patch, and five columns: patchId, C, M, Y, K. this file is extracted from any given characterization data set.

The following script is run in R-programming language:
####################################
Create a function to synthesized data and calculate their principal component analysis
####################################
```
rm(list=ls())
charData <- function(x,y){
   data <- lapply(1:29, function(x)
         do.call(cbind,do.call(cbind,
                        lapply(lapply(1:36,function(y)
                                 cbind(rnorm(29,mean=0,sd=1))), data.frame)) ))
   pVariatesNames <- paste("s", seq(from=380, to=730, by=10),sep="")
   patchId <- paste("s", seq(from=1, length.out=29),sep=" ")
   names(data) <- c("Paper", "C10", "C20", "C30", "C40", "C70", "C85", "C100",
                "M10", "M20", "M30", "M40", "M70", "M85","M100",
                "Y10", "Y20", "Y30", "Y40", "Y70", "Y85","Y100",
                "K10", "K20", "K40", "K60", "K80", "K100", "Mix")
   for(i in 1:29) {
      colnames(data[[i]]) <- pVariatesNames
      rownames(data[[i]]) <- names(data)
   }
   return(data)
```

```
}
## Create the first dataset
dupAMatrixRandom <- charData(1,1)

## Create the second dataset
dupBMatrixRandom <- charData(1,1)

## Create helper function to bootstrap the data and then calculate the mean and standard
deviation
sampleOne   <- function(x) x[sample(seq_len(nrow(x)), replace = TRUE), ]
sampleBoot  <- function(x, n) replicate(n, sampleOne(x), simplify = FALSE)
applyMean <- function(l) do.call(rbind, lapply(l, apply, 2, mean))
applySd <- function(l) do.call(rbind, lapply(l, apply, 2, sd))

## Bootstrap
samplingMeansCharDataDupARandom   <-   lapply(lapply(dupAMatrixRandom,
sampleBoot, n = 1000), applyMean)
samplingMeansCharDataDupBRandom   <-   lapply(lapply(dupBMatrixRandom,
sampleBoot, n = 1000), applyMean)

## Calculate the mean of each itiration of the bootstrapp
bootMeansDupARandom <- applyMean(samplingMeansCharDataDupARandom)
bootMeansDupBRandom <- applyMean(samplingMeansCharDataDupBRandom)

## Calculate the standard error
stdErrDupARandom <- applySd(samplingMeansCharDataDupARandom)
stdErrDupBRandom <- applySd(samplingMeansCharDataDupBRandom)
diffSpectraRandom <- bootMeansDupARandom - bootMeansDupBRandom

## Calculate the principal component analysis for patch A
pcaARandom <- prcomp(bootMeansDupARandom)

## Calculate the principal component analysis for the differenced population
pcaDiffRandom <- prcomp(diffSpectraRandom)
####################################
## Load the iT8.csv into R
iT8 <-read.csv("iT8.csv", header=T)
## Determines and store the first and second duplicated patches, dupA and dupB, in
two separate objects:
dupFirstIndex <- duplicated(iT8[ , c("C","M","Y","K")], fromLast = TRUE)
dupLastIndex <- duplicated(iT8[ , c("C","M","Y","K")], fromLast = FALSE)
dupOrder   <-   order(iT8[dupFirstIndex,   ]$C,   iT8[dupFirstIndex,   ]$M,
iT8[dupFirstIndex, ]$Y, iT8[dupFirstIndex, ]$K)
dup1 <- iT8[dupFirstIndex, ][dupOrder, ]
```

```
dup2 <- iT8[dupLastIndex, ][dupOrder, ]

## Create patch IDs,  (Mix is an abbreviation for: C=100, M=85, Y=85)
idColor <- c("Paper", "K10", "K20", "K40", "K60", "K80", "K100", "Y10", "Y20",
"Y30",  "Y40",  "Y70",  "Y85",  "Y100",  "M10","M20","M30","M40",
"M70","M85","M100", "C10", "C20", "C30", "C40", "C70", "C85", "C100", "Mix")

## Combine the duplicated patches, their color identification as well as their color
percentages in a new object
Duplicated <- cbind.data.frame(dup1=dup1$patchId, dup2=dup2$patchId, idColor,
                        C = dup1$C, M = dup2$M, Y = dup1$Y, K = dup2$K)
rm("dup1", "dup2", "dupFirstIndex", "dupLastIndex", "dupOrder","iT8")

## Load the Characterization dataset in R
fileNames <- c("characterizationdata1", "characterizationdata2")
charDataList <- lapply(fileNames, function(x) read.csv(paste(x,"csv", sep='.'),
header=TRUE))
charData <- do.call(rbind, charDataList) ## combines the two lists into ONE
data.frame

## Create two objects to hold the duplicated characterization datasets and order the data
dup1 <- with(charData, charData[patchId %in% Duplicated[,"dup1"],])
dup1 <- with(dup1, dup1[order(Customer, C, M, Y, K, patchId),])
dup2 <- with(charData, charData[patchId %in% Duplicated[,"dup2"],])
dup2 <- with(dup2, dup2[order(Customer, C, M, Y, K, patchId),])
dup1ToBeMatched <- dup1$patchId
dup1ToBeMatchedAgainst <-  as.numeric(Duplicated[,"dup1"])
dup1Index <- match(dup1ToBeMatched, dup1ToBeMatchedAgainst)
dup1$patchId <- Duplicated[dup1Index ,"idColor"]
dup2ToBeMatched <- dup2$patchId
dup2ToBeMatchedAgainst <-  as.numeric(Duplicated[,"dup2"])
dup2Index <- match(dup2ToBeMatched, dup2ToBeMatchedAgainst)
dup2$patchId <- Duplicated[dup2Index ,"idColor"]
dataOrder <- c("Paper", "C10", "C20", "C30", "C40", "C70", "C85", "C100",
                "M10", "M20", "M30", "M40", "M70", "M85","M100",
                "Y10", "Y20", "Y30", "Y40", "Y70", "Y85","Y100",
                "K10", "K20", "K40", "K60", "K80", "K100", "Mix")
rm("charData",  "dup1Index",  "dup1ToBeMatched",  "dup1ToBeMatchedAgainst",
"dup2Index",  "dup2ToBeMatched",  "dup2ToBeMatchedAgainst",  "fileNames",
"idColor", "Duplicated","charDataList" )
## Create a data matrix that holds the duplicated patches A and B
dupA <- dup1[, c(-2:-5)]
dupB <- dup2[, c(-2:-5)]
rm("dup1","dup2")
```

```
## Split the data by patchId. This generates 29 lists, one list per patch which contains
all 80 customers.
dupASplit <- split(dupA, factor(dupA$patchId, levels=dataOrder))
dupBSplit <- split(dupB, factor(dupB$patchId, levels=dataOrder))
dupAMatrix <- lapply(dupASplit, "[", 2:37)
dupBMatrix <- lapply(dupBSplit, "[", 2:37)

## Bootstrap each patch independently
samplingMeanCharDataDupA <- lapply(lapply(dupAMatrix, sampleBoot, n = 1000),
applyMean)
samplingMeanCharDataDupB <- lapply(lapply(dupBMatrix, sampleBoot, n = 1000),
applyMean)
###################################
## The rigger police is out, we only show an example of the usefulness of bootstrapping
par(mar=c(5,4,3,2), oma=c(3,3,1,3), col.main="blue", col.sub=1,
   col.axis= "green4", col.lab="red", cex.main=1.2, cex.sub=1,
   cex.axis= 0.8, cex.lab= 1.2, font.main=2, font.axis=2, font.lab=2,
   xaxp=c(0,100,20), yaxp=c(0,20,10))
par(xpd=NA)
par(mfcol=c(1,2))
hist(dupA[dupA$patchId=="C10",2],  breaks="Scott",  main="Before",  xlab="",
ylab="")
box("plot", col="red3", lwd=2)
mtext("Reflectance", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Frequency", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)

hist(samplingMeanCharDataDupA[["C10"]][,1],breaks="Scott",     main="After",
xlab="", ylab="")
box("plot", col="red3", lwd=2)
mtext("Reflectance", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Frequency", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)
mtext("Figure 1:  10% Cyan Dot  Before and After Bootstrapping",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
###################################
## Check normality
shapiro.test(dupA[dupA$patchId=="C10",2])
shapiro.test(samplingMeanCharDataDupA[["C10"]][,1])
###################################
## Average each of the 1,000 bootstrap iterations to create the bootstrapped dataset.
Two datasets are formed containing the means of each bootstrap iteration.
bootMeanDupA <- applyMean(samplingMeanCharDataDupA)
bootMeanDupB <- applyMean(samplingMeanCharDataDupB)
```

```
## Calculate the standard error for each duplicate patch which is the standard deviation
of the bootstrapped data
stdErrDupA <- applySd(samplingMeanCharDataDupA)
stdErrDupB <- applySd(samplingMeanCharDataDupB)
####################################
## Create the principal components analysis (PCA) for the duplicated patches and their
error terms.
#### Principal components for the actual patch spectra
pcaA <- prcomp(bootMeanDupA)
pcaB <- prcomp(bootMeanDupB)
pcaErrA <- prcomp(stdErrDupA)
pcaErrB <- prcomp(stdErrDupB)
pcaDiff <- prcomp(bootMeanDupA-bootMeanDupB)## Bias
####################################
## Display the principal components
summary(pcaA)
summary(pcaB)
summary(pcaErrA)
summary(pcaErrB)
summary(pcaDiff)
####################################
## Plot the scree plots to identify the number of useful PCs
par(mfrow=c(1,2))
screeplot(pcaA, type="l"); box("plot", col="red3", lwd=2)
mtext("Number of PCs Before Levelling Off", col="blue4", side=1, adj=0.5, line = 2,
font=3,cex=1)

screeplot(pcaDiff, type="l");box("plot", col="red3", lwd=2)
mtext("Number of PCs Before Levelling Off", col="blue4", side=1, adj=0.5, line = 2,
font=3,cex=1)
mtext("Figure 2:  SCREE Plot of a Duplicate Patch and the Error Difference between
patches",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)

par(mfcol=c(1,2))
screeplot(pcaARandom, type="l"); box("plot", col="red3", lwd=2)
mtext("Number of PCs Before Levelling Off", col="blue4", side=1, adj=0.5, line = 2,
font=3,cex=1)
screeplot(pcaDiffRandom, type="l");box("plot", col="red3", lwd=2)
mtext("Number of PCs Before Levelling Off", col="blue4", side=1, adj=0.5, line = 2,
font=3,cex=1)
mtext("Figure 3:  SCREE Plot of synthesized Random Data",
```

```
    side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
####################################
## To detect outliers plot eigenvalue boxplots for both the random synthesized and
characterization datasets
par(mfrow=c(1,2))
boxplot(pcaA$x, col="red", main= "pcaA", ylab="", xlab="", ylim=c(-1.4,2))
box("plot", col="red3", lwd=2)
mtext("Principal Components", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Eigenvalues", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)

boxplot(pcaDiff$x, col="red", main= "pcaDiff", ylab="", xlab="", ylim=c(-1.4, 2))
box("plot", col="red3", lwd=2)
mtext("Principal Components", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Eigenvalues", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)
mtext("Figure 4:  Boxplot of Duplicate Patches and the Duplicated Patches' Error",
    side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
####################################
## compare the above to the uncorrelated random data
par(mfrow=c(1,2))
boxplot(pcaARandom$x, col="red", main= "pcaARandom", ylab="", xlab="",
ylim=c(-1.4,2))
box("plot", col="red3", lwd=2)
mtext("Principal Components", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Eigenvalues", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)

boxplot(pcaDiffRandom$x, col="red", main= "pcaDiffRandom", ylab="", xlab="",
ylim=c(-1.4, 2))
box("plot", col="red3", lwd=2)
mtext("Principal Components", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Eigenvalues", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)
mtext("Figure 5:  Boxplot of Synthesized Uncorrelated data and their Difference",
    side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
####################################
## Create the biplot: PC1 vs PC2 for the ORIGINAL Patch
par(mfrow=c(1,2))
plot(pcaA$x[,1], pcaA$x[,2], xlab="", ylab="", type="n",xlim=c(-1.5,2.0), ylim=c(-
1.5,1.5),main="pcA")
arrows(0,0, pcaA$rotation[,1]*4, pcaA$rotation[,2]*4, length=0.1, angle=20,
col="red")
text( pcaA$rotation[,1]*4*1.2,  pcaA$rotation[,2]*4*1.2, rownames( pcaA$rotation),
col="red", cex=0.7)
```

```
text(pcaA$x[,1], pcaA$x[,2], rownames(pcaA$x), col="blue", cex=0.7)
box("plot", col="red3", lwd=2)
mtext("First Principal Component", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Second Principal Component", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)

plot(pcaDiff$x[,1], pcaDiff$x[,2], xlab="", ylab="", type="n",xlim=c(-1.5,1.5), ylim=c(-1.5,1.5),main="pcaDiff")
arrows(0,0, pcaDiff$rotation[,1]*4, pcaDiff$rotation[,2]*4, length=0.1, angle=20, col="red")
text( pcaDiff$rotation[,1]*4*1.2, pcaDiff$rotation[,2]*4*1.2, rownames( pcaDiff$rotation), col="red", cex=0.7)
text(pcaDiff$x[,1], pcaDiff$x[,2], rownames(pcaDiff$x), col="blue", cex=0.7)
box("plot", col="red3", lwd=2)
mtext("First Principal Component", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Second Principal Component", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)
mtext("Figure 6:  Biplot of the First Two Principal Components",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
#####################################
par(mfrow=c(1,2))
plot(pcaARandom$x[,1], pcaARandom$x[,2], xlab="", ylab="", type="n",xlim=c(-2,2), ylim=c(-2,2),main="pcARandom")
arrows(0,0, pcaARandom$rotation[,1]*4, pcaARandom$rotation[,2]*4, length=0.1, angle=20, col="red")
text( pcaARandom$rotation[,1]*4*1.2, pcaARandom$rotation[,2]*4*1.2, rownames( pcaARandom$rotation),
    col="red", cex=0.7)
text(pcaARandom$x[,1], pcaARandom$x[,2], rownames(pcaARandom$x), col="blue", cex=0.7)
box("plot", col="red3", lwd=2)
mtext("First Principal Component", col="blue4", side=1, adj=0.5, line = 2, font=3,cex=1)
mtext("Second Principal Component", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)

plot(pcaDiffRandom$x[,1], pcaDiffRandom$x[,2], xlab="", ylab="", type="n",xlim=c(-2,2), ylim=c(-2,2),main="pcaDiffRandomRandom")
arrows(0,0, pcaDiffRandom$rotation[,1]*4, pcaDiffRandom$rotation[,2]*4, length=0.1, angle=20, col="red")
```

```
text( pcaDiffRandom$rotation[,1]*4*1.2, pcaDiffRandom$rotation[,2]*4*1.2,
rownames( pcaDiffRandom$rotation),
    col="red", cex=0.7)
text(pcaDiffRandom$x[,1], pcaDiffRandom$x[,2], rownames(pcaDiffRandom$x),
col="blue", cex=0.7)
box("plot", col="red3", lwd=2)
mtext("First Principal Component", col="blue4", side=1, adj=0.5, line = 2,
font=3,cex=1)
mtext("Second Principal Component", side=2, col="red", adj=0.5, line = 2, font=3,
cex=1)
mtext("Figure 7:  Biplot of Synthesized Random Data for PC1 and PC2",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
####################################
## Plot the eigenvectors
par(mfrow=c(1,1))
plot(pcaA$rotation[,1],col='red',pch=15,ylim=c(-.4,.4),xaxt='n',main="A Duplicate
Set", type="b", ylab="", xlab="")
points(pcaA$rotation[,2], col='blue',pch=16, type="b")
points(pcaA$rotation[,3], col='green',pch=17, type="b")
axis(side=1,at=(1:36),labels=names(pcaA$rotation[,1]),las=2)

box("plot", col="red3", lwd=2)
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Eigenvectors", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)
mtext("Figure 8:  Eigenvectors plot of a Duplicate Patch Set",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
legend("topright", legend = c("PC1", "PC2", "PC3"), col= c("red","blue", "green"),
pch=15:17,
      bg="gray96", horiz=F, cex=1, bty ="n")
####################################
## Eigenvalues comparison between each PC from patch A and patch B
par(mfrow=c(1,1))
plot(pcaA$x[,1], col='red', pch=16, ylim=c(-1.2,1.9), main="The First Principal
Component", xaxt='n', type="b", xlab="", ylab="")
points(pcaB$x[,1], col='blue', pch=16, type="b", xaxt='n')
axis(side=1,at=(1:29),labels=names(pcaA$x[,1]),las=2)
box("plot", col="red3", lwd=2)
mtext("Patch ID", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Eigenvalues", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)
mtext("Figure 9:  Eigenvalues plot of PC1 for Both Duplicate Patches",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
```

```
legend("topleft" , legend = c("Eigenvalues Patch A, PC1", "Eigenvalues Patch B, PC1"), col= c("red","blue"),
       pch=15:16, bg="gray96", horiz=F, cex=1, bty ="n")

plot(pcaA$x[,2], col='red', pch=16, ylim=c(-1.2,1.9),
     main="The Second Principal Component", xaxt='n', type="b", xlab="", ylab="")
points(pcaB$x[,2], col='blue', pch=16, type="b", xaxt='n')
axis(side=1,at=(1:29),labels=names(pcaA$x[,1]),las=2)
box("plot", col="red3", lwd=2)
mtext("Patch ID", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Eigenvalues", side=2, col="red", adj=0.5, line = 2, font=3, cex=1)
mtext("Figure 10:  Eigenvalues plot of PC2 for Both Duplicate Patches",
      side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
legend("topleft" , legend = c("Eigenvalues Patch A, PC2", "Eigenvalues Patch B, PC2"), col= c("red","blue"),
       pch=15:16, bg="gray96", horiz=F, cex=1, bty ="n")
####################################
## subtract the spectra of patch B from patch A
diffSpectra <- bootMeanDupA - bootMeanDupB
pcaDiff <- prcomp(diffSpectra)
summary(pcaDiff)
####################################
## Box plot of the diff population
spectrumColor <- hsv(c(seq(from=0.750, to=0.694, by=-1/108), seq(from=0.667, to=0.639, by=-1/72),

0.556,0.528,0.500,0.472,0.444,0.417,0.389,0.361,0.278,0.167,0.139,0.083,0.058,0.028,
       0.000,0.972,0.944,0.944,0.944,0.944,0.9305,0.917,0.917,0.917,0.917,0.889), 1.0, 1.0)
par(mar=c(5,5,3,2), oma=c(3,3,1,3), col.main="blue", col.sub=1,
    col.axis= "green4", col.lab="red", cex.main=1.2, cex.sub=1,
    cex.axis= 0.8, cex.lab= 1.2, font.main=2, font.axis=2, font.lab=2,
    xaxp=c(0,100,20), yaxp=c(0,20,10))
spectralError <- bootMeanDupA-bootMeanDupB
par(mfcol=c(2,2))
spectralErrorCyan <- spectralError[ c("C10", "C20", "C30", "C40", "C70", "C85", "C100", "Mix", "Paper"),]
boxplot(spectralErrorCyan, main="Cyan", notch=TRUE, col=spectrumColor, at=seq(1,36,1), xlim=c(1,36),las=2)
############
box("plot", col="red3", lwd=2)
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
```

```
mtext("Spectral Error", side=2, col="red", adj=0.5, line = 3, font=3, cex=1)
############
spectralErrorMagenta <- spectralError[ c("M10", "M20", "M30", "M40", "M70",
"M85", "M100", "Mix", "Paper"),]
boxplot(spectralErrorMagenta, main="Magenta", notch=TRUE, col=spectrumColor,
at=seq(1,36,1), xlim=c(1,36),las=2)
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Spectral Error", side=2, col="red", adj=0.5, line = 3, font=3, cex=1)
############
spectralErrorYellow <- spectralError[ c("Y10", "Y20", "Y30", "Y40", "Y70", "Y85",
"Y100", "Mix", "Paper"),]
boxplot(spectralErrorYellow, main="Yellow", notch=TRUE, col=spectrumColor,
at=seq(1,36,1), xlim=c(1,36),las=2)
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Spectral Error", side=2, col="red", adj=0.5, line = 3, font=3, cex=1)
############
spectralErrorBlack <- spectralError[ c("K10", "K20", "K40", "K60", "K80", "K100",
"Mix", "Paper"),]
boxplot(spectralErrorBlack, main="Black", notch=TRUE, col=spectrumColor,
at=seq(1,36,1), xlim=c(1,36),las=2)
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Spectral Error", side=2, col="red", adj=0.5, line = 3, font=3, cex=1)
mtext("Figure 11:  Boxplot of Spectral Error between Duplicate Patches",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
####################################
### Cyan Plot For the Standard Error
xlabs <- as.character(seq(from=380,to=730,by=10))
listColor <- rep("cyan", each=9)
listpatchName <- c("C10", "C20", "C30", "C40", "C70", "C85", "C100", "Mix",
"Paper")
listplotColor <-1:9
List <- list(Color=listColor, patchName=listpatchName, plotColor=listplotColor)
scatterPlot <- function(i, listColor, listpatchName, listplotColor){
   points(stdErrDupA[List[[2]][i],], col=List[[3]][i], type="b", pch=14+i, cex=1.2)
   axis(1, labels=xlabs , at=1:36, las=2)}
par(mar=c(5,5,3,2), oma=c(3,3,3,3), col.main="blue", col.sub=1,
   col.axis= "green4", col.lab="red", cex.main=1.2, cex.sub=1,
   cex.axis= 0.8, cex.lab= 1.2, font.main=2, font.axis=2, font.lab=2,
   xaxp=c(0,100,20), yaxp=c(0,20,10))
par(mfcol=c(1,1))
plot(stdErrDupA, type="n", xlim=c(1,36), ylim=c(0,0.008), xlab="", ylab="",
    main="Cyan", xaxt="n", las=2)
box("plot", col="red3", lwd=2)
```

```
lapply(1:9, scatterPlot)
par(xpd=FALSE)
abline(v=seq(from=1,to=36, by=1), h=seq(from= 0.000, to=0.008, by=0.001),  lty=1,
col="darkgray")
par(xpd=NA)
legend("topleft" , legend = listpatchName, col= listplotColor, pch=15:23,
       bg="gray96", horiz=F, cex=1, bty ="n")
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Reflectance Standard Error", side=2, col="red", adj=0.5, line = 3, font=3,
cex=1)
mtext("Figure 12a:  Plot of the Standard Error Per Patch for Duplicate Set A: Cyan
Tone Ramp",
      side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
###################################
### Magenta Plot for the Standard Error
listColor <- rep("magenta", each=9)
listpatchName                                                                    <-
c("M10","M20","M30","M40","M70","M85","M100","Mix","Paper")
List <- list(Color=listColor, patchName=listpatchName, plotColor=listplotColor)
par(mfcol=c(1,1))
plot(stdErrDupA, type="n", xlim=c(1,36),ylim=c(0,0.008), xlab="", ylab="",
     main="Magenta", xaxt="n", las=2)## extreme tick marks and the number of inter-
vals
box("plot", col="red3", lwd=2)
lapply(1:9, scatterPlot)
par(xpd=FALSE)
abline(v=seq(from=1,to=36, by=1), h=seq(from= 0.000, to=0.008, by=0.001),  lty=1,
col="darkgray")
par(xpd=NA)
legend("topleft" , legend = listpatchName, col= listplotColor, pch=15:23,
       bg="gray96", horiz=F, cex=1, bty ="n")
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Reflectance Standard Error", side=2, col="red", adj=0.5, line = 3, font=3,
cex=1)
mtext("Figure 12b:  Plot of the Standard Error Per Patch for Duplicate Set A: Magenta
Tone Ramp",
      side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
###################################
### Yellow Plot for the Standard Error
listColor <- rep("Yellow", each=9)
listpatchName <- c("Y10","Y20","Y30","Y40","Y70","Y85","Y100","Mix","Paper")
List <- list(Color=listColor, patchName=listpatchName, plotColor=listplotColor)
```

```
par(mfcol=c(1,1))
plot(stdErrDupA, type="n", xlim=c(1,36),ylim=c(0,0.01), xlab="", ylab="",
    main="Yellow", xaxt="n", las=2)## extreme tick marks and the number of intervals
box("plot", col="red3", lwd=2)
lapply(1:9, scatterPlot)
par(xpd=FALSE)
abline(v=seq(from=1,to=36, by=1), h=seq(from= 0.000, to=0.008, by=0.001),  lty=1,
col="darkgray")
par(xpd=NA)
legend("topleft" , legend = listpatchName, col= listplotColor, pch=15:23,
      bg="gray96", horiz=F, cex=1, bty ="n")
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Reflectance Standard Error", side=2, col="red", adj=0.5, line = 3, font=3,
cex=1)
mtext("Figure 12c:  Plot of the Standard Error Per Patch for Duplicate Set A: Yellow
Tone Ramp",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
####################################
### Black Plot for the Standard Error
listColor <- rep("Black", each=8)
listpatchName <- c("K10", "K20","K40","K60","K80","K100" ,"Mix","Paper")
List <- list(Color=listColor, patchName=listpatchName, plotColor=listplotColor)
par(mfcol=c(1,1))
plot(stdErrDupA, type="n", xlim=c(1,36),ylim=c(0,0.008), xlab="", ylab="",
    main="Black", xaxt="n", las=2)## extreme tick marks and the number of intervals
box("plot", col="red3", lwd=2)
lapply(1:8, scatterPlot)
par(xpd=FALSE)
abline(v=seq(from=1,to=36, by=1), h=seq(from= 0.000, to=0.008, by=0.001),  lty=1,
col="darkgray")
par(xpd=NA)
legend("topleft" , legend = listpatchName, col= listplotColor, pch=15:23,
      bg="gray96", horiz=F, cex=1, bty ="n")
mtext("Wavelength", col="blue4", side=1, adj=0.5, line = 3, font=3,cex=1)
mtext("Reflectance Standard Error", side=2, col="red", adj=0.5, line = 3, font=3,
cex=1)
mtext("Figure 12d:  Plot of the Standard Error Per Patch for Duplicate Set A: Black
Tone Ramp",
     side=1, line=1, cex=1.4, col="firebrick", font=2, outer=TRUE)
box(which="inner", col="blue4", lwd=2)
```