Introduction

Alexander Walker
Alexander.Walker1989@gmail.com

July 3, 2015

1 Installation

The openxlsx package requires a zip application to be available to R, such as the one that comes with Rtools.

If the command

```
shell("zip")
```

returns "zip is not recognised..." install Rtools from: http://cran.r-project.org/bin/windows/Rtools/ and add the R tools bin directory (defaults to c:\Rtools\bin on windows) to the system path.

If the command returns

```
Warning message: running command '"zip" -r1 ... had status 127
```

it is most likely because R cannot find a zip application. First check if the Rtools/bin directory is in the system PATH variable. This can be checked from within R using the command

```
shell("PATH")
```

The output should contain a path to Rtools/bin somewhere in the string.

2 Basic Examples

2.1 write.xlsx

The simplest way to write to a workbook is write.xlsx().
By default, write.xlsx calls writeData. If asTable is TRUE write.xlsx will write x as an Excel table.

```
## write to working directory
write.xlsx(iris, file = "writeXLSX1.xlsx")
write.xlsx(iris, file = "writeXLSXTable1.xlsx", asTable = TRUE)

## write a list of data.frames to individual worksheets using list names as worksheet names
l <- list("IRIS" = iris, "MTCARS" = mtcars)
write.xlsx(l, file = "writeXLSX2.xlsx")
write.xlsx(l, file = "writeXLSXTable2.xlsx", asTable = TRUE)

## write.xlsx also accepts styling parameters.
## The simplest way is to set default options and set column class
options("openxlsx.borderColour" = "#4F80BD")
options("openxlsx.borderStyle" = "thin")
options("openxlsx.dateFormat" = "mm/dd/yyyy")
options("openxlsx.datetimeFormat" = "yyyy-mm-dd hh:mm:ss")
options("openxlsx.numFmt" = NULL) ## For default style rounding of numeric columns

df <- data.frame("Date" = Sys.Date()-0:19, "LogicalT" = TRUE,
                 "Time" = Sys.time()-0:19*60*60,
                 "Cash" = paste("$",1:20), "Cash2" = 31:50,
                 "hLink" = "http://cran.r-project.org/",
                 "Percentage" = seq(0, 1, length.out=20),
                 "TinyNumbers" = runif(20) / 1E9,  stringsAsFactors = FALSE)

class(df$Cash) <- "currency"
class(df$Cash2) <- "accounting"
class(df$hLink) <- "hyperlink"
class(df$Percentage) <- "percentage"
class(df$TinyNumbers) <- "scientific"

write.xlsx(df, "writeXLSX3.xlsx")
write.xlsx(df, file = "writeXLSXTable3.xlsx", asTable = TRUE)



## Additional styling
## define a style for column headers
hs <- createStyle(fontColour = "#ffffff", fgFill = "#4F80BD",
                  halign = "center", valign = "center", textDecoration = "Bold",
                  border = "TopBottomLeftRight", textRotation = 45)

write.xlsx(iris, file = "writeXLSX4.xlsx", borders = "rows", headerStyle = hs)
write.xlsx(iris, file = "writeXLSX5.xlsx", borders = "columns", headerStyle = hs)

write.xlsx(iris, "writeXLSXTable4.xlsx", asTable = TRUE,
headerStyle = createStyle(textRotation = 45))
```

```
## When writing a list, the stylings will apply to all list elements
l <- list("IRIS" = iris, "colClasses" = df)
write.xlsx(l, file = "writeXLSX6.xlsx", borders = "columns", headerStyle = hs)
write.xlsx(l, file = "writeXLSXTable5.xlsx", asTable = TRUE, tableStyle = "TableStyleLight2")

openXL("writeXLSX6.xlsx")
openXL("writeXLSXTable5.xlsx")

## write.xlsx returns the workbook object for further editing
wb <- write.xlsx(iris, "writeXLSX6.xlsx")
setColWidths(wb, sheet = 1, cols = 1:5, widths = 20)
saveWorkbook(wb, "writeXLSX6.xlsx", overwrite = TRUE)
```

2.2 Basic Workbook

```
require(ggplot2)

## set default border Colour and style
wb <- createWorkbook()
options("openxlsx.borderColour" = "#4F80BD")
options("openxlsx.borderStyle" = "thin")
modifyBaseFont(wb, fontSize = 10, fontName = "Arial Narrow")

addWorksheet(wb, sheetName = "Motor Trend Car Road Tests", gridLines = FALSE)
addWorksheet(wb, sheetName = "Iris", gridLines = FALSE)

## sheet 1
freezePane(wb, sheet = 1, firstRow = TRUE, firstCol = TRUE) ## freeze first row and column
writeDataTable(wb, sheet = 1, x = mtcars,
colNames = TRUE, rowNames = TRUE,
tableStyle = "TableStyleLight9")

setColWidths(wb, sheet = 1, cols = "A", widths = 18)

## write iris data.frame as excel table
writeDataTable(wb, sheet = 2, iris, startCol = "K", startRow = 2)

qplot(data=iris, x = Sepal.Length, y= Sepal.Width, colour = Species)
insertPlot(wb, 2, xy=c("B", 16)) ## insert plot at cell B16

means <- aggregate(x = iris[,-5], by = list(iris$Species), FUN = mean)
vars <- aggregate(x = iris[,-5], by = list(iris$Species), FUN = var)

## write group means
headSty <- createStyle(fgFill="#DCE6F1", halign="center", border = "TopBottomLeftRight")
writeData(wb, 2, x = "Iris dataset group means", startCol = 2, startRow = 2)
writeData(wb, 2, x = means, startCol = "B", startRow=3, borders="rows", headerStyle = headSty)

## write group variances
writeData(wb, 2, x = "Iris dataset group variances", startCol = 2, startRow = 9)
writeData(wb, 2, x= vars, startCol = "B", startRow=10, borders="columns",
headerStyle = headSty)

setColWidths(wb, 2, cols=2:6, widths = 12) ## width is recycled for each col
setColWidths(wb, 2, cols=11:15, widths = 15)

# style mean & variance table headers
s1 <- createStyle(fontSize=14, textDecoration=c("bold", "italic"))
addStyle(wb, 2, style = s1, rows=c(2,9), cols=c(2,2))

saveWorkbook(wb, "basics.xlsx", overwrite = TRUE) ## save to working directory
```

2.3 Gallery

```
## inspired by xtable gallery
#http://cran.r-project.org/web/packages/xtable/vignettes/xtableGallery.pdf

## Create a new workbook
wb <- createWorkbook()
data(tli, package = "xtable")

## data.frame
test.n <- "data.frame"
my.df <- tli[1:10, ]
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = my.df, borders = "n")

## matrix
test.n <- "matrix"
design.matrix <- model.matrix(~ sex * grade, data = my.df)
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = design.matrix)

## aov
test.n <- "aov"
fm1 <- aov(tlimth ~ sex + ethnicty + grade + disadvg, data = tli)
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = fm1)

## lm
test.n <- "lm"
fm2 <- lm(tlimth ~ sex*ethnicty, data = tli)
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = fm2)

## anova 1
test.n <- "anova"
my.anova <- anova(fm2)
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = my.anova)

## anova 2
test.n <- "anova2"
fm2b <- lm(tlimth ~ ethnicty, data = tli)
my.anova2 <- anova(fm2b, fm2)
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = my.anova2)

## glm
test.n <- "glm"
fm3 <- glm(disadvg ~ ethnicty*grade, data = tli, family = binomial())
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = fm3)

## prcomp
test.n <- "prcomp"
```

```
pr1 <- prcomp(USArrests)
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = pr1)

## summary.prcomp
test.n <- "summary.prcomp"
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = summary(pr1))

## simple table
test.n <- "table"
data(airquality)
airquality$OzoneG80 <- factor(airquality$Ozone > 80,
levels = c(FALSE, TRUE),
labels = c("Oz <= 80", "Oz > 80"))
airquality$Month <- factor(airquality$Month,
levels = 5:9,
labels = month.abb[5:9])
my.table <- with(airquality, table(OzoneG80,Month) )
addWorksheet(wb = wb, sheetName = test.n)
writeData(wb = wb, sheet = test.n, x = my.table)

## survdiff 1
library(survival)
test.n <- "survdiff1"
addWorksheet(wb = wb, sheetName = test.n)
x <- survdiff(Surv(futime, fustat) ~ rx, data = ovarian)
writeData(wb = wb, sheet = test.n, x = x)

## survdiff 2
test.n <- "survdiff2"
addWorksheet(wb = wb, sheetName = test.n)
expect <- survexp(futime ~ ratetable(age=(accept.dt - birth.dt),
      sex=1,year=accept.dt,race="white"), jasa, cohort=FALSE,
      ratetable=survexp.usr)
x <- survdiff(Surv(jasa$futime, jasa$fustat) ~ offset(expect))
writeData(wb = wb, sheet = test.n, x = x)

## coxph 1
test.n <- "coxph1"
addWorksheet(wb = wb, sheetName = test.n)
bladder$rx <- factor(bladder$rx, labels = c("Pla","Thi"))
x <- coxph(Surv(stop,event) ~ rx, data = bladder)
writeData(wb = wb, sheet = test.n, x = x)

## coxph 2
test.n <- "coxph2"
addWorksheet(wb = wb, sheetName = test.n)
x <- coxph(Surv(stop,event) ~ rx + cluster(id), data = bladder)
writeData(wb = wb, sheet = test.n, x = x)

## cox.zph
test.n <- "cox.zph"
```

```
addWorksheet(wb = wb, sheetName = test.n)
x <- cox.zph(coxph(Surv(futime, fustat) ~ age + ecog.ps,  data=ovarian))
writeData(wb = wb, sheet = test.n, x = x)

## summary.coxph 1
test.n <- "summary.coxph1"
addWorksheet(wb = wb, sheetName = test.n)
x <- summary(coxph(Surv(stop,event) ~ rx, data = bladder))
writeData(wb = wb, sheet = test.n, x = x)

## summary.coxph 2
test.n <- "summary.coxph2"
addWorksheet(wb = wb, sheetName = test.n)
x <- summary(coxph(Surv(stop,event) ~ rx + cluster(id), data = bladder))
writeData(wb = wb, sheet = test.n, x = x)

## view without saving
openXL(wb)
```

3 Further Examples

3.1 Stock Price

```
require(ggplot2)

wb <- createWorkbook()

## read historical prices from yahoo finance
ticker <- "CBA.AX"
csv.url <- paste("http://ichart.finance.yahoo.com/table.csv?s=",
ticker, "&a=01&b=9&c=2009&d=01&e=9&f=2014&g=d&ignore=.csv")
prices <- read.csv(url(csv.url), as.is = TRUE)
prices$Date <- as.Date(prices$Date)
close <- prices$Close
prices$logReturns = c(0, log(close[2:length(close)]/close[1:(length(close)-1)]))

## Create plot of price series and add to worksheet
ggplot(data = prices, aes(as.Date(Date), as.numeric(Close))) +
geom_line(colour="royalblue2") +
labs(x = "Date", y = "Price", title = ticker) +
geom_area(fill = "royalblue1",alpha = 0.3) +
coord_cartesian(ylim=c(min(prices$Close)-1.5, max(prices$Close)+1.5))

## Add worksheet and write plot to sheet
addWorksheet(wb, sheetName = "CBA")
insertPlot(wb, sheet = 1, xy = c("J", 3))

## Histogram of log returns
ggplot(data = prices, aes(x = logReturns)) + geom_bar(binwidth=0.0025) +
labs(title = "Histogram of log returns")

## currency
class(prices$Close) <- "currency" ## styles as currency in workbook

## write historical data and  histogram of returns
writeDataTable(wb, sheet = "CBA", x = prices)
insertPlot(wb, sheet = 1, startRow=25, startCol = "J")

## Add conditional formatting to show where logReturn > 0.01 using default style
conditionalFormat(wb, sheet = 1, cols = 1:ncol(prices), rows = 2:(nrow(prices)+1),
rule = "$H2 > 0.01")

## style log return col as a percentage
logRetStyle <- createStyle(numFmt = "percentage")

addStyle(wb, 1, style = logRetStyle, rows = 2:(nrow(prices) + 1),
cols = "H", gridExpand = TRUE)

setColWidths(wb, sheet=1, cols = c("A", "F", "G", "H"), widths = 15)

## save workbook to working directory
saveWorkbook(wb, "stockPrice.xlsx", overwrite = TRUE)
openXL("stockPrice.xlsx")
```

3.2 Image Compression using PCA

```
require(openxlsx)
require(jpeg)
require(ggplot2)

plotFn <- function(x, ...){
  colvec <- grey(x)
  colmat <- array(match(colvec, unique(colvec)), dim = dim(x)[1:2])
  image(x = 0:(dim(colmat)[2]), y = 0:(dim(colmat)[1]), z = t(colmat[nrow(colmat):1, ]),
    col = unique(colvec), xlab = "", ylab = "", axes = FALSE, asp = 1,
    bty ="n", frame.plot=F, ann=FALSE)
}

## Create workbook and add a worksheet, hide gridlines
wb <- createWorkbook("Einstein")
addWorksheet(wb, "Original Image", gridLines = FALSE)

A <- readJPEG(file.path(path.package("openxlsx"), "einstein.jpg"))
height <- nrow(A); width <- ncol(A)

## write "Original Image" to cell B2
writeData(wb, 1, "Original Image", xy = c(2,2))

## write Object size to cell B3
writeData(wb, 1, sprintf("Image object size: %s bytes",
                         format(object.size(A+0)[[1]], big.mark=',')),
          xy = c(2,3))  ## equivalent to startCol = 2, startRow = 3

## Plot image
par(mar=rep(0, 4), xpd = NA); plotFn(A)

## insert plot currently showing in plot window
insertPlot(wb, 1, width, height, units="px", startRow= 5, startCol = 2)

## SVD of covariance matrix
rMeans <- rowMeans(A)
rowMeans <- do.call("cbind", lapply(1:ncol(A), function(X) rMeans))
A <- A - rowMeans
E <- svd(A %*% t(A) / (ncol(A) - 1)) # SVD on covariance matrix of A
pve <- data.frame("Eigenvalues" = E$d,
                  "PVE" = E$d/sum(E$d),
                  "Cumulative PVE" = cumsum(E$d/sum(E$d)))

## write eigenvalues to worksheet
addWorksheet(wb, "Principal Component Analysis")
hs <- createStyle(fontColour = "#ffffff", fgFill = "#4F80BD",
                  halign = "CENTER", textDecoration = "Bold",
                  border = "TopBottomLeftRight", borderColour = "#4F81BD")

writeData(wb, 2, x="Proportions of variance explained by Eigenvector" ,startRow = 2)
mergeCells(wb, sheet=2, cols=1:4, rows=2)

setColWidths(wb, 2, cols = 1:3, widths = c(14, 12, 15))
```

```
writeData(wb, 2, x=pve, startRow = 3, startCol = 1, borders="rows", headerStyle=hs)

## Plots
pve <- cbind(pve, "Ind" = 1:nrow(pve))
ggplot(data = pve[1:20,], aes(x = Ind, y = 100*PVE)) +
  geom_bar(stat="identity", position = "dodge") +
  xlab("Principal Component Index") + ylab("Proportion of Variance Explained") +
  geom_line(size = 1, col = "blue") + geom_point(size = 3, col = "blue")

## Write plot to worksheet 2
insertPlot(wb, 2, width = 5, height = 4, startCol = "E", startRow = 2)

## Plot of cumulative explained variance
ggplot(data = pve[1:50,], aes(x = Ind, y = 100*Cumulative.PVE)) +
  geom_point(size=2.5) + geom_line(size=1) + xlab("Number of PCs") +
  ylab("Cumulative Proportion of Variance Explained")
insertPlot(wb, 2, width = 5, height = 4, xy= c("M", 2))


## Reconstruct image using increasing number of PCs
nPCs <- c(5, 7, 12, 20, 50, 200)
startRow <- rep(c(2, 24), each = 3)
startCol <- rep(c("B", "H", "N"), 2)

## create a worksheet to save reconstructed images to
addWorksheet(wb, "Reconstructed Images", zoom = 90)

for(i in 1:length(nPCs)){

  V <- E$v[, 1:nPCs[i]]
  imgHat <- t(V) %*% A  ## project img data on to PCs
  imgSize <- object.size(V) + object.size(imgHat) + object.size(rMeans)

  imgHat <- V %*% imgHat + rowMeans  ## reconstruct from PCs and add back row means
  imgHat <- round((imgHat - min(imgHat)) / (max(imgHat) - min(imgHat))*255) # scale
  plotFn(imgHat/255)

  ## write strings to worksheet 3
  writeData(wb, "Reconstructed Images",
            sprintf("Number of principal components used:  %s",
                    nPCs[[i]]), startCol[i], startRow[i])

  writeData(wb, "Reconstructed Images",
            sprintf("Sum of component object sizes: %s bytes",
                    format(as.numeric(imgSize), big.mark=',')), startCol[i], startRow[i]+1)

  ## write reconstruced image
  insertPlot(wb, "Reconstructed Images", width, height, units="px",
             xy = c(startCol[i], startRow[i]+3))

}

# hide grid lines
```

```
showGridLines(wb, sheet = 3, showGridLines = FALSE)

## Make text above images BOLD
boldStyle <- createStyle(textDecoration="BOLD")

## only want to apply style to specified cells (not all combinations of rows & cols)
addStyle(wb, "Reconstructed Images", style=boldStyle,
         rows = c(startRow, startRow+1), cols = rep(startCol, 2),
         gridExpand = FALSE)

## save workbook to working directory
saveWorkbook(wb, "Image dimensionality reduction.xlsx", overwrite = TRUE)
```