```
# R utility for correcting for plate/batch/lot and nonspecific binding artifacts (Updated 20 January 2022.)
# See detailed documentation in Maecker et al. (2020). Journal of Immunology 204:3425-3433.
# We are grateful to Dr. Haiming Zhou (Northern Illinois University) for helpful suggestions on proper use of R package lpme.
# NOTE: Throughout, CHEX4 is a legacy term for NC. NC is nonspecific control bead for measuring nonspecific binding.
############################
# USE OF SUPPLIED FUNCTIONS #
############################
# 1) Install libraries emmeans, lattice, lpme, and nlme (cran.r-project.org) before running CHEX4Detrend function. Supply name of directory for output,
# bounded by quotation marks (e.g., Directory <- "C:\\MyData\\SP Project"). Supply resolution in DPI for output image (e.g., Res <- 1100). Supply quantity
# of SPs (e.g., QtyC <- 40). Supply input dataset as shown in example below (e.g., RawData <- read.csv(file = "C:\\MyData\\SP Project\\My SP Data.csv", header = T)).
# Must use variable names "Directory", "Res", "QtyC", and "RawData" with left assignment arrow "<-", as shown in example below.
#
# 2) Any rows of data that contain any missing values are excluded prior to all processing.
#
# 3) The algorithm also assumes the experiment has at least two plates. If not, consider splitting single plate into two "pseudo-plates" in input data set.
#
# 4) Supply the following arguments to CHEX4Detrend function (see example below).
# 4a) "InputData" is dataframe where 1st column (named "Specimen") contains specimen IDs, 2nd column (named "Plate") contains plate/batch/lot IDs (1, 2,
# 3,...), 3rd column (named "Well") contains well IDs, 4th column (named "CHEX4") contains nonspecific binding MFI values,
# and remaining columns contain MFI values by SP, one column per each of the SPs, with each column named for that SP. Do not use
# any special characters (including blanks) in SP names (e.g., "IL17F" instead of "IL-17F"). Any remaining columns (ALWAYS to the RIGHT of the
# SP data) are all covariates of interest.
# 4b) "Method" is for function meanreg in R package lpme. See papers cited therein. Method = "HZ" employs the most recent method.
# 4c) Input bandwidth sequence (e.g., Seq <- 2:6). The values in this sequence are DIVISORS; so, for example, shifting this entire sequence higher
# (e.g., Seq <- 3:7) allows the fit to be more "local," while shifting the sequence lower (e.g., Seq <- 2:5) can be useful when gaps appear in
# the sample distribution of the plate/batch/lot-detrended nonspecific binding MFI values.
# 4d) "Cont" gives column locations (numbered from 1st column on left) of interval or ratio scale covariates (e.g., Cont = c(45, 48)); otherwise set to 0.
# 4e) "Categ" gives column locations (numbered from 1st column on left) of categorical covariates (e.g., Categ = c(46, 47, 49)); otherwise set to 0. We recommend
# treating variables as categorical whether unordered (e.g., "Fe", "Mg", "Na") or ordered ("Low", "Med", "High").
# 4f) "Repeats" is quantity of repeated cross-validations. Smaller values of Repeats reduce run times and larger values increase reliability of results.
# 4g) "Covs" gives covariates as a linear model formula (e.g., Covs <- "Plate + Treatment + BMI").
# 4f) "VSpecimen" and "VPipette" are average volume to dilution per specimen and volume pipetted into each well, respectively. Must be in same units (e.g., microliters).
# 4g) "Fold" is fold dilution of specimen.
#################
# START SCRIPT #
#################
# Load libraries.
LibLoader <- function() {
library(emmeans)
library(lattice)
library(lpme)
library(nlme)
library(parallel)
}
#Function for finite-population variance.
FVar <- function(x) {
Tau <- (VSpecimen * Fold) / VPipette
m. <- length(x)
Sigma.Mu <- ((1 - m./Tau) / m.) * var(x)
return(Sigma.Mu)
}
# The main function.
CHEX4Detrend <- function(InputData, Method, Seq, Cont, Categ) {
# Set seed for random number generator for reproducible results.
set.seed(9893)
# Calculate quantity of CPUs to allocate to parallel processing.
CPUs <- ceiling(detectCores(all.tests = T, logical = T) / 3)
# Read input data removing any rows with missing data.
InputData <- na.exclude(InputData[order(as.numeric(InputData$Plate)),])
# Add sequential numbering for all wells.
NoMissing <- data.frame(InputData, WellNo = 1:nrow(InputData))
# Natural-logarithm transform all MFI data.
Stdz <- data.frame(log(NoMissing[, 4:(4 + QtyC)]))
# Create empty column for construction of following dataframe.
Fill <- rep(NA, (1 + QtyC) * nrow(Stdz))
# Long-format dataframe to receive SP name, the logarithm-transformed MFI data, specimen ID, plate ID, and sequential well number.
DF0 <- data.frame(SP = Fill, RawMFI = Fill, Specimen = Fill, Plate = Fill, WellNo = Fill)
```

```
# Populate this data frame.
for (cy in 1:(1 + QtyC)) {
DF0[(1 + (cy - 1) * nrow(Stdz)):(cy * nrow(Stdz)), 1] <- rep(names(Stdz)[cy], nrow(Stdz))
DF0[(1 + (cy - 1) * nrow(Stdz)):(cy * nrow(Stdz)), 2] <- Stdz[, cy]
DF0[(1 + (cy - 1) * nrow(Stdz)):(cy * nrow(Stdz)), 3] <- as.character(NoMissing$Specimen)
DF0[(1 + (cy - 1) * nrow(Stdz)):(cy * nrow(Stdz)), 4] <- NoMissing$Plate
DF0[(1 + (cy - 1) * nrow(Stdz)):(cy * nrow(Stdz)), 5] <- NoMissing$WellNo
}
# Sort data frame on names of SPs.
DF0 <- DF0[order(DF0$SP), ]
# Calculate the mean of the logarithm of MFI by soluble protein.
OverallMn <- aggregate(DF0$RawMFI, by = list(SP = DF0$SP), FUN = mean, simplify = T)
# Name columns of resultant data set.
names(OverallMn) <- c("SP", "OMn")
# Merge long-format data set with means by SP.
DF1 <- merge(x = DF0, y = OverallMn, by.x = c("SP"), by.y = c("SP"))
# Next 19 lines generate Figure 1.
# Create a pallete for labeling different plates with different colors in Figure 1.
palette(terrain.colors(max(NoMissing$Plate)))
# Calculate observed minimum and maximum for vertical axis.
RngY <- range(DF1$RawMFI)
# Order data set by sequential well number.
DF1 <- DF1[order(DF1$WellNo), ]
# Create Figure 1 plot of raw data.
Fig1 <- xyplot(RawMFI ~ WellNo | SP, DF1, as.table = T,
# Scales on both axes are designed to extend slightly beyond data. User may want to adjust this for x-axis in small data sets.
scales = list(alternating = 1, cex = 0.6), xlim = range(DF1$WellNo, na.rm = T) + c(-10, 10),
ylim = RngY + c(-0.1 * RngY[1], 0.1 * RngY[2]),
subscripts = T, xlab = "Well (Ordered by Plate)", ylab = "Logarithm of Raw Median Fluorescence Intensity",
panel = function(subscripts) {
# First of next two lines creates black outlines on data points. Same procedure is used in other three figures.
panel.xyplot(DF1$WellNo[subscripts], DF1$RawMFI[subscripts], type = "p", col = "black", pch = 1, cex = 0.35)
panel.xyplot(DF1$WellNo[subscripts], DF1$RawMFI[subscripts], type = "p", col = DF1$Plate[subscripts], pch = 16, cex = 0.25)
# Add smooth curve.
panel.loess(DF1$WellNo[subscripts], DF1$RawMFI[subscripts], col = "red", span = 2/3, degree = 2, family = "symmetric", lwd = 2)
})
# Next 50 lines remove plate artifact.
# Format specimen and plate as factors for use in linear mixed model.
Char <- data.frame(Stdz, Specimen = as.factor(NoMissing$Specimen), Plate = as.factor(NoMissing$Plate))
# Format categorical covariates for use in linear mixed model.
if (sum(Categ) > 0) {
Fctrs <- as.data.frame(apply(X = as.matrix(NoMissing[, Categ]), MARGIN = 2, FUN = as.factor))
names(Fctrs) <- names(NoMissing)[Categ]
Char <- data.frame(Stdz, Specimen = as.factor(NoMissing$Specimen), Plate = as.factor(NoMissing$Plate), Fctrs)
}
# Format ratio/interval-scale covariates for use in linear mixed model.
if (sum(Cont) > 0) {
RatioInterval <- as.data.frame(NoMissing[, Cont])
names(RatioInterval) <- names(NoMissing)[Cont]
Char <- data.frame(Stdz, Specimen = as.factor(NoMissing$Specimen), Plate = as.factor(NoMissing$Plate), RatioInterval)
}
# Allow for admixture of categorical and ratio/interval-scale covariates.
if ((sum(Categ) > 0) && (sum(Cont) > 0)) {
Char <- data.frame(Stdz, Specimen = as.factor(NoMissing$Specimen), Plate = as.factor(NoMissing$Plate), Fctrs, RatioInterval)
}
# Model formula for linear mixed model.
fml <- as.formula(paste(names(Stdz)[1]," ~ ", Covs))
# Fit linear mixed model with random intercept for each specimen for NC MFI.
cy.lme <- lme(fml, data = Char, random = ~ 1 | Specimen)
# Estimate population marginal means by plate.
LSM <- data.frame(SP = rep(names(Stdz)[1], times = length(unique(as.character(NoMissing$Plate)))), data.frame(lsmeans(cy.lme, "Plate")))
# Each iteration of loop fits linear mixed model and estimates population marginal means for MFI data of each separate SP.
for (cy in 2:(1 + QtyC)) {
fml <- as.formula(paste(names(Stdz)[cy]," ~ ", Covs))
cy.lme <- lme(fml, data = Char, random = ~ 1 | Specimen)
# Create dataframe consisting of SP name, plate ID, and population marginal mean for each plate.
cy.df <- data.frame(SP = rep(names(Stdz)[cy], times = length(unique(as.character(NoMissing$Plate)))), data.frame(lsmeans(cy.lme, "Plate")))
```

```
LSM <- rbind(LSM, cy.df)
}
# Create data set containing only estimated population marginal means by plate.
LSM.Only <- data.frame(SP = LSM[, 1], Plate = as.numeric(LSM[, 2]), lsmean = LSM[, 3])
# Sort these data by SP and then by plate.
LSM.Sort <- LSM.Only[order(LSM.Only[, 1], LSM.Only[, 2]), ]
# Sort data set of logarithm transformed MFI data and their means by SP.
DF1.Sort <- DF1[order(DF1$SP, DF1$Plate), ]
# Calculate overall mean of population marginal means by SP.
OverallLSM <- aggregate(LSM.Sort$lsmean, by = list(SP = LSM.Sort$SP), FUN = mean, simplify = T)
# Name columns of this data set.
names(OverallLSM) <- c("SP", "OM")
# Merge logarithm MFI data with population marginal means data by SP and plate.
With.LSM <- merge(x = DF1.Sort, y = LSM.Sort, by.x = c("SP","Plate"), by.y = c("SP","Plate"))
# Merge by SP this new file with average population marginal means data by SP.
With.Overall <- merge(x = With.LSM, y = OverallLSM, by.x = c("SP"), by.y = c("SP"))
# Sort resultant file by sequential well number.
Sort.W.LSM <- With.Overall[order(as.numeric(With.Overall$WellNo)), ]
# Create data frame of SP, plate, specimen ID, overall population marginal mean by SP, plate-detrended MFI, logarithm MFI, and well number.
FLSM <- data.frame(SP = Sort.W.LSM$SP, Plate = as.numeric(Sort.W.LSM$Plate), Specimen = Sort.W.LSM$Specimen, OM = Sort.W.LSM$OM,
PDMFI = Sort.W.LSM$RawMFI - Sort.W.LSM$lsmean + Sort.W.LSM$OM, RawMFI = Sort.W.LSM$RawMFI, WellNo = Sort.W.LSM$WellNo)
# Next 14 lines generate Figure 2. Start by getting minimum and maximum values for vertical axis (MFI).
# Calculate observed minimum and maximum for vertical axis.
RngY <- range(c(DF1$RawMFI, FLSM$PDMFI))
Fig2 <- xyplot(PDMFI ~ WellNo | SP, FLSM, as.table = T,
# Scales on both axes are designed to extend slightly beyond data. User may want to adjust this for x-axis in small data sets.
scales = list(alternating = 1, cex = 0.6), xlim = range(FLSM$WellNo, na.rm = T) + c(-10, 10),
ylim = RngY + c(-0.1 * RngY[1], 0.1 * RngY[2]),
subscripts = T, xlab = "Well (Ordered by Plate)", ylab = "Logarithm of Plate-Detrended Median Fluorescence Intensity (pMFI)",
panel = function(subscripts) {
panel.xyplot(FLSM$WellNo[subscripts], FLSM$PDMFI[subscripts], type = "p", col = "black", pch = 1, cex = 0.35)
panel.xyplot(FLSM$WellNo[subscripts], FLSM$PDMFI[subscripts], type = "p", col = FLSM$Plate[subscripts], pch = 16, cex = 0.2)
# Fit separate before and after detrending smooth curves.
panel.loess(DF1$WellNo[subscripts], DF1$RawMFI[subscripts], col = "blue", span = 2/3, degree = 2, family = "symmetric", lwd = 2)
panel.loess(FLSM$WellNo[subscripts], FLSM$PDMFI[subscripts], col = "red", span = 2/3, degree = 2, family = "symmetric", lwd = 2)
})
# Next 16 lines perform data processing in preparation for estimating technical variance.
# Using plate-detrended data, create wide data set (each SP in separate column, beginning with CHEX4).
LogLSMCyto <- matrix(FLSM[FLSM$SP == names(NoMissing)[4], 5], ncol = 1)
for (cy in 1:QtyC) {
LogLSMCyto <- cbind(LogLSMCyto, matrix(FLSM[FLSM$SP == names(NoMissing)[4 + cy], 5], ncol = 1))
}
# Transform back to original data scale.
BatchOut <- data.frame(exp(LogLSMCyto))
# Name columns by CHEX4 and SP.
names(BatchOut) <- names(NoMissing)[4:(4 + QtyC)]
# Create dataframe with specimen ID, plate ID, well ID, and plate-detrended, back-transformed data.
PlateDetrend <- data.frame(Specimen = as.character(NoMissing$Specimen), Plate = NoMissing$Plate, Well = NoMissing$Well, BatchOut)
# Next 11 lines calculate technical standard deviation (variance).
# Calculate mean by specimen of plate-detrended CHEX4 (NC, nonspecific binding) values.
PLM <- aggregate(x = PlateDetrend$CHEX4, by = list(ID = PlateDetrend$Specimen), FUN = mean, simplify = T, drop = T)
PreLogMean <- data.frame(Specimen = PLM[,1], Avg = PLM$x)
# Calculate finite-population variance by specimen of plate-detrended CHEX4 (NC, nonspecific binding) values.
PLV <- aggregate(x = PlateDetrend$CHEX4, by = list(ID = PlateDetrend$Specimen), FUN = FVar, simplify = T, drop = T)
PreLogVar <- data.frame(Specimen = PLV[,1], Var = PLV$x)
# Merge means and finite-population variances.
CHEX4WellAvgs <- merge(x = PreLogMean, y = PreLogVar, by.x = c("Specimen"), by.y = c("Specimen"))
# Calculate the approximate technical standard deviations per Eqn. 10 in Maecker et al. (2020).
ApproxTechVar <- CHEX4WellAvgs$Var / (CHEX4WellAvgs$Avg ^ 2)
CHEXWithinWellSDs <- sqrt(sum(na.exclude(ApproxTechVar)) / length(na.exclude(ApproxTechVar)))
# Next 11 lines are preparation for fitting error-in-variables regression model.
# Calculate average MFI for each SP and CHEX4 (NC, nonspecific binding).
WellAvgs <- aggregate(x = PlateDetrend[, -c(1:3)], by = list(Specimen = PlateDetrend$Specimen), FUN = mean, simplify = T, drop = T)
# Take natural logarithm of means.
ScaledData <- data.frame(Specimen = as.character(WellAvgs$Specimen), log(WellAvgs[, -1]))
# Sort data by specimen ID.
WellAvgsDF2 <- ScaledData[order(ScaledData$Specimen),]
```

```
# Create empty column for construction of following dataframe.
Fill <- rep(NA, QtyC * length(unique(WellAvgsDF2$Specimen)))
# Create dataframe for SP, CHEX4 (NC, nonspecific binding) plate-detrended MFI, fitted ONLS regression line, SP plate-detrended MFI,
# SP plate and nonspecific-binding detrended MFI, Specimen ID, and Specimen order number.
DF2 <- data.frame(SP = Fill, CHEX4pMFI = Fill, SPFit = Fill, SPpMFI = Fill, dpMFI = Fill, Specimen = Fill, OrderNo = Fill)
# Next 35 lines fit error-in-variables regression model.
# Calculate observed range of NC MFI values.
Support <- diff(range(WellAvgsDF2$CHEX4))
BW.obj <- vector(mode = "numeric", length = Repeats)
Crnt.Env <- environment(NULL)
# Each iteration of loop fits error-in-variables regression model to another SP.
for (cy in 1:QtyC) {
Repeater <- function(b) {
library(lpme)
# Bootstrap resampling.
Indx <- sample(1:nrow(WellAvgsDF2))
# Cross-validated fitting of error-in-variables regression model.
# Laplace distribution used per Huang and Zhou (2017).
BW.obj[b] <- meanregbwSIMEX(Y = WellAvgsDF2[Indx, 2 + cy], W = WellAvgsDF2[Indx, 2], method = Method, sig = CHEXWithinWellSDs,
error = "laplace", k_fold = 5, B = 2, h1 = Support/Seq, h2 = Support/Seq, length.h = length(Support/Seq), lconst = NULL,
rconst = NULL)$bw
}
# Parallel process cross-validated fitting of error-in-variables regression model.
cl <- makeCluster(getOption("cl.cores", CPUs))
# Set up random number generator across CPUs.
clusterSetRNGStream(cl, cy * 10)
clusterExport(cl = cl, varlist = list("BW.obj", "WellAvgsDF2", "Method", "CHEXWithinWellSDs", "Support", "Seq"), envir = Crnt.Env)
BW.obj.final <- parSapply(cl = cl, X = 1:Repeats, FUN = Repeater, simplify = T)
stopCluster(cl)
# Final fit of cross-validation optimized error-in-variables regression model
lpme.fit <- meanreg(Y = WellAvgsDF2[, 2 + cy], W = WellAvgsDF2[, 2], bw = mean(BW.obj.final), xgrid = WellAvgsDF2[, 2], method = Method,
sig = CHEXWithinWellSDs, error = "laplace")
# Place error-in-variables regression model results in a dataframe.
DF2[(1 + (cy - 1) * length(unique(WellAvgsDF2$Specimen))):(cy * length(unique(WellAvgsDF2$Specimen))), 1] <- rep(names(PlateDetrend[, -c(1:4)])[cy],
length(unique(WellAvgsDF2$Specimen)))
DF2[(1 + (cy - 1) * length(unique(WellAvgsDF2$Specimen))):(cy * length(unique(WellAvgsDF2$Specimen))), 2] <- lpme.fit$xgrid
DF2[(1 + (cy - 1) * length(unique(WellAvgsDF2$Specimen))):(cy * length(unique(WellAvgsDF2$Specimen))), 3] <- lpme.fit$yhat
DF2[(1 + (cy - 1) * length(unique(WellAvgsDF2$Specimen))):(cy * length(unique(WellAvgsDF2$Specimen))), 4] <- WellAvgsDF2[, cy + 2]
DF2[(1 + (cy - 1) * length(unique(WellAvgsDF2$Specimen))):(cy * length(unique(WellAvgsDF2$Specimen))), 5] <- WellAvgsDF2[, cy + 2] - lpme.fit$yhat
DF2[(1 + (cy - 1) * length(unique(WellAvgsDF2$Specimen))):(cy * length(unique(WellAvgsDF2$Specimen))), 6] <- as.character(WellAvgsDF2$Specimen)
DF2[(1 + (cy - 1) * length(unique(WellAvgsDF2$Specimen))):(cy * length(unique(WellAvgsDF2$Specimen))), 7] <- 1:length(unique(WellAvgsDF2$Specimen))
}
# Next 14 lines generate Figure 3.
DF2 <- DF2[order(DF2$SP, DF2$CHEX4pMFI),]
# Calculate observed minimum and maximum for each axis.
RngX <- range(DF2$CHEX4pMFI, na.rm = T)
RngY <- range(c(DF1$RawMFI, FLSM$PDMFI, DF2$SPFit, DF2$SPpMFI), na.rm = T)
Fig3 <- xyplot(SPpMFI ~ CHEX4pMFI | SP, DF2, as.table = T,
# Scales on both axes are designed to extend slightly beyond data. User may want to adjust this for x-axis in small data sets.
scales = list(alternating = 1, cex = 0.6), xlim = RngX + c(-0.1 * RngX[1], 0.1 * RngX[2]),
ylim = RngY + c(-0.1 * RngY[1], 0.1 * RngY[2]),
subscripts = T, xlab = "Log Plate-Detrended Preprocessed Median Fluorescence Intensity for NC Microbeads",
ylab = "Log Plate-Detrended Preprocessed Median Fluorescence Intensity (pMFI)",
panel = function(subscripts) {
panel.xyplot(DF2$CHEX4pMFI[subscripts], DF2$SPpMFI[subscripts], type = "p", col = "black", pch= 1, cex = 0.35)
panel.xyplot(DF2$CHEX4pMFI[subscripts], DF2$SPFit[subscripts], type = "l", col = "red", lwd = 2)
})
# Next 14 lines generate Figure 4.
DF3 <- DF2[order(DF2$OrderNo),]
# Calculate observed minimum and maximum for vertical axis.
RngY <- range(c(DF1$RawMFI, FLSM$PDMFI, DF2$SPFit, DF2$SPpMFI, DF3$dpMFI), na.rm = T)
Fig4 <- xyplot(dpMFI ~ OrderNo | SP, DF3, as.table = T,
# Scales on both axes are designed to extend slightly beyond data. User may want to adjust this for x-axis in small data sets.
scales = list(alternating = 1, cex = 0.6), xlim = range(DF3$OrderNo, na.rm = T) + c(-10, 10),
ylim = RngY + c(0.1 * RngY[1], 0.1 * RngY[2]),
subscripts = T, xlab = "Specimen Order (Ordered by Plate)", ylab = "Fully-Detrended Median Fluorescence Intensity (dpMFI)",
panel = function(subscripts) {
```

```
panel.xyplot(DF3$OrderNo[subscripts], DF3$dpMFI[subscripts], type = "p", col = "black", pch = 1, cex = 0.35)
panel.xyplot(DF3$OrderNo[subscripts], DF3$dpMFI[subscripts], type = "p", col = "turquoise", pch = 16, cex = 0.25)
panel.loess(DF3$OrderNo[subscripts], DF3$dpMFI[subscripts], col = "red", span = 2/3, degree = 2, family = "symmetric", lwd = 2)
})
# Close out by returning all generated figures plus dpMFI output data set.
DF3 <- DF2[order(DF2$Specimen, DF2$SP),]
OutAll <- list(Fig1 = Fig1, Fig2 = Fig2, Fig3 = Fig3, Fig4 = Fig4, DF3 = DF3[, -7])
return(OutAll)
}
##############
# END SCRIPT #
##############
########################
# EXAMPLE APPLICATION #
########################
# Settings.
# Set directory for output figures and dataset.
Directory <- "C:\\MyData\\SP Project"
# Set resolution of output figures in dots per inch (DPI).
Res <- 1100
# Set quantity of SPs (excludes nonspecific binding).
QtyC <- 40
# Read in input data set.
RawData <- read.csv(file = "C:\\MyData\\SP Project\\My SP Data.csv", header = T)
# Set quantity of repeats of crossvalidation.
Repeats <- 12
# Specify covariates.
Covs <- "Plate + Treatment + BMI"
# Provide volume of specimen.
VSpecimen <- 100
# Provide fold dilution.
Fold <- 2
# Provide volume of pipette.
VPipette <- 10
# Run utility that loads installed libraries needed for script.
LibLoader()
# Detrend data for plate and nonspecific binding artifacts.
# proc.time used to calculate and output total run time.
Start <- proc.time()
OutAll <- CHEX4Detrend(InputData = RawData, Method = "HZ", Seq <- 2:6, Cont = c(45, 48), Categ = c(46, 47, 49))
proc.time() - Start
# Activate graphics device for production of figures, separately for each figure.
# User may need to adjust height and width.
png(filename = paste0(Directory, "\\Figure 1 Raw MFI Data.png"),
width = ceiling(sqrt(QtyC)), height = ceiling(sqrt(QtyC)), units = "in", bg = "white", res = Res, restoreConsole = T)
OutAll[1]
dev.off()
png(filename = paste0(Directory, "\\Figure 2 pMFI Data.png"),
width = ceiling(sqrt(QtyC)), height = ceiling(sqrt(QtyC)), units = "in", bg = "white", res = Res, restoreConsole = T)
OutAll[2]
dev.off()
png(filename = paste0(Directory, "\\Figure 3 Fit of Error in Variables Regression Model to SP pMFI and Nonspecific Binding pMFI.png"),
width = ceiling(sqrt(QtyC)), height = ceiling(sqrt(QtyC)), units = "in", bg = "white", res = Res, restoreConsole = T)
OutAll[3]
dev.off()
png(filename = paste0(Directory, "\\Figure 4 Final dpMFI.png"),
width = ceiling(sqrt(QtyC)), height = ceiling(sqrt(QtyC)), units = "in", bg = "white", res = Res, restoreConsole = T)
OutAll[4]
dev.off()
# Write to comma-delimited file the SP MFI data that have been detrended for plate and nonspecific binding artifacts.
Final <- na.exclude(data.frame(OutAll[5]))
names(Final) <- c("SP", "CHEX4pMFI", "SPFit", "SPpMFI", "dpMFI", "Specimen")
# To scale dpMFI properly, add to mean SPpMFI.
Final <- Final[order(Final$SP),]
SPpMFI.Means <- aggregate(x = Final$SPpMFI, by = list(Final$SP), FUN = mean)
names(SPpMFI.Means) <- c("SP", "SPMean")
Final.wMeans <- merge(x = Final, y = SPpMFI.Means, by.x = c("SP"), by.y = c("SP"))
```

```
names(Final.wMeans) <- c("SP", "NCpMFI", "SPFit", "SPpMFI", "Old.dpMFI", "Specimen", "SPMean")
Final.wnewdpMFI <- data.frame(Final.wMeans[,c(1:4,6)], dpMFI = Final.wMeans$Old.dpMFI + Final.wMeans$SPMean)
Final.wnewdpMFI <- Final.wnewdpMFI[order(Final.wnewdpMFI$Specimen, Final.wnewdpMFI$SP),]
# Write to output file.
write.table(Final.wnewdpMFI, file = paste0(Directory, "\\dpMFI.csv"), append = F, quote = F, sep = ",", row.names = F, col.names = T, na = '.')
```