**Supplementary Data**

**Fig. S1**: The distances between nuclei using triplicates or single measurements (experiment 2 and 3)

CHO cells were exposed to 2.8, 11, 45, 181 and 725 mIU/mL for 48 hours and nuclei were stained with DAPI. Fluorescence images were acquired, stitched (10x10 images) and converted to binary images. Shown is the ratio of the geometric mean distance between nuclei for the indicated PTx concentration divided by the geometric mean distance between nuclei of cells grown in medium, and the confidence interval for each concentration of PTx. Measures of three wells were combined (A and B) or analyzed individually (C and D).

---

Hoonakker et al.: The Nearest Neighbor Nuclei Method to Objectify Analysis of Pertussis Toxin-Induced Clustering

Supplementary Data

doi:10.14573/altex.2012171s

This is an Open Access article distributed under the terms of the Creative Commons Attribution 4.0 International license (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is appropriately cited.
Fig. S2: The effect of different cell densities
CHO cells were seeded at a concentration of 1.5x 10^4 (A), 3 x 10^4 (B) or 6x 10^4 (C) cells/mL, exposed to 0, 2.8, 11, 45, 181 and 725 mIU/mL PTx for 48 hours and nuclei were visualized with DAPI. Fluorescence images were acquired, stitched (10x10) and converted to binary images (using ImageJ). Shown is the ratio of the geometric mean distance between nuclei for the indicated PTx concentration divided by the geometric mean distance between nuclei grown in medium, and the confidence interval for each concentration of PTx for one experiment.

---

Fig. S3: Statistical analysis of the distance to one, two, three or four nearest neighboring nuclei in three experiments
CHO cells were exposed to 0, 2.8, 3.8, 5.7, 11, 45, 181 and 725 mIU/mL PTx for 48 hours and nuclei were visualized with DAPI. Fluorescent images were acquired, stitched (10x10) and converted to binary images. Shown is the ratio of the geometric mean distance between nuclei for the indicated PTx concentration divided by the geometric mean distance between nuclei grown in medium, and the confidence interval for each concentration of PTx for three independent experiments (A-C).
Text S1: KNN script

```python
from ij.gui import Overlay, Line, OvalRoi
from ij import IJ
from ij.measure import ResultsTable
from ij.measure import Calibration
from java.awt import Color
import math

### Some functions ###

# Brute-Force KNN
def knn(data, k):
    # Just take the data, find the K nearest neighbors to each point

    the_knn=[]
    for i in range(len(data)):
        d = []
        for j in range(len(data)):
            d.append((j, dist(data[i], data[j])))
        # Sort
        d_sort = sorted(d, key=lambda thed: thed[1])
        # Keep only the k nearest
        the_knn.append(d_sort[1:(k+1)])
    return the_knn

# Euclidean Distance 2D
def dist(p0, p1):
    return math.sqrt((p0[0] - p1[0])**2 + (p0[1] - p1[1])**2)

# Convenience function to go from a ROI to points
def roiToPointList(image):
    rt = ResultsTable.getResultsTable()
    xcolidx = rt.getColumnIndex("X")
    x = rt.getColumn(xcolidx)
    ycolidx = rt.getColumnIndex("Y")
    y = rt.getColumn(ycolidx)
    p = [x, y]
    points = map(list, zip(*p))
    return points

def showKNNResults(the_knn):
    # Overlay
    ov = Overlay()
    the_avg=[]
    # Draw each neighbor as a line and draw the average as a circle
    for i in range(len(the_knn)):
        avg = sum([d[1] for d in the_knn[i]]) / k
        the_avg.append(avg)
        o = OvalRoi(points[i][0] - avg, points[i][1] - avg, 2*avg, 2*avg)
        o.setStrokeColor(Color.decode("#00FFFF"))
        ov.add(o)
    for j in range(k):
        # XY Coordinates of current point
        a = points[i];
        b = points[the_knn[i][j][0]]
        # XY Coordinates of neighbor k
        l = Line(a[0], a[1], b[0], b[1])
```
l.setStrokeColor(Color.decode("#FF00FF"))
ov.add(l)

# Build Results Table
cal = imp.getCalibration()
frt = ResultsTable()

for i in range(len(the_knn)):
    frt.incrementCounter()
    frt.addValue("Point", i)
    if( cal.scaled() ):
        frt.addValue("Average distance \[px\]", cal.get(cal.getX(avg[i])))
    frt.addValue("Average distance \[" + cal.getXUnit() + "]", cal.getX(avg[i]))

frt.show(str(k) + " Nearest Neightbors Average Distances")
imp.setOverlay(ov)

### Starting the script ###

# Get Coordinates
points = roiToPointList(imp)

# Compute K Nearest Neighbors
the_knn = knn(points,k)

# Display the result table and an overlay
showKNNResults(the_knn)

Text S2:

library(tidyverse)
library(readxl)
library(pwr)
library(openxlsx)

# Define function to calculate power
tpwr <- function(Mut, SDt, Nt, Muc, SDc, Nc){
    ES <- abs(Mut - Muc)
    SDpop <- sqrt(((Nt - 1) * SDt^2 + (Nc - 1) * SDc^2) / (Nt + Nc - 2))
    zb <- sqrt((ES^2 * (Nc / Nt)) / (SDpop^2 * (Nc / Nt + 1)))
    return(zb)
}

###############################
# Read data for Exp 4 BRP PTx #
###############################

# Loop through # neighbour cells
FileIn <- "EXP4_PL2_20180515_all_BRP PTx.xlsx"
sheetIn <- "EXP4_20180515_NB"
WsSaveName <- "Exp4B.RData"

# Read data from file and store in Exp4B as tibble
Exp4B <- NULL
for (i in 1:4){
    tX <- read_xlsx(FileIn, sheet = paste(sheetIn, i, sep = ""))
    tY <- pivot_longer(tX, cols = 2:25, names_to = "Well", values_to = "Distance")
    tY$PTx <- str_sub(tY$Well,2,2)
}

# Read data for Exp 4 BRP PTx #

# Loop through # neighbour cells
FileIn <- "EXP4_PL2_20180515_all_BRP PTx.xlsx"
sheetIn <- "EXP4_20180515_NB"
WsSaveName <- "Exp4B.RData"

# Read data from file and store in Exp4B as tibble
Exp4B <- NULL
for (i in 1:4){
    tX <- read_xlsx(FileIn, sheet = paste(sheetIn, i, sep = ""))
    tY <- pivot_longer(tX, cols = 2:25, names_to = "Well", values_to = "Distance")
    tY$PTx <- str_sub(tY$Well,2,2)
}
tY$Repl <- str_sub(tY$Well, 3, 3)
tY$NB <- i
    # Re order columns
tY <- tY[, c(4:6, 1:3)]
Exp4B <- bind_rows(Exp4B, tY)
}
    # Clean up and recode PTx value
rm(tX, tY, i)
Exp4B$PTx[Exp4B$PTx == "M"] <- "0"
Exp4B$PTx <- as.numeric(Exp4B$PTx)
    # Calculate PTx concentration
Exp4B$PTx[Exp4B$PTx > 0] <- 16/ (4^ (Exp4B$PTx[Exp4B$PTx > 0] - 1))
    # Multiply distance x 1000
Exp4B$Distance <- Exp4B$Distance * 1000
    # Vector with PTx concentrations > 0
concPTx <- sort(unique(Exp4B$PTx))[-1]
    # Save workspace
save.image(WsSaveName)
    # Summaries by PTx conc all replicates combined
sumExp4BAll <- Exp4B %>%
    group_by(NB, PTx) %>%
    summarise(count = sum(!is.na(Distance)), mean.l2 = mean(log2(Distance)), sd.l2 = sd(log2(Distance)))
    # Calculate summary values
sumExp4BAll$geomean <- 2*sumExp4BAll$mean.l2
sumExp4BAll$geolow <- 2^(sumExp4BAll$mean.l2 - qt(0.975, sumExp4BAll$count-1) * sumExp4BAll$sd.l2 / sqrt(sumExp4BAll$count))
sumExp4BAll$geoup <- 2^(sumExp4BAll$mean.l2 + qt(0.975, sumExp4BAll$count-1) * sumExp4BAll$sd.l2 / sqrt(sumExp4BAll$count))
    # Make ID Variable for left join later
sumExp4BAll$ID <- paste(sumExp4BAll$NB, sumExp4BAll$PTx)
    # Calculate post-hoc power for combined replicates
Exp4BPwr <- NULL
for (i in 1:4){
    tX <- Exp4B[Exp4B$NB == i & Exp4B$PTx == 0, ]
    nc <- length(na.omit(tX$Distance))
    mc <- mean(log2(tX$Distance))
    sc <- sd(log2(tX$Distance))
    for (j in 1:7){
        tY <- Exp4B[Exp4B$NB == i & Exp4B$PTx == concPTx[j], ]
        tT <- t.test(log2(tX$Distance), log2(tY$Distance), var.equal = TRUE)
        np <- length(na.omit(tY$Distance))
        mp <- mean(log2(tY$Distance))
        sp <- sd(log2(tY$Distance))
        Exp4BPwr <- bind_rows(Exp4BPwr, bind_cols(ID = paste(i, concPTx[j]), NB = i, PTx = concPTx[j], Cf = paste(0, "vs", concPTx[j]), MeanCtrl = tT$estimate[1], MeanPTx = tT$estimate[2], SDCtrl = sc, SDPTx = sp, NCtrl = nc, NPTx = np, Delta = tT$estimate[1] - tT$estimate[2], DeltaLow = tT$conf.int[1], DeltaUpp = tT$conf.int[2], Df = tT$parameter, tValue = tT$statistic, PValue = tT$p.value,
            # Reverse order for ratio to obtain PTx/Medium
            Ratio = 2^(mp - mc), RatLow = 2^(mp - tT$conf.int[2]), RatUpp = 2^(mp - tT$conf.int[1]), Power = tpwr(mp, sp, np, mc, sc, nc)))
    }
}
rm(tT, tX, tY, nc, sc, mc, np, sp, mp, i, j)
    # Summaries by PTx conc replicates separated
sumExp4BRep <- Exp4B %>%

group_by(NB, Repl, PTx) %>%
summarise(count = sum(!is.na(Distance)), mean.l2 = mean(log2(Distance)), sd.l2 = sd(log2(Distance)))

# Calculate summary values
sumExp4BRep$geomean <- 2^sumExp4BRep$mean.l2
sumExp4BRep$geolow <- 2^(sumExp4BRep$mean.l2 - qt(0.975, sumExp4BRep$count) * sumExp4BRep$sd.l2 / sqrt(sumExp4BRep$count))
sumExp4BRep$geoupp <- 2^(sumExp4BRep$mean.l2 + qt(0.975, sumExp4BRep$count) * sumExp4BRep$sd.l2 / sqrt(sumExp4BRep$count))

# Make ID Variable for left join later
sumExp4BRep$ID <- paste(sumExp4BRep$Repl, sumExp4BRep$NB, sumExp4BRep$PTx)
sumExp4BRep$Repl <- unclass(factor(sumExp4BRep$Repl))

# Calculate Post-Hoc power for separate replicates
reps <- c("A", "B", "C")
Exp4BPwrReplic <- NULL
for (k in 1:3){
  for (i in 1:4){
    tW <- Exp4B[Exp4B$NB == i & Exp4B$Repl == reps[k], ]
    tX <- tW[tW$NB == i & tW$PTx == 0, ]
    nc <- length(na.omit(tX$Distance))
    mc <- mean(log2(tX$Distance))
    sc <- sd(log2(tX$Distance))
    for (j in 1:7){
      tY <- tW[tW$NB == i & tW$PTx == concPTx[j], ]
      tT <- t.test(log2(tY$Distance), log2(tX$Distance), var.equal = TRUE)
      np <- length(na.omit(tY$Distance))
      mp <- mean(log2(tX$Distance))
      sp <- sd(log2(tY$Distance))
      Exp4BPwrReplic <- bind_rows(Exp4BPwrReplic, bind_cols(ID = paste(k, i, concPTx[j]), Replic = k, NB = i, PTx = concPTx[j]),
                                 CI = paste(0, " vs", concPTx[j]), MeanCtrl = tT$estimate[1], MeanPTx = tT$estimate[2], SDCtrl = sc, SDPTx = sp,
                                 NCtrl = nc, NPTx = np, Delta = tT$estimate[2] - tT$estimate[1], DeltaLow = tT$conf.int[1], DeltaUpp = tT$conf.int[2],
                                 DI = tT$parameter, tValue = tT$statistic, PValue = tT$p.value,
                                 # Reverse order for ratio to obtain PTx/Medium
                                 Ratio = 2^(mp - mc), RatLow = 2^-(tT$conf.int[2], RatUpp = 2^-tT$conf.int[1], Power = tpwr(mp, sp, np, mc, sc, nc)))
    }
  }
}
rm(tT, tW, tX, tY, nc, sc, mc, np, sp, mp, i, j, k)

########################################################################
# Figures #
########################################################################
# Define labels for PTx concentration scale
xlabs <- c(0.001, 0.01, 0.1, 1, 10, 100)

# Data for Figure with geometric nucleus distance vs PTx, separate facets for Neighbour
# get data for distance PTx (tX) and medium (tY)
tX <- sumExp4BAll[sumExp4BAll$PTx > 0, ]
tY <- sumExp4BAll[sumExp4BAll$PTx == 0, ]

# Figure with geometric nucleus distance vs PTx, separate facets for Neighbour
Exp4BFig1 <- ggplot(data = tX, aes(x = PTx, y = geomean))+facet_wrap(~NB, nrow=1)+
  labs(y="Geometric mean nucleus distance", x="Pertussis Toxin (IU/mL)")+
scale_x_log10(breaks = xlabs, labels = xlabs)+annotation_logticks(sides="b")+
geom_hline(data = tY, aes(x = NULL, yintercept = geolow), colour="red")+
geom_hline(data = tY, aes(x = NULL, yintercept = geoupp), colour="red")+
geom_point()+geom_errorbar(aes(ymin = geolow, ymax = geoupp))
theme(panel.grid.minor=element_blank())

rm(tX, tY)

# Data for Figure with geometric nucleus distance vs PTx, separate facets for Neighbour and Replicate
# tX has summaries for all PTx > 0
# tY has the lower and upper bounds for PTx = 0 per Neighbour count
# Prepare figures with delta distance for combined replicates for comparisons per PTX > 0
# Get data for figure from Exp4BPwr
# Define vertical position for Power values.
tX <- Exp4BPwr

tX$Replic <- factor(tX$Replic, labels = c("A", "B", "C"))
textAt <- max(tX$RatUpp) + 0.05 * (max(tX$RatUpp) - min(tX$RatLow))
Exp4BFig4 <- ggplot(tX, aes(x = PTx, y = Ratio))+facet_grid(Replic ~ NB)+
  labs(x = "Pertussis Toxin (IU/mL)", y = "Ratio Exposed / Ctrl")+
  scale_x_log10(breaks = xlabs, labels = xlabs)+annotation_logticks()+
  scale_y_log10(breaks = seq(from = 0.6, to = 2, by = 0.05))+
  geom_hline(yintercept = 1, colour = "red", lty = 2)+
  geom_point()+geom_errorbar(aes(ymin = RatLow, ymax = RatUpp))+
  geom_text(aes(y = textAt, label = round(Power*100,0)), size = 2)+
  theme(panel.grid.minor = element_blank())

Exp4BFig4 <- ggplot(tX, aes(x = PTx, y = Ratio))+facet_grid(Replic ~ NB)+
  labs(x = "Pertussis Toxin (IU/mL)", y = "Ratio Exposed / Ctrl")+
  scale_x_log10(breaks = xlabs, labels = xlabs)+annotation_logticks()+
  scale_y_log10(breaks = seq(from = 0.6, to = 2, by = 0.05))+
  geom_hline(yintercept = 1, colour = "red", lty = 2)+
  geom_point()+geom_errorbar(aes(ymin = RatLow, ymax = RatUpp))+
  geom_text(aes(y = textAt, label = round(Power*100,0)), size = 2)+
  theme(panel.grid.minor = element_blank())

save.image(WsSaveName)

# Export data
load(WsSaveName)
savePath <- paste(getwd(), "/Results/", sep = "")

# Save Figures as svg
ggsave(paste(savePath, "Exp4B Figure 1.svg", sep = ""), Exp4BFig1, width = 9, height = 4)
ggsave(paste(savePath, "Exp4B Figure 2.svg", sep = ""), Exp4BFig2, width = 9, height = 8)
ggsave(paste(savePath, "Exp4B Figure 3.svg", sep = ""), Exp4BFig3, width = 9, height = 8)
ggsave(paste(savePath, "Exp4B Figure 4.svg", sep = ""), Exp4BFig4, width = 9, height = 8)

# Save as pdf
Example of table containing the raw data (S1A: sample 1A, S1B: sample 1B etc. SMA: sample medium A)

|     | S1A | S1B | S1C | S2A | S2B | S2C | S3A | S3B | S3C | S4A | S4B | S4C | S5A | S5B | S5C | S6A | S6B | S6C | S7A | S7B | S7C | SMA | SMB | SMC |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1   | 0.0098| 0.0115| 0.0122| 0.0077| 0.0082| 0.0087| 0.0095| 0.0109| 0.0127| 0.0118| 0.0094| 0.0128| 0.0174| 0.0136| 0.0151| 0.0053| 0.0089| 0.0137| 0.0114| 0.0203| 0.0177| 0.0017| 0.0071| 0.0077|
| 2   | 0.0082| 0.0097| 0.0101| 0.0073| 0.0078| 0.0074| 0.0083| 0.0085| 0.0115| 0.0112| 0.0112| 0.0114| 0.0107| 0.0137| 0.0144| 0.0251| 0.0086| 0.0137| 0.0214| 0.0168| 0.0125| 0.0125| 0.0187| 0.0175|
| 3   | 0.0096| 0.0072| 0.0091| 0.0107| 0.0096| 0.0074| 0.0131| 0.0104| 0.0187| 0.0135| 0.0039| 0.0148| 0.0110| 0.0204| 0.0115| 0.0207| 0.0214| 0.0142| 0.0159| 0.0088| 0.0103| 0.0075| 0.0100|
| 4   | 0.0095| 0.0094| 0.0111| 0.0107| 0.0096| 0.0084| 0.0106| 0.0058| 0.0086| 0.0125| 0.0059| 0.0084| 0.0078| 0.0084| 0.1066| 0.0166| 0.0124| 0.0293| 0.1217| 0.0399| 0.0121| 0.0044|
| 5   | 0.0095| 0.0086| 0.0175| 0.0066| 0.0202| 0.0772| 0.0051| 0.0056| 0.0131| 0.0082| 0.0185| 0.0131| 0.0086| 0.1059| 0.0087| 0.0148| 0.0294| 0.0281| 0.0280| 0.0196| 0.0193| 0.0076| 0.0089|
| 6   | 0.0100| 0.0057| 0.0048| 0.0064| 0.0086| 0.012| 0.0156| 0.0010| 0.0103| 0.0111| 0.0091| 0.0077| 0.0285| 0.0125| 0.0159| 0.0115| 0.0119| 0.0064| 0.0242| 0.0172| 0.0176| 0.0123| 0.0111|
| 7   | 0.0100| 0.0127| 0.0129| 0.0116| 0.0096| 0.0084| 0.0068| 0.0146| 0.0060| 0.0102| 0.0111| 0.0077| 0.0285| 0.0125| 0.0159| 0.0115| 0.0119| 0.0064| 0.0242| 0.0172| 0.0176| 0.0123| 0.0111|
| 8   | 0.0064| 0.0015| 0.0076| 0.0194| 0.0086| 0.0551| 0.0312| 0.0161| 0.0126| 0.0532| 0.0064| 0.0157| 0.0183| 0.0253| 0.0102| 0.0187| 0.0144| 0.1157| 0.0172| 0.0199| 0.0264| 0.0164|
| 9   | 0.0085| 0.0137| 0.0003| 0.0008| 0.0194| 0.0184| 0.0112| 0.0082| 0.0237| 0.0087| 0.0088| 0.0077| 0.0138| 0.0062| 0.0136| 0.0095| 0.0248| 0.0249| 0.0397| 0.1137| 0.0130| 0.0148|
| 10  | 0.0052| 0.0274| 0.0129| 0.0077| 0.0099| 0.0109| 0.0084| 0.0076| 0.0076| 0.0143| 0.0133| 0.0107| 0.0232| 0.0116| 0.0213| 0.0237| 0.0293| 0.0081| 0.0116| 0.0139| 0.0095| 0.0103|
| 11  | 0.0062| 0.0086| 0.0186| 0.0086| 0.0076| 0.0074| 0.0087| 0.0102| 0.0139| 0.0109| 0.0076| 0.0165| 0.0121| 0.0187| 0.0103| 0.0255| 0.0080| 0.0176| 0.0218| 0.0131| 0.0097| 0.0078|
| 12  | 0.0128| 0.0062| 0.0278| 0.0004| 0.0037| 0.0052| 0.0434| 0.0141| 0.0090| 0.0096| 0.0107| 0.0113| 0.0006| 0.0109| 0.0061| 0.0204| 0.0177| 0.0143| 0.0121| 0.0097| 0.0047|
| 13  | 0.0088| 0.0072| 0.0046| 0.0067| 0.0096| 0.0052| 0.0048| 0.0177| 0.0074| 0.0125| 0.0111| 0.0048| 0.0082| 0.0071| 0.0111| 0.0077| 0.0132| 0.0073| 0.0147| 0.0095| 0.0091|

ALTEX 39(x), SUPPLEMENTARY DATA