ddPCR

J.Schuy

30 4 2021

ddPCR Script The following script takes raw calculated copy number / µL from ddPCR analysed in Quanta Soft software as input.

Built with 4.0.3

```
## Warning: package 'ggplot2' was built under R version 4.0.4

## Warning: package 'dplyr' was built under R version 4.0.4

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```
#Working directory
# setwd(paste(rstudioapi::getSourceEditorContext()$path, "/../..", sep = ""))

#create folder for output graphs
if (!dir.exists(paste(getwd(), "/Graphs_", ProjectName, "/", sep=""))) {
  dir.create(paste(getwd(), "/Graphs_", ProjectName, "/", sep=""))      }
#create folder for extracted data like lists and dataframes
if (!dir.exists(paste(getwd(), "/ExtractedData_", ProjectName, "/", sep=""))) {
  dir.create(paste(getwd(), "/ExtractedData_", ProjectName, "/", sep=""))     }

#load all samples
```

```
if (file.exists(paste(getwd(), "/ExtractedData_", ProjectName, "/saveEnvironment.RData", sep=""))) {
  load(paste(getwd(), "/ExtractedData_", ProjectName, "/saveEnvironment.RData", sep=""))} else {


#import data, genes for current Project (here: allChr)
data_raw <- read_excel(paste0(getwd(), "/../",inputname), tabname)

#clean input
data_clean <- data_raw[1:24,1:18]

#remove unnecessary columns
data_reduced <- data_clean[,c(1:4,6:8, 14)]



#save cleaned df
write_xlsx(data_reduced, paste0(getwd(),"/ExtractedData_", ProjectName, "/raw_cleaned.xlsx"))

#clean env
rm(data_raw)

save.image(file = paste(getwd(), "/ExtractedData_", ProjectName, "/saveEnvironment.RData", sep=""))

}

#show
head(data_reduced)
```

```
## # A tibble: 6 x 8
##   Well  Sample Target `Conc(copies/µL)` `Accepted Droplets` Positives Negatives
##   <chr> <chr>  <chr>              <dbl>               <dbl>     <dbl>     <dbl>
## 1 A01   N1     jct1                85.2               19420      1356     18064
## 2 A02   N1     jct1                90.0               17416      1282     16134
## 3 A03   N1     jct1                90.5               19202      1422     17780
## 4 A04   N1     rpp30              238.                17898      3281     14617
## 5 A05   N1     rpp30              249.                18511      3528     14983
## 6 A06   N1     rpp30              210.                18900      3083     15817
## # ... with 1 more variable: Threshold1 <dbl>
```

Normalise

Reference genes is avaraged per sample, then the conc of target is divided by that ref value yielding one normalized fraction of amplified structure per input. Here: biolg. triplicates -> three values

A second step includes averaging these three values per sample, returns mean and sd

```
## normalize ----
PlotName <- "ddPCR8"

#get backup
dd8 <- data_reduced

#derive reference value for each sample
```

```
dd8_mean <- dd8 %>%
  group_by(Sample, .add = TRUE) %>%
  group_by(Target, .add = TRUE) %>%
  summarise(mean = mean(`Conc(copies/µL)`),
            SD = sd(`Conc(copies/µL)`))
```

```
## `summarise()` has grouped output by 'Sample'. You can override using the `.groups` argument.
```

```
#normalize each target with respective ref value
dd8_target <- dd8[dd8$Target == target,]

##loop for normalization
dd8_norm <- vector()
for (i in levels(factor(dd8_target$Sample))){

  subdat <- dd8_target[dd8_target$Sample == i,]

  #add ref value
  subdat$refmean <- dd8_mean$mean[dd8_mean$Target == reference &
                                  dd8_mean$Sample == i]

  #normalize for "i"
  subdat$concNorm <- subdat$`Conc(copies/µL)` / subdat$refmean

  #mean and sd for plot
  subdat$mean <- mean(subdat$concNorm)
  subdat$sd <- sd(subdat$concNorm)

  #output
  dd8_norm <- rbind(dd8_norm, subdat)
}

head(dd8_norm)
```

```
## # A tibble: 6 x 12
##   Well  Sample Target `Conc(copies/µL)` `Accepted Droplets` Positives Negatives
##   <chr> <chr>  <chr>              <dbl>               <dbl>     <dbl>     <dbl>
## 1 D01   blood  jct1                70.7               17892      1043     16849
## 2 D02   blood  jct1                80.6               16921      1121     15800
## 3 D03   blood  jct1                69.8               17904      1032     16872
## 4 C01   F1     jct1               120.                17444      1696     15748
## 5 C02   F1     jct1               131.                16743      1765     14978
## 6 C03   F1     jct1               113.                18458      1684     16774
## # ... with 5 more variables: Threshold1 <dbl>, refmean <dbl>, concNorm <dbl>,
## #   mean <dbl>, sd <dbl>
```

Data render and export

Before continuing, we need to manipulate the data. The group *Sample* becomes a factor. The sd and mean will be manually calculated per Sample. This is for exporting the data

```
#plot preparation
dd8_norm$Sample <- factor(dd8_norm$Sample, levels = levels(factor(dd8_norm$Sample))[1:4])
dd8_norm2 <- dd8_norm %>% group_by(Sample, .add = TRUE) %>%
              group_by(Target, .add = TRUE) %>%
              summarise(mean = mean(concNorm), SD = sd(concNorm))
```

```
## `summarise()` has grouped output by 'Sample'. You can override using the `.groups` argument.
```

```
#export numbers
write_xlsx(dd8_norm, path=paste(getwd(), "/ExtractedData_", ProjectName, "/", "numbersForPlot.xlsx", sep
write_xlsx(dd8_norm2, path=paste(getwd(), "/ExtractedData_", ProjectName, "/", "numbersForPlot_short.xls

head(dd8_norm2)
```

```
## # A tibble: 4 x 4
## # Groups:   Sample [4]
##   Sample Target  mean      SD
##   <fct>  <chr>  <dbl>   <dbl>
## 1 blood  jct1   0.437 0.0356
## 2 F1     jct1   0.449 0.0344
## 3 I1     jct1   0.244 0.00875
## 4 N1     jct1   0.381 0.0127
```

Statistics

We use the one-way anova, because of 4 groups. First tests will check for normal distributed data and homogeneity. Followed by data exploring statistics.

```
#check for normality
for (i in levels(dd8_norm$Sample)){
  subdat <- dd8_norm[dd8_norm$Sample == i,]
  # subdat$normality <- shapiro.test(subdat$concNorm)$p.value
  print(paste(i, shapiro.test(subdat$concNorm)$p.value, sep = ": "))
  }
```

```
## [1] "blood: 0.129735509517597"
## [1] "F1: 0.82518178713395"
## [1] "I1: 0.518509102187032"
## [1] "N1: 0.183197587004605"
```

```
  # all tests are insignificant -> samples are normality distributed

#homogeneity of variance
leveneTest(dd8_norm$concNorm, dd8_norm$Sample, center = mean)
```

```
## Levene's Test for Homogeneity of Variance (center = mean)
##       Df F value Pr(>F)
## group  3  2.5912 0.1252
##        8
```

```
  #not significant -> variances are not different=there are homogenous

#anova, one way anova
celltype<-aov(dd8_norm$concNorm ~ dd8_norm$Sample, data = dd8_norm)
summary(celltype)
```

```
##                 Df  Sum Sq  Mean Sq F value    Pr(>F)
## dd8_norm$Sample  3 0.07911 0.026368   39.24 0.0000393 ***
## Residuals        8 0.00538 0.000672
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#yes, there is a difference

#post-hoc, bonferroni (smaller sample size)
pairwise.t.test(dd8_norm$concNorm, dd8_norm$Sample, p.adjust.method = "bonferroni")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  dd8_norm$concNorm and dd8_norm$Sample
##
##    blood  F1       I1
## F1 1.0000 -        -
## I1 0.0001 0.000066 -
## N1 0.1830 0.0780   0.0012
##
## P value adjustment method: bonferroni
```

```
pairwise.t.test(dd8_norm$concNorm, dd8_norm$Sample, p.adjust.method = "BH")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  dd8_norm$concNorm and dd8_norm$Sample
##
##    blood    F1       I1
## F1 0.59295  -        -
## I1 0.000051 0.000051 -
## N1 0.03661  0.01950  0.00038
##
## P value adjustment method: BH
```

```
pairwise.t.test(dd8_norm$concNorm, dd8_norm$Sample, p.adjust.method = "none")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  dd8_norm$concNorm and dd8_norm$Sample
##
##    blood    F1       I1
```

```
## F1 0.59295  -        -
## I1 0.000017 0.000011 -
## N1 0.03051  0.01300  0.00019
##
## P value adjustment method: none
```

```
#linear model (=baseline blood sample, only three tests, no correction)
summary(lm(dd8_norm$concNorm ~dd8_norm$Sample))
```

```
##
## Call:
## lm(formula = dd8_norm$concNorm ~ dd8_norm$Sample)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.032408 -0.015466 -0.000554  0.007537  0.041045
##
## Coefficients:
##                    Estimate Std. Error t value      Pr(>|t|)
## (Intercept)         0.43689    0.01497  29.191 0.00000000205 ***
## dd8_norm$SampleF1   0.01178    0.02117   0.557        0.5929
## dd8_norm$SampleI1 -0.19266    0.02117  -9.102 0.00001705075 ***
## dd8_norm$SampleN1 -0.05552    0.02117  -2.623        0.0305 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02592 on 8 degrees of freedom
## Multiple R-squared:  0.9364, Adjusted R-squared:  0.9125
## F-statistic: 39.24 on 3 and 8 DF,  p-value: 0.00003931
```

```
#t-test for comparison
test <- dd8_norm[dd8_norm$Sample %in% c("blood", "N1"),]
t.test(test$concNorm~test$Sample, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  test$concNorm by test$Sample
## t = 2.5425, df = 4, p-value = 0.06381
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.00510799  0.11614376
## sample estimates:
## mean in group blood    mean in group N1
##           0.4368895           0.3813716
```

```
#statistics for the comparison with the expected 0.5 perfect heterozygosity
stats <- as.data.frame(matrix(nrow = length(levels(dd8_norm$Sample)), ncol = 4))
names(stats)<- c("Sample", "pval", "padj", "sign")

for(m in 1:length(levels(dd8_norm$Sample))){
  subdat <- dd8_norm[dd8_norm$Sample == levels(dd8_norm$Sample)[m],]
```

```
  stats$Sample[m]<- levels(dd8_norm$Sample)[m]
  stats$pval[m]<- t.test(subdat$concNorm, mu = 0.5, alternative = "less", var.equal = TRUE)$p.value
  stats$padj[m] <- p.adjust(stats$pval[m], method = "bonferroni", n = length(levels(dd8_norm$Sample)))

  #add pval ID
  if(stats$padj[m] < 0.001){
    stats$sign[m] <- "***"
  }else if (stats$padj[m] < 0.01){
    stats$sign[m] <- "**"
  }else if (stats$padj[m] < 0.05){
    stats$sign[m] <- "*"
  }else{
    stats$sign[m] <- "ns"
  }
  # print(paste(levels(dd8_norm$Sample)[m], stats$sign[m], stats$pval[m], stats$padj[m], sep = ": "))
}
tibble(stats)
```

```
## # A tibble: 4 x 4
##   Sample     pval     padj sign
##   <chr>     <dbl>    <dbl> <chr>
## 1 blood  0.0459   0.184    ns
## 2 F1     0.0613   0.245    ns
## 3 I1     0.000195 0.000780 ***
## 4 N1     0.00190  0.00759  **
```

Plot

```
#plot ----
plot_norm_stat <- ggplot(dd8_norm,aes(x=Sample, y=concNorm)) +
  theme_classic(base_size=20) +
  geom_hline(yintercept = 0.5, linetype =2,size=0.5, colour = "black")+
  # geom_point(aes(col=Sample),position=position_dodge(width=0.7), size=8, alpha=0.6, shape=16)+
  geom_jitter(aes(col=Sample),size=8, alpha=0.6, shape=16, width = 0.1)+
  scale_color_manual(values=colorsForLegend[c(5,9,3,8)])+
  scale_x_discrete(labels = c("Blood", "Fibroblasts", "iPSCs", "NESCs"))+
  stat_summary(fun.data="mean_se", geom="errorbar", width=0.1, size=1)+
  geom_boxplot(aes(x=Sample, y=mean), width=0.4, size=1, colour="black")+
  labs(x="", y = "Fraction of ring chromosome", fill = element_blank()) +
  scale_y_continuous(breaks = seq(0, 0.6, by=0.1), limits = c(0,0.6))+
  annotate(geom = "text", label = "expected fraction for perfect heterozygosity", x=0.2, y = 0.5, size =
  theme(
    plot.subtitle = element_text(colour="black",face="italic", size = 15),
    axis.ticks = element_line(colour="black"),
    axis.text.x = element_text(colour="black", hjust=0.5, vjust=1, angle=0),
    axis.text.y = element_text(colour="black", hjust=0.5),
    legend.title = element_text(colour = "black", size = 15),
    strip.background = element_rect(size=1.5, linetype="solid", colour ="black"),
    strip.text = element_text(size = 14),
    legend.position = "none"
  )+
```

```
  annotate("text", label =paste(stats$sign)[1:2],x = stats$Sample[1:2], y = 0.58, size = 5, colour = "bl
  annotate("text", label =paste(stats$sign)[3:4],x = stats$Sample[3:4], y = 0.58, size = 7, colour = "bl
  annotate("text", label ="p-value:",x = 0.25, y = 0.58, size = 5, colour = "black", hjust=0, vjust=0.5
  ggtitle(paste("Mosaic state of the ring chromosome in RD_P26", sep=""),
          subtitle = paste("ddPCR (n=3), Anova: p<0.0001, bonferroni corrected post-hoc(t.test) for H0:r

ggsave(filename = paste("Graphs_", ProjectName, "/", PlotName, "_stats", gsub(":", "-", format(Sys.time
       height = 20, width = 25, unit="cm", limitsize = FALSE)
plot_norm_stat
```