Test replicate function

Shuyu Zheng

2021-01-15

1. Functions

```
repResponse <- function(data){
  sum.data <- data %>%
    dplyr::group_by(PairIndex, Conc1, Conc2) %>%
    dplyr::summarise(mean = mean(Response),
                     sd = sd(Response),
                     n = n(), .groups = "keep") %>%
    mutate(sem = sd/sqrt(n)) %>%
    mutate(error = qt(0.975,df=n-1)*sem,
           lower_95CI = mean - error,
           upper_95CI = mean + error)
}

repSynergy <- function(data, method){
  pair <- unique(data$PairIndex)
  scores <- NULL
  for (p in pair){
    tmp <- dplyr::filter(data, PairIndex == p)
    repN <- tmp %>%
      dplyr::group_by(PairIndex, Conc1, Conc2) %>%
      dplyr::summarise(count = n(), .groups = "keep") %>%
      ungroup() %>%
      dplyr::select(count) %>%
      unique() %>%
      unlist()
    if (length(unique(repN)) == 1){
      rest <- tmp %>%
        mutate(index = seq(1, n()))
      response <- NULL
      for (i in seq(1, (repN - 1))){
        t <- rest %>%
          dplyr::group_by(PairIndex, Conc1, Conc2) %>%
          dplyr::sample_n(1) %>%
          dplyr::mutate(block_id = i)
        response <- rbind.data.frame(response, t)
        rest <- rest %>%
          dplyr::filter(!index %in% t$index)
      }
      response <- rbind.data.frame(response, dplyr::mutate(rest, block_id = repN))
      # Calculate scoree
```

```
      blocks <- unique(response$block_id)
      score <- NULL
      for (i in blocks){
        response.mat <- reshape2::acast(Conc1 ~ Conc2, data = response[which(response$block_id == i),],
                                        value.var = "Response")
        t <- switch(method,
                        ZIP = ZIP(response.mat),
                        HSA = HSA(response.mat),
                        Bliss = Bliss(response.mat),
                        Loewe = Loewe(response.mat))
        t <- reshape2::melt(t)
        colnames(t) <- c("Conc1", "Conc2", "synergy")
        score <- rbind.data.frame(score, t)
      }
      sum.score <- score %>%
        dplyr::group_by(Conc1, Conc2) %>%
        dplyr::summarise(mean = mean(synergy),
                         sd = sd(synergy),
                         n = n(), .groups = "keep") %>%
        mutate(sem = sd/sqrt(n)) %>%
        mutate(error = qt(0.975,df=n-1)*sem,
               lower_95CI = mean - error,
               upper_95CI = mean + error) %>%
        mutate(PairIndex = p)
      scores <- rbind.data.frame(scores, sum.score)
    }
  }
  return(scores)
}

plot2DrugRes <- function(data.plot, block = 1, metric = NULL, color.conc = "black",
                         color.low.response = "green", color.high.response = "red"){
  drug.pairs <- data.plot$drug.pair

  response.df<- data.plot$response
    # Transform conc into factor
  response.df[, c("conc_r", "conc_c")] <-
    lapply(response.df[, c("conc_r", "conc_c")], factor)

  # Round the response values and their statistics
  response.df[, !grepl("(block_id|conc)", colnames(response.df), perl = TRUE)] <-
    sapply(response.df[, !grepl("(block_id|conc)", colnames(response.df), perl = TRUE)],
           signif, digits = 3)
  if (!is.null(metric)){
    avail_metrics <- grep("(response|conc_r|conc_c|block_id)",
                          colnames(response.df), perl = TRUE, value = TRUE,
                          invert = TRUE)
    if (length(avail_metrics) == 0){
      warning("The input dataset doesn't contain the statistic metrics for replicates.")
      response.df <- response.df %>%
        dplyr::mutate(text = response)
    } else {
      if (metric %in% colnames(response.df)){
```

```
        response.df <- response.df %>%
          dplyr::mutate(text = paste0(response, "\n \u00B1 ", !!as.name(metric)))
      } else {
        warning("Specified metric '", metric, "' is not available. Available ",
                "metrics are: '", paste(avail_metrics, collapse = "', '"), "'")
        response.df <- response.df %>%
          dplyr::mutate(text = response)
      }
    }
  } else {
    response.df <- response.df %>%
      dplyr::mutate(text = response)
  }

  plot.title <- paste("Dose-response matrix (inhibition)", "\nBlockID:",
                      block, sep = " ")

  # plot heatmap for dose-response matrix
  axis.x.text <- signif(as.numeric(levels(response.df$conc_c)), 4)
  axis.y.text <- signif(as.numeric(levels(response.df$conc_r)), 4)
  dose.response.p <- ggplot2::ggplot(data = response.df,
                                     aes(x=conc_c, y=conc_r, fill=response)) +
    ggplot2::geom_tile() +
    ggplot2::geom_text(ggplot2::aes(label = text), size = 2.4) +
    ggplot2::scale_fill_gradient2(low = color.low.response,
                                  high = color.high.response,
                                  midpoint = 0, name = "Inhibition (%)") +
    ggplot2::scale_x_discrete(labels = axis.x.text) +
    ggplot2::scale_y_discrete(labels = axis.y.text) +
    ggplot2::xlab(paste0(drug.pairs$drug_col, " (", drug.pairs$conc_c_unit, ")")) +
    ggplot2::ylab(paste0(drug.pairs$drug_row, " (", drug.pairs$conc_r_unit, ")"))

  # Add the title for heatmap
  dose.response.p <- dose.response.p +
    ggplot2::ggtitle(plot.title) +
    ggplot2::theme(plot.title = ggplot2::element_text(size = 20))

  # Set label's style of heatmap
  dose.response.p <- dose.response.p +
    ggplot2::theme(axis.text.x = ggplot2::element_text(color = color.conc,
                                                       face = "bold",
                                                       size = 15),
                   axis.text.y = ggplot2::element_text(color = color.conc,
                                                       face = "bold",
                                                       size = 15),
                   axis.title = ggplot2::element_text(size = 15))
  return(dose.response.p)
}
```

1. Caculate statistics for response (%inhibition) value

2. Visualize dose response

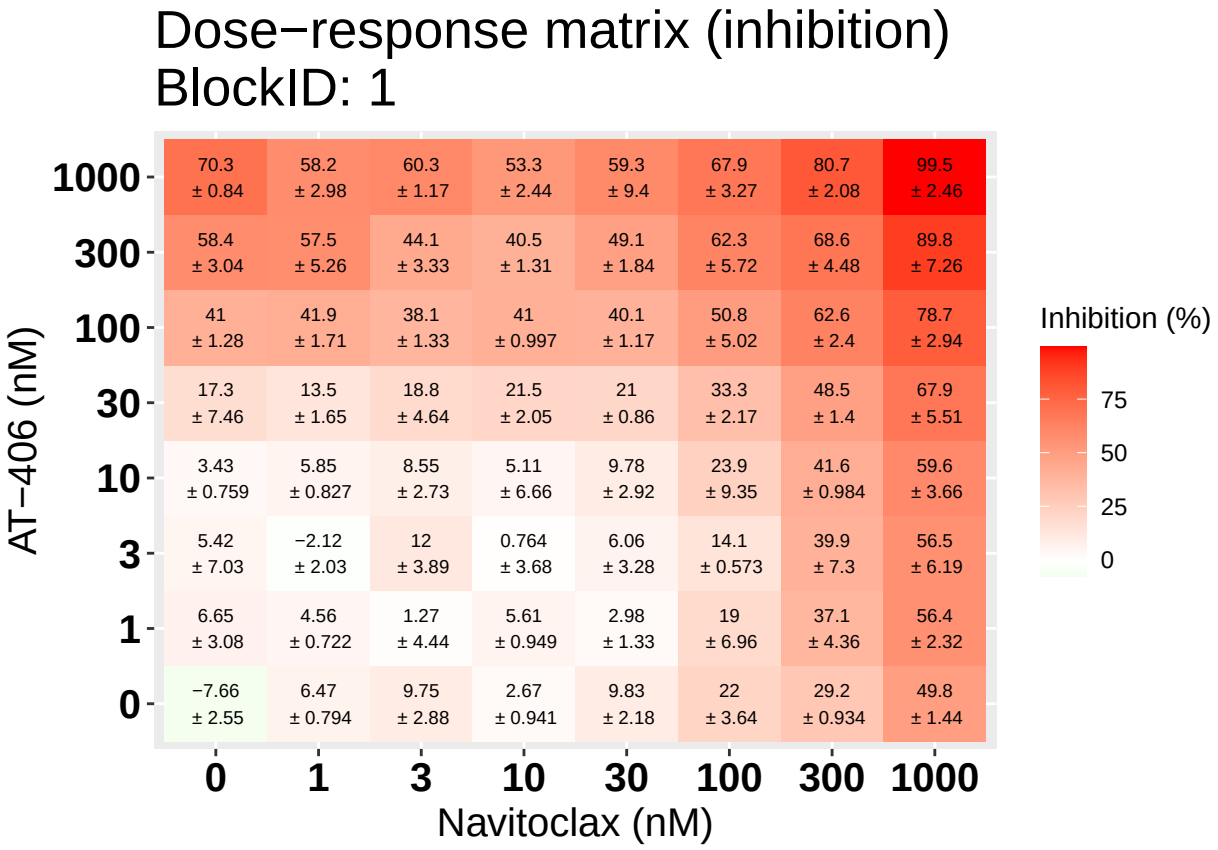
Plotting function requires a input data (`data.plot`). It is a list contains 2 data frames for **only one block**:

1. **data.pairs** must contain columns:

- drug_row
- drug_col
- conc_r_unit
- conc_c_unit

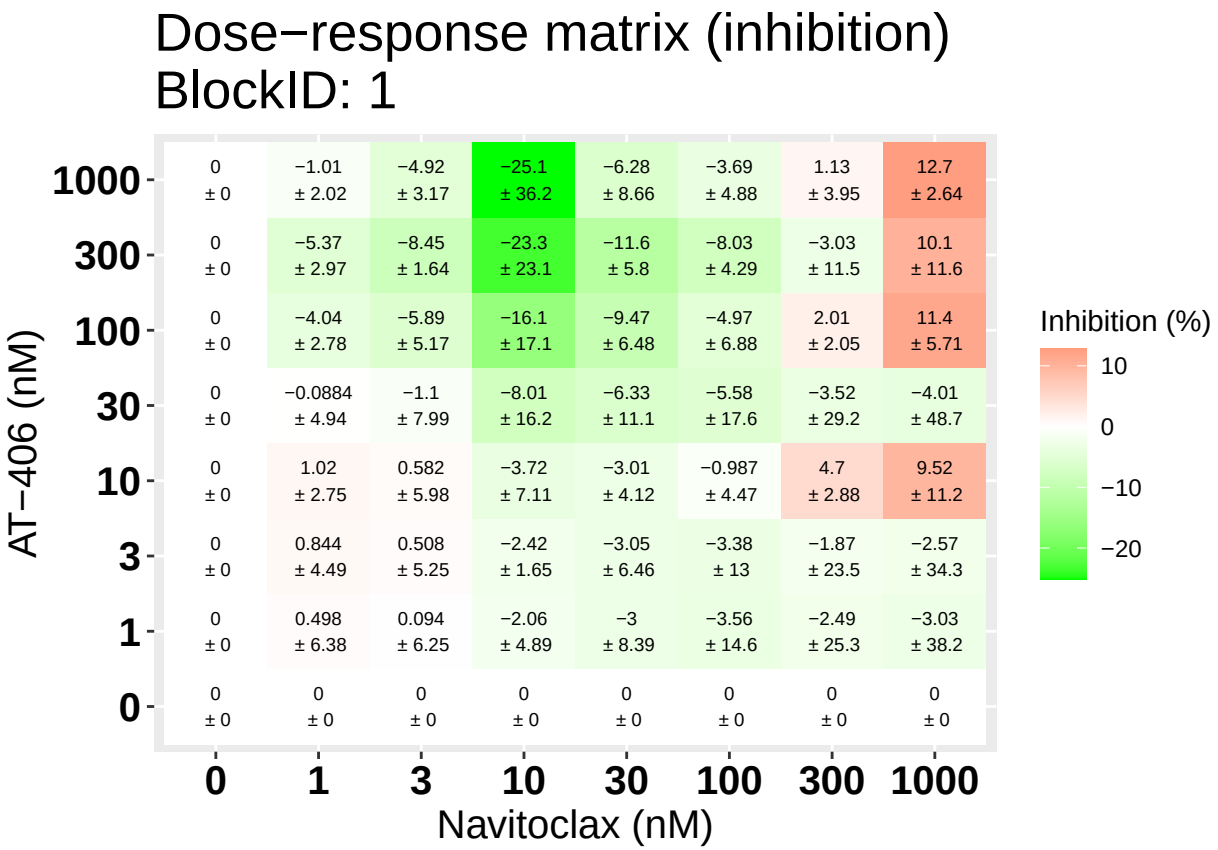
2. **response** must contain columns:

- conc_r
- conc_c
- response (main value which will be plotted in the heatmap)
- ... several columns for the response value.



3. Calculate synergy scores

4. Visualize synergy scores



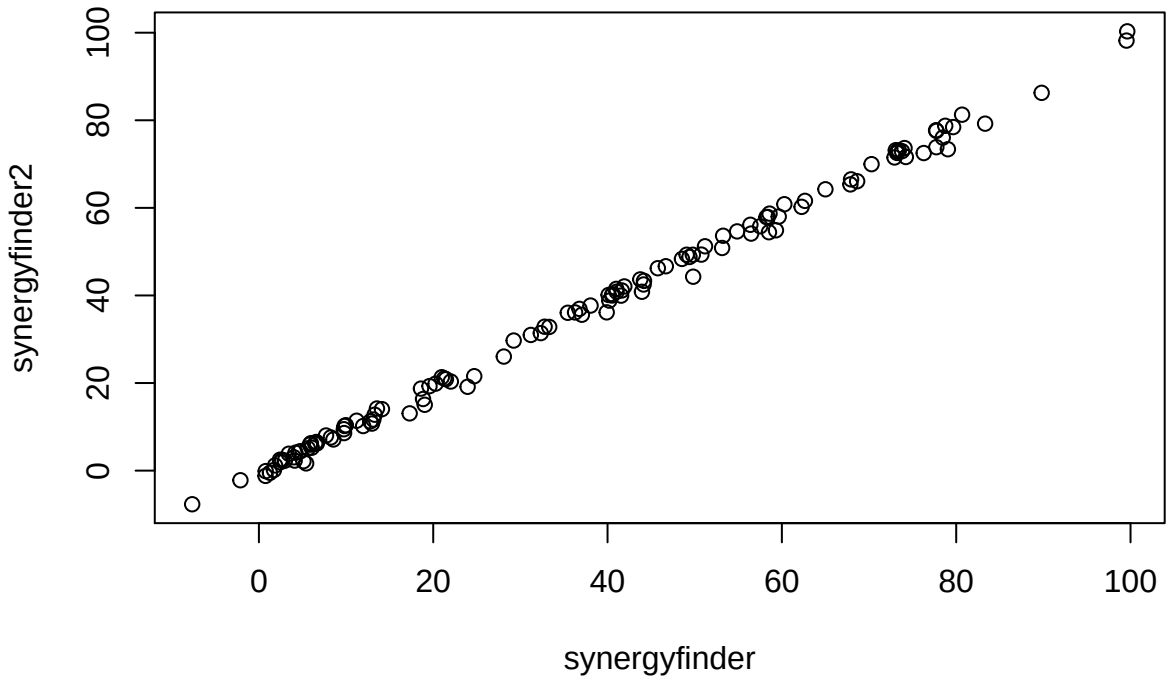
5. table from SynergyFinder 2

6. Compare

Response mean

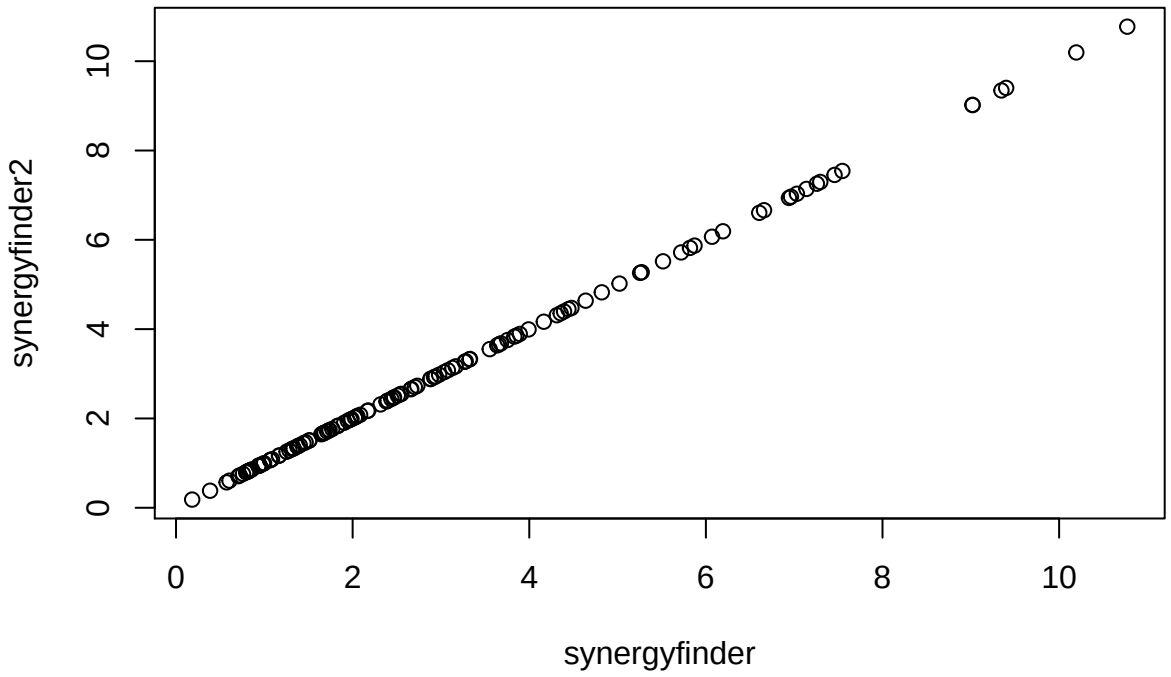
```
## Maximum absolute difference:  5.656749
```

```
##
##  Pearson's product-moment correlation
##
## data:  diff$mean and diff$Relative_inhibition
## t = 219.33, df = 126, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9981446 0.9990793
## sample estimates:
##       cor
## 0.9986929
```



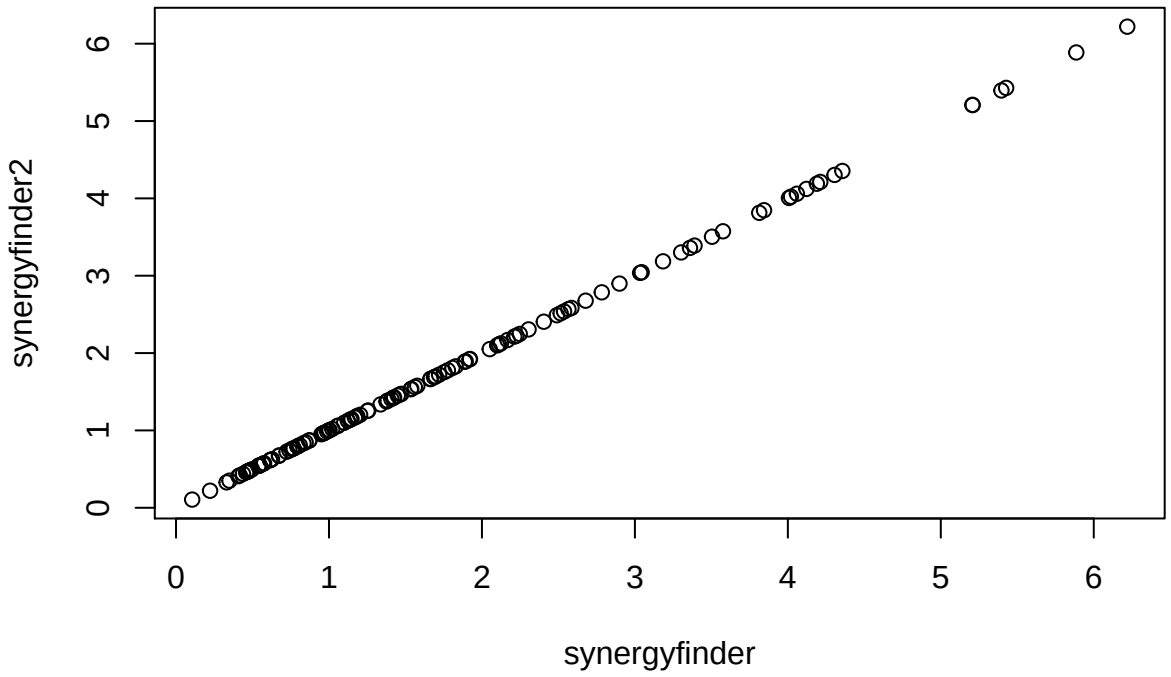
Response SD

```
## SD:

## Maximum absolute difference:  0.004813518

##
##  Pearson's product-moment correlation
##
## data:  diff$sd and diff$SD
## t = 12850, df = 126, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9999995 0.9999997
## sample estimates:
##       cor
## 0.9999996
```



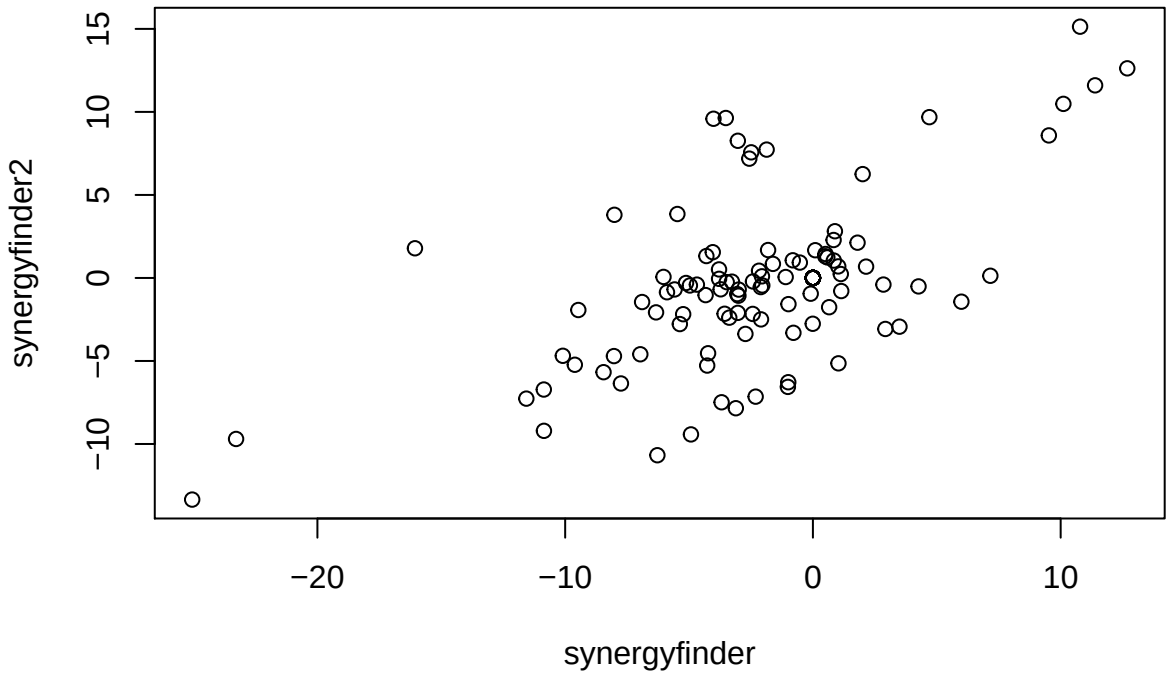
Response SEM

```
## SEM:

## Maximum absolute difference:  0.002779086

##
##  Pearson's product-moment correlation
##
## data:  diff$sem and diff$SEM
## t = 12850, df = 126, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9999995 0.9999997
## sample estimates:
##       cor
## 0.9999996
```



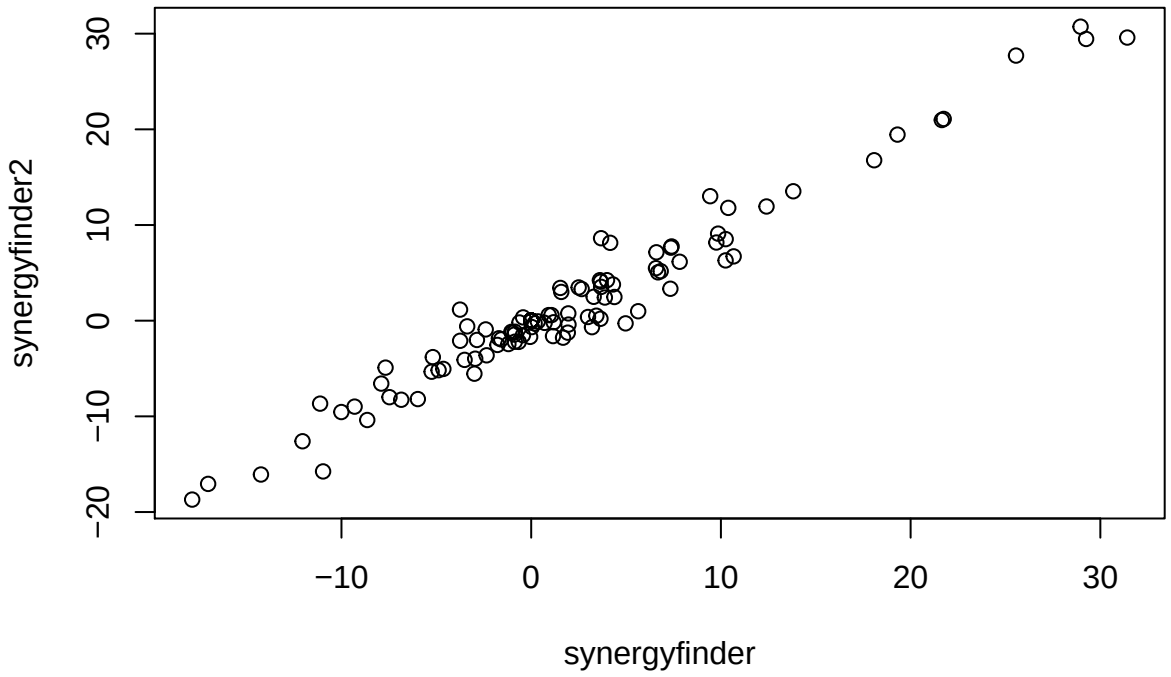
ZIP

```
## ZIP:
## Maximum absolute difference:  17.84971
##
##  Pearson's product-moment correlation
##
## data:  diff$ZIP_mean and diff$ZIP_Synergy
## t = 8.5815, df = 126, p-value = 2.946e-14
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4849256 0.7064260
## sample estimates:
##       cor
## 0.6073481
```



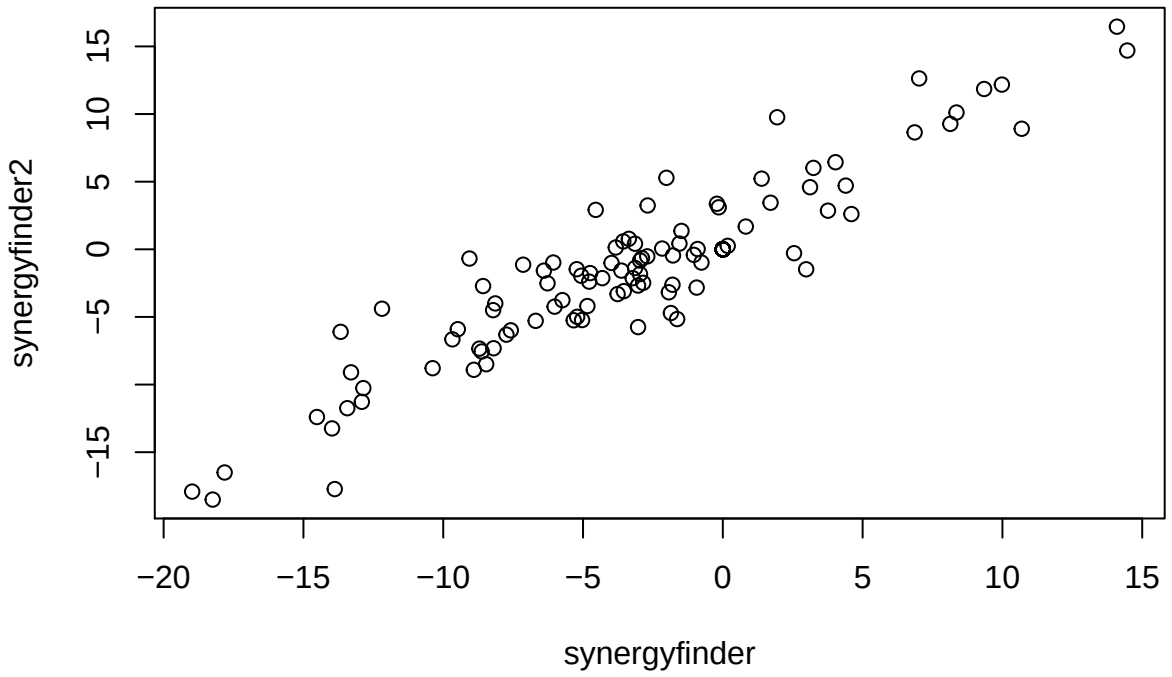
HSA

```
## HSA:

## Maximum absolute difference:  5.255413

##
##  Pearson's product-moment correlation
##
## data:  diff$HSA_mean and diff$HSA_Synergy
## t = 52.619, df = 126, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9688967 0.9844515
## sample estimates:
##      cor
## 0.977994
```



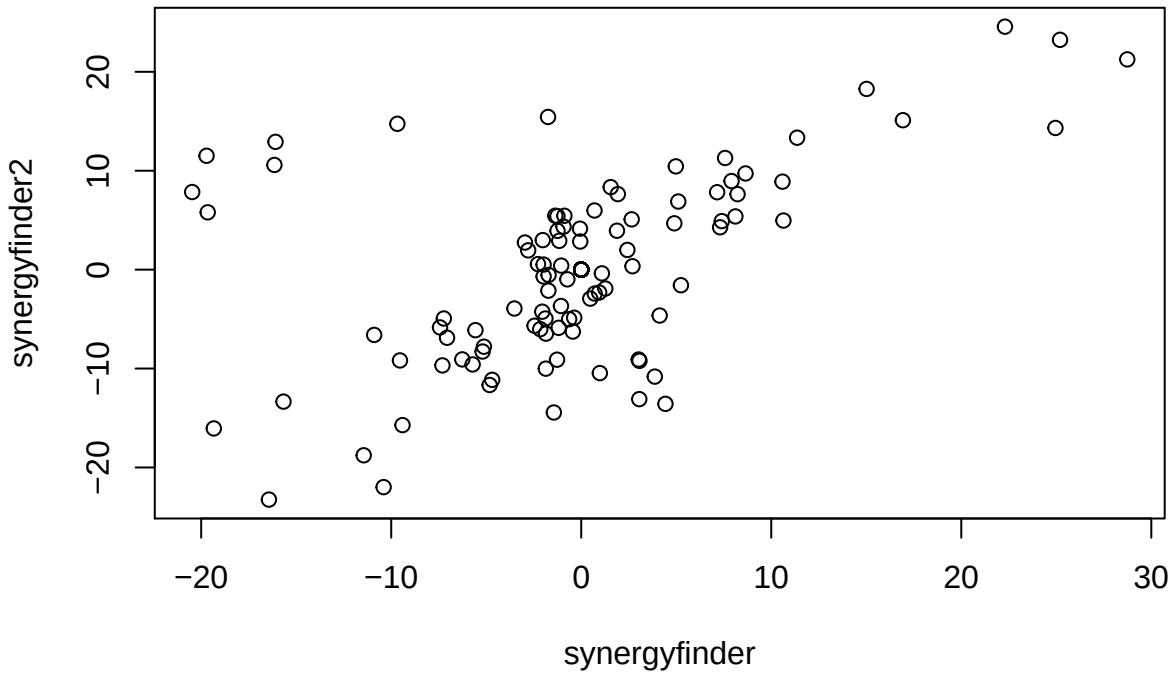
Bliss

```
## Bliss:

## Maximum absolute difference:  8.378764

##
##  Pearson's product-moment correlation
##
## data:  diff$Bliss_mean and diff$Bliss_Synergy
## t = 26.374, df = 126, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8884610 0.9430788
## sample estimates:
##       cor
## 0.9201303
```



Loewe

```
## Loewe:

## Maximum absolute difference:  31.23033

##
##  Pearson's product-moment correlation
##
## data:  diff$Loewe_mean and diff$Loewe_Synergy
## t = 7.1272, df = 126, p-value = 7.015e-11
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3996675 0.6491696
## sample estimates:
##       cor
## 0.5360224
```


synergyfinder2
20
10
0
−10
−20
−20
−10
0
10
20
30
synergyfinder