Reproducibility metrics for context-specific CRISPR screens

Maximilian Billmann[1,2,8,#], Henry N. Ward[3], Michael Aregger[4,5], Michael Costanzo[5], Brenda J. Andrews[5,6], Charles Boone[5,6], Jason Moffat[5,6,7], Chad L. Myers[1,3,#]

[1]Department of Computer Science and Engineering, University of Minnesota – Twin Cities, Minneapolis, Minnesota 55455, USA
[2]Institute of Human Genetics, University of Bonn, School of Medicine and University Hospital Bonn, Bonn, 53127, Germany
[3]Bioinformatics and Computational Biology Graduate Program, University of Minnesota – Twin Cities, Minneapolis, Minnesota 55455, USA
[4]National Cancer Institute, National Institutes of Health, Frederick, Maryland 21702, USA
[5]Donnelly Centre, University of Toronto, Toronto, Ontario M5S3E1, Canada
[6]Department of Molecular Genetics, University of Toronto, Toronto, Ontario M5S1A8, Canada
[7]Program in Genetics and Genome Biology, The Hospital for Sick Children, Peter Gilgan Research and Learning Centre, 686 Bay Street, Toronto, ON M5G0A4, Canada
[8]Lead contact

[#]Correspondance: chadm@umn.edu, maximilian.billmann@gmail.com

Supplementary Information Table of Contents

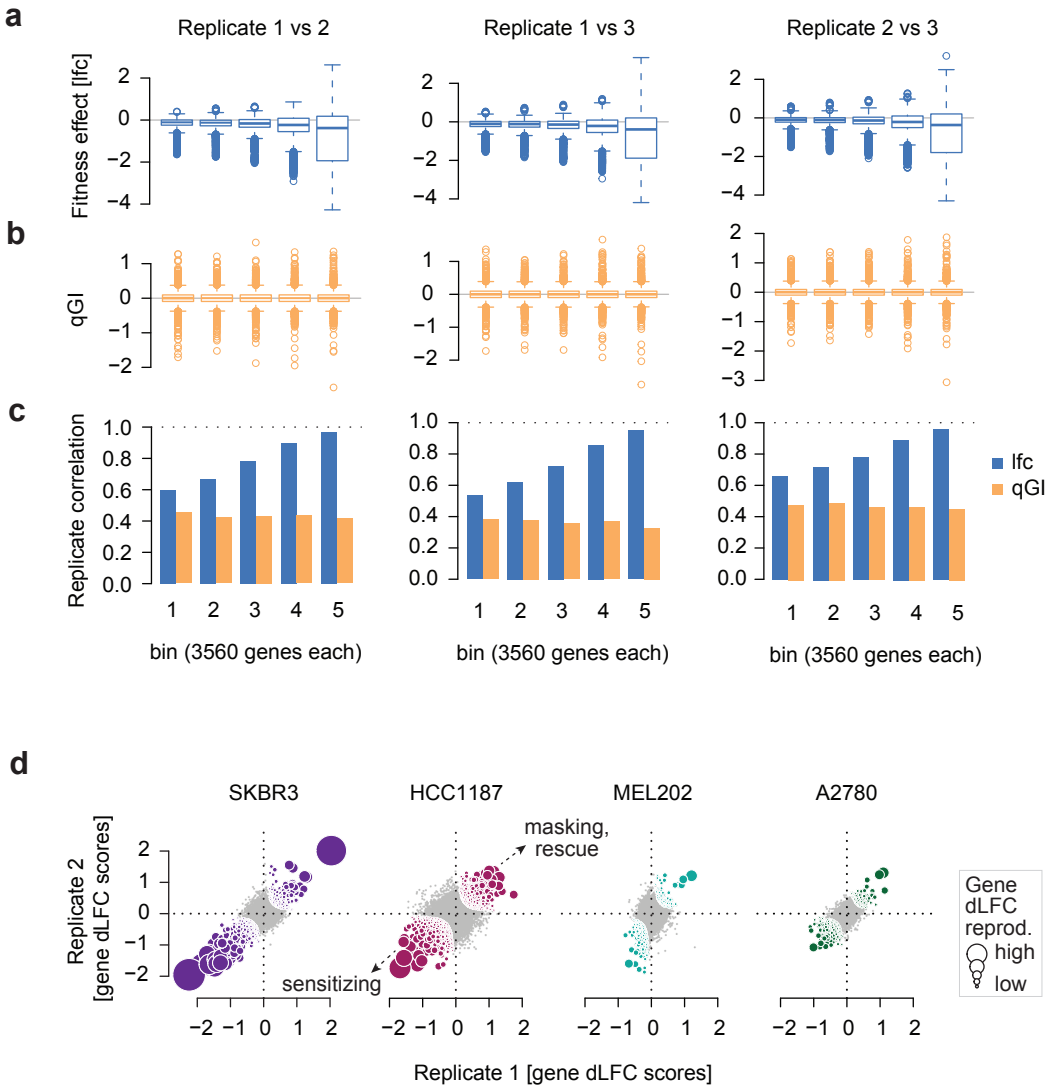


Figure S1: Relation of qGI and lfc-based screen replicate correlation. (a) Fitness effects of genes in equally sized bins. The fitness effects represent the mean LFC scores across the two FASN KO replicates screens indicated. **(b)** qGI effects of genes in equally sized bins. qGI scores represent the mean across the two FASN KO screens indicated. **(c)** Replicate Pearson correlation coefficients per bin based on LFC (blue) and qGI (orange) scores. Each bin contains exactly 3560 genes that were all taken from the same screen (see methods). **(d)** Reproducibility of dLFC effects in four cell lines with different sets of replicate PCCs and WBCs. Circle size indicates each gene's dLFC reproducibility and corresponds to the per-gene dLFC product between replicate screens. Colored dots are the most reproducible genes for a given cell line and match the visible dots in Figure 1i.

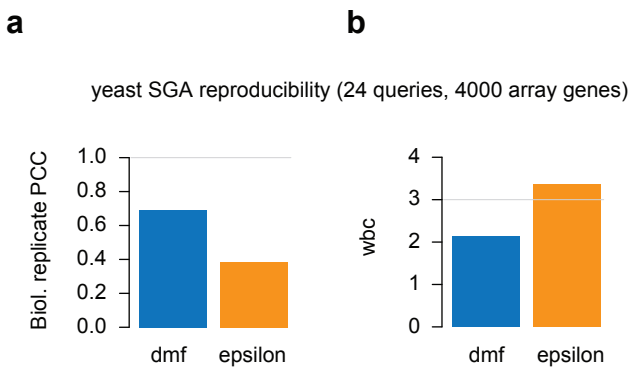


Figure S2: Reproducibility metrics applied to yeast SGA genetic interaction data. (a) Between-biological replicate correlation of double mutant fitness(dmf) and genetic interaction scores (epsilon). The mean Pearson correlation coefficient between 24 query genes screened in biological duplicates against ~4000 array genes is shown. **(b)** WBC scores show the difference between each of the query screens contrasted to the remaining 23 queries by using dmf and epsilon scores.

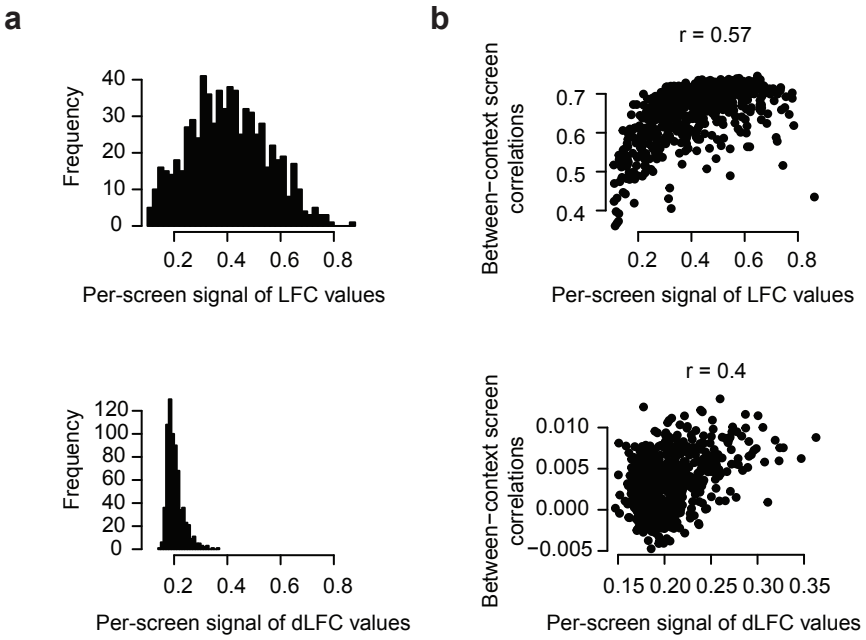


Figure S3: Signal sparsity and background correlation in DepMap CRISPR screens. (a) Per-screen signal across 693 DepMap screens. The signal is represented by the standard deviation across all gene values in a given cell line and is shown at the LFC (top) and dLFC (bottom) data processing level. **(b)** The dependency between per-screen signal (x-axis) and the background (between-context) screen correlation. For each cell line, the mean of the background correlation distribution is plotted. The dependency is measured using the Pearson correlation coefficient.

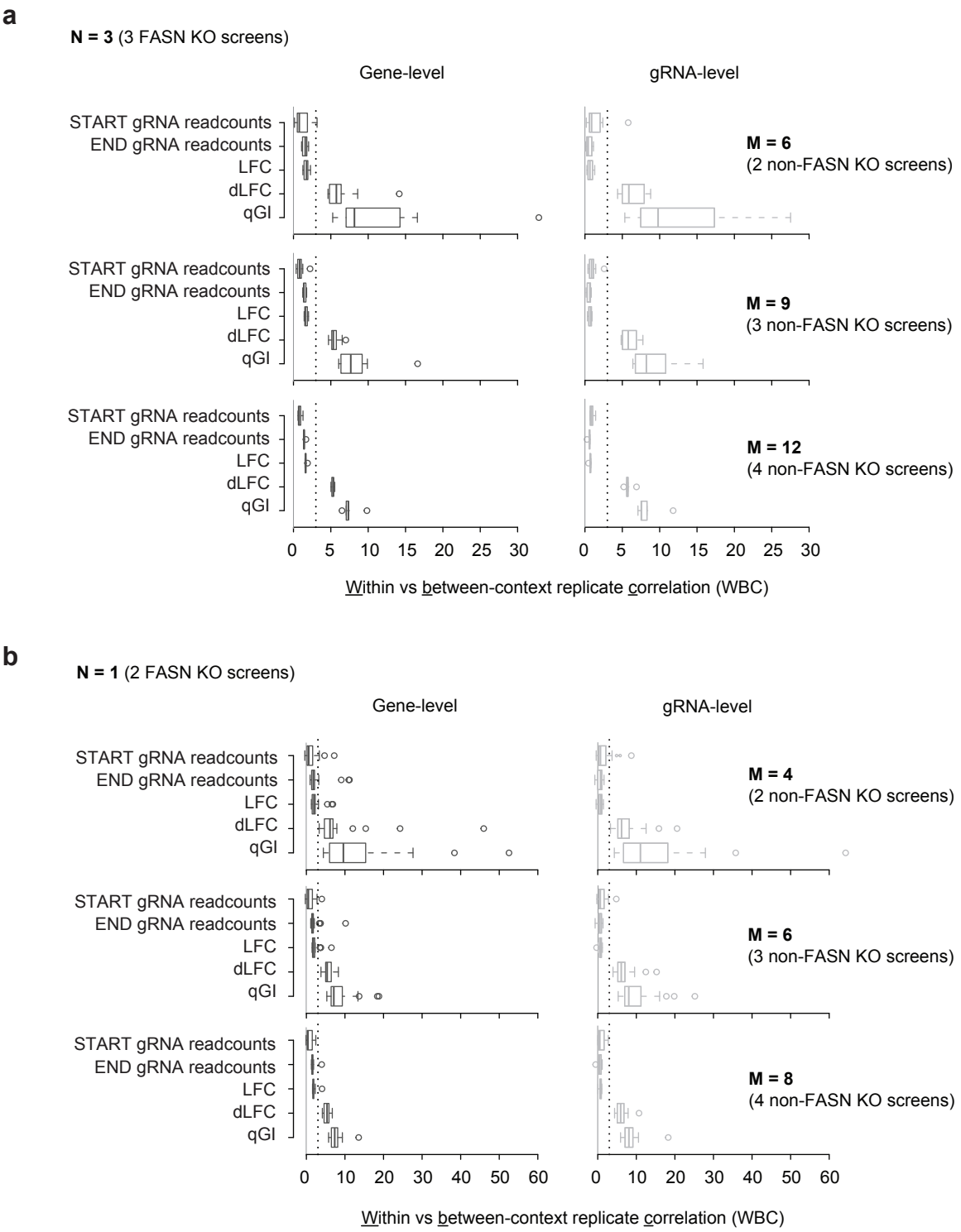


Figure S4: WBC stability at different numbers of FASN and non-FASN KO screens. (a) Within-FASN KO replicate to between FASN KO and non-FASN KO screen ratio of PCCs (WBC; see methods for details). The 3 FASN KO screens were compared to all possible 2, 3 or 4 screen sets of the 5 non-FASN KO (control) screens. Boxplots show the 10 (2 screen possibilities), 10 (3) and 5 (4) WBC gene-level estimates. **(b)** Within-FASN KO replicate to between FASN KO and non-FASN KO screen ratio of PCCs. Done as in (a) but with all possible pairs of 2 FASN KO screens.

Method S1: Implementation of the WBC score

Maximilian Billmann

2023-02-20

Load data

CRISPR screening data from Aregger et al., Nature Metabolismn 2020 loaded. See description of each object below.

```
load(file = "data/crispr_data_input/t0d.rda")
load(file = "data/crispr_data_input/t0_gene.rda")
load(file = "data/crispr_data_input/t18d.rda")
load(file = "data/crispr_data_input/t18_gene.rda")
load(file = "data/crispr_data_input/fcGuides.rda")
load(file = "data/crispr_data_input/dLFC_rawGI_gRNA.rda")
load(file = "data/crispr_data_input/dLFC_rawGI.rda")
load(file = "data/crispr_data_input/gi.rda")
load(file = "data/crispr_data_input/giStats.rda")
```

Different data objects are loaded for the 8 screens (FASN x3, SREBF1, SREBF2, LDLR, C12orf49/LUR1, ACACA) reported in Aregger et al., Nature Metabolism 2020. *t0d* and *t18d* contain normalized gRNA readcounts at the start and end of the experiment, respectively. *t0_gene* and *t18_gene* contain those values summarized per gene for each gene with at least 2 gRNAs. *fcGuides* contains gRNA-level log2-foldchange (lfc) data. *dLFC_rawGI_gRNA* contains gRNA-level differential lfc (dLfc) data (where lfc values measured in a wildtype HAP1 screen are subtracted from a query HAP1 screen). *dLFC_rawGI* contains gene-level dLfc data. *gi* contains gRNA-level genetic interaction (GI) scores. *giStats* contains gene-level lfc and quantitative GI (qGI) scores.

Within-vs-Between context replicate Correlation (WBC) score

Implement Within-vs-Between context replicate Correlation (WBC) score function. The input data object contains screens as columns and measurements (gRNAs, genes) as rows. This data object can be readcounts, lfc values or GI scores.

```
wbc_func <- function(x, # data object, contains screens as columns, measurements as rows
                     qoi, # IDs of replicated screens to test
                     q_anno, # IDs of all screens or just non-qoi screens
                     metric = "WBC") { # output WBC or PCC

  if(missing(q_anno)) {
    q_anno <- colnames(x)
  } else {
    q_anno <- unique(c(qoi, q_anno)) #relevant if qoi not in q_anno
```

```
  }

  x <- cor(x[,q_anno], use = "pairwise.complete.obs") #Pearson cor. coeff. (default)

  qoi <- qoi[qoi %in% q_anno]
  cor_in <- x[qoi,qoi]
  cor_in <- cor_in[lower.tri(cor_in, diag = F)] #within replicate correlation

  cor_out <- as.vector(x[qoi, q_anno[!q_anno %in% qoi]]) #between replicate correlation

  if(metric == "WBC") {  # for each screen of interest gets separate WBC
    x <- (cor_in - mean(cor_out, na.rm = T)) / sd(cor_out, na.rm = T)
    x <- c(x, mean(x, na.rm=T)) # for each screen plus their mean
  }
  if(metric == "PCC") {
    x <- c(cor_in, mean(cor_in, na.rm = T))
  }

  x
}
```

Prepare array for FASN reproducibility stats. Here, the 3 possible per-FASN screen-pair reproducibility is reported as well as the mean reproducibility. Both, the WBC and the Pearson correlation coefficient (PCC) are reported.

```
wbc_stats <- array(NA, dim = c(5, 4, 2, 2),
                   dimnames = list(c("t0","t18","lfc","dlfc","qGI"),
                                   c("12","13","23","mean"),
                                   c("gRNA","gene"), c("pcc","wbc")))
```

Define control screens or use all 5 non-FASN screens (default). Note: since those query genes have biological functions related to FASN, they are likely more similar to FASN than expected by chance. Therefore, the true reproducibility of the FASN screens, expressed as WBC score, are likely higher.

FASN screening IDs are the first 3 `colnames` in this object.

```
Qoi <- colnames(giStats)[1:3]
```

Run the WBC implementation for all data processing levels at gRNA and gene-level.

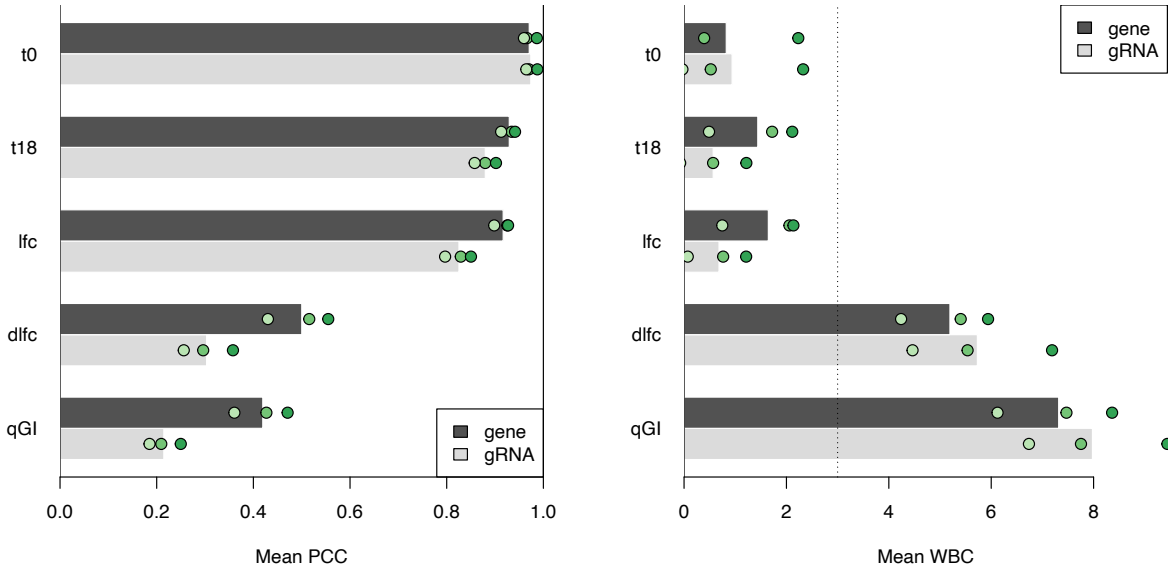
```
for(j in 1:dim(wbc_stats)[4]) {
  m <- c("PCC","WBC")[j]
  wbc_stats["t0",,"gRNA",j] <- wbc_func(x = t0d[,,"norm"], qoi = Qoi, metric = m)
  wbc_stats["t0",,"gene",j] <- wbc_func(x = t0_gene[,,"mean"], qoi = Qoi, metric = m)

  wbc_stats["t18",,"gRNA",j] <- wbc_func(x = t18d, qoi = Qoi, metric = m)
  wbc_stats["t18",,"gene",j] <- wbc_func(x = t18_gene[,,"mean"], qoi = Qoi, metric = m)

  wbc_stats["lfc",,"gRNA",j] <- wbc_func(x = fcGuides, qoi = Qoi, metric = m)
  wbc_stats["lfc",,"gene",j] <- wbc_func(x = giStats[,,"mean","lfc"], qoi = Qoi, metric = m)

  wbc_stats["dlfc",,"gRNA",j] <- wbc_func(x = dLFC_rawGI_gRNA, qoi = Qoi, metric = m)
  wbc_stats["dlfc",,"gene",j] <- wbc_func(x = dLFC_rawGI, qoi = Qoi, metric = m)
```

```
  wbc_stats["qGI",,"gRNA",j] <- wbc_func(x = gi, qoi = Qoi, metric = m)
  wbc_stats["qGI",,"gene",j] <- wbc_func(x = giStats[,,"mean","qGI"], qoi = Qoi, metric = m)
}
```

Show WBC scores

Plot WBC scores and PCCs to generate Figure 1b and c.

```
scol <- c("#74c476","#bae4b3","#31a354")

par.p <- par(mfrow = c(1,2))

y <- barplot(t(wbc_stats[5:1,"mean",,"pcc"]), beside = T, xlim = c(0,1), las = 1,
             xlab = "Mean PCC", horiz = T, col = c("#dbdbdb","#525252"), border = "white")

for(i in 1:3) {
  points(wbc_stats[5:1,i,"gRNA","pcc"], y[1,], pch = 21, bg = scol[i], cex = 1.3) #gRNA
  points(wbc_stats[5:1,i,"gene","pcc"], y[2,], pch = 21, bg = scol[i], cex = 1.3) #gene
}
abline(v = c(0,1))

legend("bottomright", legend = c("gene","gRNA"), fill = c("#525252","#dbdbdb"))


y <- barplot(t(wbc_stats[5:1,"mean",,"wbc"]), beside = T, las = 1,
             xlim = c(0, max(wbc_stats[,,,"wbc"])), xlab = "Mean WBC",
             horiz = T, col = c("#dbdbdb","#525252"), border = "white")
abline(v = c(0,3), col = "black", lty = c(1,3), lwd = 1)

for(i in 1:3) {
  points(wbc_stats[5:1,i,"gRNA","wbc"], y[1,], pch = 21, bg = scol[i], cex = 1.3) #gRNA
  points(wbc_stats[5:1,i,"gene","wbc"], y[2,], pch = 21, bg = scol[i], cex = 1.3) #gene
}

legend("topright", legend = c("gene","gRNA"), fill = c("#525252","#dbdbdb"))
```




```
par(par.p)
```

```
## R version 4.2.1 (2022-06-23)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Monterey 12.6.3
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.2.1  magrittr_2.0.3  fastmap_1.1.0   cli_3.3.0
##  [5] tools_4.2.1     htmltools_0.5.3 rstudioapi_0.14 yaml_2.3.5
##  [9] stringi_1.7.8   rmarkdown_2.16  highr_0.9       knitr_1.40
## [13] stringr_1.4.1   xfun_0.32       digest_0.6.29   rlang_1.0.5
## [17] evaluate_0.16
```