sgRNA-Donor Design_Script

Luca Torello Pianale

February 2023

SCOPE OF THE MARKDOWN

This script is used to select the best sgRNA coming from three websites:
(1) CHOPCOP (https://chopchop.cbu.uib.no/).
(2) CRISPR-ERA (http://crispr-era.stanford.edu/index.jsp).
(3) YeastCrispri (https://lp2.github.io/yeast-crispri/) -OPTIONAL-.

IMPORTANT: Place this R markdown file in the same folder as a "sgRNAs_input.xlsx" file containing all the possible sgRNAs.

LIBRARIES and FUNCTIONS

Automatic installation of packages (if required) and loading.

```
#Install and load packages
requiredPackages <- c("tidyverse", "ggpubr", "readxl", "writexl")

ipak <- function(pkg){
        new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
        if (length(new.pkg))
                install.packages(new.pkg, dependencies = TRUE)
        sapply(pkg, require, character.only = TRUE)
}

ipak(requiredPackages)
```

```
## tidyverse    ggpubr    readxl   writexl
##      TRUE      TRUE      TRUE      TRUE
```

```
#Function to reverse-complement a DNA sequence
rev_compl <- function(seq) {
  tmp <- strsplit(seq, split = "")[[1]]
  rev <- rev(tmp)
  compl <- sapply(rev, function(base) {
    switch(base, "A" = "T", "C" = "G", "G" = "C", "T" = "A",
                 "a" = "t", "c" = "g", "g" = "c", "t" = "a",
                 "N" = "N", "n" = "n")})
  return(paste(compl, collapse = ""))
}
```

IMPORT DATA.

Data are imported. The .xlsx file should contain (1) One sheet for each gene with the sgRNAs coming from CHOPCHOP.
(2) One sheet for each gene with the sgRNAs coming from CRISPR-ERA.
(3) One sheet with all sgRNAs for all the genes coming from YeastCRISPRi (optional).
(4) One sheet with all the genomic sequences (CDS plus/minus 1000 bp), to check if the sgRNA is present in your strain.

```
#Set new directory
folder<- getwd()
setwd(folder)
file <- paste0("/", list.files(folder)[endsWith(list.files(folder), "input.xlsx")])

#Select all sgRNA in all the dataframes
sheet_list <- excel_sheets(paste0(folder, file))
sheet_list <- sheet_list[!sheet_list %in% c('INFO', 'YeastCRISPRi', 'CDS_plusminus1000')]
gene_names <- unique(gsub("(.*)_.*", "\\1", sheet_list))
gene_number <- length(gene_names)

message("Looking for sgRNAs for ", gene_number, " genes.")
```

```
## Looking for sgRNAs for 16 genes.
```

```
gene_names
```

```
##  [1] "RPS14A" "RPS14B" "FRT1"   "AIF1"   "QDR1"   "MET28"  "MRP13"  "HCM1"
##  [9] "GBP2"   "OCA4"   "MSH3"   "BCH1"   "WWM1"   "HLJ1"   "TIR3"   "SMA2"
```

```
#Upload Genomic sequences
genomic_data <- read_excel(paste0(folder, file), sheet= "CDS_plusminus1000", col_names =
TRUE) %>%
  group_by(Gene) %>%
  mutate(GenomicDNA_RV = rev_compl(GenomicDNA),
         FW_RV = paste(GenomicDNA, GenomicDNA_RV, sep = "n")) %>%
  select(Gene, FW_RV)
```

CHOPCHOP

Description for each column:
(1) Target sequence –> Sequence of 23 bp (20bp-NGG, so NGG IN THE SEQUENCE!).
(2) Self-complementarity –> The number indicates the number of regions of self-complementary predicted within the gRNA, and between the gRNA and either a standard backbone (AGGCTAGTCCGT), an extended backbone (AGGCTAGTCCGT, ATGCTGGAA) or a custom backbone.
(3) MM0, MM1, MM2, MM3 –> Number of off-targets (with 0, 1, 2 or 3 mismatches, respectively) outside of your target gene.
(4) Efficiency –> Predicted eddiciency based on some publications (check website).

```
#CHOPCHOP sgRNAs
chop <- data.frame()

for (i in sheet_list[endsWith(sheet_list, "_chop")]) {
  df1 <- read_excel(paste0(folder, file), sheet= i, col_names = TRUE) %>%
    as.data.frame() %>%
    rename(Sequence = 'Target sequence',
           GC_content = 'GC content (%)',
           Self_compl = 'Self-complementarity') %>%
    mutate(Offtargets = MM0 + MM1 + MM2 + MM3,
           Sequence = gsub(".{0,3}$", "", Sequence),
           Sequence = gsub("(A{4,}|C{4,}|G{4,}|T{4,})", "", Sequence)) %>%  #Poly-Ns
    cbind(Gene = gsub("(.+)_chop", "\\1", i)) %>%
    filter(nchar(Sequence) == 20, #Remove sgRNAs with poly-Ns (shorter than 20)
           Efficiency > 50,
           GC_content > 25,
           Self_compl == 0,
           Offtargets == 0) %>%
    mutate(Poly_N = "No") %>%
    select(-c(Strand, Rank, starts_with("MM")))

  chop <- rbind(chop, df1)
  rm(df1)
  }

head(chop)
```

```
##               Sequence Genomic location GC_content Self_compl Efficiency
## 1 GGTGGAGTCAGATGGAACTG    chrIII:177536         55          0      74.89
## 2 TGGTGGAGTCAGATGGAACT    chrIII:177535         50          0      51.97
## 3 TCTGGCCAAAGCTCTCAAAG       chrX:74496         50          0      79.37
## 4 TACAAAGGTATCGTTAAAGG       chrX:74259         35          0      65.62
## 5 TGGAGATGATTCGTCTCTGT       chrX:74346         45          0      58.82
## 6 TAAATCGGTAACATGTACAA       chrX:74274         30          0      56.62
##   Offtargets   Gene Poly_N
## 1          0 RPS14A     No
## 2          0 RPS14A     No
## 3          0 RPS14B     No
## 4          0 RPS14B     No
## 5          0 RPS14B     No
## 6          0 RPS14B     No
```

CRISPR-ERA

Description for each column:
(1) sgRNA sequence –> Sequence of 20 bp (NO NGG IN THE SEQUENCE!).
(2) E score –> "Efficiency Score" (Max 20, lowering if GC content and PolyT are present).
(3) S score –> "Specificity Score" (max 0, negative the more offtargets there are).
(4) E+S score –> Sum of E score and S score. Maximal Value is 20.
(5) #[GC,Polyt] –> [Number between 0-1 referring to GC content, Yes/No for polyTs presence].
(6) #Offtarget detail –> Example: "PAM=NGG:[0mismatch,1mismatch,2mismatches]=[0,0,0] [.,.,.] PAM=NAG:[0mismatch,1mismatch,2mismatches]=[0,1,0] [.,chr11:123572734,.]" means that this sgRNA has only one offtarget with 1 mismatch and an NAG PAM sequence with location is chr11:123572734.
Note that for the S score, the values were changed to 0 if there were offtargets with NAG as PAM sequence, but no offtargets with NGG as PAM sequence. This because the Cas9 in YN2_1 targets only NGG as PAM sequence.

```
#CRISPR-ERA sgRNAs
ERA <- data.frame()

for (i in sheet_list[endsWith(sheet_list, "_era")]) {
  df1 <- read_excel(paste0(folder, file), sheet= i, col_names = TRUE) %>%
    as.data.frame() %>%
    rename(Sequence = 'sgRNA sequence', Target_exon = 'Target Exon',
           Gene = 'Target Gene', Transcript_ID = 'Transcript ID',
           Poly_T = '#[GC,Polyt]', Offtarget_details = '#Offtarget detail',
           E_score = 'E score', S_score = 'S score',
           ES_score = 'E+S score',) %>%
    mutate(GC_content = gsub(".*=[[](.+),", "\\1", Poly_T),
           Poly_T = gsub(".*,(.+)]", "\\1", Poly_T),
           Sequence = gsub("(A{4,}|C{4,}|G{4,}|T{4,})", "", Sequence), #Poly-Ns
           S_score = case_when(
            startsWith(Offtarget_details,
            "(PAM=NGG)[0mismatch,1mismatch,2mismatches,3mismatches]=[0,0,0,0]")~0)) %>%
    filter(nchar(Sequence) == 20, #Remove sgRNAs with poly-Ns (shorter than 20)
           GC_content > 0.25,
           Poly_T == "No",
           E_score >= 15,
           S_score == 0) %>%
    select(-c(Poly_T, GC_content, Target_exon, ID, Strand,
              Offtarget_details, ends_with("_score")))

  ERA <- rbind(ERA, df1)
  rm(df1)
  }

head(ERA)
```

```
##                Sequence   Gene Transcript_ID Chromosome Location
## 1 ATCGCTATGAATTACTTTGG RPS14A       YCR031C          3   178061
## 2 CTAATCGCTATGAATTACTT RPS14A       YCR031C          3   178058
## 3 GTGATTAAACATACTAGACA RPS14A       YCR031C          3   178199
## 4 TATACTGCCATTGACACAAT RPS14A       YCR031C          3   178030
## 5 AAACATTCAACAACAATCAA RPS14A       YCR031C          3   178145
## 6 ACCACCAGTAACTCTGGCGA RPS14A       YCR031C          3   177778
```

YeastCRISPRi

The sgRNAs coming from this website are not mandatory to have, as nucleosome position and chromatin accessibility might change based on the condition the cell is in. It is possible to skip this chunk without affecting the next chunks.
Description for each column:
(1) Seq –> Sequence of 20 bp (NO NGG IN THE SEQUENCE!).
(2) Nucleosome –> Posterior probability of nucleosome occupancy as defined in Schep et al. (2015) Genome Research.
(3) Chromatin –> This features encodes how open the chromatin is relative to the rest of the region. Data from Schep et al. ATAC-seq experiment.
(4) score –> Values can be: 0 (NEITHER good chromatin accessibility NOR low nucleosome presence), 1 (EITHER good chromatin accessibility OR low nucleosome presence), 2 (BOTH good chromatin accessibility AND low nucleosome presence).

```
#YeastCRISPRi sgRNAs filtered based on parameters
Best_yCRISPRi <- read_excel(paste0(folder, file),
                            sheet= "YeastCRISPRi", col_names = TRUE) %>%
  as.data.frame() %>%
  rename(Sequence = Seq,
         Transcript_ID = ORF,
         TSS_dist = Midpoint_TSS_dist) %>%
  filter(Nucleosome < 0.2,  #The closer to 0, the better
         Chromatin > 0.7,   #The closer to 1, the better
         TSS_dist > 0) %>%  #Only sgRNAs after the TSS
  select(-c(Chrm, PAM_mid, Length, Gene_name))

#All YeastCRISPRi sgRNAs
Unfiltered_yCRISPRi <- read_excel(paste0(folder, file),
                                  sheet= "YeastCRISPRi", col_names = TRUE) %>%
  as.data.frame() %>%
  rename(Sequence = Seq,
         Transcript_ID = ORF,
         TSS_dist = Midpoint_TSS_dist) %>%
  select(-c(Chrm, PAM_mid, Length, Gene_name))

head(Unfiltered_yCRISPRi)
```

```
##               Sequence Transcript_ID TSS_dist Nucleosome Chromatin score
## 1 AAGCATCGAGAAACGGCGCT       YOR324C      -30       0.06      0.89     2
## 2 ACCAGATAAGCATCGAGAAA       YOR324C      -37       0.04         1     2
## 3 GCCGTTTCTCGATGCTTATC       YOR324C      -32       0.05      0.89     2
## 4 TCGAAAATCTGTTGGGGTTT       YOR324C      -79          0       0.7     2
## 5 TTAGGGTCGAAAATCTGTTG       YOR324C      -85          0      0.73     2
## 6 GTTAGGGTCGAAAATCTGTT       YOR324C      -86          0      0.73     2
```

SELECT sgRNAs

Here, the best sgRNAs are selected.
If YeastCRISPRi was not included, the common sgRNAs between CHOPCHOP and CRISPR-ERA are selected for each gene.
If YeastCRISPRi was included, common sgRNAs between the three websites are picked. If no common sgRNA are found for some genes, the common sgRNAs between CHOPCHOP and CRISPR-ERA with highest efficiencies are picked.

```
#Remove all the sgRNA whose sequence is not present in the genomic DNA (useful if using a reference strain to design the sgRNAs)
ERA <- merge(ERA, genomic_data, by = "Gene", all = TRUE) %>%
  mutate(sgRNA_presence = ifelse(str_detect(FW_RV, fixed(Sequence)), "Yes", "No")) %>%
  filter(sgRNA_presence == "Yes") %>%
  select(-FW_RV, -sgRNA_presence)

chop <- merge(chop, genomic_data, by = "Gene", all = TRUE) %>%
  mutate(sgRNA_presence = ifelse(str_detect(FW_RV, fixed(Sequence)), "Yes", "No")) %>%
  filter(sgRNA_presence == "Yes") %>%
  select(-FW_RV, -sgRNA_presence)

#Select common sgRNAs
Good_sgRNAs <- merge(chop, ERA, by = c("Sequence", "Gene"), all = FALSE) %>%
  as.data.frame()

Good_gene_number <- length(unique(Good_sgRNAs$Gene))
Good_gene_names <- unique(Good_sgRNAs$Gene)

message("GOOD sgRNAs for ", Good_gene_number, " genes out of ", gene_number)
```

```
## GOOD sgRNAs for 15 genes out of 16
```

```
if(ls()[ls() == "Best_yCRISPRi"] == "Best_yCRISPRi") {
  Best_sgRNAs <- merge(Good_sgRNAs, Best_yCRISPRi,
                       by = c("Sequence", "Transcript_ID"), all = FALSE) %>%
    as.data.frame()

  Best_gene_number <- length(unique(Best_sgRNAs$Gene))
  Best_gene_names <- unique(Best_sgRNAs$Gene)

  message("BEST sgRNAs for ", Best_gene_number, " genes out of ", gene_number)
  message("BEST sgRNAs for ", paste(Best_gene_names, collapse =", "))
  }
```

```
## BEST sgRNAs for 7 genes out of 16

## BEST sgRNAs for MSH3, AIF1, MET28, MRP13, QDR1, HCM1, WWM1
```

```
if (Good_gene_number/gene_number < 1) {
  message("NO common sgRNAs for ", paste(setdiff(gene_names, Good_gene_names), collapse
  =", "))
  message("The sgRNA will be taken from CRISPR-ERA or CHOPCHOP ONLY.")
```

```
  message("Use less stringent conditions if you prefer to have common ones!")
  }
```

```
## NO common sgRNAs for RPS14A

## The sgRNA will be taken from CRISPR-ERA or CHOPCHOP ONLY.

## Use less stringent conditions if you prefer to have common ones!
```

FINAL sgRNAs

If multiple sgRNAs fulfilling the requirements are present for a gene, the one with the highest efficiency will be picked.

```
#Select Final sgRNAs
if(ls()[ls() == "Best_yCRISPRi"] == "Best_yCRISPRi") {
  final_best <- Best_sgRNAs %>%
    group_by(Gene) %>%
    filter(Efficiency == max(Efficiency))

  final_good <- subset(Good_sgRNAs, Gene %in% setdiff(gene_names, Best_gene_names)) %>%
    group_by(Gene) %>%
    filter(Efficiency == max(Efficiency)) %>%
    merge(., Unfiltered_yCRISPRi, all.x = TRUE, by = c("Sequence", "Transcript_ID"))

  Final_sgRNAs <- final_best %>%
    as.data.frame() %>%
    rbind(final_good)
  } else {
  Final_sgRNAs <- Good_sgRNAs %>%
    as.data.frame() %>%
    group_by(Gene) %>%
    filter(Efficiency == max(Efficiency))
  }

#Select only one sgRNA from CRISPR-ERA for genes with no common sgRNA (possible to swap
to CHOPCHOP).
if(length(setdiff(gene_names, Good_gene_names)) > 0) {
  df1 <- subset(ERA, Gene %in% setdiff(gene_names, Good_gene_names)) %>%
    group_by(Gene) %>%
    slice_sample(n = 1)

  Final_sgRNAs <- bind_rows(Final_sgRNAs, df1)

  rm(df1)
  }

#Export file excel
write_xlsx(Final_sgRNAs, path = paste0(folder, "/Final_sgRNAs.xlsx"), use_zip64 = TRUE)
```

OLIGOS

Design Oligos to be ordered.
The sgRNA target sequence should be preceded by "gact" (Forward oligo), while its reverse-complement should be preceded by "aaac" (Reverse oligo). This would allow, after the annealing procedure, to create a double-stranded DNA oligo with sticky-ends compatible with the plasmid YN2_1. WIth a Restriction-ligation reaction, the sgRNA sequence is added to the scaffold sequence of the sgRNA (raplacing the GFP-dropout cassette). The resulting plasmid will contain both the sgRNA- and Cas9 casettes.

```
Final_Oligos <- Final_sgRNAs %>%
  select(Gene, Sequence) %>%
  group_by(Gene) %>%
  mutate(ReverseComplement = rev_compl(Sequence),
         FW = paste0("gact", Sequence),
         RV = paste0("aaac", ReverseComplement)) %>%
  pivot_longer(c("FW", "RV"), names_to = "Direction", values_to = "Oligo_seq") %>%
  mutate(Name = paste(Gene, "sgRNA", Direction, sep = "_")) %>%
  ungroup() %>%
  select(Name, Oligo_seq)

write.table(Final_Oligos, paste0(folder, "/sgRNA_Oligos.txt"), sep = " ", quote = FALSE,
col.names = FALSE, row.names = FALSE)

head(Final_Oligos)
```

```
## # A tibble: 6 x 2
##   Name           Oligo_seq
##   <chr>          <chr>
## 1 MRP13_sgRNA_FW gactATAACAGTGGTGATTAATGA
## 2 MRP13_sgRNA_RV aaacTCATTAATCACCACTGTTAT
## 3 QDR1_sgRNA_FW  gactATAGTAGATAGACCCTGCCA
## 4 QDR1_sgRNA_RV  aaacTGGCAGGGTCTATCTACTAT
## 5 MET28_sgRNA_FW gactGCGGAGAAGAAAGAACACAG
## 6 MET28_sgRNA_RV aaacCTGTGTTCTTTCTTCTCCGC
```

DONOR DNA for deletion

Design donor DNA sequences to be ordered.
The donor DNA sequence is designed to replace the whole CDS of the target gene with "TAACTAGCTGA", sequence containing 3 stop codons in 3 different frame-shifts. It is possible to replace the donor sequence upon need. The input data should be a data frame with 3 columns: (1) "Gene", the name of the target gene. (2) "Sequence", the CDS of the gene with the 1000 bp preceeding and following the CDS (easily found in https://www.yeastgenome.org/).

From "Sequence", the homology arms will be generated automatically and stored in 2 new columns:
(1) "upstream", 50 bp in the region preceding the ATG of the target gene used as 5' homology arm.
(2) "downstream", 50 bp following the stop codon of the target gene used as 3' homology arm. As these region in rich in As and Ts, also the end of the gene can be used as homology to get a higher GC content. Note that the homology arm lengths can be changed upon need (longer or shorter). Moreover, a message will appear if the GC content of any homology arms is lower than 0.3 (30%), since the ends of the promoters and beginning of terminators are generally rich in As and Ts. In this case, change the position from where picking the homology arms.

```
donor_sequence <- toupper("TAACTAGCTGA")
arms_length <- 49

donor_data <- read_excel(paste0(folder, file), sheet= "CDS_plusminus1000", col_names =
TRUE) %>%
  group_by(Gene) %>%
  mutate(upstream = substr(GenomicDNA, 930-arms_length, 930),
         downstream = substr(GenomicDNA, nchar(GenomicDNA)-1045,
         nchar(GenomicDNA)-1045+arms_length),
         upstream = tolower(upstream),
         downstream = tolower(downstream),
         GC_content_up = sum(str_count(upstream, c("c", "g")))/nchar(upstream),
         GC_content_down = sum(str_count(downstream, c("c", "g")))/nchar(downstream),
         across(where(is.numeric), ~ round(.,3)))

if(length(donor_data$GC_content_up[donor_data$GC_content_up < 0.3]) > 0){
  message("Low GC content in upstream sequences!")
  message("Genes with low GC content in upstream sequences: ",
  paste(donor_data$Gene[donor_data$GC_content_up < 0.3], collapse = ", "))
} else {
  message("Good GC content in upstream sequences!")
  }
```

```
## Low GC content in upstream sequences!

## Genes with low GC content in upstream sequences: GBP2, SMA2
```

```
if(length(donor_data$GC_content_down[donor_data$GC_content_down < 0.3]) > 0){
  message("Low GC content in downstream sequences!")
  message("Genes with low GC content in downstream sequences: ",
  paste(donor_data$Gene[donor_data$GC_content_down < 0.3], collapse = ", "))
} else {
  message("Good GC content in downstream sequences!")
  }
```

```
## Low GC content in downstream sequences!

## Genes with low GC content in downstream sequences: HLJ1, BCH1, FRT1
```

```
final_donor <- donor_data %>%
  select(-c(starts_with("GC_content"), GenomicDNA)) %>%
  mutate(FW = paste0(upstream, donor_sequence, downstream),
         RV = rev_compl(FW)) %>%
  pivot_longer(c("FW", "RV"), names_to = "Direction", values_to = "Full_donor") %>%
  mutate(Name = paste(Gene, "Donor", Direction, sep = "_")) %>%
  ungroup() %>%
  select(Name, Full_donor)

write.table(final_donor, paste0(folder, "/Donor_Oligos.txt"), sep = " ", quote = FALSE,
col.names = FALSE, row.names = FALSE)

head(final_donor)
```

```
## # A tibble: 6 x 2
##   Name            Full_donor
##   <chr>           <chr>
## 1 GBP2_Donor_FW   aggtgaaaaaaattgaaaaaaaatgaaagcttggatgaaatcaattatatTAACTAGCTGA~
## 2 GBP2_Donor_RV   ggaattaatcacgtctagcataagagatctgtaaactacaaccaccataaTCAGCTAGTTA~
## 3 RPS14A_Donor_FW gcttcctcaggtagacagttgaaatgaattggttattaatcactatatatTAACTAGCTGA~
## 4 RPS14A_Donor_RV taactcataatcttctacctcttctaccacccttctttctggtggagtctTCAGCTAGTTA~
## 5 HCM1_Donor_FW   attctattctttttccccccttttattatagttacatctactatttgagctTAACTAGCTGA~
## 6 HCM1_Donor_RV   aaagtcaattcttttcattaccgctatcgttggaagggtgattatgataaTCAGCTAGTTA~
```

CITATIONS

Citations of R Studio and packages used.

```
print(citation(), style = "text")
```

```
## R Core Team (2021). _R: A Language and Environment for Statistical
## Computing_. R Foundation for Statistical Computing, Vienna, Austria.
## <URL: https://www.R-project.org/>.
```

```
for(i in c("rmarkdown", requiredPackages)) {
  print(i); print(citation(i), style = "text"); cat('\n')
}
```

```
## [1] "rmarkdown"
## Allaire J, Xie Y, McPherson J, Luraschi J, Ushey K, Atkins A, Wickham
## H, Cheng J, Chang W, Iannone R (2022). _rmarkdown: Dynamic Documents
## for R_. R package version 2.17, <URL:
## https://github.com/rstudio/rmarkdown>.
##
## Xie Y, Allaire J, Grolemund G (2018). _R Markdown: The Definitive
## Guide_. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 9781138359338,
## <URL: https://bookdown.org/yihui/rmarkdown>.
##
## Xie Y, Dervieux C, Riederer E (2020). _R Markdown Cookbook_. Chapman
## and Hall/CRC, Boca Raton, Florida. ISBN 9780367563837, <URL:
## https://bookdown.org/yihui/rmarkdown-cookbook>.
##
## [1] "tidyverse"
## Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
## Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E,
## Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi
## K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the
## tidyverse." _Journal of Open Source Software_, *4*(43), 1686. doi:
## 10.21105/joss.01686 (URL: https://doi.org/10.21105/joss.01686).
##
## [1] "ggpubr"
## Kassambara A (2020). _ggpubr: 'ggplot2' Based Publication Ready Plots_.
## R package version 0.4.0, <URL:
## https://CRAN.R-project.org/package=ggpubr>.
##
## [1] "readxl"
## Wickham H, Bryan J (2022). _readxl: Read Excel Files_. R package
## version 1.4.1, <URL: https://CRAN.R-project.org/package=readxl>.
##
## [1] "writexl"
## Ooms J (2022). _writexl: Export Data Frames to Excel 'xlsx' Format_. R
## package version 1.4.1, <URL:
## https://CRAN.R-project.org/package=writexl>.
```