sds

Mladen Cucak

April 19, 2019

# Libraries

Packages needed for the analysis are loaded. If the libraries do not exist locally, they will be downloaded.

list.of.packages <-  
 c(  
 "tidyverse",  
 "readxl",  
 "data.table",  
 "lubridate",  
 "knitr",  
 "padr",  
 "devtools",  
 "pracma",  
 "remotes",  
 "parallel",  
 "pbapply",  
 "ggrepel",  
 "ggthemes",  
 "egg",  
 "rsm",  
 "GGally",  
 "R.utils",  
 "rcompanion",  
 "mgsub",  
 "here",  
 "stringr",  
 "pander"  
 )  
  
new.packages <-  
 list.of.packages[!(list.of.packages %in% installed.packages()[, "Package"])]  
  
#Download packages that are not already present  
if (length(new.packages))  
 install.packages(new.packages)  
  
if ("gt" %in% installed.packages() == FALSE)  
 remotes::install\_github("rstudio/gt")  
  
list.of.packages <- c(list.of.packages, "gt")  
packages\_load <-  
 lapply(list.of.packages, require, character.only = TRUE)  
  
#Print warning if there is a problem with installing/loading some of packages  
if (any(as.numeric(packages\_load) == 0)) {  
 warning(paste("Package/s", paste(list.of.packages[packages\_load != TRUE]), "not loaded!"))  
} else {  
 print("All packages were successfully loaded.")  
}

## [1] "All packages were successfully loaded."

rm(list.of.packages, new.packages, packages\_load)

# Cut-off dates

Soil temperature data.

soildf<- read\_csv(here::here("tmp", "agridata\_375.csv"),col\_types = cols(X1 = col\_skip(), X10 = col\_skip()),   
 skip = 7)  
soildf$date = as.Date(soildf$date,format="%d-%b-%Y", tz = "UTC")  
soildf <- separate(soildf, "date", c("year\_var", "month", "day"), sep = "-", remove = F )  
soildf$doy <- yday(soildf$date)  
soildf[,2:5] <- lapply(soildf[,2:5], as.numeric)

We assume that the planting starts day after the daily average soil temperature becomes higher than 8°C for three consecutive days in the beginning of April. This is common practice in Ireland, recomended by Teagasc.

soildf <- soildf[with(soildf, month >= 4),]  
dfls <- base::split(soildf, soildf$year\_var)  
dates\_cut <-   
lapply(dfls, function(x) {  
 criteria <- as.numeric(x$soil>8)  
 #cumulative sum days matching the criteria  
 criteria\_sum <- stats::ave(criteria, cumsum(criteria == 0), FUN = cumsum)  
 pl\_date <- x[match(3, criteria\_sum)+1, "date"]  
 data.frame( "year" = unique(year(x$date)),  
 date = pl\_date[1,1])  
   
}  
) %>% bind\_rows() %>%   
rename( "plant\_date" = date)  
rm(soildf, dfls)

The data for 2007 is missing so we shall use the mean date from other years.

dates\_cut$plant\_doy <-yday(dates\_cut$plant\_date)  
dates\_cut <-   
bind\_rows(  
 list(  
 dates\_cut,  
 data.frame(  
 year = 2007,  
 plant\_date = as.Date(strptime(paste("2007", mean(dates\_cut$plant\_doy, na.rm = T )), format="%Y %j")),  
 plant\_doy = mean(dates\_cut$plant\_doy)  
 )  
 )  
)

We will assume that the full emergence takes 3 and additional minimum of 2 weeks for full emergence and for the first plants sto startme meeting in the rows.

meet\_in\_rows <- 5\*7  
dates\_cut$prot\_start <- with(dates\_cut, plant\_doy + meet\_in\_rows)  
dates\_cut$prot\_start <- paste(dates\_cut$year, dates\_cut$prot\_start, sep = "-") %>% strptime(format="%Y-%j") %>% as.Date()

For the purposes of this study we will assume that the growing season lasts 120 days. However, the pesticide protection lasts for at least another three weeks to alow the tuber growth.

duration\_of\_season <- 120 + 21   
dates\_cut$season\_end <- with(dates\_cut, plant\_doy + duration\_of\_season)  
dates\_cut$season\_end <- paste(dates\_cut$year, dates\_cut$season\_end, sep = "-") %>% strptime(format="%Y-%j") %>% as.Date()  
dates\_cut %>% kable()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| year | plant\_date | plant\_doy | prot\_start | season\_end |
| 2008 | 2008-04-04 | 95.00000 | 2008-05-09 | 2008-08-23 |
| 2009 | 2009-04-04 | 94.00000 | 2009-05-09 | 2009-08-23 |
| 2010 | 2010-04-08 | 98.00000 | 2010-05-13 | 2010-08-27 |
| 2011 | 2011-04-04 | 94.00000 | 2011-05-09 | 2011-08-23 |
| 2012 | 2012-04-04 | 95.00000 | 2012-05-09 | 2012-08-23 |
| 2013 | 2013-04-12 | 102.00000 | 2013-05-17 | 2013-08-31 |
| 2014 | 2014-04-04 | 94.00000 | 2014-05-09 | 2014-08-23 |
| 2015 | 2015-04-05 | 95.00000 | 2015-05-10 | 2015-08-24 |
| 2016 | 2016-04-07 | 98.00000 | 2016-05-12 | 2016-08-26 |
| 2007 | 2007-04-06 | 96.11111 | 2007-05-11 | 2007-08-25 |

Load the weather data with model outputs.

load(file = here::here("data", "op\_2007\_16", "OP\_for\_analysis.RData"))  
  
years <- unique(OP$year\_var)  
  
df <- filter(OP, month>=2 & month <=9 );rm(OP)  
df <-  
 df[, c("date",  
 "year\_var",  
 "short\_date",  
 "week\_var",  
 "doy",  
 "rain",  
 "temp",  
 "rhum",  
 "90\_10\_12\_rain\_ebh",  
 "88\_12\_10\_rainrh\_ebh",  
 "88\_10\_10\_rainrh\_ebh")] #Keep only columns needed for faster code execution  
df <- dplyr::rename(df,   
 ir = "90\_10\_12\_rain\_ebh",  
 ir\_opt = "88\_12\_10\_rainrh\_ebh",  
 ir\_low\_risk = "88\_10\_10\_rainrh\_ebh"  
 )

# Treatment frequency: Model vs. Calendar

Calculate the number of calendar treatments, assuming they would start 5 weeks and end 141 days after the planting

(calendar\_five <- round( c(duration\_of\_season-5\*7)/5,0))

## [1] 21

(calendar\_seven <- round( c(duration\_of\_season-5\*7)/7,0))

## [1] 15

We are asuming that a fungicide treatment is activated everytime the warning threshold is reached. Cumulative sum of recomanded treatments is calculated per season.

ControlFreqFun <-  
 function(y,  
 weather\_data,  
 dates\_cut,  
 min\_prot\_dur = NULL,  
 max\_prot\_dur = NULL) {  
 #Set the warning threshold and run the rest of the script  
 warning\_threshold <- y  
 #Weather and model output data  
 fun\_df <- weather\_data  
   
 #A function to subset the data for the period of interest in each year  
 test.overlap = function(vals, start\_date, end\_date) {  
 rowSums(mapply(function(a, b)  
 between(vals, a, b),  
 start\_date, end\_date)) > 0  
 }  
   
 #Subset the of the data for the duration of 'protection' period for each year  
 trt\_df <-  
 fun\_df %>%  
 filter(test.overlap(short\_date,   
 dates\_cut$prot\_start,   
 dates\_cut$season\_end)  
 ) %>%  
 select(  
 ends\_with("year\_var"),  
 ends\_with("week\_var"),  
 ends\_with("doy"),  
 starts\_with("ir")  
 ) %>%  
 group\_by(year\_var) %>%  
 #if there was an accumulation from previous day, it would triger a warning  
 #Check all of the first five rows because of possible break of 5 hours  
 mutate\_at(., .vars = colnames(.[grep("ir", colnames(.))]),  
 funs(  
 ifelse(row\_number() <= 5 & . >= warning\_threshold,  
 warning\_threshold, .)  
 )) %>%  
 #all five values all changed so we have to delete 4 of them and leave only 1  
 mutate\_at(., .vars = colnames(.[grep("ir", colnames(.))]),  
 funs(ifelse(  
 row\_number() <= 4 & . == warning\_threshold, 0, .  
 ))) %>%  
 # Change values coresponding to the warning threshold to 1 for calculating the sum  
 mutate\_at(., .vars = colnames(.[grep("ir", colnames(.))]),  
 funs(ifelse(. == warning\_threshold, 1, 0))) %>%  
 group\_by(year\_var, week\_var, doy) %>%  
 summarise\_at(., .vars = colnames(.[grep("ir", colnames(.))]), .funs = sum)  
   
   
 #####################################################################  
 #Calculate the number of treatments with 5-10 day sliding treatment interval  
 TreatmentWindow <- function(model\_output, min\_prot\_dur = NULL,max\_prot\_dur = NULL) {  
 #Each warning would cause treatment that will keep the plants protected for a period of time  
 min\_prot\_dur <-  
 ifelse(is.null(min\_prot\_dur), 5, min\_prot\_dur)#If not defined default value is 5 days  
 max\_prot\_dur <-  
 ifelse(is.null(max\_prot\_dur), 10, max\_prot\_dur)#If not defined default value is 10 days  
   
  
 z <- model\_output  
 x <- c(z, seq(1,max\_prot\_dur, 1))  
 #attach vector, 10 days of 0  
 y <- vector(mode = "numeric", length = length(x) )  
 # find the first treatment  
 y[which(x == 1)[1]] <- 1  
   
   
 for (i in seq\_along(y)) {  
   
 #Following treatment will be implemented on day 5 if there is 1 in the next five days  
 if (y[i] == 1 & sum(x[c(i+1):c(i+5)])>0 ) {y[i+5] <- 1}  
 #Or on any following day with risk threshold reached  
 if (y[i] == 1 & sum(x[c(i+1):c(i+5)])==0 & x[i+6]==1 ) {y[i+6] <- 1}  
 if (y[i] == 1 & sum(x[c(i+1):c(i+6)])==0 & x[i+7]==1 ) {y[i+7] <- 1}  
 if (y[i] == 1 & sum(x[c(i+1):c(i+7)])==0 & x[i+8]==1 ) {y[i+8] <- 1}  
 if (y[i] == 1 & sum(x[c(i+1):c(i+8)])==0 & x[i+9]==1 ) {y[i+9] <- 1}  
 #If warning threshold was not reached at all or 10th day, then spray on 10th day   
 if (y[i] == 1 & sum(x[c(i+1):c(i+9)])==0 ) {y[i+10] <- 1}  
 #if there was no treatment   
 if(i== length(z)) break  
 }  
 y[1:length(z)]   
 }  
   
 trt\_df\_window <-   
 split(trt\_df, trt\_df$year\_var)  
   
   
 for (year in seq\_along(trt\_df\_window)) {  
 x <- trt\_df\_window[[year]]  
 trt\_df\_window[[year]] <-   
   
 lapply(x[grep("ir", colnames(x))], function(x)  
 TreatmentWindow(x, min\_prot\_dur,max\_prot\_dur)) %>%  
 bind\_cols() %>%   
 rename\_all(., .funs = funs(paste0("w\_", .)))  
 }  
   
 trt\_df\_window <- bind\_rows(trt\_df\_window)  
 #Triger a treatment window  
  
   
   
 trt\_df <- bind\_cols(trt\_df, trt\_df\_window)  
  
   
   
 #Calculate number of treatmetns per year  
 sum\_df <-   
 trt\_df %>%   
 group\_by(year\_var) %>%   
 summarise\_at(colnames(trt\_df[, grep("ir", colnames(trt\_df))]), .funs = sum)   
   
   
   
 sum\_df$warning\_thres <- warning\_threshold  
   
 return(sum\_df)  
 }

Calculate the number of treatments for each warning threshold (EBH).

# this chunk not evaluated  
#select max warning threshold  
warning\_thresholds <- 1:18  
begin <- Sys.time()  
#Detect the number of cores and set it to total minus 1, if there are multiple cores, to avoid overload  
cores <- ifelse(detectCores() > 1, detectCores()-1, 1)   
cl <- makeCluster(cores)  
clusterExport(cl, c("df", "dates\_cut", "ControlFreqFun"))  
  
clusterEvalQ(cl, library("tidyverse"))

## [[1]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr"   
## [6] "tidyr" "tibble" "ggplot2" "tidyverse" "stats"   
## [11] "graphics" "grDevices" "utils" "datasets" "methods"   
## [16] "base"   
##   
## [[2]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr"   
## [6] "tidyr" "tibble" "ggplot2" "tidyverse" "stats"   
## [11] "graphics" "grDevices" "utils" "datasets" "methods"   
## [16] "base"   
##   
## [[3]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr"   
## [6] "tidyr" "tibble" "ggplot2" "tidyverse" "stats"   
## [11] "graphics" "grDevices" "utils" "datasets" "methods"   
## [16] "base"

control\_eval <- pbapply::pblapply(warning\_thresholds, function(x)  
 {  
 xx <-  
 ControlFreqFun(x,  
 df,  
 dates\_cut,  
 min\_prot\_dur = 5,  
 max\_prot\_dur = 10)  
 return(xx)  
 },  
 cl = cl  
 )  
begin- Sys.time() #check the duration of the pcontrol\_evaless in the console

## Time difference of -5.249967 secs

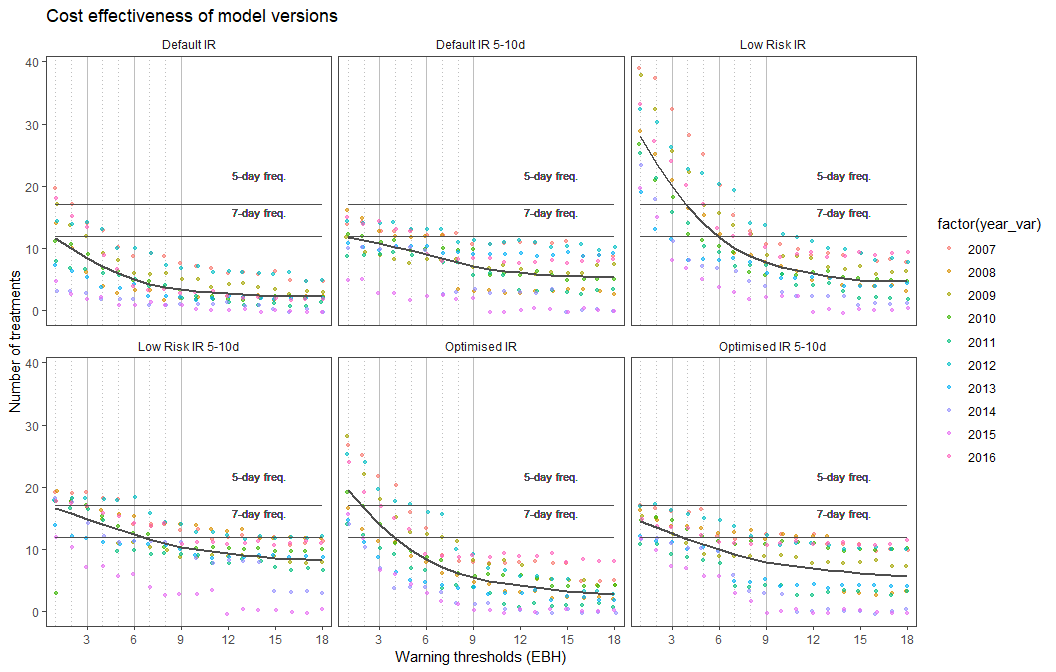
stopCluster(cl)

Melt the data frame.

control\_eval\_long <-   
control\_eval %>%   
 bind\_rows() %>%   
 reshape2::melt(  
 id.vars = c("year\_var", "warning\_thres"),  
 variable.name = "programme",  
 value.name = "no\_of\_treatments"  
 )  
control\_eval\_long %>% head() %>% pander()

|  |  |  |  |
| --- | --- | --- | --- |
| year\_var | warning\_thres | programme | no\_of\_treatments |
| 2007 | 1 | ir | 20 |
| 2008 | 1 | ir | 14 |
| 2009 | 1 | ir | 17 |
| 2010 | 1 | ir | 11 |
| 2011 | 1 | ir | 8 |
| 2012 | 1 | ir | 14 |

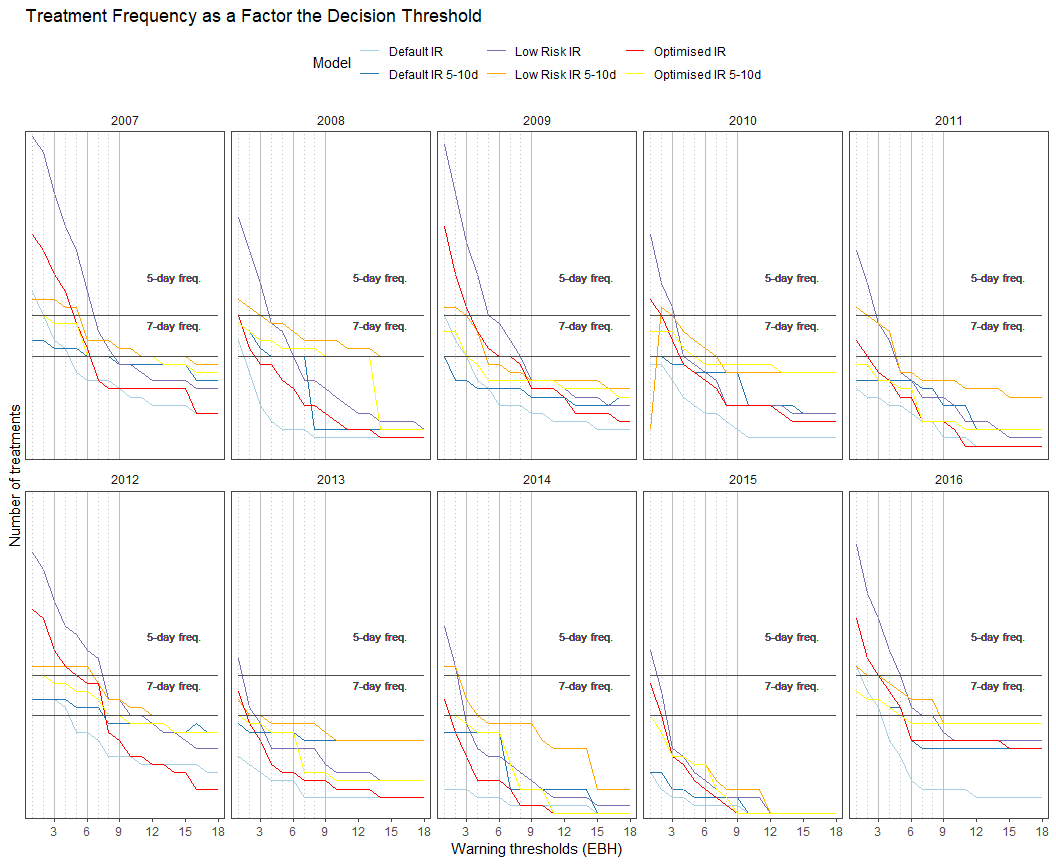
control\_eval\_long %>%   
# control\_eval\_long%>%   
 mutate(programme = ifelse(programme== "ir", "Default IR",   
 ifelse(programme== "ir\_opt", "Optimised IR",  
 ifelse(programme == "ir\_low\_risk", "Low Risk IR",  
 ifelse(programme== "w\_ir", "Default IR 5-10d",  
 ifelse(programme== "w\_ir\_opt", "Optimised IR 5-10d",  
 ifelse(programme == "w\_ir\_low\_risk", "Low Risk IR 5-10d",""))))))) %>%   
mutate(programme = factor(programme  
 # , levels = ("Default IR","Optimised IR", "Low Risk IR","Default IR 5-10d","Optimised IR 5-10d","Low Risk IR 5-10d")  
 )) %>%   
ggplot(., aes(factor(warning\_thres), no\_of\_treatments)) +  
 geom\_vline(xintercept=seq(1,9,1), size = 0.2, color = "gray", linetype="dotted")+  
 geom\_vline(xintercept=seq(3,9,3), size = 0.2, color = "gray", linetype="solid")+  
 # geom\_boxplot(width = 0.4, show.legend=FALSE) +  
 geom\_jitter(aes(color = factor(year\_var)),  
 position = position\_jitter(width = 0.1),  
 # colour = "black",  
 alpha = 0.6,  
 size = 1  
 ) +  
 geom\_line(aes(warning\_thres, 12),colour = gray.colors(2)[1])+  
 geom\_label(aes(14,calendar\_seven+0.8 , label= "7-day freq."),   
 colour = gray.colors(2)[1], fill = NA, label.size = NA, family = "Helvetica", size = 3) +  
 geom\_line(aes(warning\_thres, 17),colour = gray.colors(2)[1])+  
 geom\_label(aes(14,calendar\_five+0.8, label= "5-day freq."),   
 colour = gray.colors(2)[1], fill = NA, label.size = NA, family = "Helvetica", size = 3) +  
 geom\_smooth(method = "loess",  
 se = FALSE,  
 color = gray.colors(2)[1],  
 aes(group = 1)) +  
 facet\_wrap(~programme, ncol = 3) +  
 ggtitle("Cost effectiveness of model versions") +  
   
 xlab("Warning thresholds (EBH)") +  
 ylab("Number of treatments") +  
 scale\_x\_discrete(breaks=seq(0, 18, 3)) +  
 theme\_article()



The figure shows how do the decision thresholds of the three model versions differ from the calendar spray programmes. The dots represent the number of treatments in relation to coresponding warning threshold, where each dot represents one year. If the fitted smothed line is above the estimated calendar frequency line, the model recomands more treatments than the usual standard calendar programme.

There is variation in the data and it would be interesting to see if there is an effect of year.

control\_eval\_long %>%   
# control\_eval\_long%>%   
 mutate(programme = ifelse(programme== "ir", "Default IR",   
 ifelse(programme== "ir\_opt", "Optimised IR",  
 ifelse(programme == "ir\_low\_risk", "Low Risk IR",  
 ifelse(programme== "w\_ir", "Default IR 5-10d",  
 ifelse(programme== "w\_ir\_opt", "Optimised IR 5-10d",  
 ifelse(programme == "w\_ir\_low\_risk", "Low Risk IR 5-10d",""))))))) %>%   
mutate(programme = factor(programme  
 # , levels = ("Default IR","Optimised IR", "Low Risk IR","Default IR 5-10d","Optimised IR 5-10d","Low Risk IR 5-10d")  
 )) %>%   
ggplot(., aes(factor(warning\_thres), no\_of\_treatments, group =programme, colour =programme)) +  
 geom\_vline(xintercept=seq(1,9,1), size = 0.2, color = "gray", linetype="dotted")+  
 geom\_vline(xintercept=seq(3,9,3), size = 0.2, color = "gray", linetype="solid")+  
  
 geom\_line(aes())+  
 geom\_line(aes(warning\_thres, 12),colour = gray.colors(2)[1])+  
 geom\_label(aes(14,calendar\_seven+0.8 , label= "7-day freq."),   
 colour = gray.colors(2)[1], fill = NA, label.size = NA, family = "Helvetica", size = 3) +  
 geom\_line(aes(warning\_thres, 17),colour = gray.colors(2)[1])+  
 geom\_label(aes(14,calendar\_five+0.8, label= "5-day freq."),   
 colour = gray.colors(2)[1], fill = NA, label.size = NA, family = "Helvetica", size = 3) +  
 facet\_wrap(~year\_var, ncol = 5)+  
 ggtitle("Treatment Frequency as a Factor the Decision Threshold") +  
 theme(strip.text.y = element\_blank())+  
 xlab("Warning thresholds (EBH)") +  
 ylab("Number of treatments") +  
 theme\_article() +   
 geom\_vline(xintercept=c(2,3,4), size = 0.5, color = "gray", linetype="dotted")+  
 scale\_x\_discrete(breaks=seq(0, 18, 3))+  
 scale\_y\_discrete(breaks=seq(0, 30, 5))+  
 scale\_colour\_manual( "Model",values = c("#a6cee3", "#1f78b4", "#7570b3","orange", "red", "yellow" ))+  
 theme(legend.position="top",  
 text = element\_text( size = 11))



It is evident that setting a threshold to even a minimum accumulation of risk, in other words a single effective blight hour, is trigerring less treatments in certain years.

# Dose reduction based on warning outputs

Producers often lack the trust environmental risk predicon. Calculate the dose reductions based on the

Packages used

sessionInfo()

## R version 3.5.1 (2018-07-02)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 10 x64 (build 17134)  
##   
## Matrix products: default  
##   
## locale:  
## [1] LC\_COLLATE=English\_United States.1252   
## [2] LC\_CTYPE=English\_United States.1252   
## [3] LC\_MONETARY=English\_United States.1252  
## [4] LC\_NUMERIC=C   
## [5] LC\_TIME=English\_United States.1252   
##   
## attached base packages:  
## [1] parallel stats graphics grDevices utils datasets methods   
## [8] base   
##   
## other attached packages:  
## [1] bindrcpp\_0.2.2 gt\_0.1.0 pander\_0.6.3   
## [4] here\_0.1 mgsub\_1.5.0 rcompanion\_2.0.10  
## [7] R.utils\_2.7.0 R.oo\_1.22.0 R.methodsS3\_1.7.1  
## [10] GGally\_1.4.0 rsm\_2.10 egg\_0.4.2   
## [13] gridExtra\_2.3 ggthemes\_4.0.1 ggrepel\_0.8.0   
## [16] pbapply\_1.3-4 remotes\_2.0.2 pracma\_2.2.2   
## [19] devtools\_1.13.6 padr\_0.4.1 knitr\_1.22   
## [22] lubridate\_1.7.4 data.table\_1.11.8 readxl\_1.2.0   
## [25] forcats\_0.3.0 stringr\_1.4.0 dplyr\_0.7.8   
## [28] purrr\_0.2.5 readr\_1.3.1 tidyr\_0.8.2   
## [31] tibble\_2.0.1 ggplot2\_3.1.0 tidyverse\_1.2.1   
##   
## loaded via a namespace (and not attached):  
## [1] nlme\_3.1-137 RColorBrewer\_1.1-2 httr\_1.4.0   
## [4] rprojroot\_1.3-2 tools\_3.5.1 backports\_1.1.3   
## [7] R6\_2.3.0 nortest\_1.0-4 lazyeval\_0.2.1   
## [10] colorspace\_1.3-2 withr\_2.1.2 tidyselect\_0.2.5   
## [13] emmeans\_1.3.1 compiler\_3.5.1 cli\_1.0.1   
## [16] rvest\_0.3.2 expm\_0.999-3 xml2\_1.2.0   
## [19] sandwich\_2.5-0 labeling\_0.3 sass\_0.1.0.9000   
## [22] checkmate\_1.8.5 scales\_1.0.0 lmtest\_0.9-36   
## [25] mvtnorm\_1.0-8 multcompView\_0.1-7 commonmark\_1.7   
## [28] digest\_0.6.18 foreign\_0.8-70 rmarkdown\_1.12.6   
## [31] pkgconfig\_2.0.2 htmltools\_0.3.6 manipulate\_1.0.1   
## [34] highr\_0.7 rlang\_0.3.1 rstudioapi\_0.8   
## [37] bindr\_0.1.1 generics\_0.0.2 zoo\_1.8-4   
## [40] jsonlite\_1.6 magrittr\_1.5 modeltools\_0.2-22   
## [43] Matrix\_1.2-14 Rcpp\_1.0.0 DescTools\_0.99.26   
## [46] munsell\_0.5.0 stringi\_1.2.4 multcomp\_1.4-8   
## [49] yaml\_2.2.0 MASS\_7.3-50 plyr\_1.8.4   
## [52] grid\_3.5.1 crayon\_1.3.4 lattice\_0.20-35   
## [55] haven\_2.0.0 splines\_3.5.1 hms\_0.4.2   
## [58] pillar\_1.3.1 EMT\_1.1 boot\_1.3-20   
## [61] estimability\_1.3 reshape2\_1.4.3 stats4\_3.5.1   
## [64] codetools\_0.2-15 glue\_1.3.0 evaluate\_0.13   
## [67] modelr\_0.1.2 cellranger\_1.1.0 gtable\_0.2.0   
## [70] reshape\_0.8.8 assertthat\_0.2.0 xfun\_0.6   
## [73] coin\_1.2-2 xtable\_1.8-3 broom\_0.5.1   
## [76] coda\_0.19-2 survival\_2.42-3 memoise\_1.1.0   
## [79] TH.data\_1.0-9