The model

Mladen Cucak

03 May, 2020

source(here::here("scr","lib", "pkg.R"))

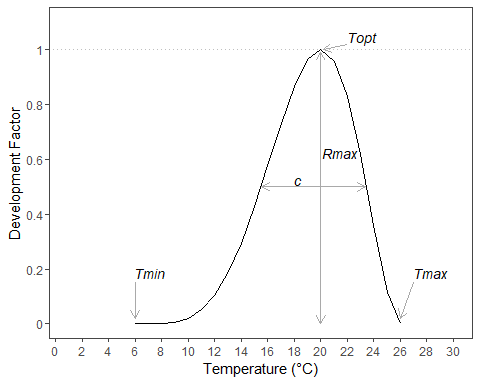
## [1] "All packages were successfully loaded."

library("grid")

## Beta function

BlightR’s temperature functions are base don Beta function.

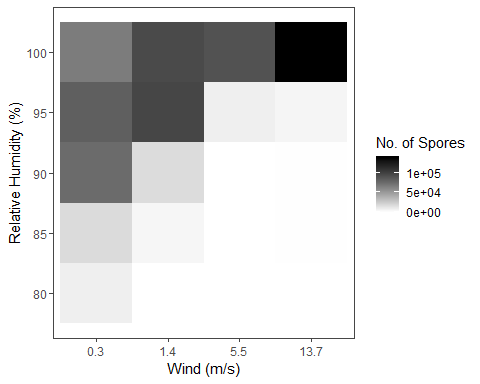
beta\_df <- data.frame(temp = c(4:28))  
  
Tmin <- 6  
Topt <- 20  
Tmax <- 26  
Rfact <- 1 #set to default 1   
Shape <-2 # parameter that determines the shape of curve  
  
#Function to cacluate the sporulation temperature factor   
temp\_dev <- sapply(beta\_df$temp, function(x) {  
 temps <-  
 Rfact \* ((Tmax - x) / (Tmax - Topt) \* ((x - Tmin) / (Topt - Tmin)) ^ ((Topt - Tmin) / (Tmax - Topt))) ^ Shape  
 temps = ifelse(x < Tmin | x > Tmax, 0, temps)  
   
})  
  
  
  
lab\_size <- 10.5  
  
grobc <- grobTree(textGrob("c", x=0.58, y=0.48, hjust=0,  
 gp=gpar(fill = "white", coll = "black",fontsize=lab\_size, fontface="italic")))  
  
grobr <- grobTree(textGrob("Rmax", x=0.645, y=0.56, hjust=0,  
 gp=gpar(fill = "white", coll = "black",fontsize=lab\_size, fontface="italic")))  
  
grobTmin <- grobTree(textGrob("Tmin", x=0.2, y=.2, hjust=0,  
 gp=gpar(fill = "white", coll = "black",fontsize=lab\_size, fontface="italic")))  
  
grobTopt <- grobTree(textGrob("Topt", x=0.705, y=.91, hjust=0,  
 gp=gpar(fill = "white", coll = "black",fontsize=lab\_size, fontface="italic")))  
  
grobTmax <- grobTree(textGrob("Tmax", x=0.86, y=0.2, hjust=0,  
 gp=gpar(fill = "white", coll = "black",fontsize=lab\_size, fontface="italic")))  
  
  
  
pbeta <-   
ggplot()+  
 geom\_line(aes(ifelse(beta\_df$temp<Tmin|beta\_df$temp>Tmax,NA, beta\_df$temp),temp\_dev))+  
 geom\_hline(yintercept = 1, linetype = "dotted", color = "gray", size = .2)+  
 labs(y = "Development Factor",  
 x = "Temperature (˚C)")+  
 #add arrows and label for parameter c  
 geom\_segment(aes(x = 15.55, y = .5, xend = 23.4, yend = .5), linetype = "solid",color = "darkgray",  
 arrow = arrow(length = unit(0.1, "inches"), ends = "both")  
 )+  
 annotation\_custom(grobc)+  
 #add arrows and label for parameter Rmax  
 geom\_segment(aes(x = 20, y = 0, xend = 20, yend = .99), linetype = "solid",color = "darkgray",  
 arrow = arrow(length = unit(0.1, "inches"), ends = "both")  
 )+  
 annotation\_custom(grobr)+  
 #add arrows and label for parameter Tmin  
 geom\_segment(aes(x = Tmin, y = 0.15, xend = Tmin, yend = .02), linetype = "solid",color = "darkgray",  
 arrow = arrow(length = unit(0.1, "inches"))  
 )+  
 annotation\_custom(grobTopt)+  
 #add arrows and label for parameter Topt  
 geom\_segment(aes(x = Topt+2, y = 1.02, xend = Topt + .2, yend =1), linetype = "solid",color = "darkgray",  
 arrow = arrow(length = unit(0.1, "inches"))  
 )+  
 annotation\_custom(grobTmax)+  
 #add arrows and label for parameter Tmax  
 geom\_segment(aes(x = Tmax+1, y = 0.15, xend = Tmax, yend = .02), linetype = "solid",color = "darkgray",  
 arrow = arrow(length = unit(0.1, "inches"))  
 )+  
 annotation\_custom(grobTmin)+  
 scale\_x\_continuous(limits = c(1,30), breaks = seq(0,30, 2),labels = seq(0,30, 2))+  
 scale\_y\_continuous(limits = c(0,1.1), breaks = seq(0,1, .2),labels = seq(0,1, .2))+  
 egg:: theme\_article()+  
 ggsave(here::here("scr", "model", "Beta\_fun.png"),  
 width = 7, height = 5, units = "in",  
 dpi = 820)  
  
 # rm(grobc, grobr, grobTmin, grobTopt, grobTmax, temp, temp\_dev, Rfact, Shape, Tmax, Tmin, Topt)  
pbeta



## The Model

### Sporulation

spor\_df <- read\_excel(here::here("dat", "Harrison data.xlsx"))  
  
names(spor\_df)[1] <- "rh"  
spor\_df <- gather(spor\_df, "wind", "spor", -rh)  
spor\_df$spor <- as.numeric(spor\_df$spor)  
spor\_df$wind <- as.numeric(spor\_df$wind)  
# spor\_df$wind <- factor(spor\_df$wind, levels = c("0.3", "1.4", "5.5", "13.7"))  
  
ggplot(spor\_df, aes(factor(spor\_df$wind, levels = c("0.3", "1.4", "5.5", "13.7")), rh))+  
 geom\_tile(aes(fill = spor))+  
 xlab("Wind (m/s)")+  
 ylab("Relative Humidity (%)")+  
 scale\_fill\_gradient(name = "No. of Spores", low = "white", high = "black")+  
 theme\_article()+  
 ggsave(  
 file = here::here("scr", "model", "fig" , "Harrison&Lowe\_Sporulation.png"),  
 width = 5.1,  
 height = 3.6,  
 dpi = 800  
 )



spor\_df <- spor\_df[spor\_df$wind == 1.4 ,] %>% arrange(., rh)

Modeling the relationship between relative humidity and sporualtion using logistic model.

start\_Ks <- c(110) #the population size at equilibrium  
start\_n0 <- c(0.001) #the population size at the begining  
start\_r<- c(0.15) #the growth rate  
rh <- spor\_df$rh - 80  
spor <- spor\_df$spor   
  
#Use minpack.lm function because nls apparently has a famouse "singular gradient issue"  
 f <- spor~K\*n0\*exp(r\*rh)/(K+n0\*(exp(r\*rh)-1))  
   
m <-  
 minpack.lm::nlsLM(  
 formula = f,  
 start = list(K = start\_Ks, n0 = start\_n0, r = start\_r),  
 control = nls.lm.control(maxiter = 500)  
 )  
m

## Nonlinear regression model  
## model: spor ~ K \* n0 \* exp(r \* rh)/(K + n0 \* (exp(r \* rh) - 1))  
## data: parent.frame()  
## K n0 r   
## 9.705e+04 6.049e-04 1.735e+00   
## residual sum-of-squares: 22310121  
##   
## Number of iterations till stop: 94   
## Achieved convergence tolerance: 1.49e-08  
## Reason stopped: Number of calls to `fcn' has reached or exceeded `maxfev' == 400.

fit\_data <- broom::augment(m)  
fit\_data[, c("spor", ".fitted")]

## # A tibble: 5 x 2  
## spor .fitted  
## <dbl> <dbl>  
## 1 69 0.000605  
## 2 4384 3.54   
## 3 17147 17073.   
## 4 98221 96972.   
## 5 95804 97050.

# return rh data to approprate scale  
fit\_data[, "rh"] <- fit\_data[, "rh"] + 80  
  
# Change to a resizing factor, as a proportion  
fit\_data[, c(".fitted", "spor")] <- lapply(fit\_data[, c(".fitted", "spor")], function(x) x/max(x))  
  
#do the same steps with the predicted data  
newdata <- data.frame( rhvec = seq(0, 20, 0.1))  
newdata$fit <- predict(m, newdata=data.frame( rh =newdata$rhvec ), interval = "prediction")  
newdata$rhvec <- newdata$rhvec+80  
  
newdata[, "fit"] <- newdata[, "fit"] / max(newdata[, "fit"])  
  
#Save the coefficents  
KSpor <- coef(m)["K"]  
n0Spor <- coef(m)["n0"]  
rSpor <- coef(m)["r"]  
KSpor <- 97049.81  
n0Spor <- 0.0006048922  
rSpor <- 1.734924  
  
newdata$rhvec <- ifelse(newdata$rhvec >=80, newdata$rhvec - 80, 0)  
sporulation\_rh <-  
 sapply(newdata$rhvec, function(x) {  
 spor\_rh\_hour <-  
 KSpor \* n0Spor \* exp(rSpor \* x) / (KSpor + n0Spor \* (exp(rSpor \* x) - 1))  
 # Change to proportion; sporulation is divided by sporulation at the equilibrium  
 spor\_rh\_hour <- as.numeric(spor\_rh\_hour) / as.numeric(KSpor)  
 return(spor\_rh\_hour)  
 })  
  
  
prhspor <-   
ggplot() +  
 geom\_hline(yintercept = 1, linetype = "dotted", color = "gray", size = .2)+  
 geom\_line(data =newdata, aes(newdata$rhvec+80, sporulation\_rh,color = "Fitted"), size = 0.8) +   
 geom\_point(data = fit\_data, aes(rh, spor,color = "Observed")) +  
 scale\_y\_continuous(limits = c(0,1.1), breaks = seq(0,1, .2),labels = seq(0,1, .2))+  
 scale\_color\_manual(values = c("Fitted" = "gray",  
 "Observed" = "black"))+  
 labs(y = "RH Sporulation Factor", x = "Relative Humidity (%)", colour = "Data:")+  
 egg::theme\_article()+  
 theme(  
 legend.position = c(.2, .64),  
 legend.text = element\_text(size = 13.3),  
 legend.title = element\_text(size = 13.3),  
 legend.key.width = unit(1, "cm")  
 ) +  
 ggsave(  
 file = here::here("scr", "model", "fig" , "spor\_rh.png"),  
 width = 6,  
 height = 4.5,  
 dpi = 800  
 )

Temperature relation to sporulation.

spor\_t <- data.frame( temp = seq(4,28, 0.1))  
  
TminSpor <- 6  
ToptSpor <- 20  
TmaxSpor <- 26  
RfactSpor <- 1 #set to default 1   
ShapeSpor <-2 # parameter that determines the shape of curve  
  
#Function to cacluate the sporulation temperature factor   
temp\_spor <- sapply(spor\_t$temp, function(x) {  
 sporulation\_temperature <-  
 RfactSpor \* ((TmaxSpor - x) / (TmaxSpor - ToptSpor) \* ((x - TminSpor) / (ToptSpor - TminSpor)) ^ ((ToptSpor - TminSpor) / (TmaxSpor - ToptSpor))) ^ ShapeSpor  
 sporulation\_temperature = ifelse(x < TminSpor | x > TmaxSpor, 0, sporulation\_temperature)  
   
})  
  
psportemp <-   
ggplot()+  
 geom\_line(aes(spor\_t$temp, temp\_spor))+  
 labs(y = "Temperature Sporulation Factor",  
 x = "Temperature (˚C)")+  
 scale\_x\_continuous(breaks = seq(0,30, 2),labels = seq(0,30, 2))+  
 scale\_y\_continuous(limits = c(0,1.1), breaks = seq(0,1, .2),labels = seq(0,1, .2))+  
 egg::theme\_article()+  
 theme(  
 legend.position = c(.2, .64),  
 legend.text = element\_text(size = 13.3),  
 legend.title = element\_text(size = 13.3),  
 legend.key.width = unit(1, "cm")  
 ) +  
 ggsave(  
 file = here::here("scr", "model", "fig" , "spor\_temp.png"),  
 width = 5.1,  
 height = 3.6,  
 dpi = 800  
 )

Save parameters into a separate file.  
We will also add the parameter for the sporulation threshold. It is considered that sporulation criteria is met if there was n (default 10) hours of sporulation.

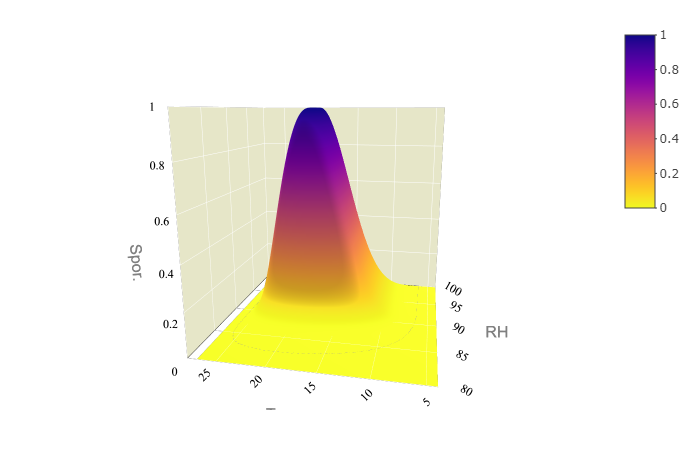
params <- data.frame(  
 TminSpor = TminSpor,  
 ToptSpor = ToptSpor,  
 TmaxSpor = TmaxSpor,  
 RfactSpor = RfactSpor,  
 ShapeSpor = ShapeSpor,  
   
 KSpor = KSpor,  
 n0Spor = n0Spor,  
 rSpor = rSpor,  
   
 spor\_dur = 10  
)  
  
write\_csv(params, here::here("scr", "model", "par", "parameters\_sporulation.csv"))

This function calculates the sporulation as a product of two vectors containing hourly measurements of relative humidity and temperature.

Sporulation <-  
 function(temp,  
 rh,  
 parameters) {  
   
 #Import parameters  
 #Temp factor  
 TminSpor <- parameters[, "TminSpor"]  
 ToptSpor <- parameters[, "ToptSpor"]  
 TmaxSpor <- parameters[, "TmaxSpor"]  
 RfactSpor <- parameters[, "RfactSpor"]  
 ShapeSpor <- parameters[, "ShapeSpor"]  
   
 #RH factor  
 KSpor <- parameters[, "KSpor"]  
 n0Spor <- parameters[, "n0Spor"]  
 rSpor <- parameters[, "rSpor"]  
   
   
   
 #Calculate temp factor  
 #Function to cacluate the sporulation temperature factor  
 sporulation\_temp <-   
 sapply(temp, function(x) {  
 sporulation\_temperature <-  
 RfactSpor \* ((TmaxSpor - x) / (TmaxSpor - ToptSpor) \* ((x - TminSpor) / (ToptSpor - TminSpor)) ^ ((ToptSpor - TminSpor) / (TmaxSpor - ToptSpor))) ^ ShapeSpor  
 sporulation\_temperature = ifelse(x < TminSpor | x > TmaxSpor, 0, sporulation\_temperature)  
 return(sporulation\_temperature)  
 }) %>% unlist()  
   
 # Calculate the RH factor  
 rh <- ifelse(rh >=80, rh - 80, 0) #resize rh scale to 1-20  
 sporulation\_rh <-  
 sapply(rh, function(x) {  
 spor\_rh\_hour <-   
 KSpor \* n0Spor \* exp(rSpor \* x) / (KSpor + n0Spor \* (exp(rSpor \* x) - 1))  
 # Change to proportion; sporulation is divided by sporulation at the equilibrium  
 spor\_rh\_hour <-as.numeric(spor\_rh\_hour)/ as.numeric(KSpor)  
 return(spor\_rh\_hour)  
 })  
  
 sporulation <-   
 round(sporulation\_temp \* sporulation\_rh, 6)   
 return(sporulation)  
 }

Visualisation of the overall effect of this function.

#Sporulation  
require(plotly)  
  
rhumt = seq(80,100, 0.1)  
tempt = seq(4,27, 0.1)  
  
spor <- outer(tempt, rhumt, Sporulation, params)  
  
f1 <- list(  
 family = "Arial, sans-serif",  
 size = 16,  
 color = "grey"  
)  
f2 <- list(  
 family = "Old Standard TT, serif",  
 size = 12,  
 color = "black"  
)  
tit<- list(  
 title = "Sporuation Function",  
 titlefont = f1,  
 tickfont = f2  
 )  
x <- list(  
 title = "RH",  
 titlefont = f1,  
 showticklabels = TRUE,  
 tickangle = 30,  
 tickfont = f2,  
 exponentformat = "E",  
 nticks= 8,  
 range= c(80,100),  
 backgroundcolor="rgb(230, 230,200)",  
 gridcolor="rgb(255,255,255)",  
 showbackground=TRUE,  
 zerolinecolor="rgb(255,255,255"  
)  
y <- list(  
 title = "Temp",  
 titlefont = f1,  
 showticklabels = TRUE,  
 tickangle = -45,  
 tickfont = f2,  
 exponentformat = "E",  
 nticks = 8,  
 range= c(4,28),  
 backgroundcolor="rgb(230, 230,200)",  
 gridcolor="rgb(255,255,255)",  
 showbackground=TRUE,  
 zerolinecolor="rgb(255,255,255"  
)  
z <- list(  
 title = "Spor.",  
 titlefont = f1,  
 showticklabels = TRUE,  
 tickangle = 0,  
 tickfont = f2,  
 exponentformat = "E",  
 nticks= 10,  
 range= c(0,1),  
 backgroundcolor="rgb(230, 230,200)",  
 gridcolor="rgb(255,255,255)",  
 showbackground=F,  
 zerolinecolor="rgb(255,255,255"  
)  
  
library(viridis)  
color\_var <- rev(plasma(256))  
  
p <-   
plot\_ly(x= rhumt,y=tempt,z=spor, type="surface",  
 colors = color\_var) %>%   
 layout(  
 title = tit,  
 scene = list(  
 xaxis = x,  
 yaxis = y,  
 zaxis = z  
 ))  
  
  
##Custom ticks  
axx <- list(  
 ticketmode = 'array',  
 ticktext = c("Huey", "Dewey", "Louie"),  
 tickvals = c(0,25,50),  
 range = c(-25,75)  
)  
  
htmlwidgets::saveWidget(p, "sporulation.html")



3D surface plot of sporulation function

## Survival

Airborne sporangia survival.

B0 <- 2.37  
B1 <- 0.45  
Survival <- function(x) {  
 Pr <- 1 / (1 + exp(-(B0 - B1 \* x)))  
}  
  
 psurv <-   
data.frame(  
 sol\_rad = seq(0,30, 0.1)  
) %>%   
 mutate(surv = Survival(sol\_rad)) %>%   
ggplot()+  
 geom\_hline(yintercept = 1, linetype = "dotted", color = "gray", size = .2)+  
 geom\_line(aes(sol\_rad, surv))+  
 scale\_y\_continuous( limits = c(0,1.1), breaks = seq(0,1.4,0.2))+  
 scale\_x\_continuous(limits = c(0,15))+  
 labs(title = "Mortality of Sporangia Due To Solar Radiation",   
 x = expression("Solar radiation"~(MJ/m^2/day)),  
 # x = paste0("Solar radiation (MJ/", expression(m^2)),  
 y = "Daily Survival Probability")+  
 theme\_article()+  
 ggsave(here::here("scr", "model", "fig", "survival.png"),  
 width = 7, height = 5, units = "in",  
 dpi = 820)

Save the parameters into a separate file.

params\_solsurv <- data.frame(  
 B0 = B0,  
 B1 = B1  
)  
  
write\_csv(params\_solsurv, here::here("scr", "model", "par", "parameters\_sol\_surv.csv"))

This function calculates the survival probabllty based on sum of total daily sola radiation.

SolSurv <-  
 function(sol,  
 params\_solsurv) {  
 B0 <- params\_solsurv[, "B0"]  
 B1 <- params\_solsurv[, "B1"]  
   
 Survival <- function(x) {  
 Pr <- 1 / (1 + exp(-(B0 - B1 \* x)))  
 }  
   
 sol\_surv <- Survival(sol)  
   
 return(round(sol\_surv, 6))  
 }

## Infection

The surface wethess on plant surface is an essential requirement for germination or zoospore release and for subsequent infection to take place. Risk of the infection is calculated as a function of temperature and leaf wetness risk factors.  
Temperature effect on Infection is estimated using two functions, one for the direct germinationa and one for zoospore infection. Minimum and maximum values are set equal for both functions.  
The leaf wetness is calculated using two linear functions.

# TODO add the rain to lw  
# Check the newforge data for leaf wetness could it be used?   
#   
  
temp <- seq(0,30,.1)  
  
#Direct infection function  
TminInf <- 6  
ToptInf <- 12  
TmaxInf <- 26  
RfactInf <- 1 #set to default 1   
ShapeInf <-15 # parameter that determines the shape of curve  
   
   
temp\_inf\_zoo <- sapply(temp, function(x) {  
 Infection\_temperature <-  
 RfactInf \* ((TmaxInf - x) / (TmaxInf - ToptInf) \* ((x - TminInf) / (ToptInf - TminInf)) ^ ((ToptInf - TminInf) / (TmaxInf - ToptInf))) ^ ShapeInf  
 Infection\_temperature = ifelse(x < TminInf | x > TmaxInf, 0, Infection\_temperature)  
   
})  
  
#Direct infection function  
TminInfDir <- 6  
ToptInfDir <- 23  
TmaxInfDir <- 26  
RfactInfDir <- 0.4   
ShapeInfDir <-.5 # parameter that determines the shape of curve  
  
  
#Function to cacluate the Infection temperature factor   
temp\_inf\_direct <- sapply(temp, function(x) {  
 Infection\_temperature <-  
 RfactInfDir \* ((TmaxInfDir - x) / (TmaxInfDir - ToptInfDir) \* ((x - TminInfDir) / (ToptInfDir - TminInfDir)) ^ ((ToptInfDir - TminInfDir) / (TmaxInfDir - ToptInfDir))) ^ ShapeInfDir  
 Infection\_temperature = ifelse(x < TminInfDir | x > TmaxInfDir, 0, Infection\_temperature)  
})  
# Find the curve intersect  
x <- which(temp\_inf\_zoo==RfactInf) #peek of zoospore germiantion  
y <- which(temp\_inf\_direct==RfactInfDir) #peek of direct germination  
  
  
  
# Straight lines (empirical)  
line1 <- data.frame(x = temp[x:y], y = temp\_inf\_direct[x:y])  
line2 <- data.frame(x = temp[x:y], y = temp\_inf\_zoo[x:y])  
  
intersect <- reconPlots::curve\_intersect(line1, line2) %>% as.data.frame()  
  
InterTemp <- intersect[, "x"]  
  
pinftemp <-   
ggplot() +  
 geom\_hline(yintercept = 1, linetype = "dotted", color = "gray", size = .2)+  
 geom\_line(aes(temp, temp\_inf\_direct, linetype = "Direct")) +  
 geom\_line(aes(temp, temp\_inf\_zoo, linetype = "Zoospore")) +  
 scale\_linetype("Infection pathway:") +  
 geom\_point(data = intersect, aes(x, y, shape = "Intercept"), size = 3) +  
 scale\_shape("") +  
 scale\_y\_continuous(limits = c(0,1.1), breaks = seq(0,1, .2),labels = seq(0,1, .2))+  
  
 labs(title = "Dependance of infection on temperature",  
 y = "Temperature Infection Factor",  
 x = "Temperature (˚C)") +  
 scale\_x\_continuous(breaks = seq(0, 30, 2), labels = seq(0, 30, 2)) +  
 egg::theme\_article()+  
 egg::theme\_article()+  
 theme(  
 legend.position = c(.82, .88),  
 # legend.text = element\_text(size = 13.3),  
 # legend.title = element\_text(size = 13.3),  
 legend.key.width = unit(1, "cm")  
 ) +  
 ggsave(here::here("scr", "model", "fig", "temp\_inf.png"),  
 width = 7, height = 5, units = "in",  
 dpi = 820)

RH factor.

RhminInf <- 86  
RhoptInf <- 95  
  
#fit linear model   
x <- c(RhminInf, RhoptInf)  
coefs <- lm(c(0,1) ~ x )[["coefficients"]]  
  
rhum <- c(RhminInf:100)  
  
inf\_rh <- sapply(rhum, function (rhum\_val) {  
 if (rhum\_val > RhminInf & rhum\_val <= RhoptInf) {  
 SporRh <- coefs[["(Intercept)"]] +coefs[["x"]] \* rhum\_val   
 } else if (rhum\_val > RhoptInf & rhum\_val <= 100) {  
 SporRh <- 1  
 } else {  
 SporRh <- 0  
 }  
})  
  
pinfrh <-   
ggplot()+  
 geom\_line(aes(rhum, inf\_rh))+  
 geom\_hline(yintercept = 1, linetype = "dotted", color = "gray", size = .2)+  
 labs(title = "Dependance of Infection On Relative Humidity",  
 y = "RH Infection Factor",  
 x = "Relative Humidity (%)") +  
 scale\_y\_continuous(limits = c(0,1.1), breaks = seq(0,1, .2),labels = seq(0,1, .2))+  
 scale\_x\_continuous(breaks = seq(RhminInf, 100, 2), labels = seq(RhminInf, 100,2 )) +  
 theme\_article()+  
 ggsave(here::here("scr", "model", "fig","rh\_inf.png"),  
 width = 7, height = 5, units = "in",  
 dpi = 820)

Save parameters into a separate file.

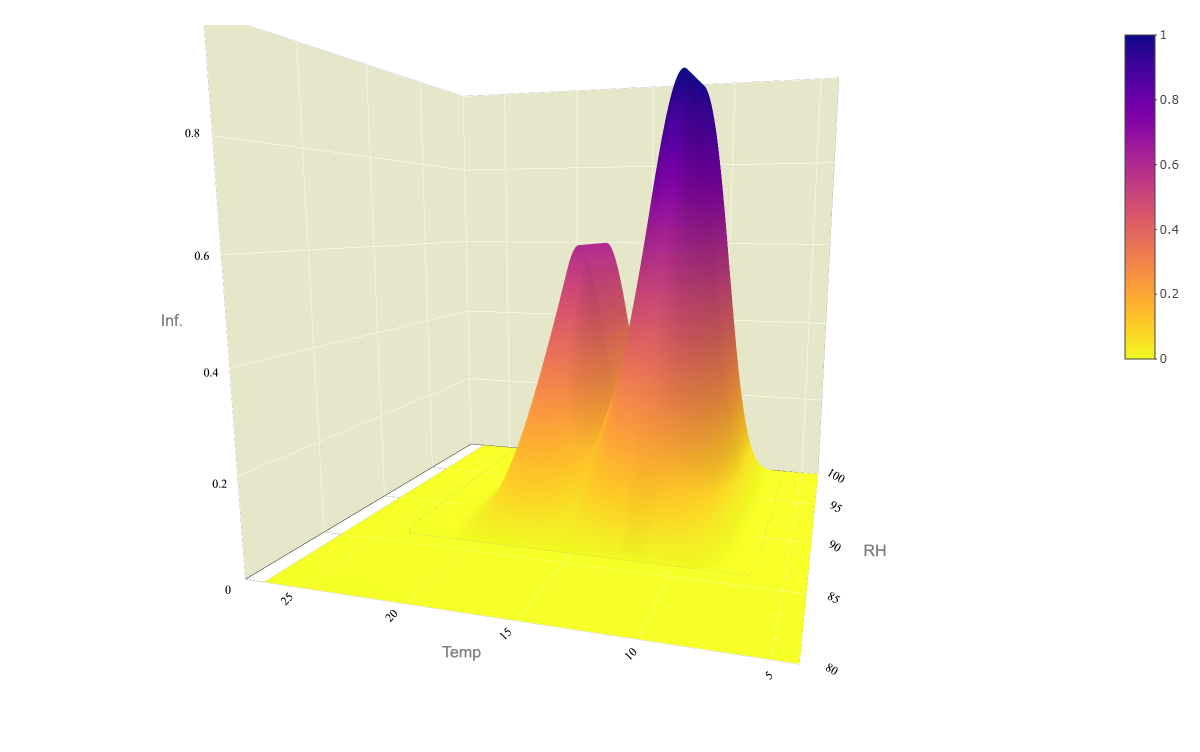
params\_inf <- data.frame(  
 #temperature factor  
 TminInf = TminInf,  
 ToptInf = ToptInf,  
 TmaxInf = TmaxInf,  
 RfactInf = RfactInf,  
 ShapeInf = ShapeInf,  
 # Direct penetrtion  
 TminInfDir = TminInf,  
 ToptInfDir = ToptInfDir,  
 TmaxInfDir = TmaxInf,   
 RfactInfDir = RfactInfDir,  
 ShapeInfDir = ShapeInfDir,  
  
 #rh factor  
 RhminInf = RhminInf,  
 RhoptInf = RhoptInf  
)  
  
write\_csv(params\_inf, here::here("scr", "model", "par", "parameters\_infection.csv"))

Here we put it all into a function. This function calculates the infection risk based on two given vectors containing hourly measurements of reltive humidity and temperature.

Infection <-  
 function(temp,  
 rh,  
 params\_inf) {  
   
 #Import parameters  
 #Temp factor zoospore  
 TminInf <- params\_inf[, "TminInf"]  
 ToptInf <- params\_inf[, "ToptInf"]  
 TmaxInf <- params\_inf[, "TmaxInf"]  
 RfactInf <- params\_inf[, "RfactInf"]  
 ShapeInf <- params\_inf[, "ShapeInf"]  
 #Temp factor direct  
 TminInfDir <- params\_inf[, "TminInf"]  
 ToptInfDir <- params\_inf[, "ToptInfDir"]  
 TmaxInfDir <- params\_inf[, "TmaxInf"]  
   
   
 RfactInfDir <- params\_inf[, "RfactInfDir"]  
 ShapeInfDir <- params\_inf[, "ShapeInfDir"]  
   
   
 # Temperature intersect between functions for mechanisms of infection  
 CalcIntersect <- function(TminInf,ToptInf,TmaxInf,RfactInf,ShapeInf,  
 TminInfDir,ToptInfDir,TmaxInfDir,RfactInfDir,ShapeInfDir){  
 temp <- c(0:34)  
   
 temp\_inf\_zoo <- sapply(temp, function(x) {  
 Infection\_temperature <-  
 RfactInf \* ((TmaxInf - x) / (TmaxInf - ToptInf) \* ((x - TminInf) / (ToptInf - TminInf)) ^ ((ToptInf - TminInf) / (TmaxInf - ToptInf))) ^ ShapeInf  
 Infection\_temperature = ifelse(x < TminInf | x > TmaxInf, 0, Infection\_temperature)  
 })  
   
 #Function to calculate the Infection temperature factor   
 temp\_inf\_direct <- sapply(temp, function(x) {  
 Infection\_temperature <-  
 RfactInfDir \* ((TmaxInfDir - x) / (TmaxInfDir - ToptInfDir) \* ((x - TminInfDir) / (ToptInfDir - TminInfDir)) ^ ((ToptInfDir - TminInfDir) / (TmaxInfDir - ToptInfDir))) ^ ShapeInfDir  
 Infection\_temperature = ifelse(x < TminInfDir | x > TmaxInfDir, 0, Infection\_temperature)  
 })  
   
 # Find the curve intersect  
 x <- which(temp\_inf\_zoo==RfactInf) #peek of zoospore germiantion  
 y <- which(temp\_inf\_direct==RfactInfDir) #peek of direct germination  
   
 # Straight lines (empirical)  
 line1 <- data.frame(x = temp[x:y], y = temp\_inf\_direct[x:y])  
 line2 <- data.frame(x = temp[x:y], y = temp\_inf\_zoo[x:y])  
   
 intersect <- reconPlots::curve\_intersect(line1, line2) %>% as.data.frame()  
   
 InterTemp <- intersect[, "x"]  
 return(InterTemp)  
 }  
   
 InterTemp <-   
 CalcIntersect(TminInf,ToptInf,TmaxInf,RfactInf,ShapeInf,  
 TminInfDir,ToptInfDir,TmaxInfDir,RfactInfDir,ShapeInfDir)  
   
   
 #Calculate temp factor  
 #Function to calculate the Infection temperature factor  
 temp\_inf\_zoo <- sapply(temp, function(x) {  
 Infection\_temperature <-  
 RfactInf \* ((TmaxInf - x) / (TmaxInf - ToptInf) \* ((x - TminInf) / (ToptInf - TminInf)) ^ ((ToptInf - TminInf) / (TmaxInf - ToptInf))) ^ ShapeInf  
 Infection\_temperature = ifelse(x < TminInf |  
 x > TmaxInf, 0, Infection\_temperature)  
 }) %>% unlist()  
   
 #Function to cacluate the Infection temperature factor  
 temp\_inf\_direct <- sapply(temp, function(x) {  
 Infection\_temperature <-  
 RfactInfDir \* ((TmaxInfDir - x) / (TmaxInfDir - ToptInfDir) \* ((x - TminInfDir) / (ToptInfDir - TminInfDir)) ^ ((ToptInfDir - TminInfDir) / (TmaxInfDir - ToptInfDir))) ^ ShapeInfDir  
 Infection\_temperature = ifelse(x < TminInfDir |  
 x > TmaxInfDir, 0, Infection\_temperature)  
 })%>% unlist()  
   
   
 inf\_temp <-   
 ifelse(temp <= InterTemp, temp\_inf\_zoo, temp\_inf\_direct )  
   
 #RH factor  
 RhminInf <- params\_inf[, "RhminInf"]  
 RhoptInf <- params\_inf[, "RhoptInf"]  
   
   
 # Calculate the RH factor  
 #fit linear model  
 x <- c(RhminInf, RhoptInf) %>% as.numeric()  
 coefs <- lm(c(0, 1) ~ x)[["coefficients"]]  
   
 inf\_rh <- sapply(rh, function (rhum\_val) {  
 if (rhum\_val > RhminInf & rhum\_val <= RhoptInf) {  
 SporRh <- coefs[["(Intercept)"]] + coefs[["x"]] \* rhum\_val  
 } else if (rhum\_val > RhoptInf & rhum\_val <= 100) {  
 SporRh <- 1  
 } else {  
 SporRh <- 0  
 }  
 })  
   
 infection = round(inf\_temp \* inf\_rh, 6)  
 return(infection)  
 }

Visualisation of the overal effect of this function.

#Infection  
require(plotly)  
  
rhumt = seq(80,100, 0.1)  
tempt = seq(4,27, 0.1)  
  
inf\_ar <- outer(tempt, rhumt, Infection, params\_inf)  
  
  
  
  
f1 <- list(  
 family = "Arial, sans-serif",  
 size = 16,  
 color = "grey"  
)  
f2 <- list(  
 family = "Old Standard TT, serif",  
 size = 12,  
 color = "black"  
)  
tit<- list(  
 title = "Infuation Function",  
 titlefont = f1,  
 tickfont = f2  
 )  
x <- list(  
 title = "RH",  
 titlefont = f1,  
 showticklabels = TRUE,  
 tickangle = 30,  
 tickfont = f2,  
 exponentformat = "E",  
 nticks= 8,  
 range= c(80,100),  
 backgroundcolor="rgb(230, 230,200)",  
 gridcolor="rgb(255,255,255)",  
 showbackground=TRUE,  
 zerolinecolor="rgb(255,255,255"  
)  
y <- list(  
 title = "Temp",  
 titlefont = f1,  
 showticklabels = TRUE,  
 tickangle = -45,  
 tickfont = f2,  
 exponentformat = "E",  
 nticks = 8,  
 range= c(4,28),  
 backgroundcolor="rgb(230, 230,200)",  
 gridcolor="rgb(255,255,255)",  
 showbackground=TRUE,  
 zerolinecolor="rgb(255,255,255"  
)  
z <- list(  
 title = "Infection",  
 titlefont = f1,  
 showticklabels = TRUE,  
 tickangle = 0,  
 tickfont = f2,  
 exponentformat = "E",  
 nticks= 10,  
 range= c(0,1),  
 backgroundcolor="rgb(230, 230,200)",  
 gridcolor="rgb(255,255,255)",  
 showbackground=F,  
 zerolinecolor="rgb(255,255,255"  
)  
  
library(viridis)  
color\_var <- rev(plasma(256))  
  
p <-   
plot\_ly(x= rhumt,y=tempt,z=inf\_ar, type="surface",  
 colors = color\_var) %>%   
 layout(  
 title = tit,  
 scene = list(  
 xaxis = x,  
 yaxis = y,  
 zaxis = z  
 ))  
  
  
##Custom ticks  
axx <- list(  
 ticketmode = 'array',  
 ticktext = c("Huey", "Dewey", "Louie"),  
 tickvals = c(0,25,50),  
 range = c(-25,75)  
)  
  
htmlwidgets::saveWidget(p, "Infection.html")

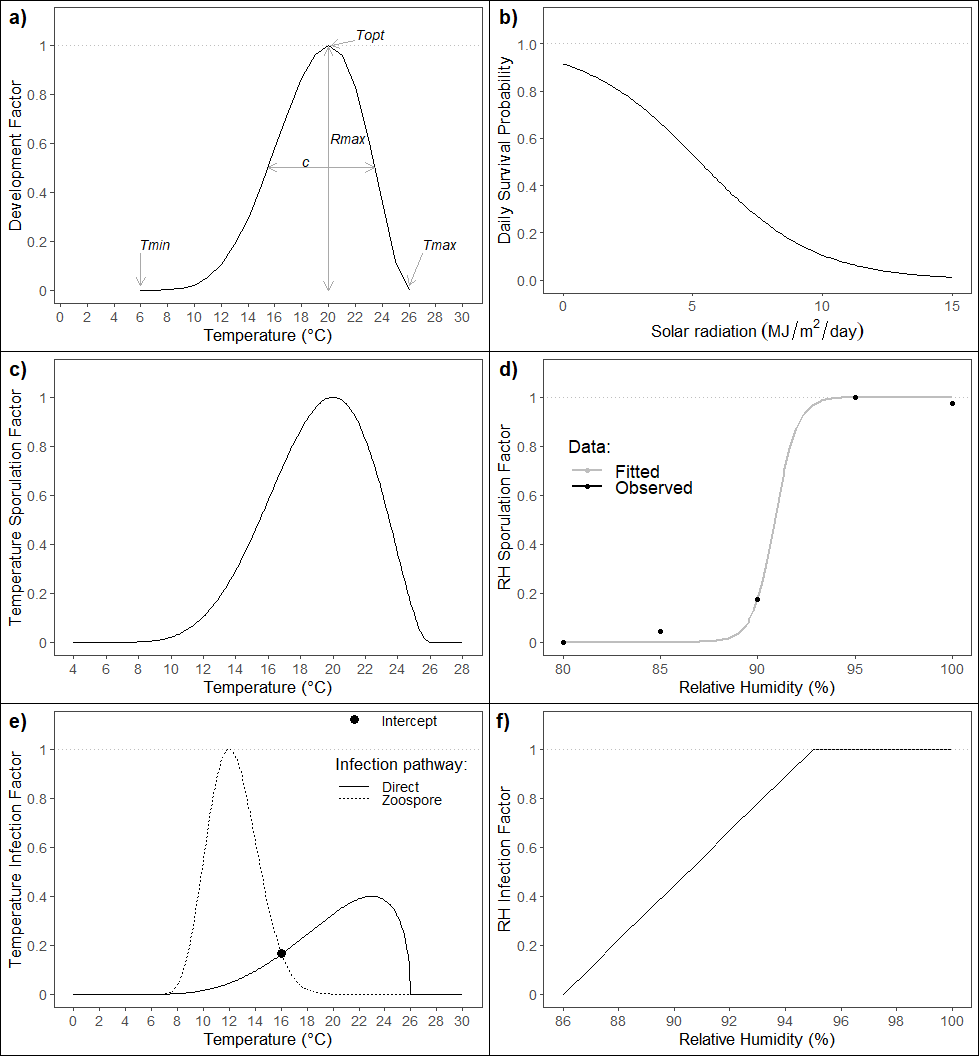


3D surface plot of Infection function

## Plot functions

Plot of functions in a single graph.

plotls <- list(pbeta, psurv, psportemp , prhspor, pinftemp, pinfrh)  
  
plotls <-   
lapply(plotls,function(x)  
 x+  
 theme(plot.title = element\_blank(),  
 text = element\_text(size= 13),  
 plot.background = element\_rect(size=.4,linetype="solid",color="black"))  
 )  
ggpubr::ggarrange(plotlist = plotls,   
 # heights = c(1.2,1,1),  
 font.label = list(size = 15, face = "bold"),  
 labels = paste0(letters[1:length(plotls)], ")"),  
 ncol= 2, nrow= 3)+  
 ggsave(filename=here::here("scr", "model", "fig","together.png"),  
 width = 10.2, height =11, dpi = 820)



# shell.exec(here::here("scr", "model", "fig","together.png"))

# Geo location

Function used to calculate sunset and sunrise times for the locations of interest.

#https://github.com/cran/HelpersMG/blob/master/R/sun.info.R  
  
  
SunInfo <- function(date, latitude, longitude,   
 #Ireland uses Irish Standard Time (IST, UTC+01:00 in the summer months  
 #and Greenwich Mean Time (UTC+0) in the winter period.  
 UTC\_zone  
 ){  
   
 d <- as.numeric(as.POSIXlt(date)$yday)+1  
 Lat <- latitude  
 Long <- longitude  
   
 ## d is the day of year  
 ## Lat is latitude in decimal degrees  
 ## Long is longitude in decimal degrees (negative == West)  
   
 ##This method is copied from:  
 ##Teets, D.A. 2003. Predicting sunrise and sunset times.  
 ## The College Mathematics Journal 34(4):317-321.  
   
 ## At the default location the estimates of sunrise and sunset are within  
 ## seven minutes of the correct times (http://aa.usno.navy.mil/data/docs/RS\_OneYear.php)  
 ## with a mean of 2.4 minutes error.  
   
 ## Function to convert degrees to radians  
 rad <- function(x) pi\*x/180  
   
 ##Radius of the earth (km)  
 R=6378  
   
 ##Radians between the xy-plane and the ecliptic plane  
 epsilon=rad(23.45)  
   
 ##Convert observer's latitude to radians  
 L=rad(Lat)  
   
 ## Calculate offset of sunrise based on longitude (min)  
 ## If Long is negative, then the mod represents degrees West of  
 ## a standard time meridian, so timing of sunrise and sunset should  
 ## be made later.  
 timezone = -4\*(abs(Long)%%15)\*sign(Long)  
   
 ## The earth's mean distance from the sun (km)  
 r = 149598000  
   
 theta = 2\*pi/365.25\*(d-80)  
   
 z.s = r\*sin(theta)\*sin(epsilon)  
 r.p = sqrt(r^2-z.s^2)  
   
 t0 = 1440/(2\*pi)\*acos((R-z.s\*sin(L))/(r.p\*cos(L)))  
   
 ##a kludge adjustment for the radius of the sun  
 that = t0+5   
   
 ## Adjust "noon" for the fact that the earth's orbit is not circular:  
 n = 720-10\*sin(4\*pi\*(d-80)/365.25)+8\*sin(2\*pi\*d/365.25)  
   
 ## now sunrise and sunset are:  
 sunrise = (n-that+timezone)/60  
 sunset = (n+that+timezone)/60  
   
 UTC <- (((7.5+Long)%%360)) %/% 15  
 if (UTC>12) {UTC <- 12-UTC;tz <- "Etc/GMT"} else {tz <- "Etc/GMT+"}  
 tz <- paste0(tz, UTC)  
   
 df <-  
 data.frame(  
 sunrise = sunrise,  
 sunset = sunset,  
 day.length = sunset - sunrise,  
 date.time.sunrise = as.POSIXlt(format(date, "%Y-%m-%d"), tz =  
 tz) + sunrise \* 60 \* 60,  
 date.time.sunset = as.POSIXlt(format(date, "%Y-%m-%d"), tz =  
 tz) + sunset \* 60 \* 60  
 )  
   
 sunrise.UTC <-  
 as.POSIXlt(  
 format(df$date.time.sunrise, format = "%Y-%m-%d %H:%M:%S"),  
 tz = "UTC",  
 use.tz = TRUE  
 )  
 sunrise.UTC.dec <-  
 sunrise.UTC$hour + sunrise.UTC$min / 60 + sunrise.UTC$sec / 3600  
 sunset.UTC <-  
 as.POSIXlt(  
 format(df$date.time.sunset, format = "%Y-%m-%d %H:%M:%S"),  
 tz = "UTC",  
 use.tz = TRUE  
 )  
 sunset.UTC.dec <-  
 sunset.UTC$hour + sunset.UTC$min / 60 + sunset.UTC$sec / 3600  
   
 #Return   
 df <-  
 data.frame(  
 # df,  
 sunrise = sunrise.UTC + c(UTC\_zone\*60\*60),  
 sunset = sunset.UTC + c(UTC\_zone\*60\*60)  
 # time.sunrise.UTC = sunrise.UTC.dec,  
 # time.sunset.UTC = sunset.UTC.dec  
 )  
   
 return(df)  
}

params\_time <- data.frame(  
 hr\_before\_spor = 5,  
 hr\_after\_spor = 5,  
 hr\_after\_inf = 5  
)  
  
write\_csv(params\_time, here::here("scr", "model", "par", "parameters\_time.csv"))

Wraper function for SunInfo function.

GetTimes <- function(fun\_df, params\_time) {  
 if(all(str\_detect(colnames(fun\_df), fixed("lon", ignore\_case=TRUE)))){ stop("No Longitude reference or it is not named: 'lon'!")}  
 if(all(str\_detect(colnames(fun\_df), fixed("lat", ignore\_case=TRUE)))){ stop("No Latitude reference or it is not named: 'lat'!")}  
 lat <- fun\_df[, str\_detect(colnames(fun\_df), fixed("lat", ignore\_case=TRUE))]   
 lon <- fun\_df[, str\_detect(colnames(fun\_df), fixed("lon", ignore\_case=TRUE))]   
   
 #Extract dates and geo-coordinates  
 tempdf <-   
 data.frame(  
 date = unique(fun\_df$short\_date),  
 lat = unique(lat),  
 lon = unique(lat)  
 )   
   
 fun\_ls <- list()  
 for (i in seq\_along(1:nrow(tempdf))) {  
 fun\_ls[[i]] <- SunInfo(tempdf[i,"date"],   
 tempdf[i, "lat"],   
 tempdf[i,"lon"],  
 UTC\_zone=1)  
 }  
   
 tempdf <-   
 fun\_ls %>%   
 bind\_rows() %>%   
 bind\_cols(tempdf, .) %>%   
 mutate(sunset\_hr = lubridate::hour(sunset),  
 sunrise\_hr = lubridate::hour(sunrise),  
 doy = lubridate::yday(date)) %>%   
 select(c("doy", "sunrise\_hr", "sunset\_hr"))  
   
 #Retun sunset and sunrise times as a data frame  
 return(tempdf)  
   
}

# Parameteres

All parameteres are stored in a single file.

all\_pars <- cbind(params\_inf,params\_solsurv, params, params\_time)  
write\_csv(all\_pars, here::here("scr", "model", "par", "par\_default.csv"))  
all\_pars

## TminInf ToptInf TmaxInf RfactInf ShapeInf TminInfDir ToptInfDir  
## 1 6 12 26 1 15 6 23  
## TmaxInfDir RfactInfDir ShapeInfDir RhminInf RhoptInf B0 B1 TminSpor  
## 1 26 0.4 0.5 86 95 2.37 0.45 6  
## ToptSpor TmaxSpor RfactSpor ShapeSpor KSpor n0Spor rSpor  
## 1 20 26 1 2 97049.81 0.0006048922 1.734924  
## spor\_dur hr\_before\_spor hr\_after\_spor hr\_after\_inf  
## 1 10 5 5 5