Ceballos-growthcurver-values

Carson Stacy

6/20/2020

```
# renv::init()

library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(DescTools)
library(MESS)
#this should normally be commented out, except when knitting. it is where I downloaded grofit b/c it is
install.packages("~/Downloads/grofit_1.1.1-1.tar.gz", repos = NULL, type = "source")
```

```
## Installing package into '/home/carson/Documents/Research/CEMB/Virulence/Index/renv/library/R-4.0/x86_
## (as 'lib' is unspecified)
```

```
library(grofit)
library(readxl)
library(ggpubr)
library(here)
```

```
## here() starts at /home/carson/Documents/Research/CEMB/Virulence/Index
```

```
set_here(path='..')
```

```
## File .here already exists in /home/carson/Documents/Research/CEMB/Virulence/Index
```

```
# Week_2_Growthcurver_S1_parameters <- read_excel("~/Downloads/Week 2 - Growthcurver S1 parameters.xlsx

# if (file.exists("FinalOutPutfigZ.Rda")) {
#   load("FinalOutPutfigZ.Rda")
```

```r
# } else {

options(scipen = 999)

#IMPORT DATA HERE AS DATA.FRAME

# dat <- read_csv("~/Downloads/Supplemental_Table_S1_ready.csv")

dat <- read_excel(here("/Data/Processed/Sulfolobus/Sulfolobus_Infection_Growth_Curves.xlsx"), sheet = "S


dat <- as.data.frame(dat[complete.cases(dat),])



trim <- 15

colnames(dat)[1] <- "time"

colnames(dat)[-1] <- str_pad(colnames(dat)[-1], trim, pad = "0", side = "left" )




# maxbyrowraw <- colnames(dat[-1])[max.col(dat[-1],ties.method="random")]
# maxbyrowconvert <- as.data.frame(table(maxbyrowraw))
# maxbyrowcount <- arrange(maxbyrowconvert,-Freq)
# maxmax <- as.character(maxbyrowcount$maxbyrowraw[1])

#note: should always be 1 (first column)

timeColumn <- 1
#can be adjsuted. average of ctl replicates is recommended based on zero science (seems like a good sta
controlColumn <- 5
#shouldn't change unless time column != 1
a <- 2
#ADJUST ME: total number of curves in table + 1
b <- ncol(dat)
#What is the timepoint at which stationary phase is reached?
t_stationary <- 26

firstRun <- TRUE
c <- 2
d <- length(dat[[timeColumn]])
#d <- 20 #this is where 'stationary phase' would traditionally be considered fully reached
for (j in c:d) {


figZ <- gcFitSpline(dat[[timeColumn]], dat[[controlColumn]], gcID = "spline",
            control = grofit.control())
lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
```

```
upperbound <- as.numeric(dat[j,1])
#upperbound <- dat[timeColumn[[j]]
#AUCraw <- AUC(dat[1:length(dat[[timeColumn]]),1], dat[1:length(dat[[controlColumn]]),controlColumn])
AUCraw <- AUC(dat[1:j,1], dat[1:j,controlColumn])
PI <- 0
IscZ <- 0

storage.vector_figZ <- data.frame( "mumax"= mumax, "K"= K, "lambda"= lambda, "UpperBound" = upperbound,


for (i in a:b) {
if(i != controlColumn) {
figZ <- gcFitSpline(dat$time, dat[[i]], gcID = "spline444",
            control = grofit.control())
#plot(figZ)

lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])

  AUCraw <- AUC(dat[1:j,1], dat[1:j,i])
  if(i == controlColumn) {
    PI <- 0
  } else {
    PI <- (1 - (AUCraw/storage.vector_figZ$AUC[1])) * 100
  }

  IscZ <-  (1 - sqrt((AUCraw*K)/(storage.vector_figZ$AUC[1]*storage.vector_figZ$K[1]))) * 100


storage.vector_figZ <- rbind(storage.vector_figZ, c( mumax, K, lambda, upperbound, as.numeric(AUCraw), I


}
}
#output <- rbind
if(firstRun == TRUE) {
  #something
  firstRun <- FALSE
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- storage.vector_figZ
} else {
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- rbind(FinalOutPutfigZ, storage.vector_figZ)
}

}
#storage.vector_figZ <- storage.vector_figZ[-c(1), ]

#storage.vector_figZ
#Below code to get rid of scientific notation:
options(scipen=999)
```

```r
#here is how to get back to scientific notation: options(scipen=0)



library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:DescTools':
##
##     %like%
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
setDT(FinalOutPutfigZ, keep.rownames = TRUE)
#here I'm fixing the group names to not say replicates
# FinalOutPutfigZ$rn <- c(str_replace_all(string=FinalOutPutfigZ$rn,pattern="\\l.*$",replacement="l"))
FinalOutPutfigZ$rn <- substr(FinalOutPutfigZ$rn, 1, trim)
# df$col1 <- strtrim(df$col, 1, 1)

IscOrderedOutputFigZ <- FinalOutPutfigZ %>%
  arrange((Isc))



if (exists("maxIsc") == TRUE) {
  save(FinalOutPutfigZ,file="FinalOutPutfigZ.Rda")
} else {
maxIsc <- IscOrderedOutputFigZ[1,1]
}
# }

# for writing a data.frame or list of data.frames to an xlsx file
#write.xlsx(FinalOutPutfigZ, 'Isc_figZISC.xlsx')
#FinalOutPUtfigZsave <- FinalOutPutfigZ

IscOrderedOutputFigZ
```

```
##                   rn      mumax     K    lambda UpperBound     AUC          PI
##   1: 00000S437CTLAVG 0.07189840 1.930 -1.112681          2   0.300 -13.3501259
##   2: 0000000S437CTL3 0.06102719 2.130 32.370296        104 136.950  -0.9558679
##   3: 0000000S437CTL3 0.06102719 2.130 32.370296         98 125.520  -0.4937151
##   4: 0000000S437CTL3 0.06102719 2.130 32.370296         92 114.150  -0.1989642
##   5: 00000S437CTLAVG 0.07189840 1.930 -1.112681          8   2.370 -10.1983881
```

```
##  ---
## 252: 000000S437SSV9A 0.05382076 0.568 23.843297         98  32.803  73.7372902
## 253: 000000S437SSV9B 0.05771198 0.640 -1.386194         92  26.160  77.0371888
## 254: 000000S437SSV9A 0.05382076 0.568 23.843297        104  34.750  74.3832318
## 255: 000000S437SSV9B 0.05771198 0.640 -1.386194         98  27.240  78.1911345
## 256: 000000S437SSV9B 0.05771198 0.640 -1.386194        104  28.380  79.0790250
##            Isc
##   1: -5.558361
##   2: -4.654659
##   3: -4.414841
##   4: -4.261603
##   5: -4.080473
##  ---
## 252: 72.435703
## 253: 72.640668
## 254: 72.776791
## 255: 73.336970
## 256: 73.885367
```

```
gdIsc <- FinalOutPutfigZ %>%
        group_by(rn, UpperBound) %>%
        #group_by(UpperBound) %>%
        summarise(Isc = mean(Isc))
```

```
## `summarise()` regrouping output by 'rn' (override with `.groups` argument)
```

```
gdVr <- gdIsc %>%
  group_by(rn) %>%
  summarize(Vr = auc(UpperBound, Isc, type = "spline")/(max(UpperBound) - min(UpperBound)))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
gdPI <- FinalOutPutfigZ %>%
        group_by(rn) %>%
        summarise(PI = mean(PI))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
AUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == max(FinalOutPutfigZ$UpperBound)) %>%
    mutate(rAUC = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select(c(-1,-4, -5, -6, -7, -8))
AUCfigZ
```

```
##          mumax         K      rAUC
##  1: 0.05644127 1.9633333  99.05318
##  2: 0.05202223 1.9200000  99.21869
##  3: 0.06102719 2.1300000 100.00000
##  4: 0.04432807 1.6541968  95.37788
##  5: 0.04134172 1.7102575  91.54436
```

```
##  6: 0.04912324 1.7400000  93.75685
##  7: 0.02831545 1.3021414  60.35049
##  8: 0.02221345 1.2586455  64.97262
##  9: 0.07608401 1.5600000  76.72873
## 10: 0.05382076 0.5680000  25.37422
## 11: 0.05771198 0.6400000  20.72289
## 12: 0.04308685 0.7800000  35.59328
## 13: 0.07189840 1.9300000  97.94085
## 14: 0.04107026 1.6846049  93.55969
## 15: 0.02831955 1.3054468  67.35061
## 16: 0.04341265 0.5496667  27.23013
```

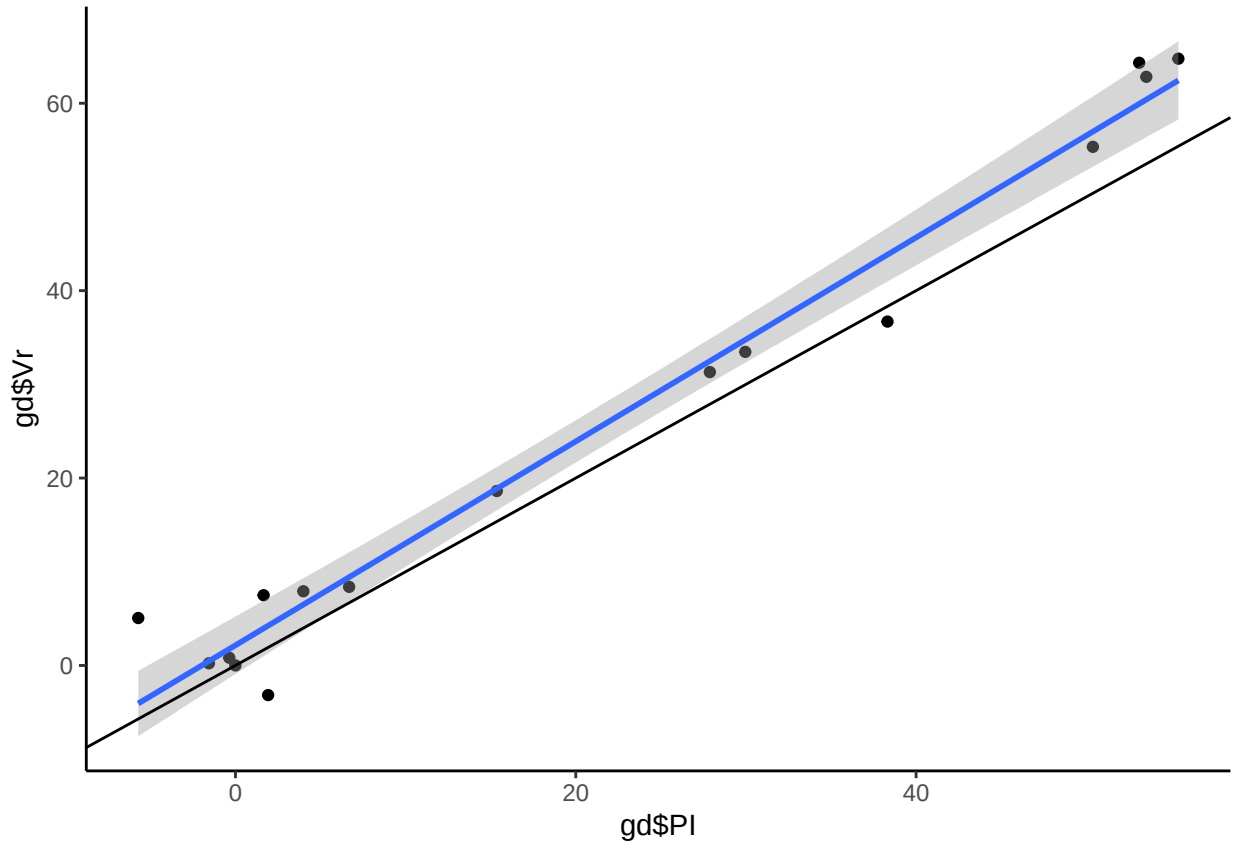
```
trimAUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == t_stationary) %>%
    mutate(AUCtrim = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select("AUCtrim")


gd <- cbind(gdVr, gdPI, AUCfigZ, trimAUCfigZ)
gd <- gd[-3]
gd
```

```
##                 rn          Vr          PI      mumax          K      rAUC
## 1  0000000S437CTL1   0.0000000   0.0000000 0.05644127 1.9633333  99.05318
## 2  0000000S437CTL2   0.8196314 -0.3603004 0.05202223 1.9200000  99.21869
## 3  0000000S437CTL3  -3.1616953   1.9197086 0.06102719 2.1300000 100.00000
## 4  000000S437SSV1A   5.0608600 -5.7112100 0.04432807 1.6541968  95.37788
## 5  000000S437SSV1B   7.9186199   3.9906215 0.04134172 1.7102575  91.54436
## 6  000000S437SSV1C   8.3869115   6.6812940 0.04912324 1.7400000  93.75685
## 7  000000S437SSV8A  36.6990402  38.3189723 0.02831545 1.3021414  60.35049
## 8  000000S437SSV8B  33.4665488  29.9625217 0.02221345 1.2586455  64.97262
## 9  000000S437SSV8C  18.6179605  15.3587025 0.07608401 1.5600000  76.72873
## 10 000000S437SSV9A  64.7555389  55.4141494 0.05382076 0.5680000  25.37422
## 11 000000S437SSV9B  62.8176249  53.5299810 0.05771198 0.6400000  20.72289
## 12 000000S437SSV9C  55.3483937  50.3896638 0.04308685 0.7800000  35.59328
## 13 00000S437CTLAVG   0.2275353  -1.5594082 0.07189840 1.9300000  97.94085
## 14 0000S437SSV1AVG   7.4969883   1.6535685 0.04107026 1.6846049  93.55969
## 15 0000S437SSV8AVG  31.3076909  27.8800655 0.02831955 1.3054468  67.35061
## 16 0000S437SSV9AVG  64.3173981  53.1112647 0.04341265 0.5496667  27.23013
##       AUCtrim
## 1    82.42499
## 2    86.31906
## 3    80.88144
## 4   100.00000
## 5    90.56487
## 6    82.74364
## 7    49.22408
## 8    59.15580
## 9    73.92924
## 10   45.87213
## 11   52.45189
## 12   49.66480
## 13   80.07449
```

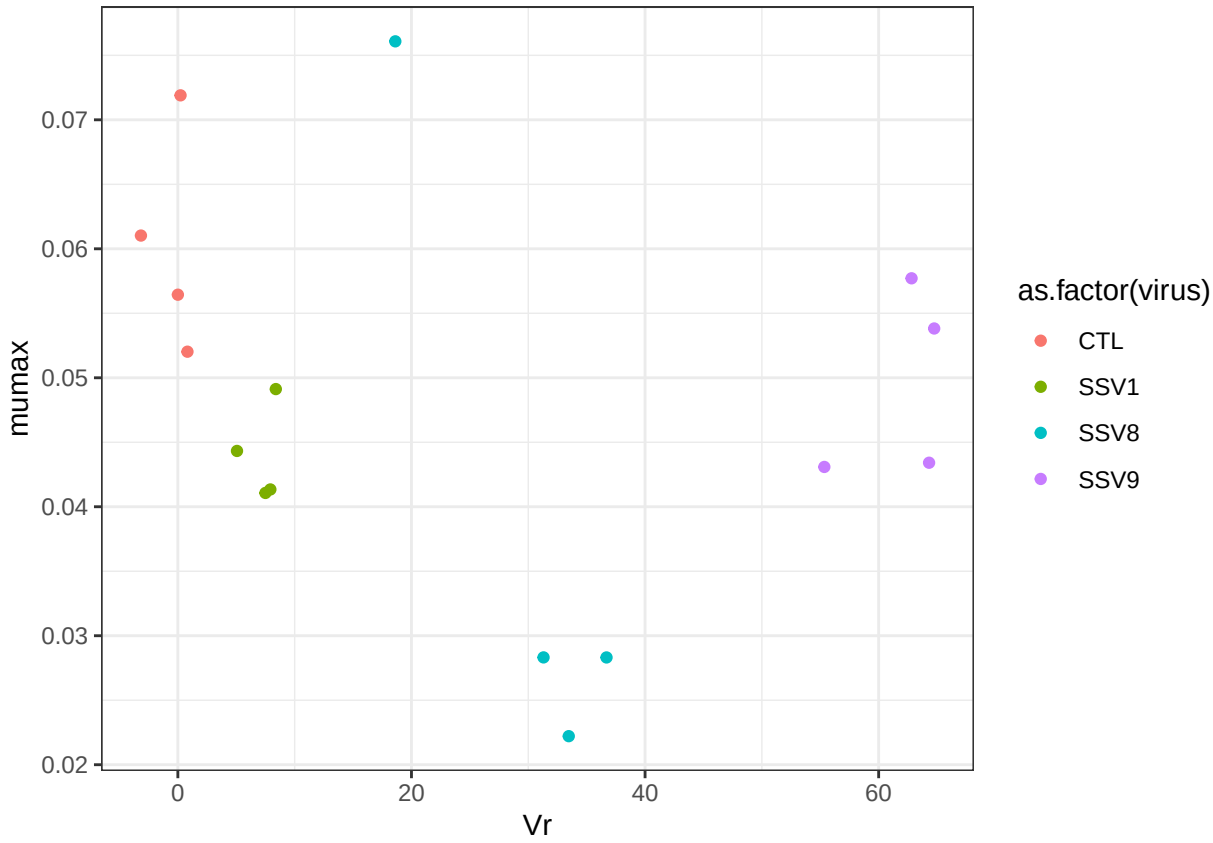
```
## 14  91.10283
## 15  60.76971
## 16  49.32961
```

```
gd$virus <- c(
    rep("CTL", 3),
    rep("SSV1", 3),
    rep("SSV8", 3),
    rep("SSV9", 3),
    rep("CTL", 1),
    rep("SSV1", 1),
    rep("SSV8", 1),
    rep("SSV9", 1)
)

# gd$virus <- c(
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12)
# )

gd$host <- c(
    rep("S437", 16)
)

gd437 <- gd

ggplot(data = NULL, aes(x = gd$PI, y = gd$Vr)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_abline() +
  theme_classic()
```

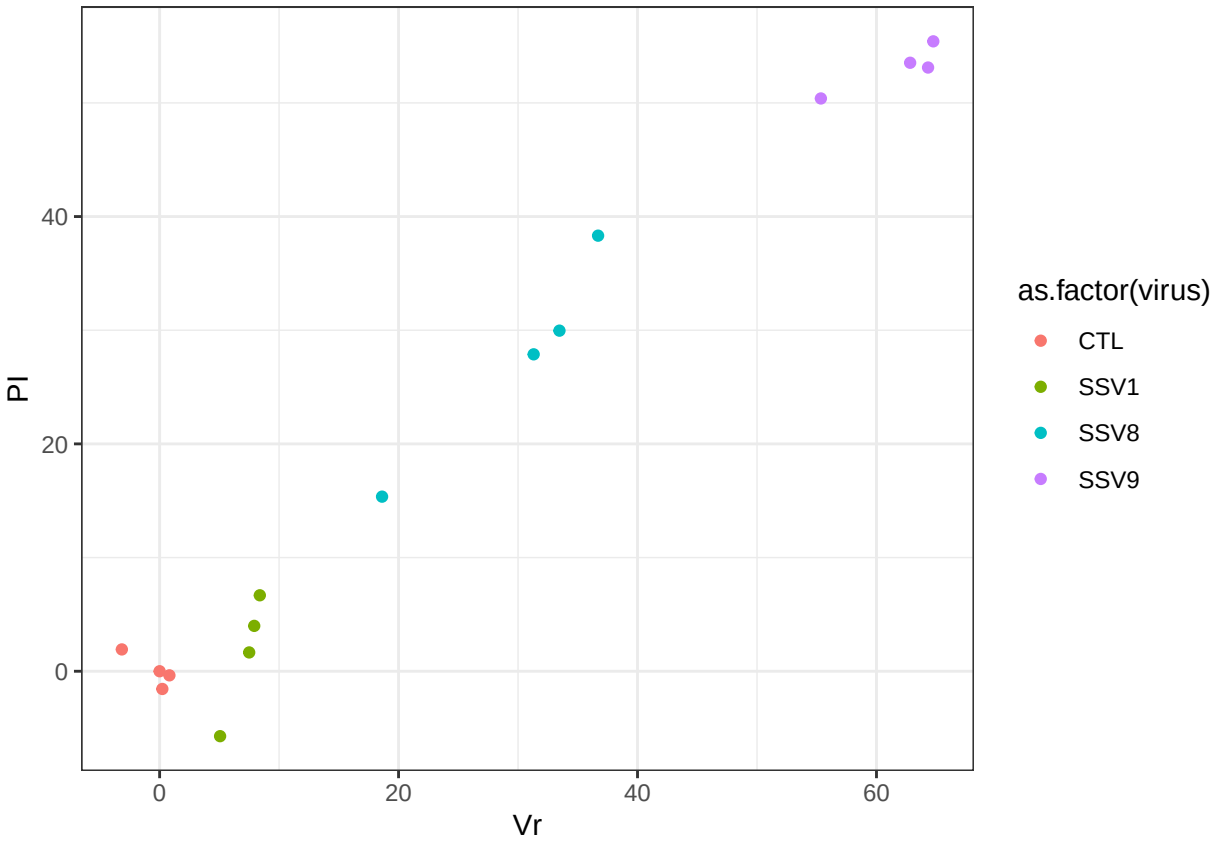
```
## `geom_smooth()` using formula 'y ~ x'
```



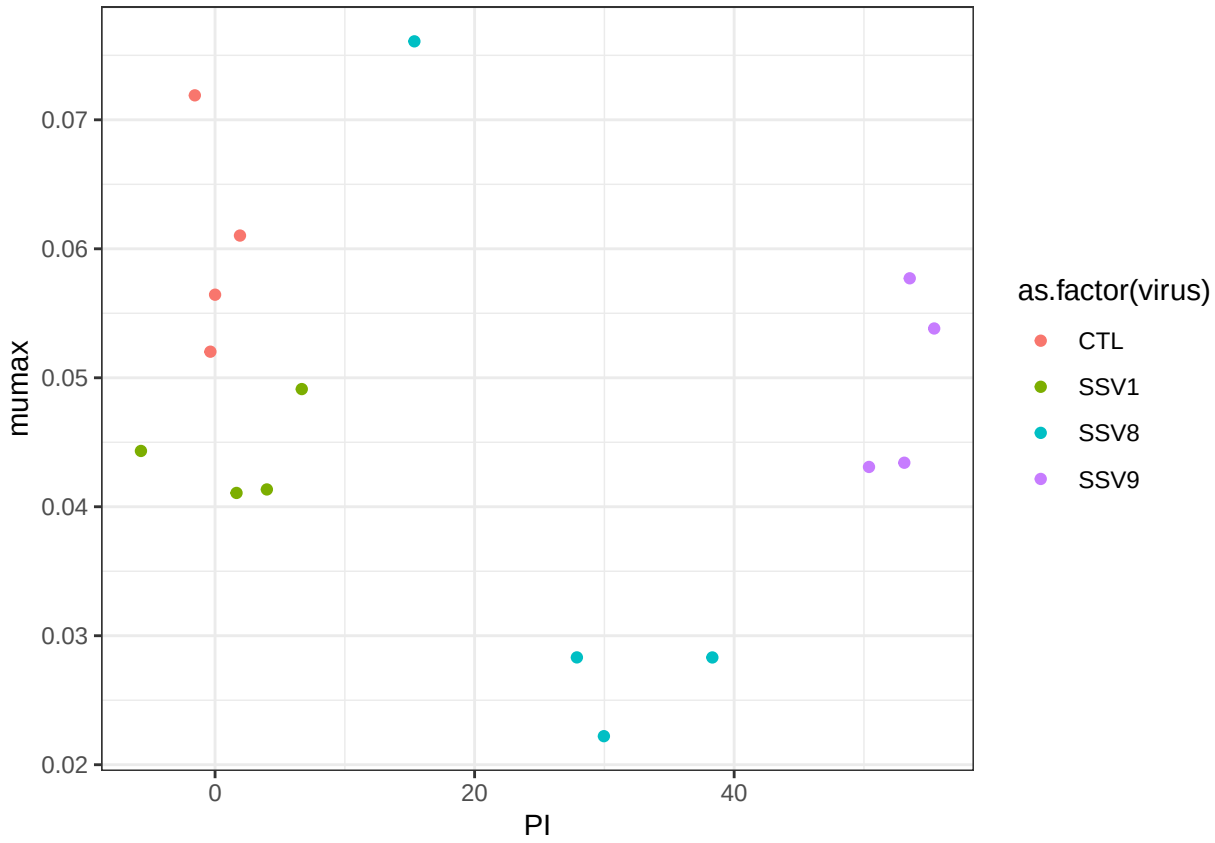

```
ggplot(data = gd, aes(x = Vr, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



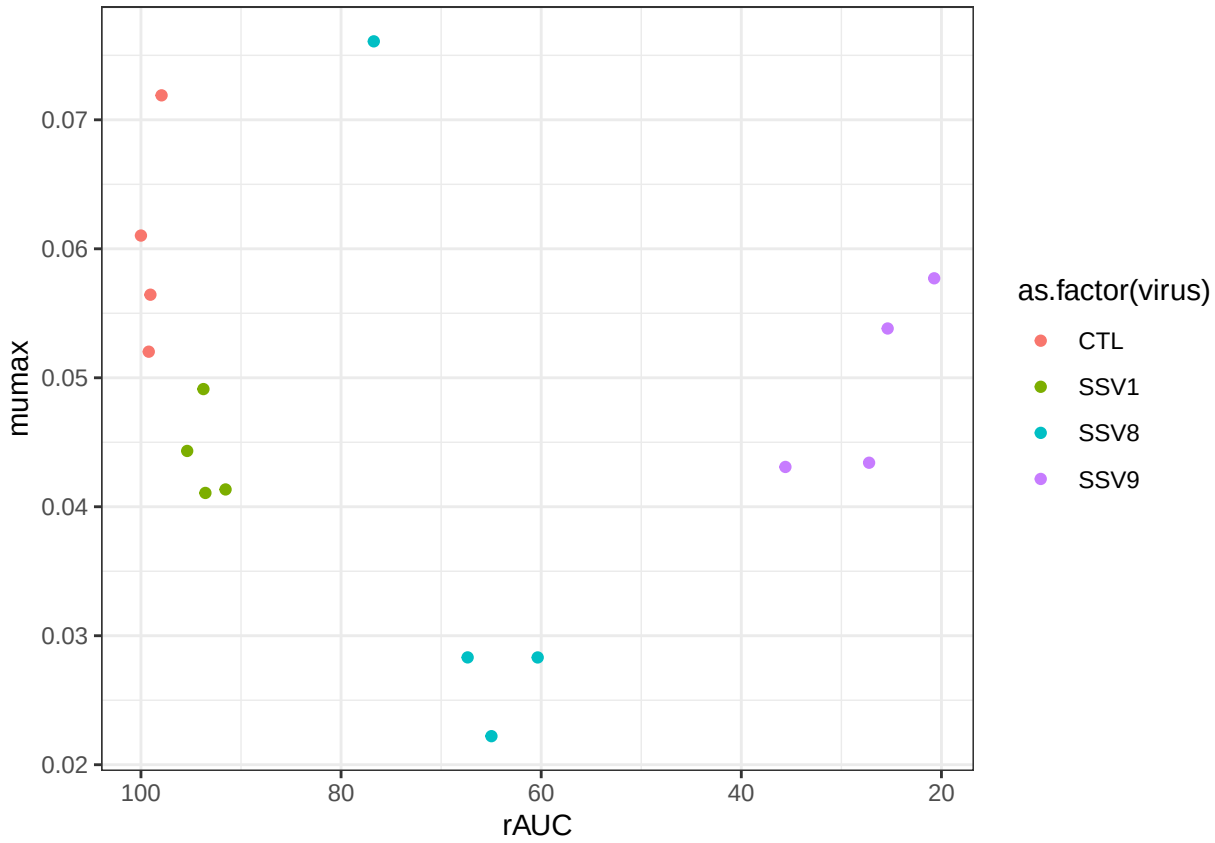

```
ggplot(data = gd, aes(y = PI, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



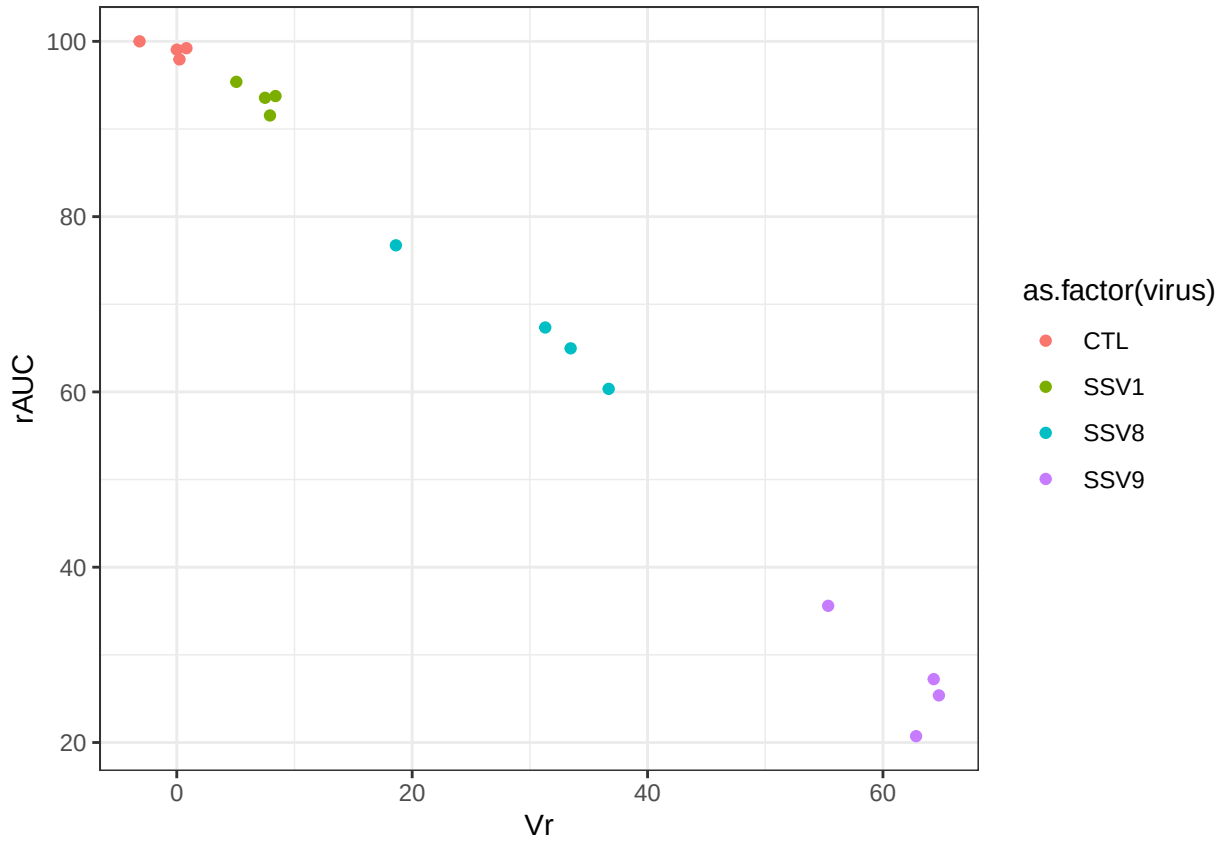

```
ggplot(data = gd, aes(y = mumax, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



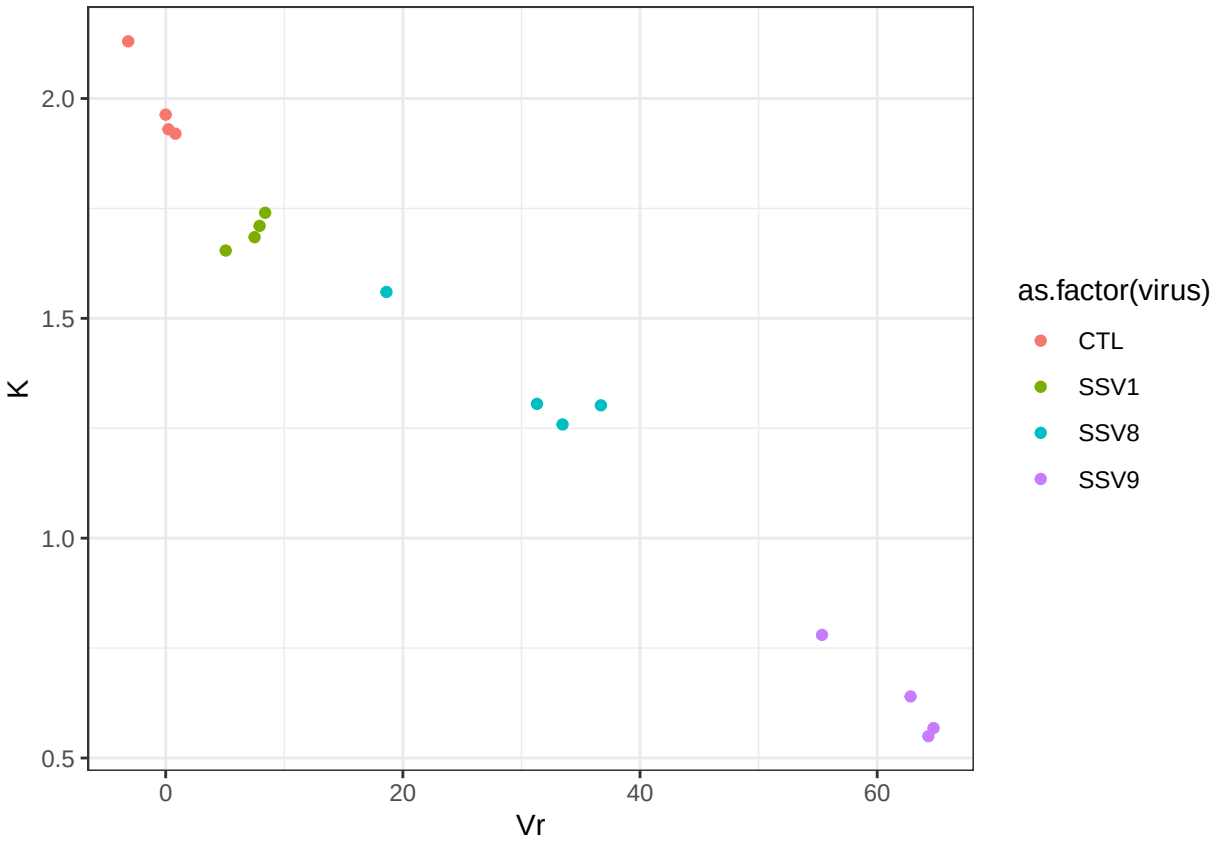

```
ggplot(data = gd, aes(y = mumax, x = rAUC, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



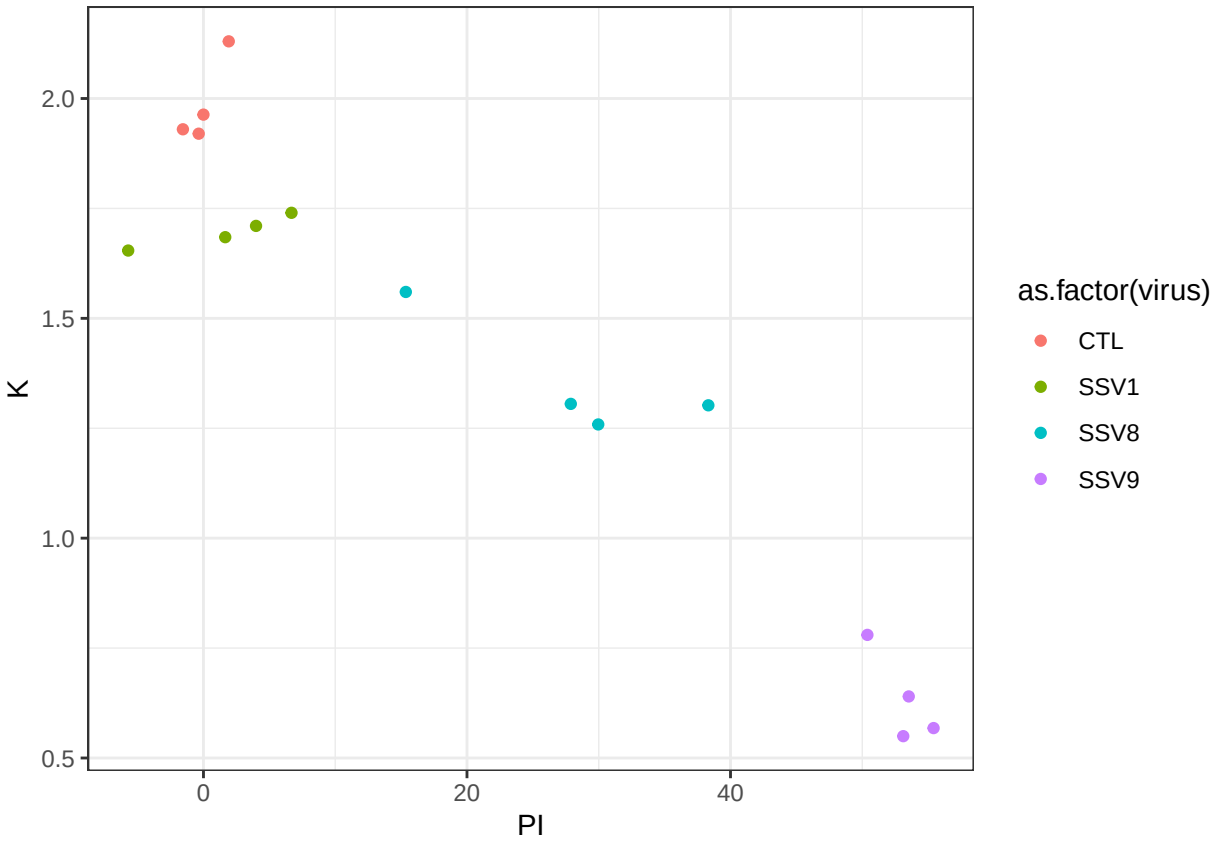

```
ggplot(data = gd, aes(y = rAUC, x = Vr, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



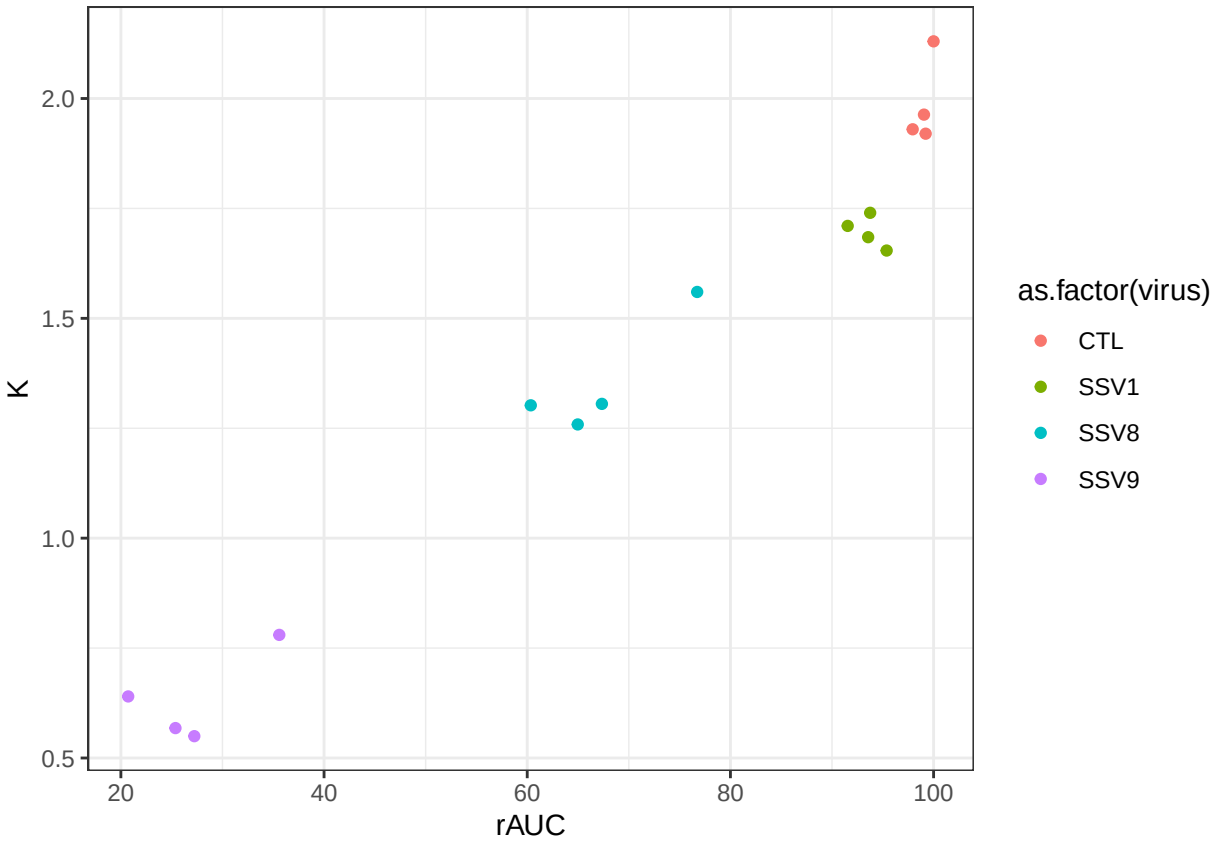

```
ggplot(data = gd, aes(y = K, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



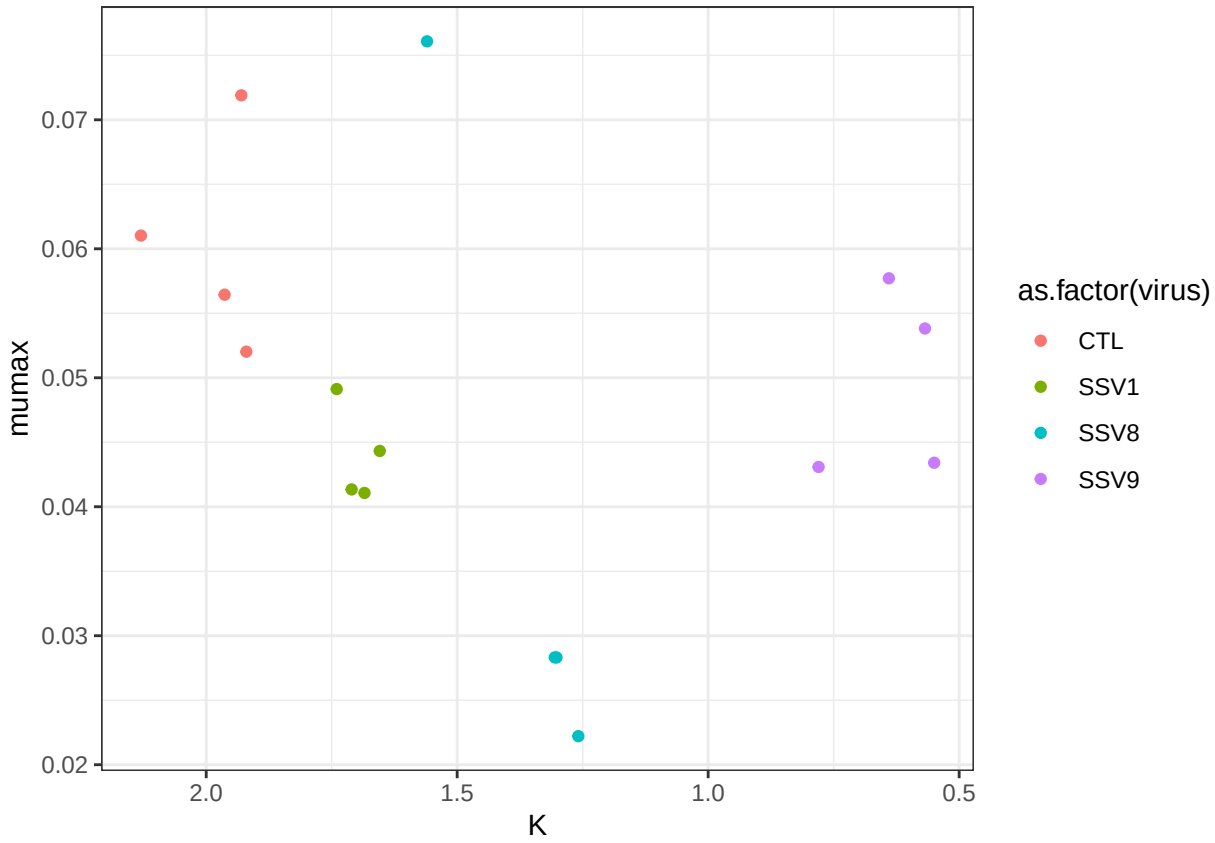

```
ggplot(data = gd, aes(y = K, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(y = K, x = rAUC, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(x = K, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



#now for other hosts (S444)

```
library(tidyverse)
library(DescTools)
library(MESS)
library(grofit)
library(readxl)

# Week_2_Growthcurver_S1_parameters <- read_excel("~/Downloads/Week 2 - Growthcurver S1 parameters.xlsx

# if (file.exists("FinalOutPutfigZ.Rda")) {
#   load("FinalOutPutfigZ.Rda")
# } else {

options(scipen = 999)

#IMPORT DATA HERE AS DATA.FRAME

# dat <- read_csv("~/Downloads/Supplemental_Table_S1_ready.csv")
dat <- read_excel(here("/Data/Processed/Sulfolobus/Sulfolobus_Infection_Growth_Curves.xlsx"), sheet = "S

dat <- as.data.frame(dat[complete.cases(dat),])

############3
####hard coded: remove!
# dat[1,11] <- 0.07
```

```
###########
##########

trim <- 15

colnames(dat)[1] <- "time"

colnames(dat)[-1] <- str_pad(colnames(dat)[-1], trim, pad = "0", side = "left" )




# maxbyrowraw <- colnames(dat[-1])[max.col(dat[-1],ties.method="random")]
# maxbyrowconvert <- as.data.frame(table(maxbyrowraw))
# maxbyrowcount <- arrange(maxbyrowconvert,-Freq)
# maxmax <- as.character(maxbyrowcount$maxbyrowraw[1])

#note: should always be 1 (first column)

timeColumn <- 1
#can be adjsuted. average of ctl replicates is recommended based on zero science (seems like a good sta
controlColumn <- 5
#shouldn't change unless time column != 1
a <- 2
#ADJUST ME: total number of curves in table + 1
b <- ncol(dat)
#What is the timepoint at which stationary phase is reached?
t_stationary <- 26

firstRun <- TRUE
c <- 2
d <- length(dat[[timeColumn]])
#d <- 20 #this is where 'stationary phase' would traditionally be considered fully reached
for (j in c:d) {


figZ <- gcFitSpline(dat[[timeColumn]], dat[[controlColumn]], gcID = "spline",
            control = grofit.control())
lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])
#upperbound <- dat[timeColumn[[j]]
#AUCraw <- AUC(dat[1:length(dat[[timeColumn]]),1], dat[1:length(dat[[controlColumn]]),controlColumn])
AUCraw <- AUC(dat[1:j,1], dat[1:j,controlColumn])
PI <- 0
IscZ <- 0

storage.vector_figZ <- data.frame( "mumax"= mumax, "K"= K, "lambda"= lambda, "UpperBound" = upperbound,


for (i in a:b) {
if(i != controlColumn) {
```

```
figZ <- gcFitSpline(dat$time, dat[[i]], gcID = "spline444",
            control = grofit.control())
#plot(figZ)

lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])

  AUCraw <- AUC(dat[1:j,1], dat[1:j,i])
  if(i == controlColumn) {
    PI <- 0
  } else {
    PI <- (1 - (AUCraw/storage.vector_figZ$AUC[1])) * 100
  }

  IscZ <-  (1 - sqrt((AUCraw*K)/(storage.vector_figZ$AUC[1]*storage.vector_figZ$K[1]))) * 100


storage.vector_figZ <- rbind(storage.vector_figZ, c( mumax, K, lambda, upperbound, as.numeric(AUCraw), I


}
}
#output <- rbind
if(firstRun == TRUE) {
  #something
  firstRun <- FALSE
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- storage.vector_figZ
} else {
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- rbind(FinalOutPutfigZ, storage.vector_figZ)
}

}
#storage.vector_figZ <- storage.vector_figZ[-c(1), ]

#storage.vector_figZ
#Below code to get rid of scientific notation:
options(scipen=999)
#here is how to get back to scientific notation: options(scipen=0)



library(data.table)
setDT(FinalOutPutfigZ, keep.rownames = TRUE)
#here I'm fixing the group names to not say replicates
# FinalOutPutfigZ$rn <- c(str_replace_all(string=FinalOutPutfigZ$rn,pattern="\\l.*$",replacement="l"))
FinalOutPutfigZ$rn <- substr(FinalOutPutfigZ$rn, 1, trim)
# df$col1 <- strtrim(df$col, 1, 1)

IscOrderedOutputFigZ <- FinalOutPutfigZ %>%
```

```
  arrange((Isc))



if (exists("maxIsc") == TRUE) {
  save(FinalOutPutfigZ,file="FinalOutPutfigZ.Rda")
} else {
maxIsc <- IscOrderedOutputFigZ[1,1]
}
# }

# for writing a data.frame or list of data.frames to an xlsx file
#write.xlsx(FinalOutPutfigZ, 'Isc_figZISC.xlsx')
#FinalOutPUtfigZsave <- FinalOutPutfigZ

IscOrderedOutputFigZ
```

```
##                    rn       mumax          K    lambda UpperBound      AUC
##   1: 0000000S444CTL3 0.048760650 1.6210964 27.619754          2  0.26000
##   2: 0000000S444CTL3 0.048760650 1.6210964 27.619754         20  6.50000
##   3: 0000000S444CTL3 0.048760650 1.6210964 27.619754         26 10.82000
##   4: 0000000S444CTL3 0.048760650 1.6210964 27.619754          8  1.46000
##   5: 0000000S444CTL3 0.048760650 1.6210964 27.619754         14  3.35000
##  ---
## 252: 0000S444SSV9AVG 0.012765338 0.3966745 -6.610997        104 30.51333
## 253: 000000S444SSV9B 0.009123167 0.3724942 -7.734721         86 24.01000
## 254: 000000S444SSV9B 0.009123167 0.3724942 -7.734721         92 25.12000
## 255: 000000S444SSV9B 0.009123167 0.3724942 -7.734721         98 26.32000
## 256: 000000S444SSV9B 0.009123167 0.3724942 -7.734721        104 27.28000
##             PI       Isc
##   1: -16.41791 -9.291345
##   2: -14.24889 -8.268435
##   3: -13.77099 -8.041756
##   4: -12.02046 -7.207347
##   5: -11.02519 -6.730030
##  ---
## 252:  71.97183 73.473131
## 253:  71.02574 73.864105
## 254:  72.66495 74.614181
## 255:  73.79704 75.145421
## 256:  74.94182 75.694422
```

```
gdIsc <- FinalOutPutfigZ %>%
        group_by(rn, UpperBound) %>%
        #group_by(UpperBound) %>%
        summarise(Isc = mean(Isc))
```

```
## `summarise()` regrouping output by 'rn' (override with `.groups` argument)
```

```
gdVr <- gdIsc %>%
  group_by(rn) %>%
  summarize(Vr = auc(UpperBound, Isc, type = "spline")/(max(UpperBound) - min(UpperBound)))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
gdPI <- FinalOutPutfigZ %>%
        group_by(rn) %>%
        summarise(PI = mean(PI))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
AUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == max(FinalOutPutfigZ$UpperBound)) %>%
    mutate(rAUC = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select(c(-1,-4, -5, -6, -7, -8))
AUCfigZ
```

```
##           mumax         K      rAUC
##  1: 0.042978219 1.5800000  96.01082
##  2: 0.035676320 1.5400000  93.18282
##  3: 0.048760650 1.6210964 100.00000
##  4: 0.058982484 1.5700000  86.62404
##  5: 0.049059869 1.3200000  81.01949
##  6: 0.038416912 1.3728120  84.40780
##  7: 0.030059456 1.1701166  70.38628
##  8: 0.016340701 1.1708469  71.83790
##  9: 0.019710960 1.2220591  72.89884
## 10: 0.016136072 0.4481003  28.87380
## 11: 0.009123167 0.3724942  24.05856
## 12: 0.045202446 0.5200000  27.79787
## 13: 0.039646808 1.6300000  94.84963
## 14: 0.043317430 1.4033333  84.01711
## 15: 0.016820273 1.1822568  71.70768
## 16: 0.012765338 0.3966745  26.91007
```

```
trimAUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == t_stationary) %>%
    mutate(AUCtrim = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select("AUCtrim")


gd <- cbind(gdVr, gdPI, AUCfigZ, trimAUCfigZ)
gd <- gd[-3]
gd
```

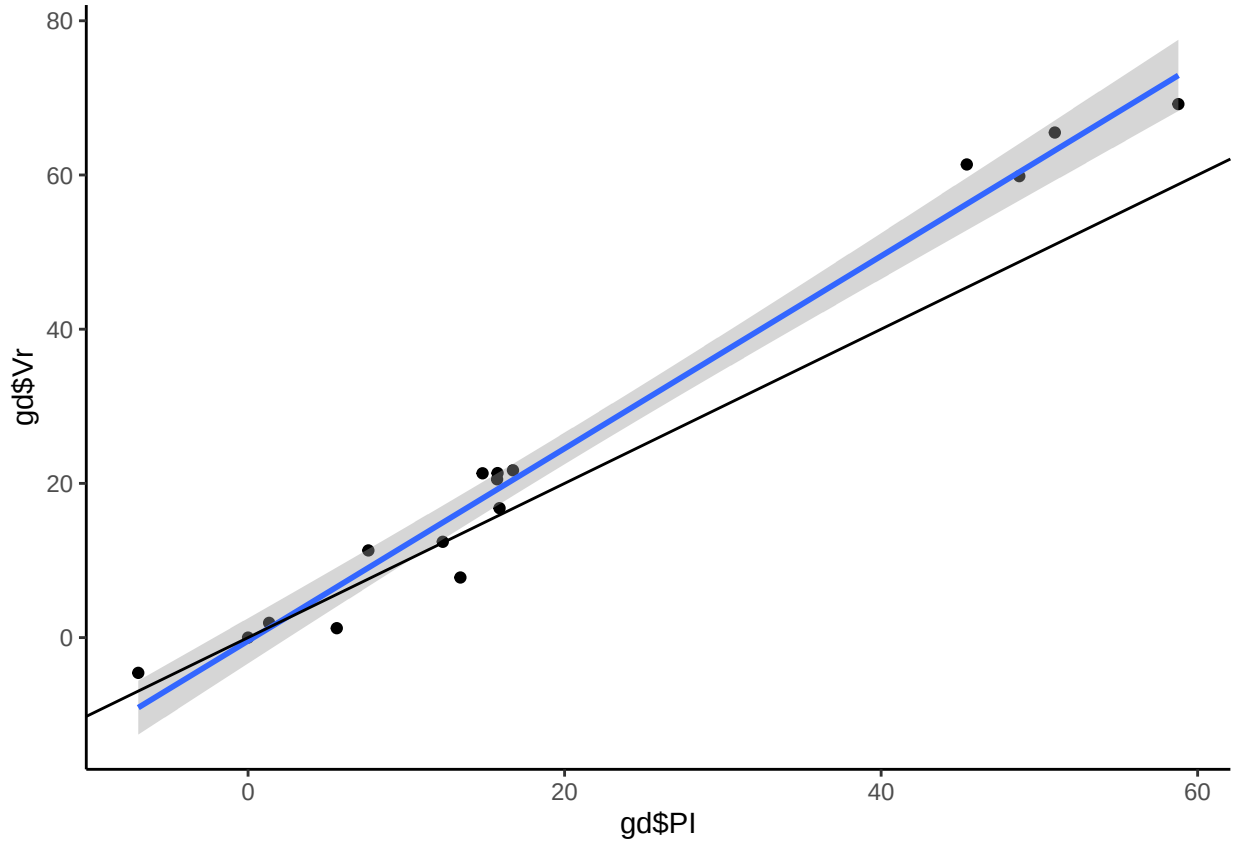
```
##                 rn        Vr        PI       mumax         K      rAUC
## 1  0000000S444CTL1  0.000000  0.000000 0.042978219 1.5800000  96.01082
## 2  0000000S444CTL2  1.909756  1.330773 0.035676320 1.5400000  93.18282
## 3  0000000S444CTL3 -4.575079 -6.936766 0.048760650 1.6210964 100.00000
## 4  000000S444SSV1A  7.794972 13.416733 0.058982484 1.5700000  86.62404
## 5  000000S444SSV1B 16.785329 15.890541 0.049059869 1.3200000  81.01949
## 6  000000S444SSV1C 11.305930  7.602623 0.038416912 1.3728120  84.40780
## 7  000000S444SSV8A 21.696435 16.739947 0.030059456 1.1701166  70.38628
```

```
## 8  000000S444SSV8B 21.309246 14.816789 0.016340701 1.1708469  71.83790
## 9  000000S444SSV8C 20.533240 15.735636 0.019710960 1.2220591  72.89884
## 10 000000S444SSV9A 61.363512 45.412939 0.016136072 0.4481003  28.87380
## 11 000000S444SSV9B 69.192399 58.782478 0.009123167 0.3724942  24.05856
## 12 000000S444SSV9C 59.846017 48.730875 0.045202446 0.5200000  27.79787
## 13 00000S444CTLAVG  1.220295  5.605993 0.039646808 1.6300000  94.84963
## 14 0000S444SSV1AVG 12.425523 12.303299 0.043317430 1.4033333  84.01711
## 15 0000S444SSV8AVG 21.333706 15.764124 0.016820273 1.1822568  71.70768
## 16 0000S444SSV9AVG 65.505047 50.975431 0.012765338 0.3966745  26.91007
##      AUCtrim
## 1   87.89587
## 2   86.05360
## 3  100.00000
## 4   76.50647
## 5   70.24030
## 6   77.63401
## 7   81.87616
## 8   77.22736
## 9   73.75231
## 10  64.91682
## 11  43.25323
## 12  69.87061
## 13  77.63401
## 14  74.79359
## 15  77.61861
## 16  59.34689
```

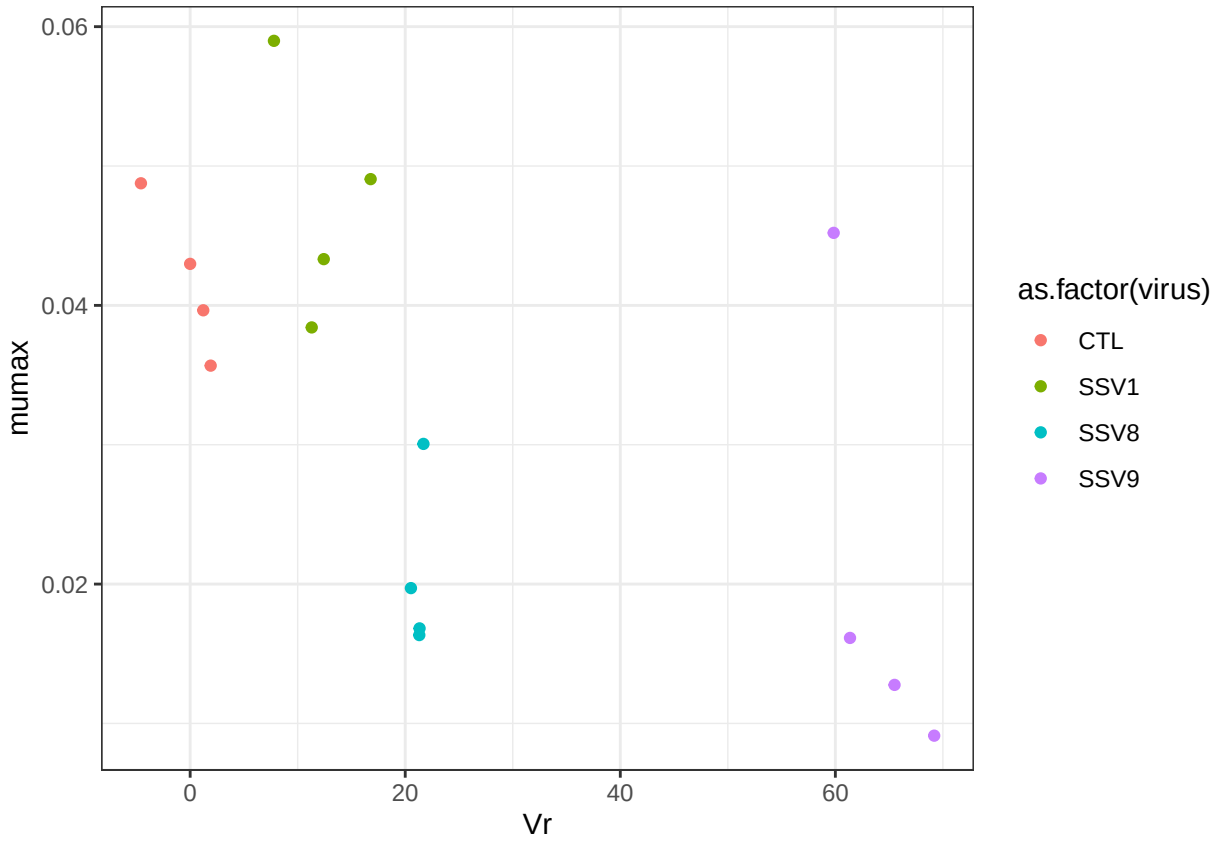
```r
gd$virus <- c(
    rep("CTL", 3),
    rep("SSV1", 3),
    rep("SSV8", 3),
    rep("SSV9", 3),
    rep("CTL", 1),
    rep("SSV1", 1),
    rep("SSV8", 1),
    rep("SSV9", 1)
)

# gd$virus <- c(
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
#     rep("rosR", 12),
```

```
#     rep("ura3_3", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12)
# )

gd$host <- c(
    rep("S444", 16)
)

gd444 <- gd

ggplot(data = NULL, aes(x = gd$PI, y = gd$Vr)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_abline() +
  theme_classic()
```

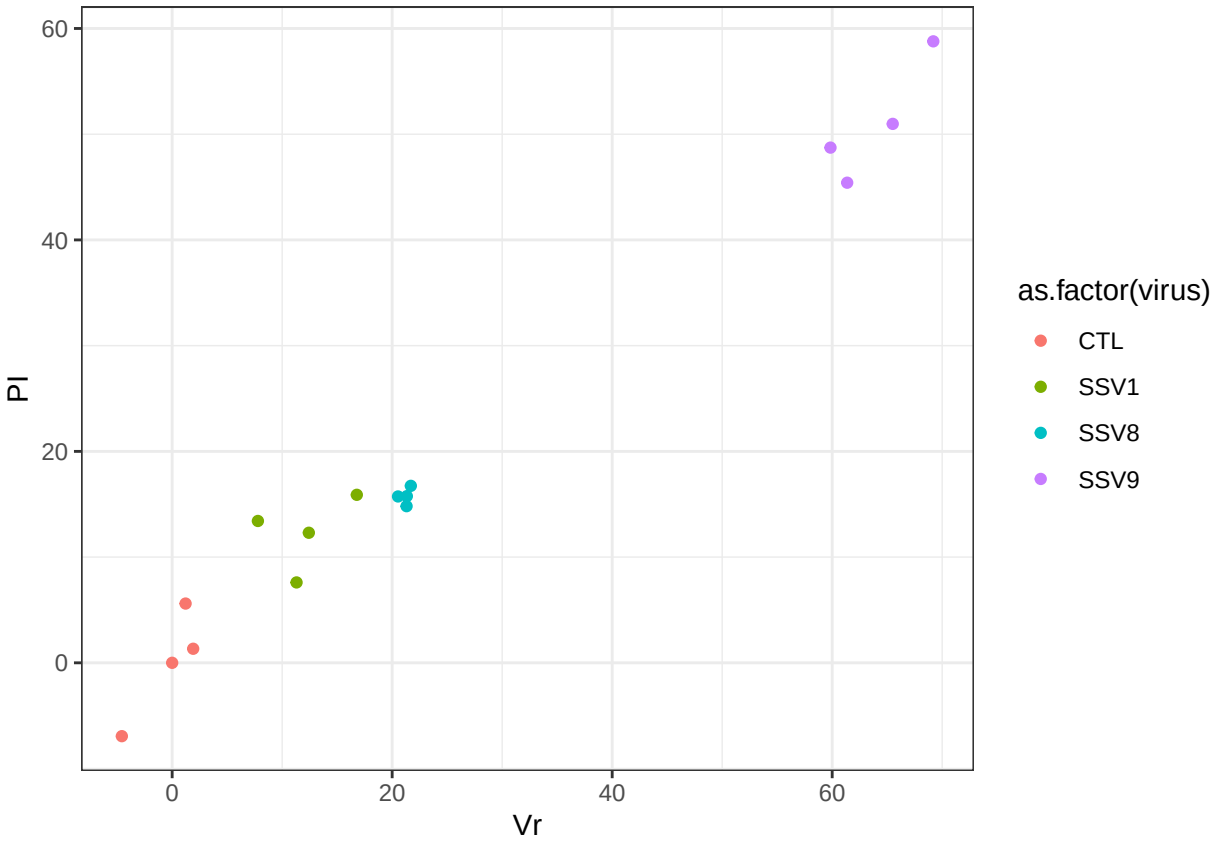
```
## `geom_smooth()` using formula 'y ~ x'
```



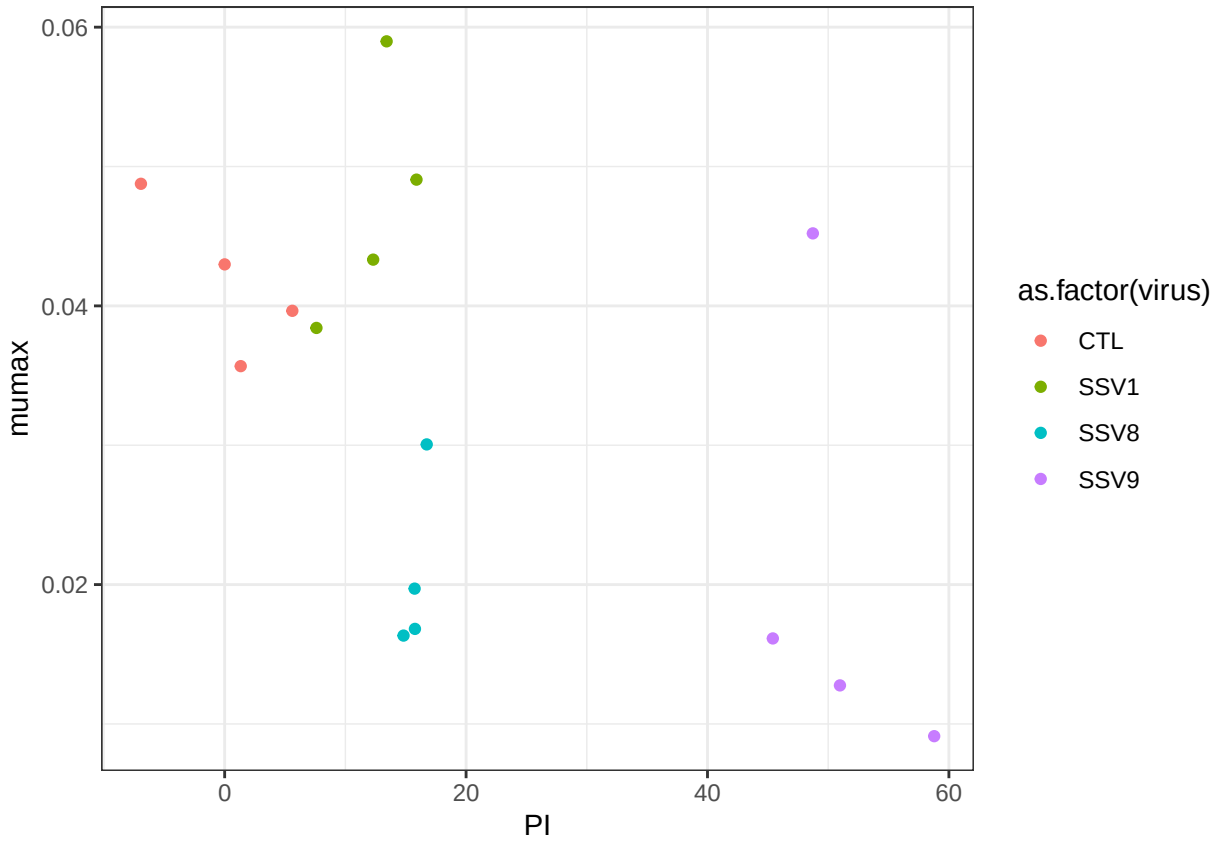
```
ggplot(data = gd, aes(x = Vr, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



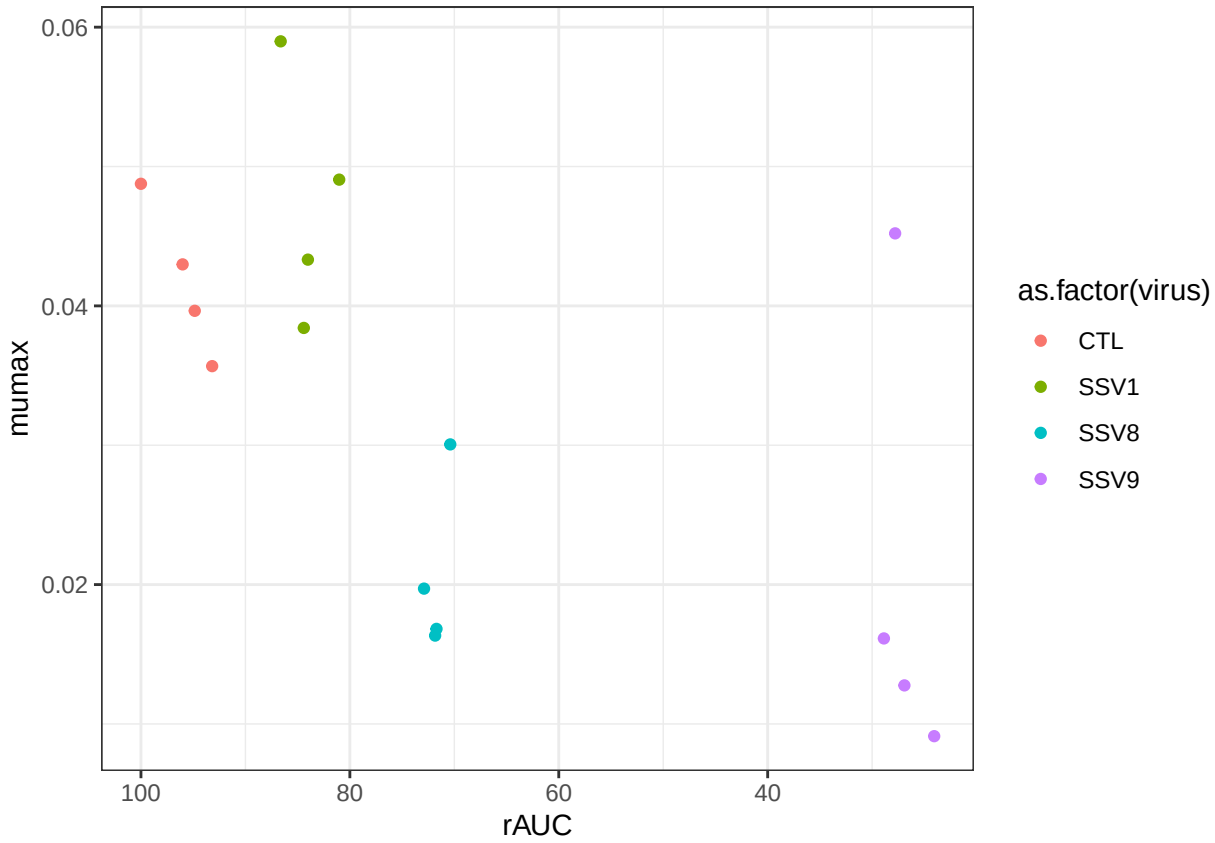

```
ggplot(data = gd, aes(y = PI, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



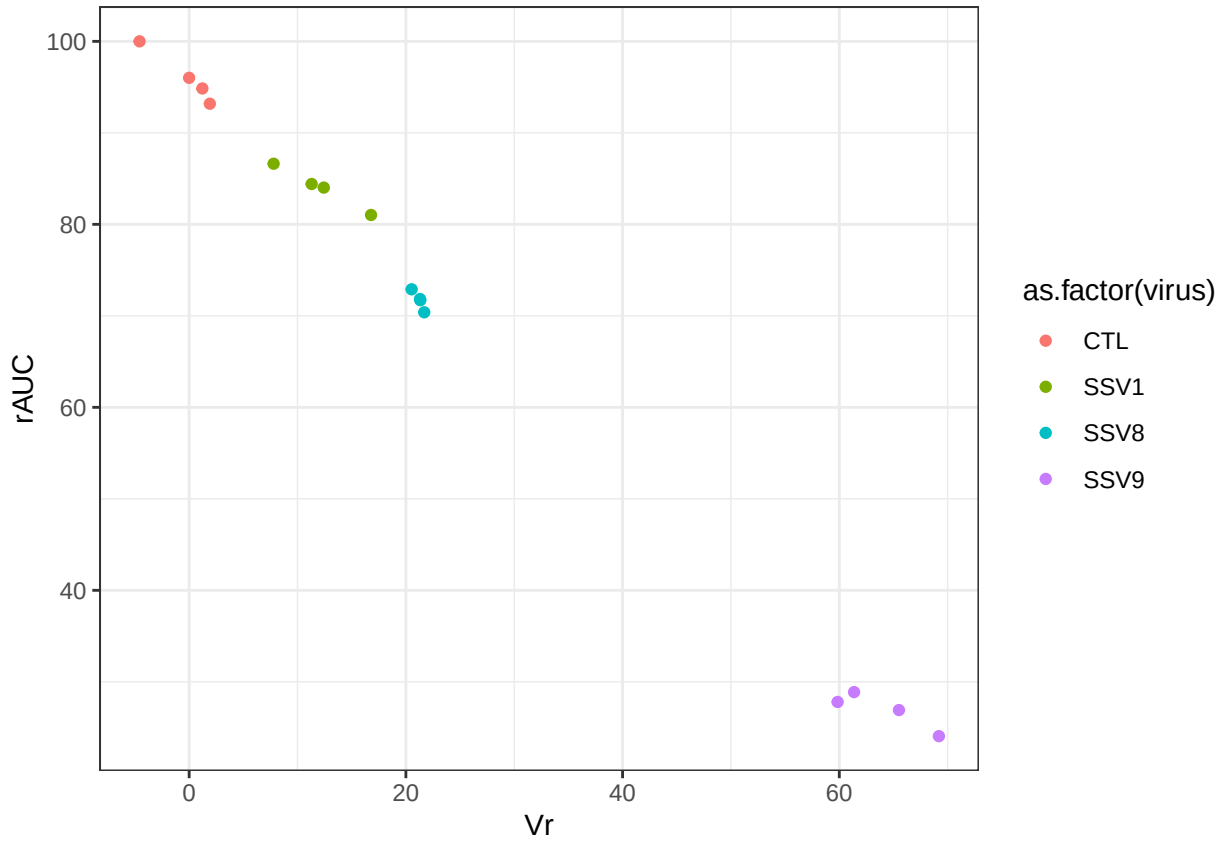

```
ggplot(data = gd, aes(y = mumax, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



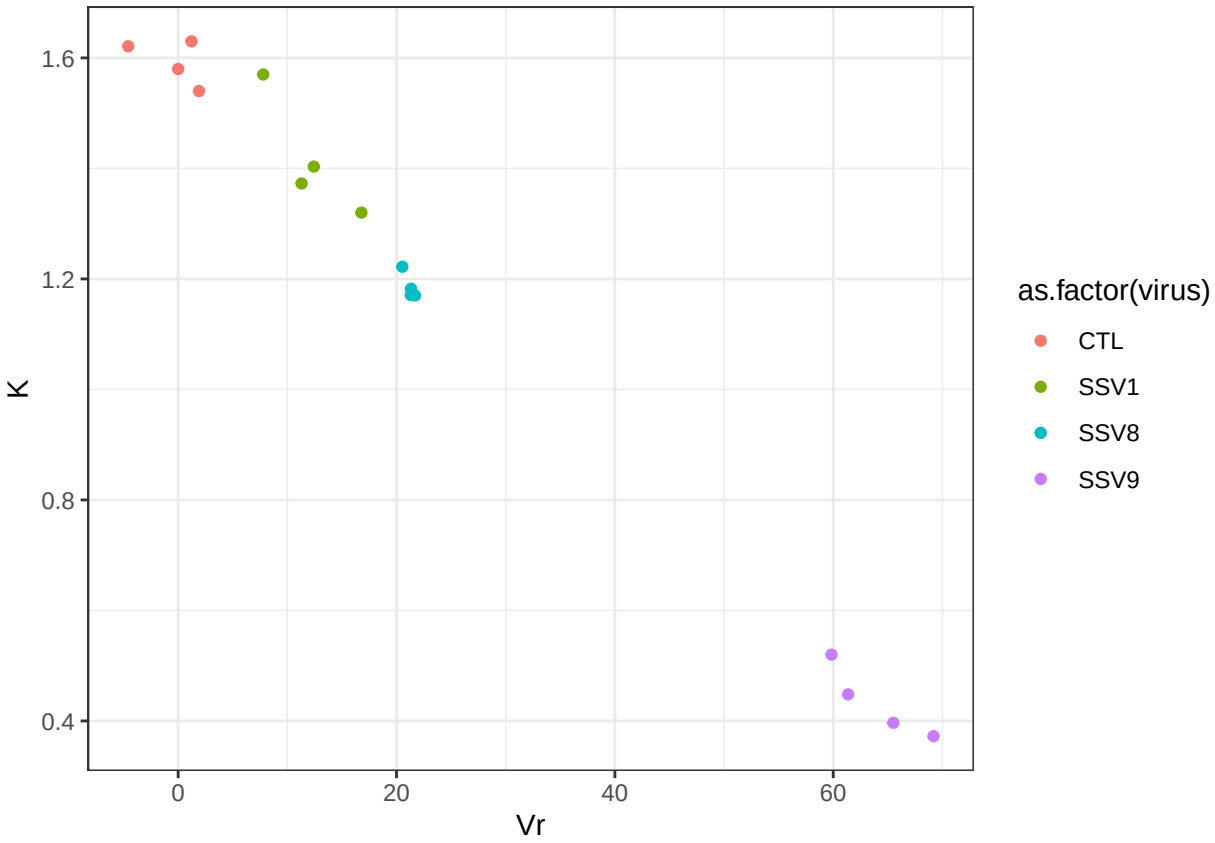

```
ggplot(data = gd, aes(y = mumax, x = rAUC, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



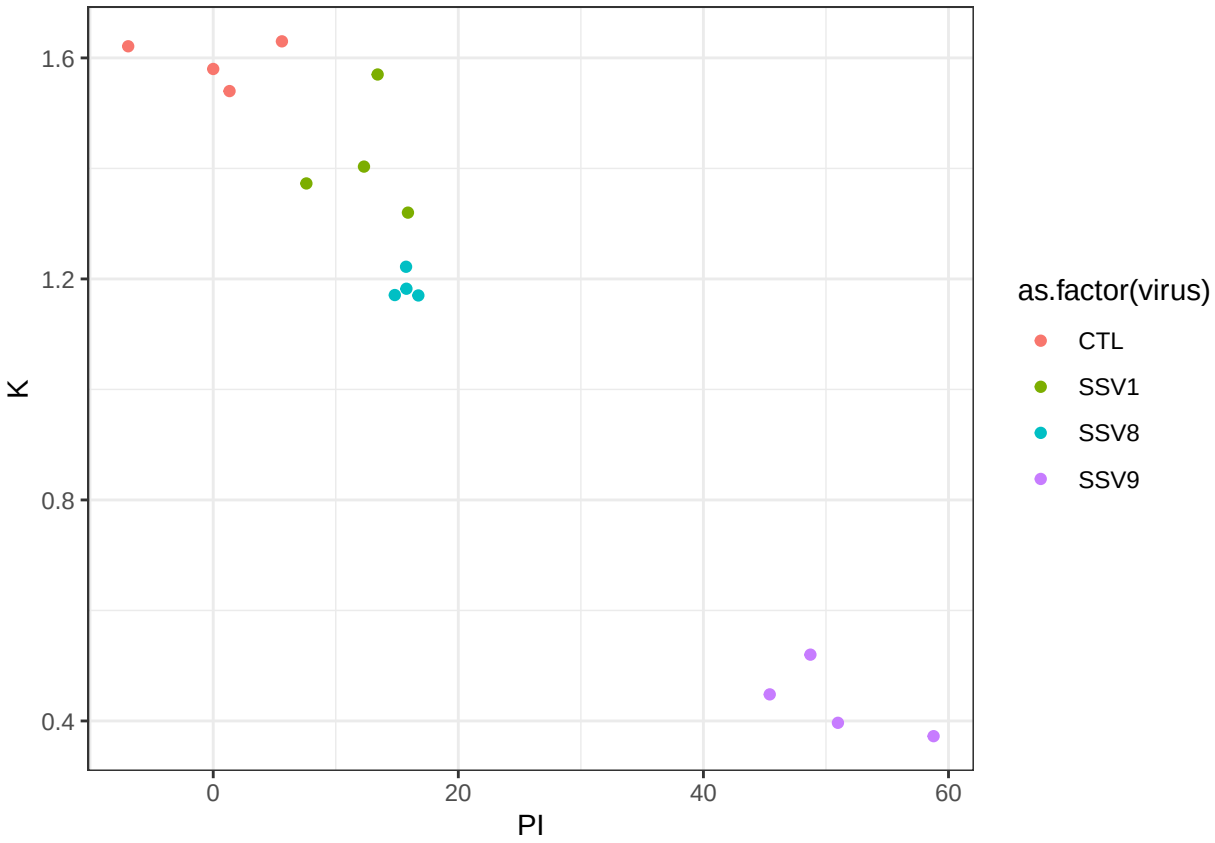

```
ggplot(data = gd, aes(y = rAUC, x = Vr, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



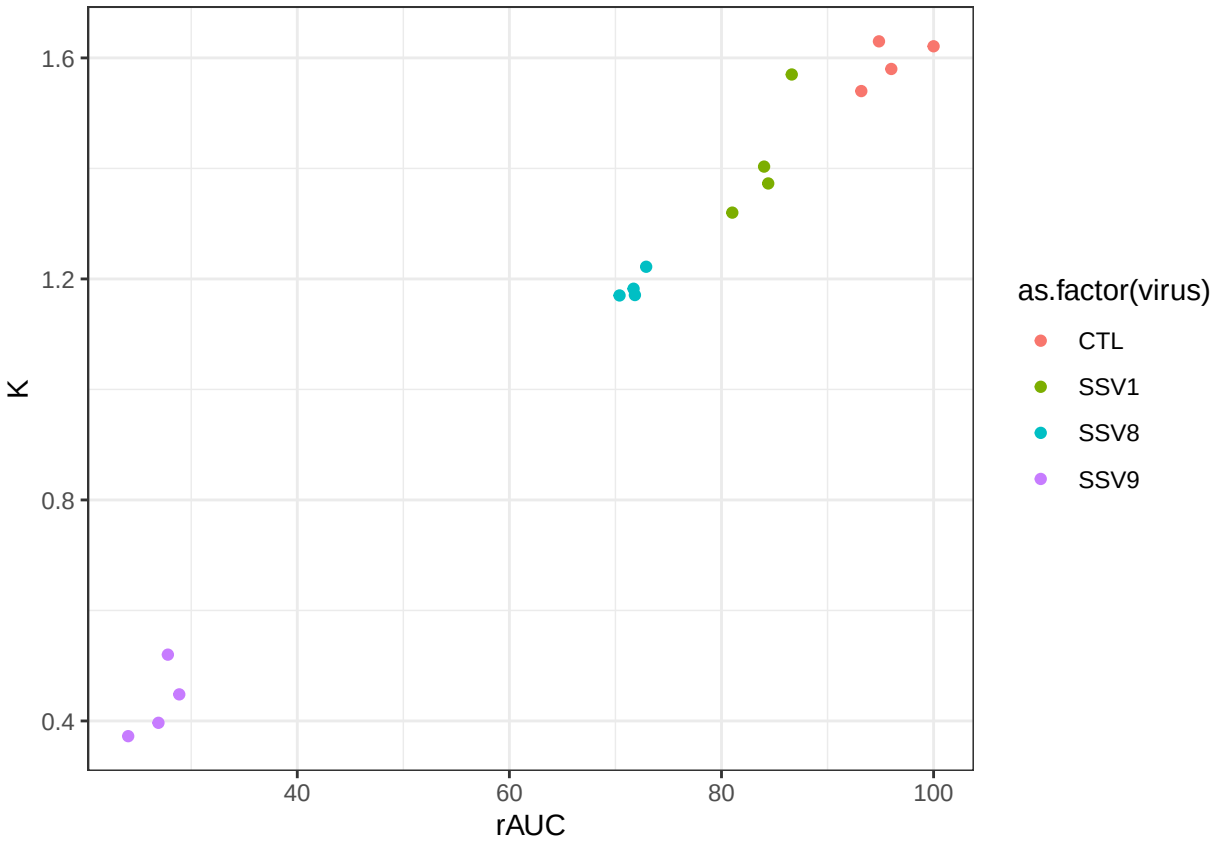

```
ggplot(data = gd, aes(y = K, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



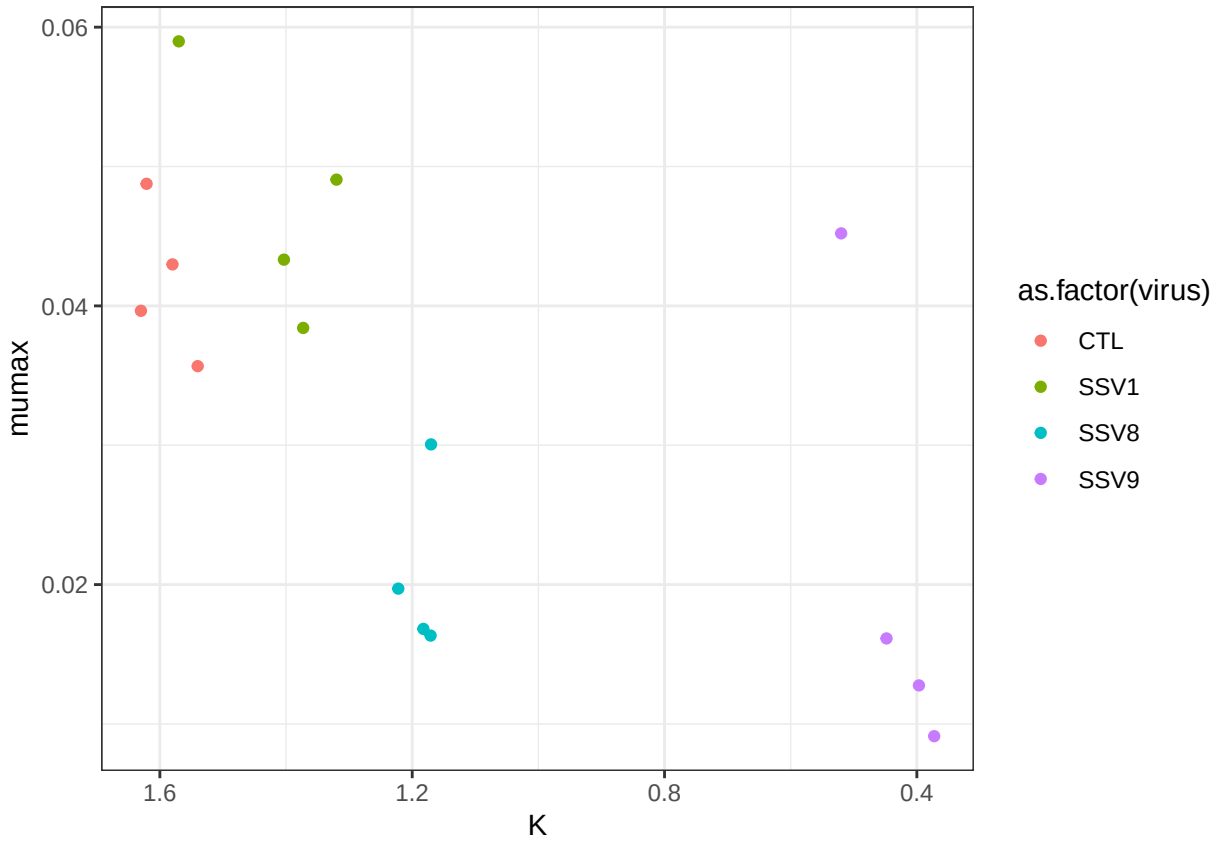

```
ggplot(data = gd, aes(y = K, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(y = K, x = rAUC, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(x = K, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



NOw for S200

```
library(tidyverse)
library(DescTools)
library(MESS)
library(grofit)
library(readxl)

# Week_2_Growthcurver_S1_parameters <- read_excel("~/Downloads/Week 2 - Growthcurver S1 parameters.xlsx

# if (file.exists("FinalOutPutfigZ.Rda")) {
#   load("FinalOutPutfigZ.Rda")
# } else {

options(scipen = 999)

#IMPORT DATA HERE AS DATA.FRAME

# dat <- read_csv("~/Downloads/Supplemental_Table_S1_ready.csv")
dat <- read_excel(here("/Data/Processed/Sulfolobus/Sulfolobus_Infection_Growth_Curves.xlsx"), sheet = "

dat <- as.data.frame(dat[complete.cases(dat),])

############3
####hard coded: remove!
# dat[1,11] <- 0.07
```

```
###########
##########

trim <- 15

colnames(dat)[1] <- "time"

colnames(dat)[-1] <- str_pad(colnames(dat)[-1], trim, pad = "0", side = "left" )




# maxbyrowraw <- colnames(dat[-1])[max.col(dat[-1],ties.method="random")]
# maxbyrowconvert <- as.data.frame(table(maxbyrowraw))
# maxbyrowcount <- arrange(maxbyrowconvert,-Freq)
# maxmax <- as.character(maxbyrowcount$maxbyrowraw[1])

#note: should always be 1 (first column)

timeColumn <- 1
#can be adjsuted. average of ctl replicates is recommended based on zero science (seems like a good sta
controlColumn <- 5
#shouldn't change unless time column != 1
a <- 2
#ADJUST ME: total number of curves in table + 1
b <- ncol(dat)
#What is the timepoint at which stationary phase is reached?
t_stationary <- 26

firstRun <- TRUE
c <- 2
d <- length(dat[[timeColumn]])
#d <- 20 #this is where 'stationary phase' would traditionally be considered fully reached
for (j in c:d) {


figZ <- gcFitSpline(dat[[timeColumn]], dat[[controlColumn]], gcID = "spline",
            control = grofit.control())
lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])
#upperbound <- dat[timeColumn[[j]]
#AUCraw <- AUC(dat[1:length(dat[[timeColumn]]),1], dat[1:length(dat[[controlColumn]]),controlColumn])
AUCraw <- AUC(dat[1:j,1], dat[1:j,controlColumn])
PI <- 0
IscZ <- 0

storage.vector_figZ <- data.frame( "mumax"= mumax, "K"= K, "lambda"= lambda, "UpperBound" = upperbound,


for (i in a:b) {
if(i != controlColumn) {
```

```
figZ <- gcFitSpline(dat$time, dat[[i]], gcID = "spline444",
            control = grofit.control())
#plot(figZ)

lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])

  AUCraw <- AUC(dat[1:j,1], dat[1:j,i])
  if(i == controlColumn) {
    PI <- 0
  } else {
    PI <- (1 - (AUCraw/storage.vector_figZ$AUC[1])) * 100
  }

  IscZ <-  (1 - sqrt((AUCraw*K)/(storage.vector_figZ$AUC[1]*storage.vector_figZ$K[1]))) * 100


storage.vector_figZ <- rbind(storage.vector_figZ, c( mumax, K, lambda, upperbound, as.numeric(AUCraw), I


}
}
#output <- rbind
if(firstRun == TRUE) {
  #something
  firstRun <- FALSE
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- storage.vector_figZ
} else {
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- rbind(FinalOutPutfigZ, storage.vector_figZ)
}

}
#storage.vector_figZ <- storage.vector_figZ[-c(1), ]

#storage.vector_figZ
#Below code to get rid of scientific notation:
options(scipen=999)
#here is how to get back to scientific notation: options(scipen=0)



library(data.table)
setDT(FinalOutPutfigZ, keep.rownames = TRUE)
#here I'm fixing the group names to not say replicates
# FinalOutPutfigZ$rn <- c(str_replace_all(string=FinalOutPutfigZ$rn,pattern="\\l.*$",replacement="l"))
FinalOutPutfigZ$rn <- substr(FinalOutPutfigZ$rn, 1, trim)
# df$col1 <- strtrim(df$col, 1, 1)

IscOrderedOutputFigZ <- FinalOutPutfigZ %>%
```

```
  arrange((Isc))



if (exists("maxIsc") == TRUE) {
  save(FinalOutPutfigZ,file="FinalOutPutfigZ.Rda")
} else {
maxIsc <- IscOrderedOutputFigZ[1,1]
}
# }

# for writing a data.frame or list of data.frames to an xlsx file
#write.xlsx(FinalOutPutfigZ, 'Isc_figZISC.xlsx')
#FinalOutPUtfigZsave <- FinalOutPutfigZ

IscOrderedOutputFigZ
```

```
##                    rn      mumax         K    lambda UpperBound     AUC
##   1: 0000000S200CTL2 0.05128115 1.8800000 -1.755031         54  39.174
##   2: 0000000S200CTL2 0.05128115 1.8800000 -1.755031          2   0.276
##   3: 0000000S200CTL2 0.05128115 1.8800000 -1.755031         48  32.181
##   4: 0000000S200CTL2 0.05128115 1.8800000 -1.755031        104 115.104
##   5: 0000000S200CTL2 0.05128115 1.8800000 -1.755031         60  47.034
##  ---
## 252: 000000S200SSV9A 0.01818307 0.4237080 -8.157385         98  30.413
## 253: 000000S200SSV9B 0.02326762 0.4270855 -4.962609         92  27.176
## 254: 000000S200SSV9A 0.01818307 0.4237080 -8.157385        104  31.883
## 255: 000000S200SSV9B 0.02326762 0.4270855 -4.962609         98  28.226
## 256: 000000S200SSV9B 0.02326762 0.4270855 -4.962609        104  29.246
##             PI       Isc
##   1: -2.049287 -7.290072
##   2: -1.970443 -7.248618
##   3: -1.689506 -7.100776
##   4: -1.460892 -6.980319
##   5: -1.415922 -6.956608
##  ---
## 252: 70.611957 72.666561
## 253: 71.106165 72.789555
## 254: 71.896045 73.270388
## 255: 72.725252 73.562924
## 256: 74.220485 74.297793
```

```
gdIsc <- FinalOutPutfigZ %>%
        group_by(rn, UpperBound) %>%
        #group_by(UpperBound) %>%
        summarise(Isc = mean(Isc))
```

```
## `summarise()` regrouping output by 'rn' (override with `.groups` argument)
```

```
gdVr <- gdIsc %>%
  group_by(rn) %>%
  summarize(Vr = auc(UpperBound, Isc, type = "spline")/(max(UpperBound) - min(UpperBound)))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
gdPI <- FinalOutPutfigZ %>%
        group_by(rn) %>%
        summarise(PI = mean(PI))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
AUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == max(FinalOutPutfigZ$UpperBound)) %>%
    mutate(rAUC = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select(c(-1,-4, -5, -6, -7, -8))
AUCfigZ
```

```
##          mumax         K      rAUC
##  1: 0.05955619 1.6666667  95.34615
##  2: 0.05128115 1.8800000  96.73906
##  3: 0.06723610 1.6400000  95.90869
##  4: 0.03585643 1.6304217 100.00000
##  5: 0.05217504 1.3573429  87.07389
##  6: 0.03598114 1.2809373  83.85329
##  7: 0.04471916 1.1800000  64.70954
##  8: 0.01894459 1.0039085  62.52942
##  9: 0.01763715 1.0573824  61.91589
## 10: 0.01818307 0.4237080  26.79604
## 11: 0.02326762 0.4270855  24.57978
## 12: 0.01396722 0.4483852  30.58983
## 13: 0.03195441 1.5659880  93.39071
## 14: 0.03544997 1.4025556  90.30906
## 15: 0.03129484 1.0736659  63.05161
## 16: 0.05959694 0.4793333  26.35004
```

```
trimAUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == t_stationary) %>%
    mutate(AUCtrim = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select("AUCtrim")

gd <- cbind(gdVr, gdPI, AUCfigZ, trimAUCfigZ)
gd <- gd[-3]
gd
```

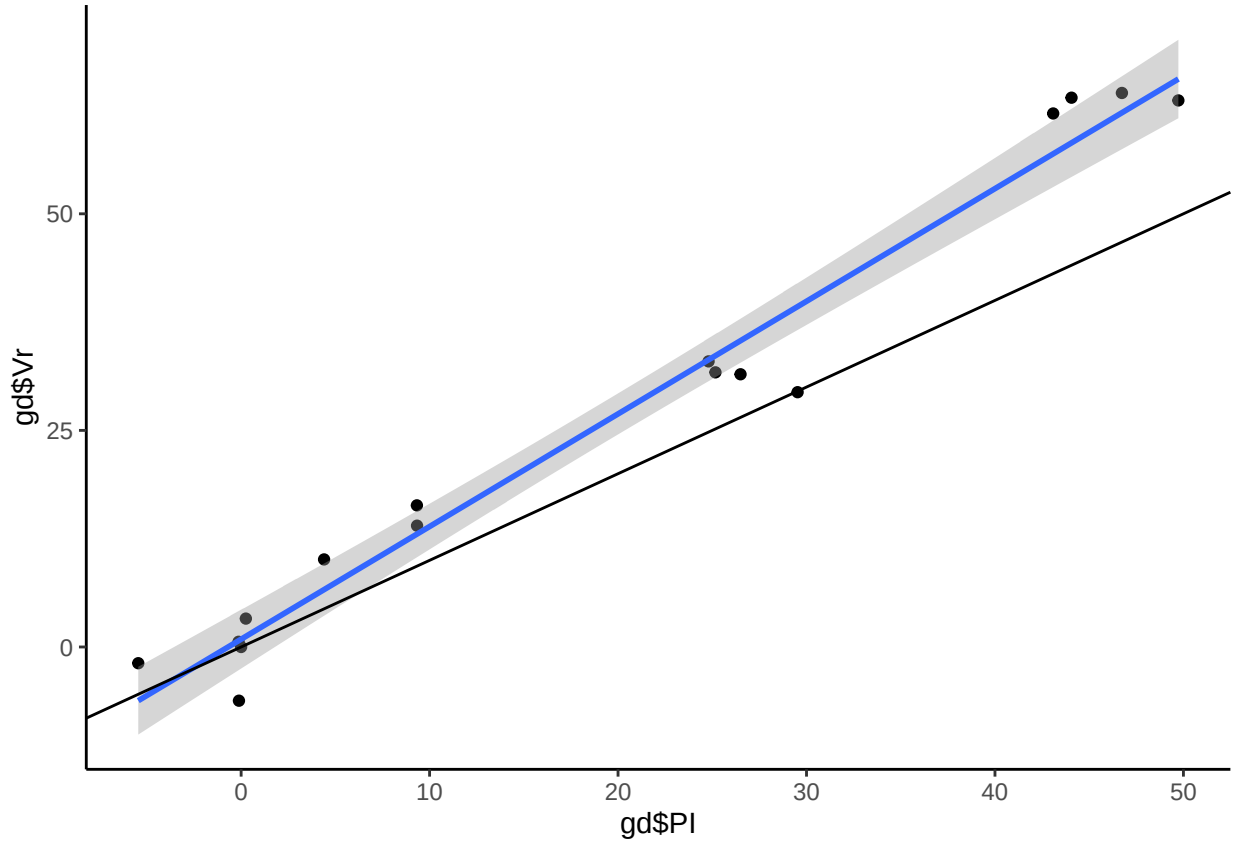
```
##                 rn         Vr         PI      mumax         K      rAUC
## 1  0000000S200CTL1  0.0000000  0.0000000 0.05955619 1.6666667  95.34615
## 2  0000000S200CTL2 -6.2007514 -0.1151505 0.05128115 1.8800000  96.73906
## 3  0000000S200CTL3  0.6004346 -0.1355918 0.06723610 1.6400000  95.90869
## 4  000000S200SSV1A -1.8722095 -5.4577265 0.03585643 1.6304217 100.00000
## 5  000000S200SSV1B 13.9993161  9.3354097 0.05217504 1.3573429  87.07389
## 6  000000S200SSV1C 16.3446366  9.3263081 0.03598114 1.2809373  83.85329
## 7  000000S200SSV8A 29.4087915 29.5266767 0.04471916 1.1800000  64.70954
## 8  000000S200SSV8B 32.9627220 24.8023613 0.01894459 1.0039085  62.52942
```

```
## 9  000000S200SSV8C 31.6993356 25.1671947 0.01763715 1.0573824  61.91589
## 10 000000S200SSV9A 63.4007321 44.0623662 0.01818307 0.4237080  26.79604
## 11 000000S200SSV9B 63.9470428 46.7360350 0.02326762 0.4270855  24.57978
## 12 000000S200SSV9C 61.5791212 43.0896278 0.01396722 0.4483852  30.58983
## 13 00000S200CTLAVG  3.2768196  0.2507422 0.03195441 1.5659880  93.39071
## 14 0000S200SSV1AVG 10.1073424  4.4013304 0.03544997 1.4025556  90.30906
## 15 0000S200SSV8AVG 31.4762400 26.4987442 0.03129484 1.0736659  63.05161
## 16 0000S200SSV9AVG 63.0834456 49.7323624 0.05959694 0.4793333  26.35004
##       AUCtrim
## 1   92.17088
## 2   89.51686
## 3   94.49127
## 4  100.00000
## 5   75.76008
## 6   77.52860
## 7   64.29862
## 8   69.61168
## 9   68.63335
## 10  64.00512
## 11  59.84347
## 12  59.27152
## 13  92.50452
## 14  84.42956
## 15  67.51455
## 16  51.09874
```

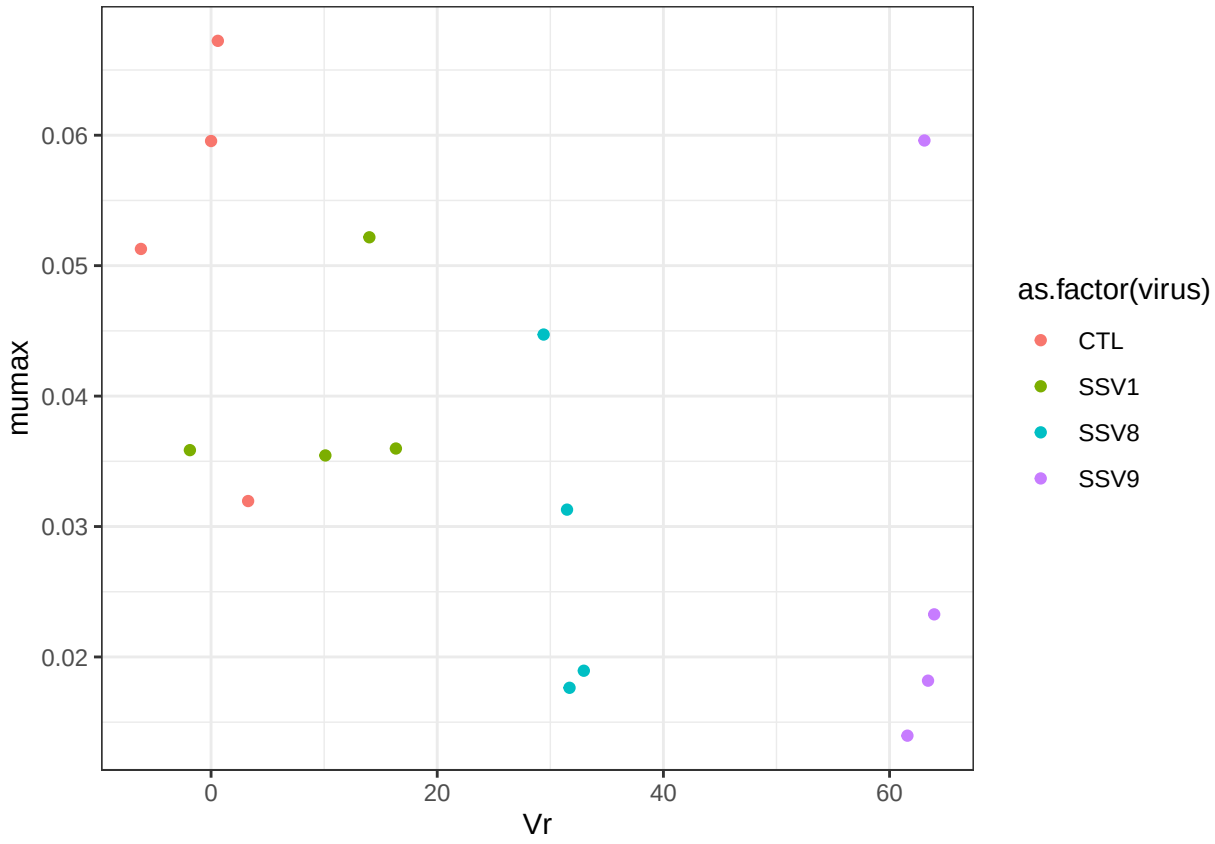
```
gd$virus <- c(
    rep("CTL", 3),
    rep("SSV1", 3),
    rep("SSV8", 3),
    rep("SSV9", 3),
    rep("CTL", 1),
    rep("SSV1", 1),
    rep("SSV8", 1),
    rep("SSV9", 1)
)

# gd$virus <- c(
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
```

```
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12)
# )

gd$host <- c(
    rep("S200", 16)
)


gd200 <- gd

ggplot(data = NULL, aes(x = gd$PI, y = gd$Vr)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_abline() +
  theme_classic()
```

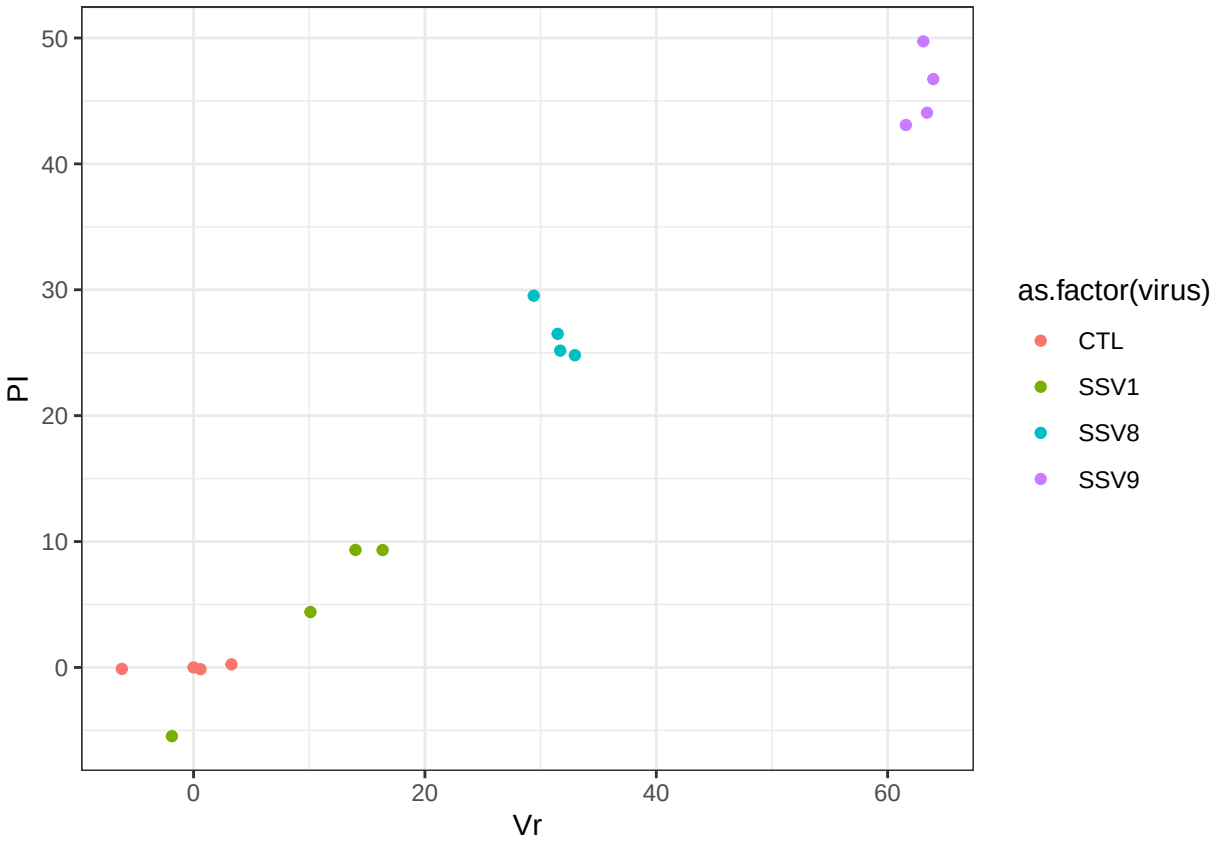
```
## `geom_smooth()` using formula 'y ~ x'
```



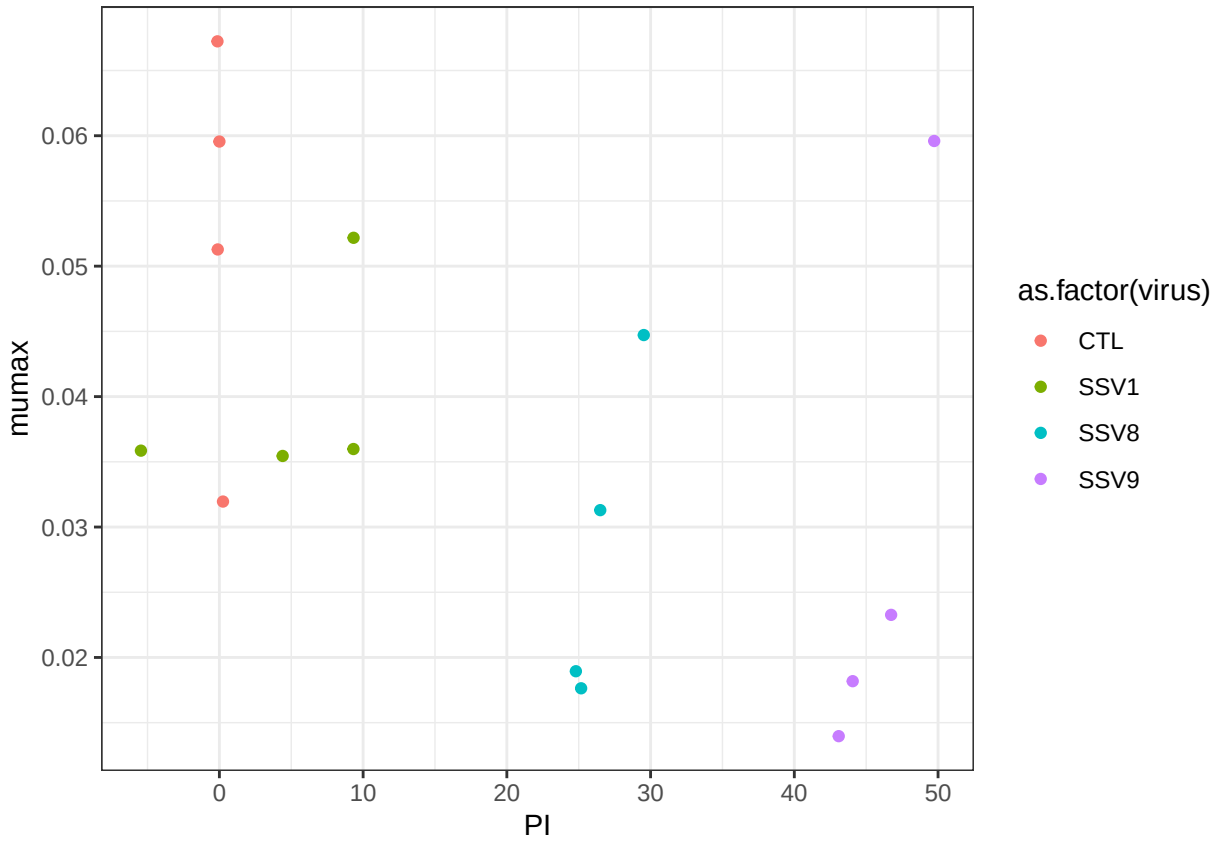
```
ggplot(data = gd, aes(x = Vr, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



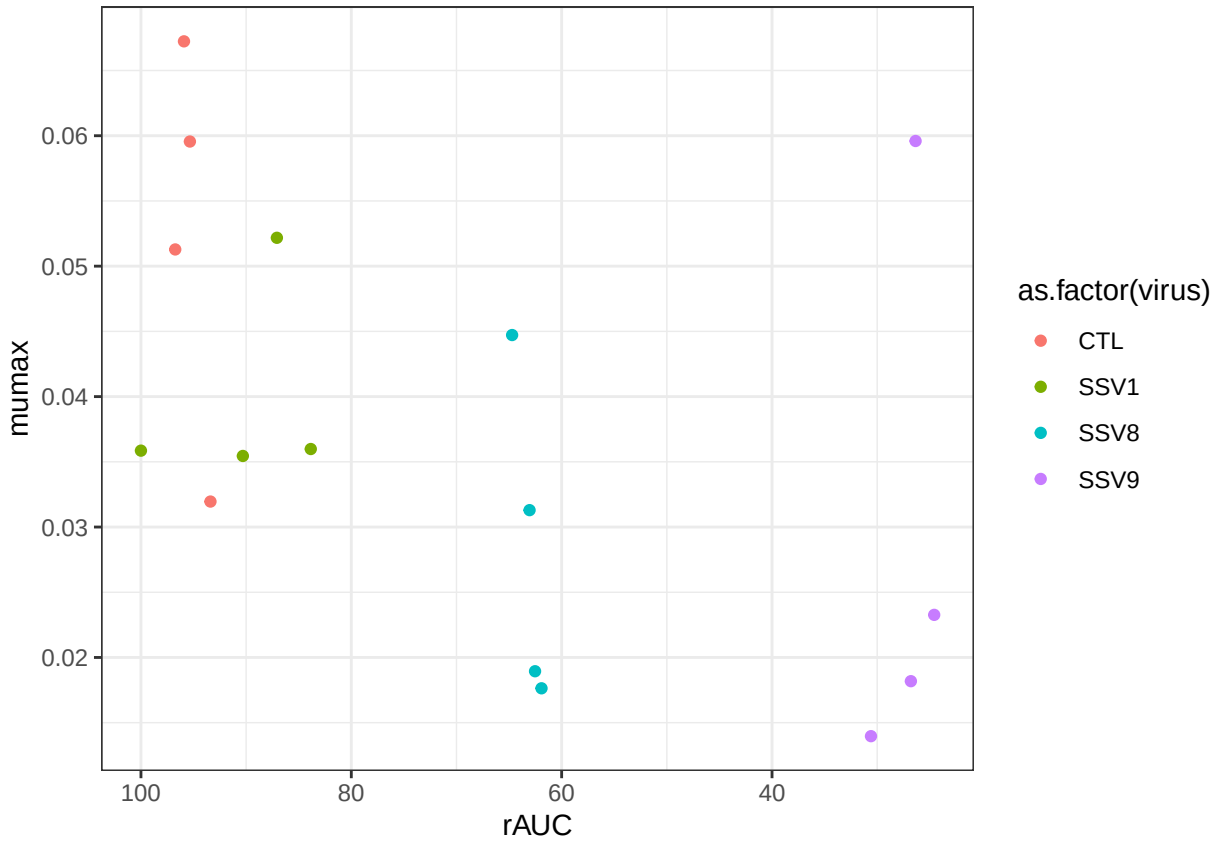

```
ggplot(data = gd, aes(y = PI, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



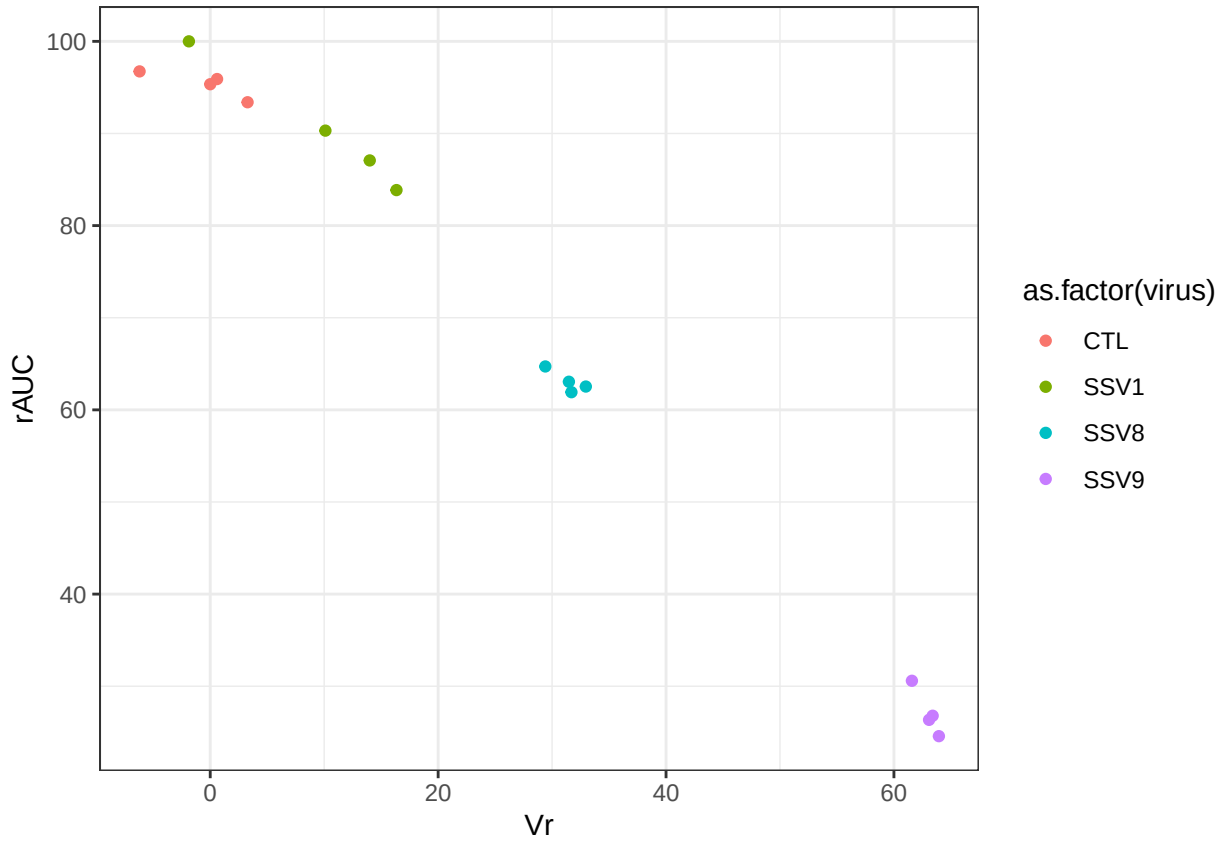

```
ggplot(data = gd, aes(y = mumax, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



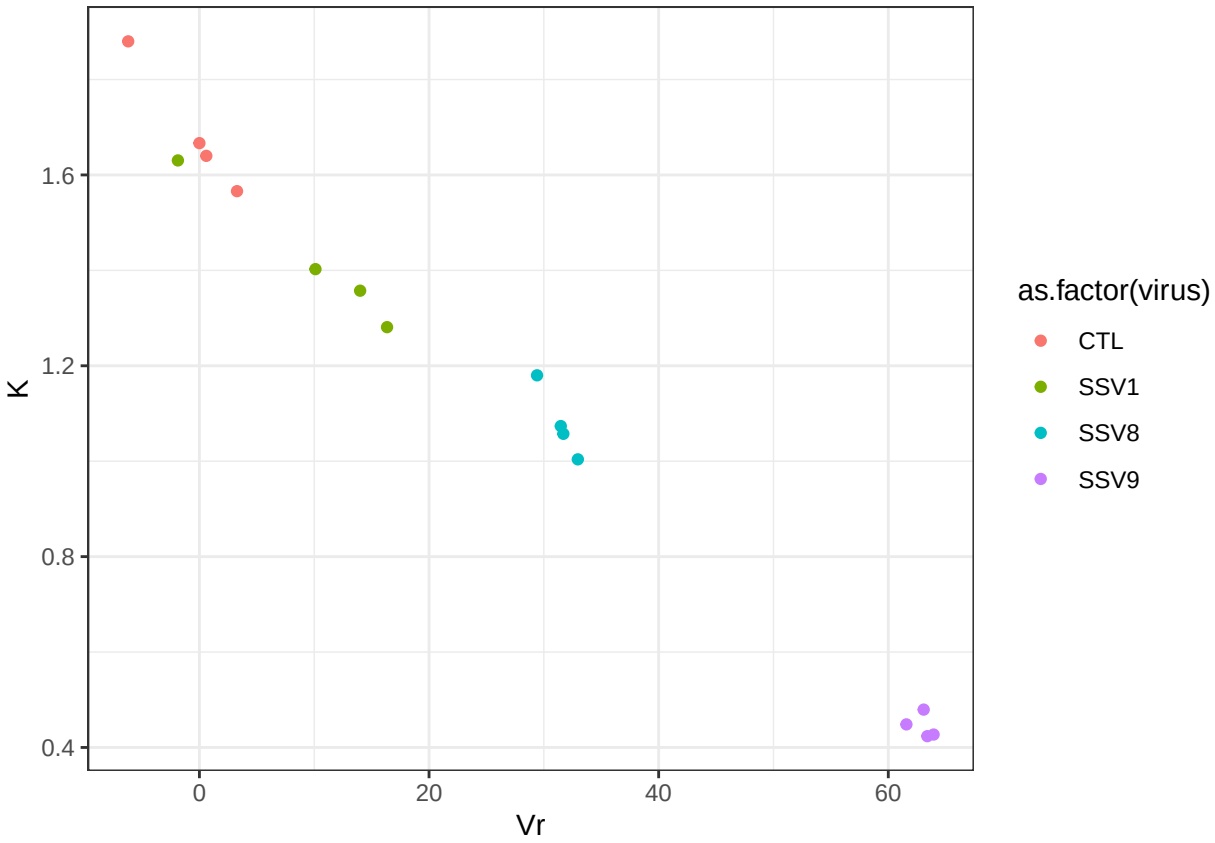

```
ggplot(data = gd, aes(y = mumax, x = rAUC, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



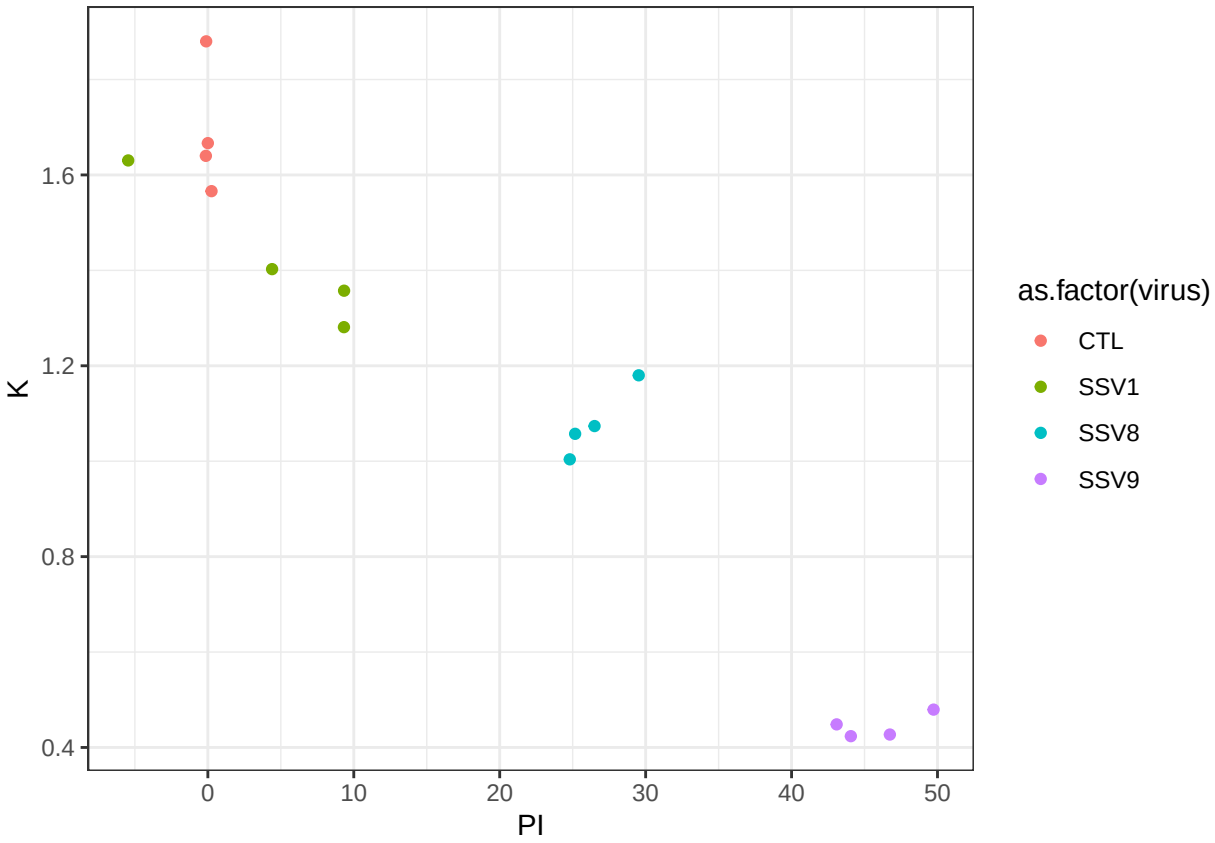

```
ggplot(data = gd, aes(y = rAUC, x = Vr, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



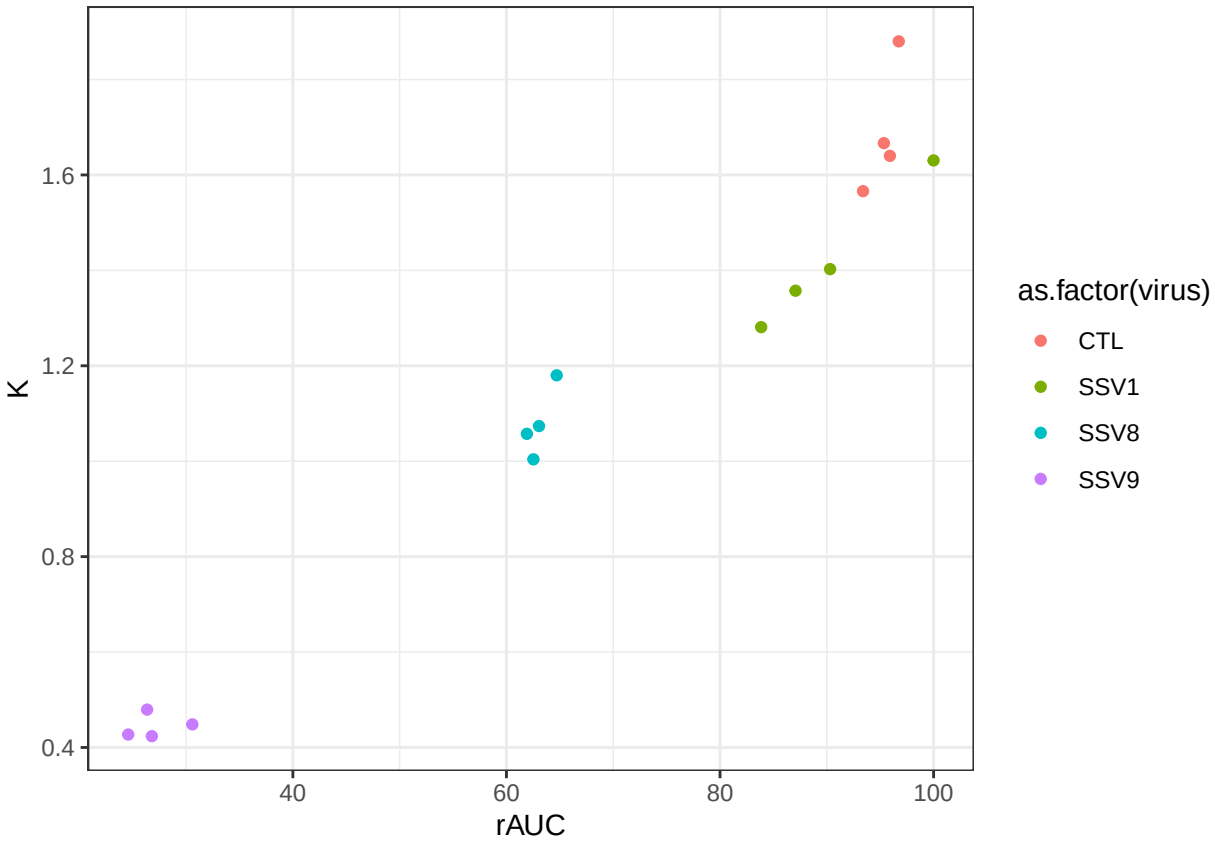

```
ggplot(data = gd, aes(y = K, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```



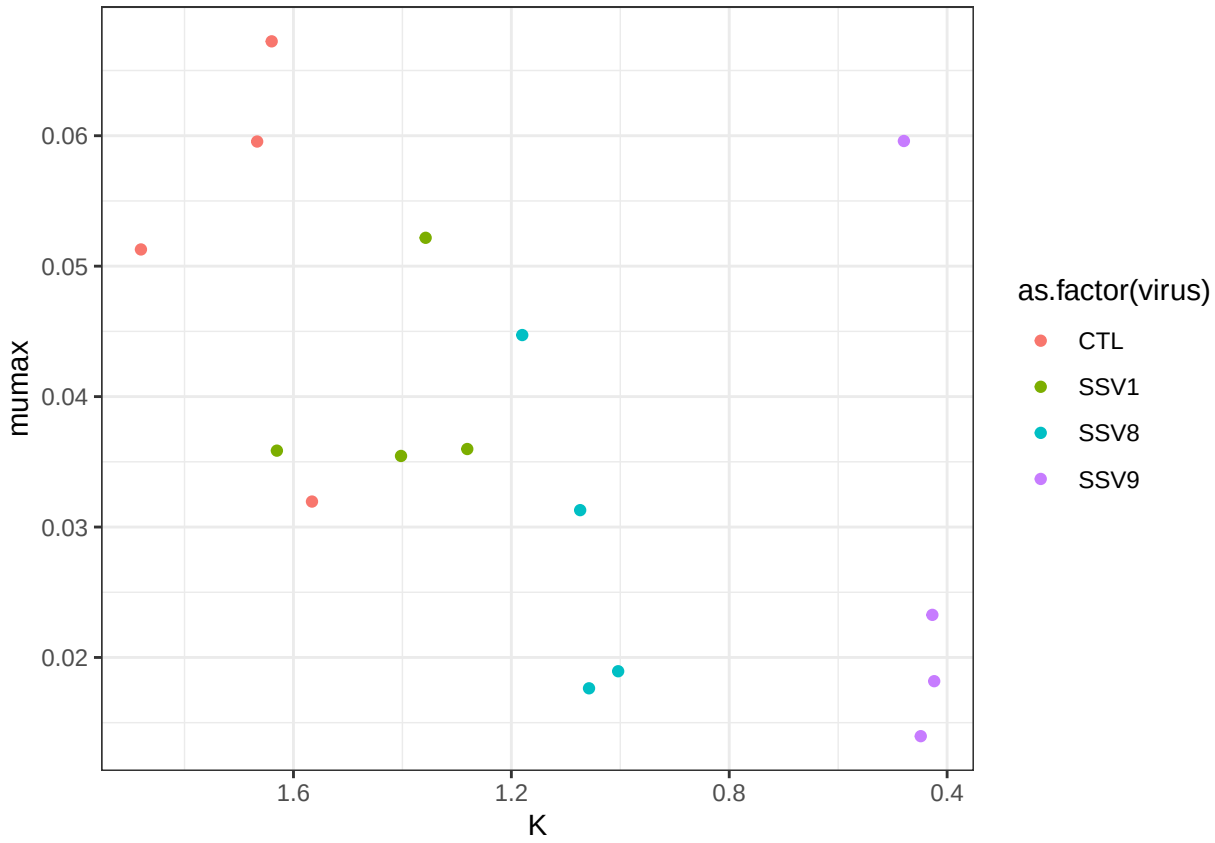

```
ggplot(data = gd, aes(y = K, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(y = K, x = rAUC, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(x = K, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



S147

```
library(tidyverse)
library(DescTools)
library(MESS)
library(grofit)
library(readxl)

# Week_2_Growthcurver_S1_parameters <- read_excel("~/Downloads/Week 2 - Growthcurver S1 parameters.xlsx

# if (file.exists("FinalOutPutfigZ.Rda")) {
#   load("FinalOutPutfigZ.Rda")
# } else {

options(scipen = 999)

#IMPORT DATA HERE AS DATA.FRAME

# dat <- read_csv("~/Downloads/Supplemental_Table_S1_ready.csv")
dat <- read_excel(here("/Data/Processed/Sulfolobus/Sulfolobus_Infection_Growth_Curves.xlsx"), sheet = "S

dat <- as.data.frame(dat[complete.cases(dat),])

#now remove S150 columns
dat <- dat[-c(2:9)]
```

```
############3
####hard coded: remove!
# dat[1,11] <- 0.07
############
##########

#tells us the number of digits to make the names, always needs to be longer than length of col names
trim <- 15

colnames(dat)[1] <- "time"

colnames(dat)[-1] <- str_pad(colnames(dat)[-1], trim, pad = "0", side = "left" )




# maxbyrowraw <- colnames(dat[-1])[max.col(dat[-1],ties.method="random")]
# maxbyrowconvert <- as.data.frame(table(maxbyrowraw))
# maxbyrowcount <- arrange(maxbyrowconvert,-Freq)
# maxmax <- as.character(maxbyrowcount$maxbyrowraw[1])

#note: should always be 1 (first column)

timeColumn <- 1
#can be adjsuted. average of ctl replicates is recommended based on zero science (seems like a good sta
controlColumn <- 5
#shouldn't change unless time column != 1
a <- 2
#ADJUST ME: total number of curves in table + 1
b <- ncol(dat)

#What is the timepoint at which stationary phase is reached?
t_stationary <- 26


firstRun <- TRUE
c <- 2
d <- length(dat[[timeColumn]])
#d <- 20 #this is where 'stationary phase' would traditionally be considered fully reached
for (j in c:d) {


figZ <- gcFitSpline(dat[[timeColumn]], dat[[controlColumn]], gcID = "spline",
            control = grofit.control())
lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])
#upperbound <- dat[timeColumn[[j]]
#AUCraw <- AUC(dat[1:length(dat[[timeColumn]]),1], dat[1:length(dat[[controlColumn]]),controlColumn])
AUCraw <- AUC(dat[1:j,1], dat[1:j,controlColumn])
PI <- 0
IscZ <- 0
```

```
storage.vector_figZ <- data.frame( "mumax"= mumax, "K"= K, "lambda"= lambda, "UpperBound" = upperbound,


for (i in a:b) {
if(i != controlColumn) {
figZ <- gcFitSpline(dat$time, dat[[i]], gcID = "spline444",
            control = grofit.control())
#plot(figZ)

lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])

  AUCraw <- AUC(dat[1:j,1], dat[1:j,i])
  if(i == controlColumn) {
    PI <- 0
  } else {
    PI <- (1 - (AUCraw/storage.vector_figZ$AUC[1])) * 100
  }

  IscZ <-  (1 - sqrt((AUCraw*K)/(storage.vector_figZ$AUC[1]*storage.vector_figZ$K[1]))) * 100


storage.vector_figZ <- rbind(storage.vector_figZ, c( mumax, K, lambda, upperbound, as.numeric(AUCraw), I


}
}
#output <- rbind
if(firstRun == TRUE) {
  #something
  firstRun <- FALSE
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- storage.vector_figZ
} else {
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- rbind(FinalOutPutfigZ, storage.vector_figZ)
}

}
#storage.vector_figZ <- storage.vector_figZ[-c(1), ]

#storage.vector_figZ
#Below code to get rid of scientific notation:
options(scipen=999)
#here is how to get back to scientific notation: options(scipen=0)



library(data.table)
setDT(FinalOutPutfigZ, keep.rownames = TRUE)
#here I'm fixing the group names to not say replicates
```

```r
# FinalOutPutfigZ$rn <- c(str_replace_all(string=FinalOutPutfigZ$rn,pattern="\\l.*$",replacement="l"))
FinalOutPutfigZ$rn <- substr(FinalOutPutfigZ$rn, 1, trim)
# df$col1 <- strtrim(df$col, 1, 1)

IscOrderedOutputFigZ <- FinalOutPutfigZ %>%
  arrange((Isc))



if (exists("maxIsc") == TRUE) {
  save(FinalOutPutfigZ,file="FinalOutPutfigZ.Rda")
} else {
maxIsc <- IscOrderedOutputFigZ[1,1]
}
# }

# for writing a data.frame or list of data.frames to an xlsx file
#write.xlsx(FinalOutPutfigZ, 'Isc_figZISC.xlsx')
#FinalOutPUtfigZsave <- FinalOutPutfigZ

IscOrderedOutputFigZ
```

```
##                  rn      mumax         K    lambda UpperBound        AUC
##  1: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         14  4.5510000
##  2: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596          8  1.8900000
##  3: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596          2  0.3000000
##  4: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         72 59.7230000
##  5: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         60 46.6760000
##  6: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         66 53.3750000
##  7: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         54 39.4370000
##  8: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         48 32.5430000
##  9: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         20  7.7280000
## 10: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         32 16.7670000
## 11: 00000S147CTLAVG 0.02606074 1.1377008 -4.951596         26 11.6100000
## 12: 0000000S147CTL2 0.05555074 1.1600000 -1.476128          8  1.6820000
## 13: 0000000S147CTL2 0.05555074 1.1600000 -1.476128         60 42.6890000
## 14: 0000000S147CTL2 0.05555074 1.1600000 -1.476128         54 36.0470000
## 15: 0000000S147CTL2 0.05555074 1.1600000 -1.476128         66 48.5360000
## 16: 0000000S147CTL2 0.05555074 1.1600000 -1.476128         48 29.6600000
## 17: 0000000S147CTL2 0.05555074 1.1600000 -1.476128         72 53.5490000
## 18: 0000000S147CTL2 0.05555074 1.1600000 -1.476128          2  0.2660000
## 19: 0000000S147CTL2 0.05555074 1.1600000 -1.476128         14  3.7400000
## 20: 0000000S147CTL2 0.05555074 1.1600000 -1.476128         32 14.7560000
## 21: 0000000S147CTL3 0.04213917 1.0278885  1.705721         26 11.6400000
## 22: 0000000S147CTL2 0.05555074 1.1600000 -1.476128         26 10.1780000
## 23: 0000000S147CTL1 0.02616651 1.0564087 -4.026511          2  0.2756667
## 24: 0000000S147CTL1 0.02616651 1.0564087 -4.026511          8  1.6656667
## 25: 0000000S147CTL1 0.02616651 1.0564087 -4.026511         14  3.9916667
## 26: 0000000S147CTL1 0.02616651 1.0564087 -4.026511         20  7.2666667
## 27: 0000000S147CTL1 0.02616651 1.0564087 -4.026511         26 11.1426667
## 28: 0000000S147CTL1 0.02616651 1.0564087 -4.026511         32 15.8336667
## 29: 0000000S147CTL1 0.02616651 1.0564087 -4.026511         48 30.3696667
## 30: 0000000S147CTL1 0.02616651 1.0564087 -4.026511         54 36.5206667
```

```
## 31: 0000000S147CTL1 0.02616651 1.0564087 -4.026511        60 43.0996667
## 32: 0000000S147CTL1 0.02616651 1.0564087 -4.026511        66 49.3166667
## 33: 0000000S147CTL1 0.02616651 1.0564087 -4.026511        72 54.9026667
## 34: 0000000S147CTL3 0.04213917 1.0278885  1.705721        20  7.4640000
## 35: 0000000S147CTL2 0.05555074 1.1600000 -1.476128        20  6.6080000
## 36: 0000000S147CTL3 0.04213917 1.0278885  1.705721        32 15.9780000
## 37: 0000000S147CTL3 0.04213917 1.0278885  1.705721        48 28.9060000
## 38: 0000000S147CTL3 0.04213917 1.0278885  1.705721         2  0.2610000
## 39: 0000000S147CTL3 0.04213917 1.0278885  1.705721        72 51.4360000
## 40: 0000000S147CTL3 0.04213917 1.0278885  1.705721        66 46.0390000
## 41: 0000000S147CTL3 0.04213917 1.0278885  1.705721        54 34.0780000
## 42: 0000000S147CTL3 0.04213917 1.0278885  1.705721        60 39.9340000
## 43: 0000000S147CTL3 0.04213917 1.0278885  1.705721        14  3.6840000
## 44: 0000000S147CTL3 0.04213917 1.0278885  1.705721         8  1.4250000
## 45: 000000S147SSV9A 0.03016752 0.7820000 -2.939088         8  1.6760000
## 46: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        14  3.9920000
## 47: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        20  6.6590000
## 48: 000000S147SSV9B 0.03982461 0.7690000 -2.134359         8  1.5140000
## 49: 000000S147SSV9B 0.03982461 0.7690000 -2.134359         2  0.2450000
## 50: 000000S147SSV9A 0.03016752 0.7820000 -2.939088         2  0.2390000
## 51: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264         8  1.4903333
## 52: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        26  9.5330000
## 53: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        14  3.4863333
## 54: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        32 13.1360000
## 55: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        26  9.3260000
## 56: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        14  3.3080000
## 57: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        20  6.1693333
## 58: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        32 13.0040000
## 59: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        26  9.3603333
## 60: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        20  5.9420000
## 61: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264         2  0.2283333
## 62: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        32 12.9803333
## 63: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        48 23.6320000
## 64: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        48 23.3960000
## 65: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        54 27.4510000
## 66: 000000S147SSV9C 0.03173204 0.6980000 -2.237486        26  9.2220000
## 67: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        60 31.7500000
## 68: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        48 23.2870000
## 69: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        66 36.2080000
## 70: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        54 27.2120000
## 71: 000000S147SSV9A 0.03016752 0.7820000 -2.939088        72 39.9610000
## 72: 000000S147SSV9C 0.03173204 0.6980000 -2.237486        20  5.9070000
## 73: 000000S147SSV9C 0.03173204 0.6980000 -2.237486        32 12.8010000
## 74: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        60 31.3610000
## 75: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        66 35.6840000
## 76: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        54 27.0600000
## 77: 000000S147SSV9B 0.03982461 0.7690000 -2.134359        72 39.5780000
## 78: 000000S147SSV9C 0.03173204 0.6980000 -2.237486        14  3.1590000
## 79: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        60 31.1560000
## 80: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        66 35.3280000
## 81: 000000S147SSV9C 0.03173204 0.6980000 -2.237486         8  1.2810000
## 82: 0000S147SSV9AVG 0.03357541 0.7496667 -2.442264        72 38.7850000
## 83: 000000S147SSV9C 0.03173204 0.6980000 -2.237486        48 22.8330000
## 84: 000000S147SSV9C 0.03173204 0.6980000 -2.237486         2  0.2010000
```

```
## 85: 000000S147SSV9C 0.03173204 0.6980000 -2.237486         54 26.5170000
## 86: 000000S147SSV9C 0.03173204 0.6980000 -2.237486         60 30.3570000
## 87: 000000S147SSV9C 0.03173204 0.6980000 -2.237486         66 34.0920000
## 88: 000000S147SSV9C 0.03173204 0.6980000 -2.237486         72 36.8160000
##                  rn      mumax         K    lambda UpperBound        AUC
##                PI          Isc
##  1: -14.012526096 -10.80882353
##  2: -13.468080849 -10.54393364
##  3:  -8.827085852  -8.25963533
##  4:  -8.779779974  -8.23610316
##  5:  -8.297821329  -7.99606207
##  6:  -8.229131463  -7.96180740
##  7:  -7.985432906  -7.84019059
##  8:  -7.156263377  -7.42536560
##  9:  -6.348623853  -7.01976603
## 10:  -5.894612745  -6.79108356
## 11:  -4.194088788  -5.93015320
## 12:  -0.980588353  -5.30087094
## 13:   0.952830262  -4.28793063
## 14:   1.296982530  -4.10659199
## 15:   1.582967219  -3.95566201
## 16:   2.336761462  -3.55678897
## 17:   2.465575443  -3.48847276
## 18:   3.506650544  -2.93467692
## 19:   6.304801670  -1.43122773
## 20:   6.806172502  -1.15948023
## 21:  -4.463324159  -0.81820273
## 22:   8.657412947  -0.14970370
## 23:   0.000000000   0.00000000
## 24:   0.000000000   0.00000000
## 25:   0.000000000   0.00000000
## 26:   0.000000000   0.00000000
## 27:   0.000000000   0.00000000
## 28:   0.000000000   0.00000000
## 29:   0.000000000   0.00000000
## 30:   0.000000000   0.00000000
## 31:   0.000000000   0.00000000
## 32:   0.000000000   0.00000000
## 33:   0.000000000   0.00000000
## 34:  -2.715596330   0.02872609
## 35:   9.064220183   0.07356066
## 36:  -0.911559757   0.91053358
## 37:   4.819501915   3.76545066
## 38:   5.320435308   4.01902516
## 39:   6.314204532   4.52406840
## 40:   6.646164245   4.69336989
## 41:   6.688450375   4.71495767
## 42:   7.344991067   5.05076250
## 43:   7.707724426   5.23680225
## 44:  14.448669202   8.76312216
## 45:  -0.620372223  13.69611292
## 46:  -0.008350731  13.95898413
## 47:   8.362385321  17.63849294
## 48:   9.105463278  18.65776889
```

```
## 49:  11.124546554  19.56628927
## 50:  13.301088271  19.88861874
## 51:  10.526315789  20.31698087
## 52:  14.445973435  20.41932401
## 53:  12.659707724  21.27268482
## 54:  17.037536052  21.63390294
## 55:  16.303697499  21.94507221
## 56:  17.127348643  22.33008911
## 57:  15.100917431  22.38071833
## 58:  17.871202712  22.67945231
## 59:  15.995572574  22.79077226
## 60:  18.229357798  22.84822993
## 61:  17.170495768  23.33261480
## 62:  18.020673249  23.72709383
## 63:  22.185514055  24.10422097
## 64:  22.962605231  25.11445689
## 65:  24.834340373  25.40716159
## 66:  17.237046787  26.05144005
## 67:  26.333536996  26.15479407
## 68:  23.321515986  26.23423009
## 69:  26.580601555  26.27873009
## 70:  25.488764353  26.35248918
## 71:  27.214828666  26.59783813
## 72:  18.711009174  26.71288976
## 73:  19.153280984  26.91252952
## 74:  27.236096180  27.22115053
## 75:  27.643122677  27.42499104
## 76:  25.904967051  27.48753509
## 77:  27.912426840  27.56017496
## 78:  20.860125261  27.68816027
## 79:  27.711737910  28.37708170
## 80:  28.364988172  28.70143500
## 81:  23.093856314  28.71596922
## 82:  29.356801127  29.19673273
## 83:  24.816428673  29.51881337
## 84:  27.085852479  30.59070506
## 85:  27.391796426  30.73647702
## 86:  29.565580554  31.78118029
## 87:  30.871240284  32.41643233
## 88:  32.943147874  33.43693482
##                PI          Isc
```

```
gdIsc <- FinalOutPutfigZ %>%
        group_by(rn, UpperBound) %>%
        #group_by(UpperBound) %>%
        summarise(Isc = mean(Isc))
```

```
## `summarise()` regrouping output by 'rn' (override with `.groups` argument)
```

```
gdVr <- gdIsc %>%
  group_by(rn) %>%
  summarize(Vr = auc(UpperBound, Isc, type = "spline")/(max(UpperBound) - min(UpperBound)))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
gdPI <- FinalOutPutfigZ %>%
        group_by(rn) %>%
        summarise(PI = mean(PI))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
AUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == max(FinalOutPutfigZ$UpperBound)) %>%
    mutate(rAUC = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select(c(-1,-4, -5, -6, -7, -8))
AUCfigZ
```

```
##         mumax         K      rAUC
## 1: 0.02616651 1.0564087  91.92885
## 2: 0.05555074 1.1600000  89.66227
## 3: 0.04213917 1.0278885  86.12427
## 4: 0.03016752 0.7820000  66.91057
## 5: 0.03982461 0.7690000  66.26928
## 6: 0.03173204 0.6980000  61.64459
## 7: 0.02606074 1.1377008 100.00000
## 8: 0.03357541 0.7496667  64.94148
```

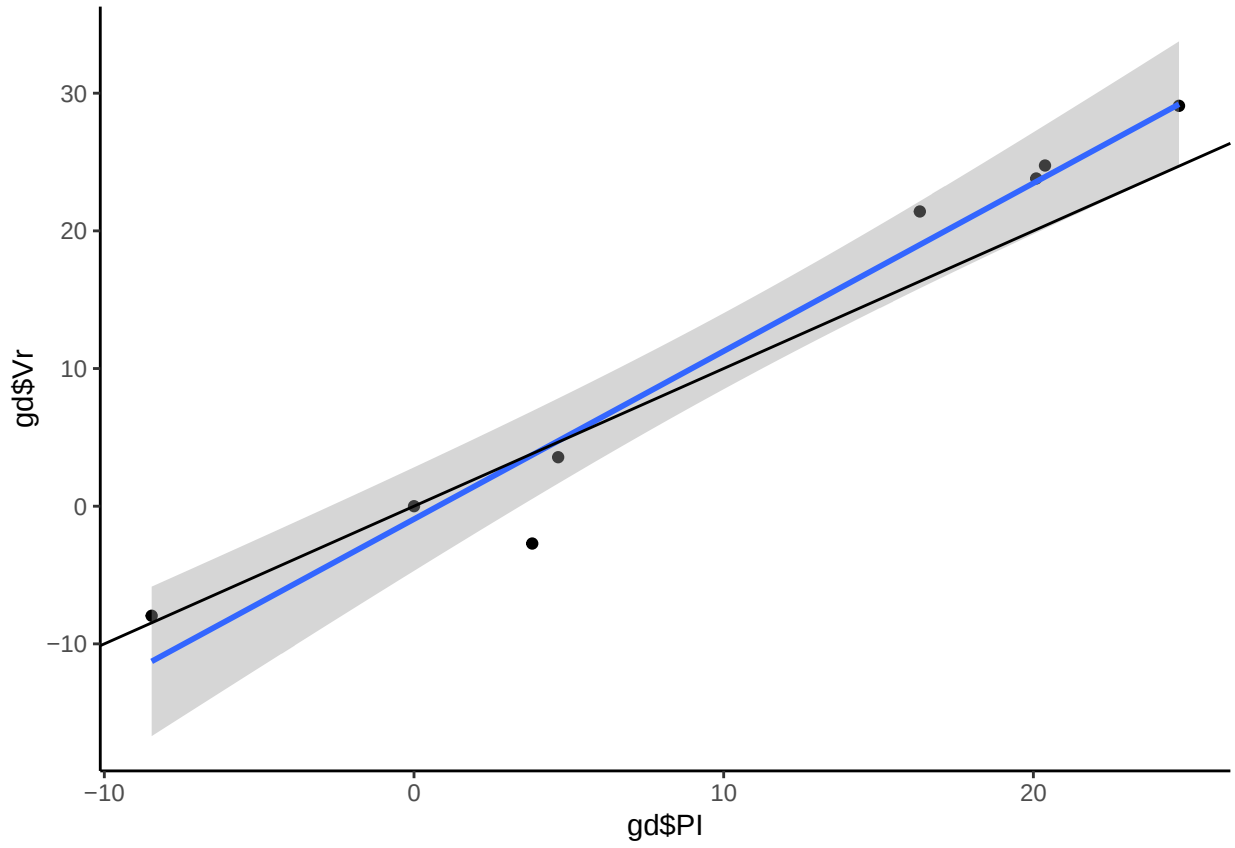
```
trimAUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == t_stationary) %>%
    mutate(AUCtrim = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select("AUCtrim")


gd <- cbind(gdVr, gdPI, AUCfigZ, trimAUCfigZ)
gd <- gd[-3]
gd
```

```
##                rn        Vr        PI      mumax         K      rAUC   AUCtrim
## 1 0000000S147CTL1  0.000000  0.000000 0.02616651 1.0564087  91.92885  95.72738
## 2 0000000S147CTL2 -2.711772  3.817617 0.05555074 1.1600000  89.66227  87.43986
## 3 0000000S147CTL3  3.559293  4.654515 0.04213917 1.0278885  86.12427 100.00000
## 4 000000S147SSV9A 21.408342 16.333371 0.03016752 0.7820000  66.91057  81.89863
## 5 000000S147SSV9B 23.804341 20.091330 0.03982461 0.7690000  66.26928  80.12027
## 6 000000S147SSV9C 29.083708 24.702670 0.03173204 0.6980000  61.64459  79.22680
## 7 00000S147CTLAVG -7.959805 -8.472132 0.02606074 1.1377008 100.00000  99.74227
## 8 0000S147SSV9AVG 24.747432 20.375790 0.03357541 0.7496667  64.94148  80.41523
```

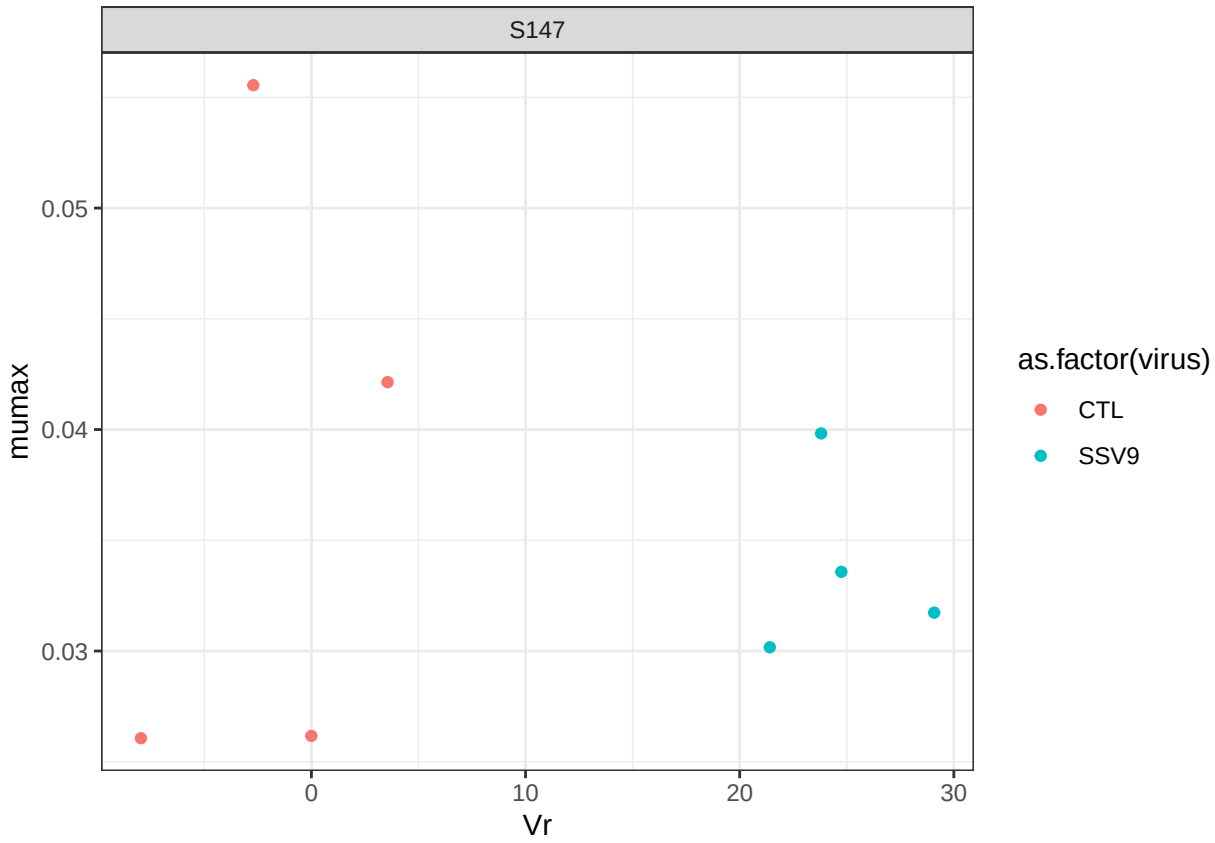
```
gd$virus <- c(
    rep("CTL", 3),
    rep("SSV9", 3),
    rep("CTL", 1),
    rep("SSV9", 1)
)
```

```r
# gd$virus <- c(
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12)
# )

gd$host <-  rep("S147", 8)



gd147 <- gd


ggplot(data = NULL, aes(x = gd$PI, y = gd$Vr)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_abline() +
  theme_classic()
```

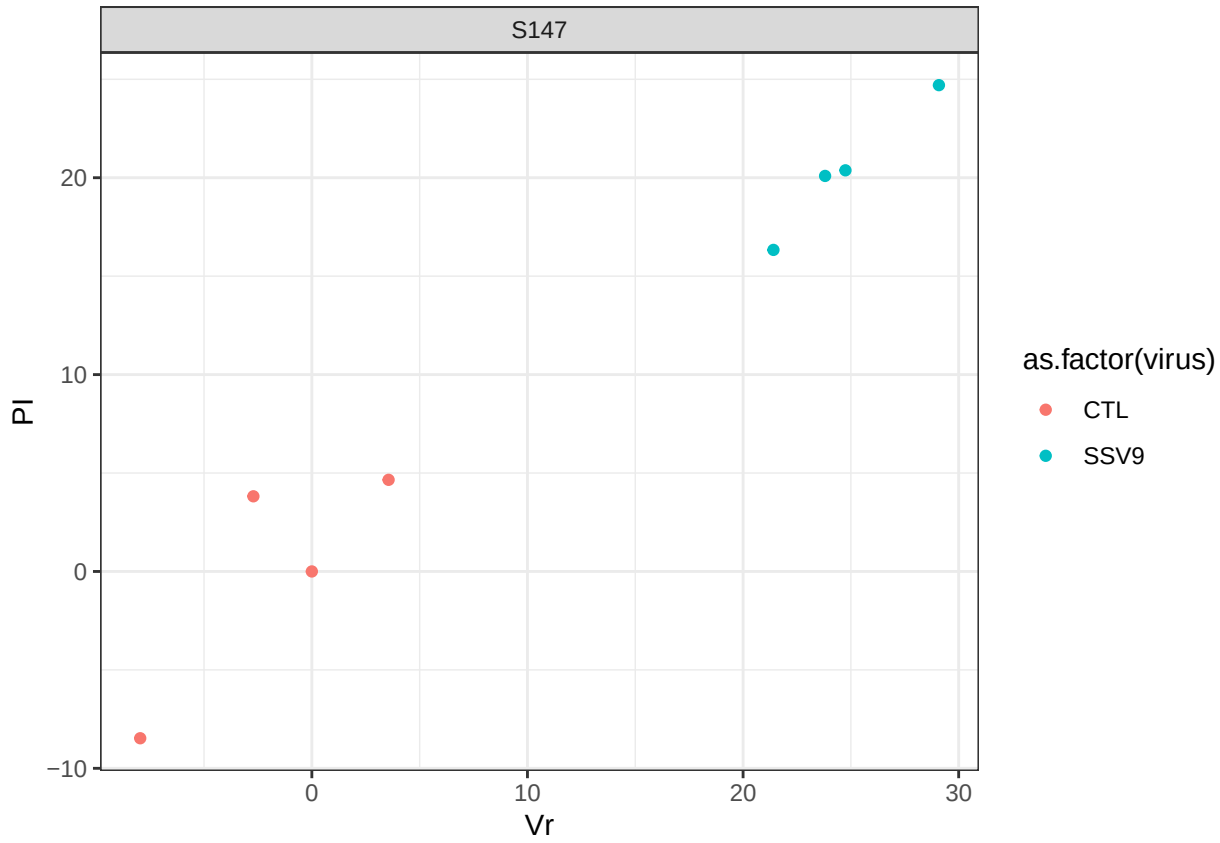
```
## `geom_smooth()` using formula 'y ~ x'
```



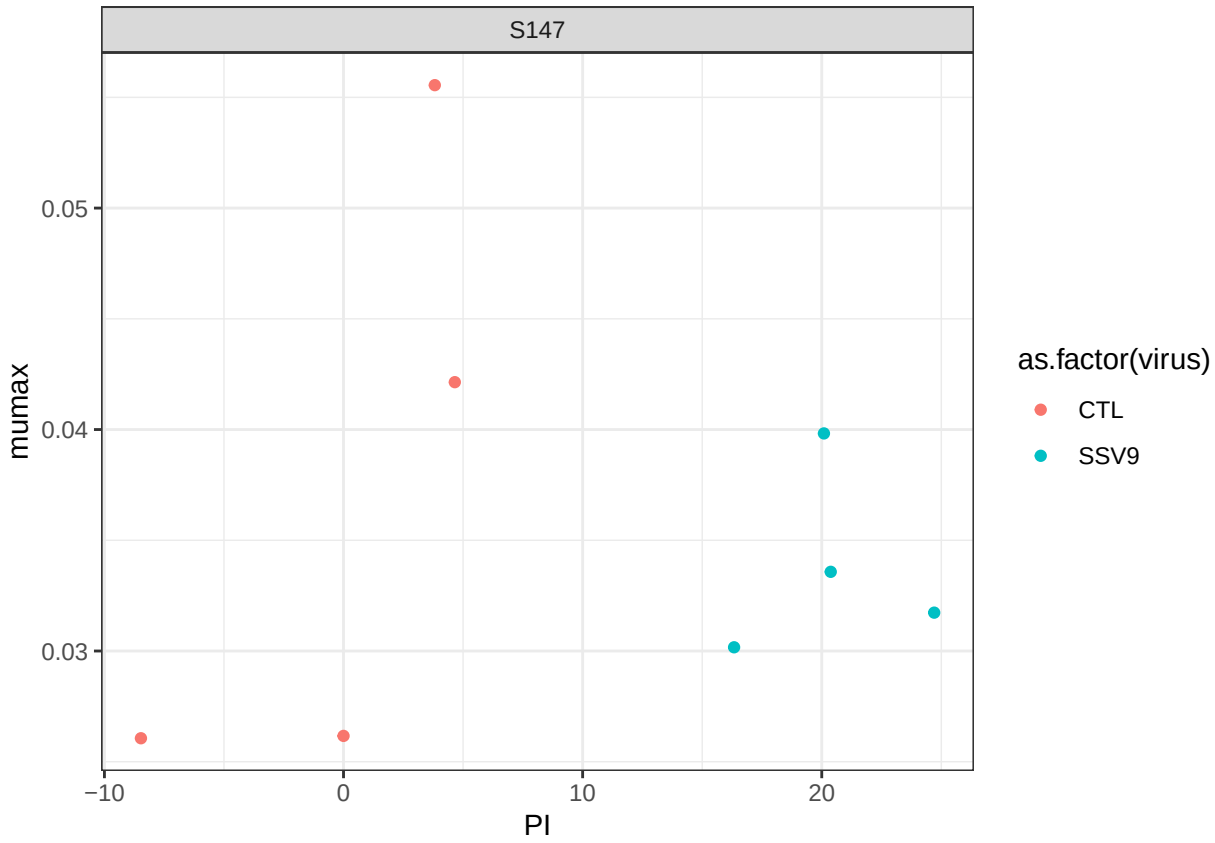

```
ggplot(data = gd, aes(x = Vr, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



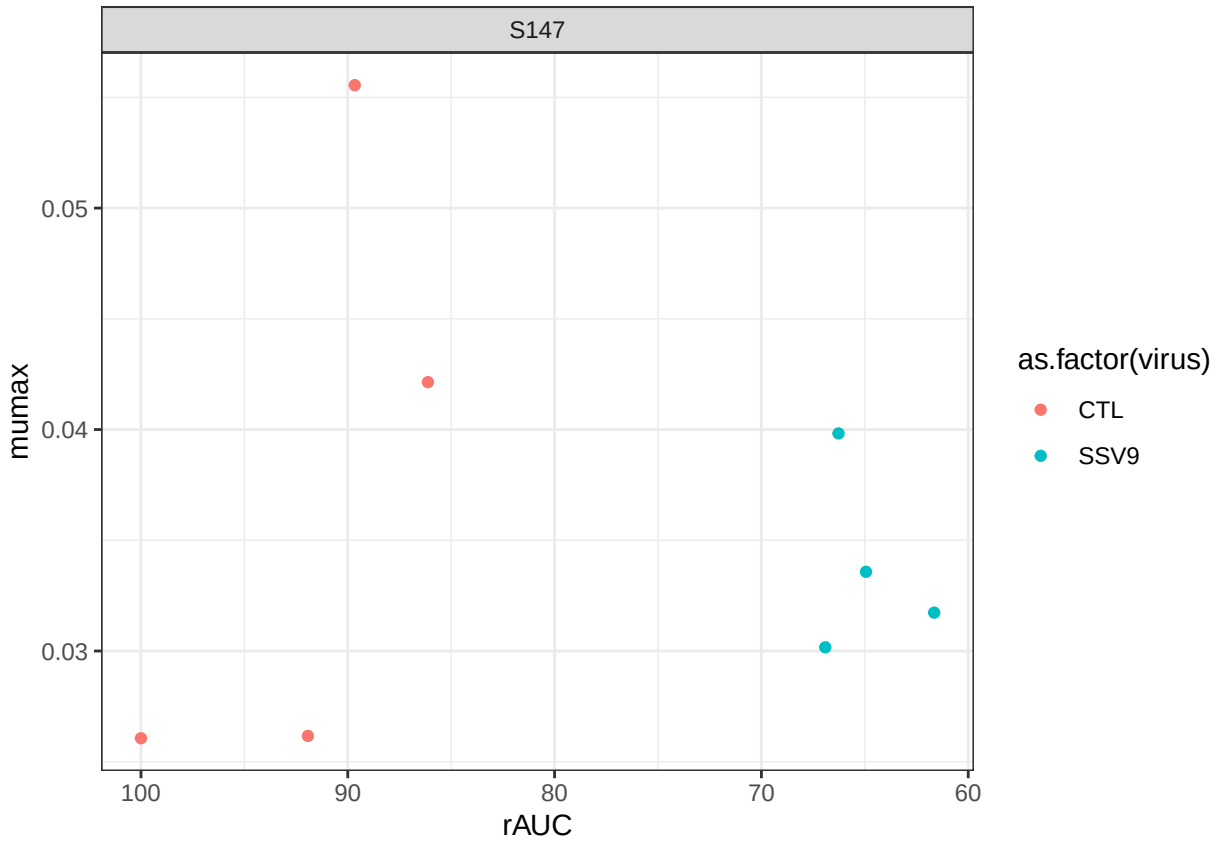

```
ggplot(data = gd, aes(y = PI, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



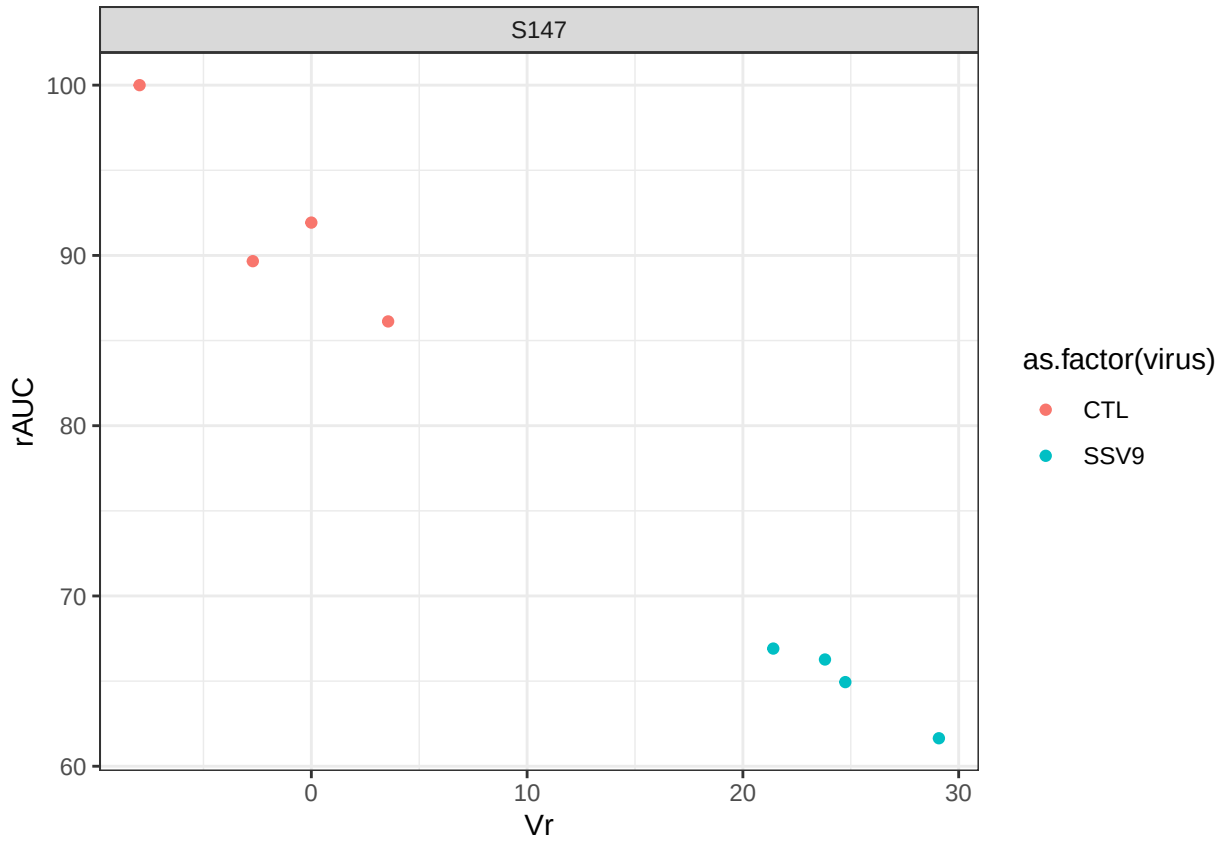

```
ggplot(data = gd, aes(y = mumax, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



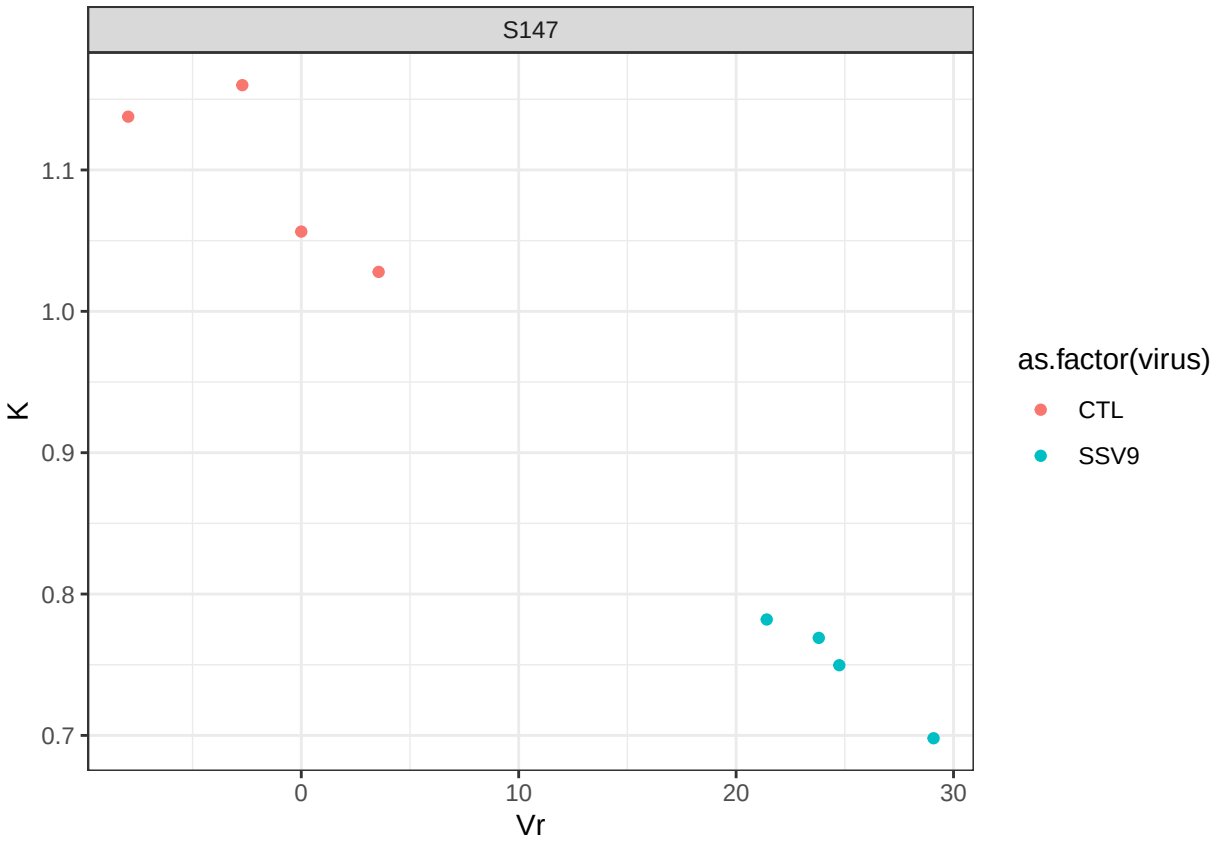

```
ggplot(data = gd, aes(y = mumax, x = rAUC, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



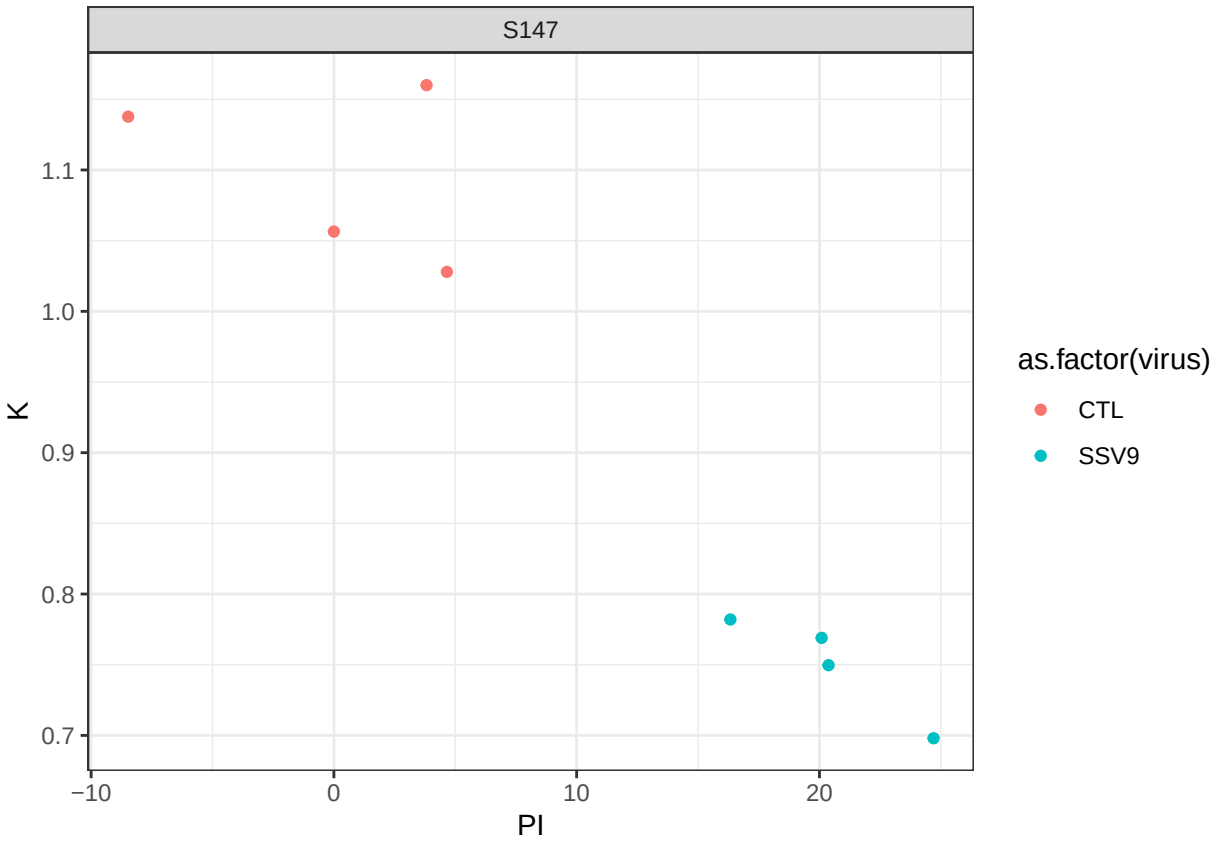

```
ggplot(data = gd, aes(y = rAUC, x = Vr, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



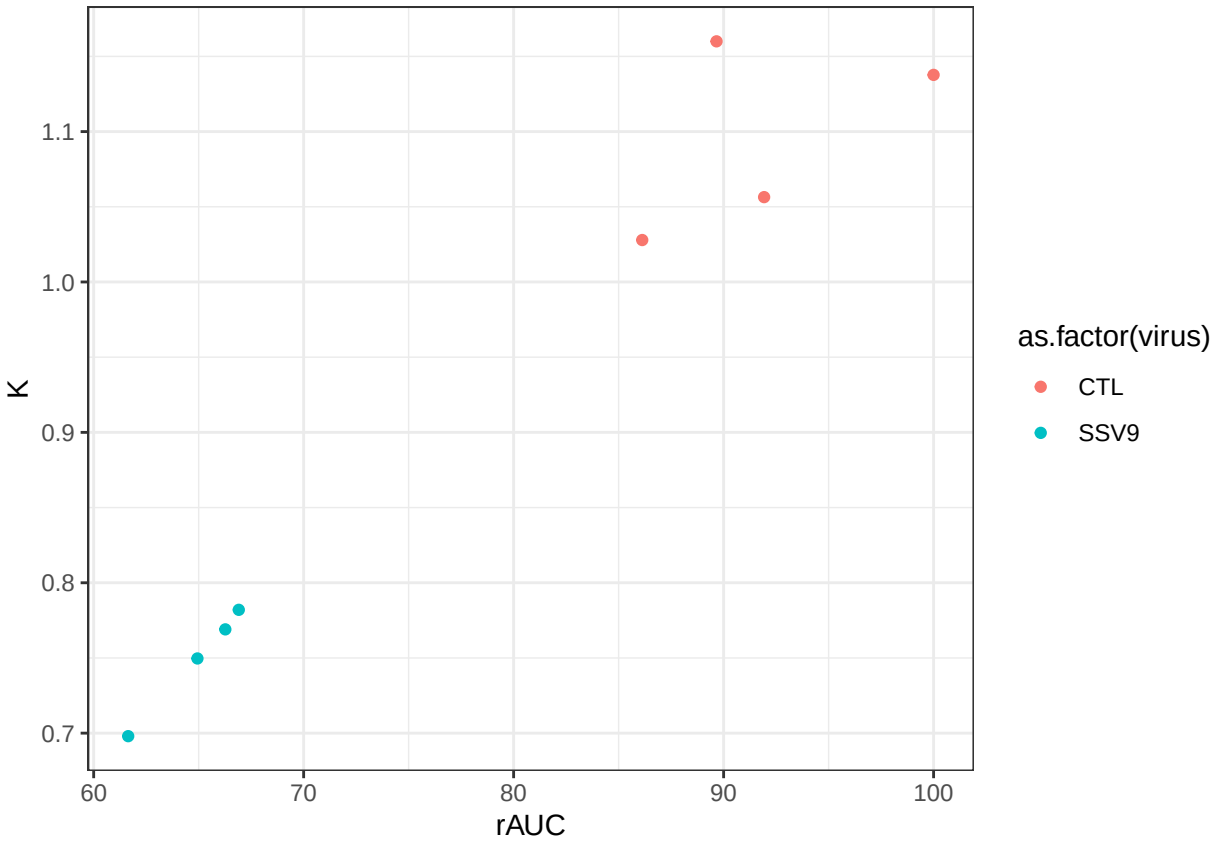

```
ggplot(data = gd, aes(y = K, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



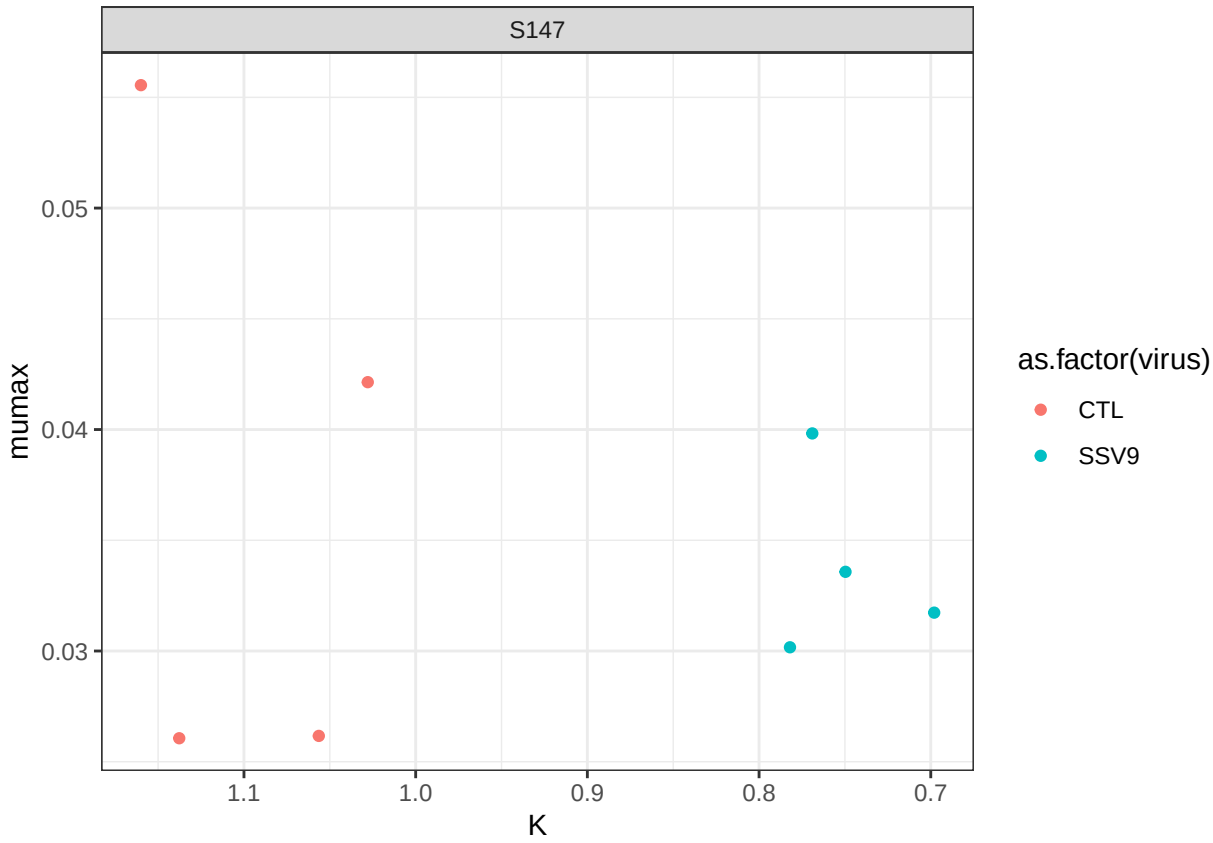

```
ggplot(data = gd, aes(y = K, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(y = K, x = rAUC, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(x = K, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



S150

```
library(tidyverse)
library(DescTools)
library(MESS)
library(grofit)
library(readxl)

# Week_2_Growthcurver_S1_parameters <- read_excel("~/Downloads/Week 2 - Growthcurver S1 parameters.xlsx

# if (file.exists("FinalOutPutfigZ.Rda")) {
#   load("FinalOutPutfigZ.Rda")
# } else {

options(scipen = 999)

#IMPORT DATA HERE AS DATA.FRAME

# dat <- read_csv("~/Downloads/Supplemental_Table_S1_ready.csv")
dat <- read_excel(here("/Data/Processed/Sulfolobus/Sulfolobus_Infection_Growth_Curves.xlsx"), sheet = "S

dat <- as.data.frame(dat[complete.cases(dat),])

#now remove S147 columns
dat <- dat[-c(10:18)]
```

```
############3
####hard coded: remove!
# dat[1,11] <- 0.07
############
##########

#tells us the number of digits to make the names, always needs to be longer than length of col names
trim <- 15

colnames(dat)[1] <- "time"

colnames(dat)[-1] <- str_pad(colnames(dat)[-1], trim, pad = "0", side = "left" )




# maxbyrowraw <- colnames(dat[-1])[max.col(dat[-1],ties.method="random")]
# maxbyrowconvert <- as.data.frame(table(maxbyrowraw))
# maxbyrowcount <- arrange(maxbyrowconvert,-Freq)
# maxmax <- as.character(maxbyrowcount$maxbyrowraw[1])

#note: should always be 1 (first column)

timeColumn <- 1
#can be adjsuted. average of ctl replicates is recommended based on zero science (seems like a good sta
controlColumn <- 5
#shouldn't change unless time column != 1
a <- 2
#ADJUST ME: total number of curves in table + 1
b <- ncol(dat)

#What is the timepoint at which stationary phase is reached?
t_stationary <- 26


firstRun <- TRUE
c <- 2
d <- length(dat[[timeColumn]])
#d <- 20 #this is where 'stationary phase' would traditionally be considered fully reached
for (j in c:d) {


figZ <- gcFitSpline(dat[[timeColumn]], dat[[controlColumn]], gcID = "spline",
            control = grofit.control())
lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])
#upperbound <- dat[timeColumn[[j]]
#AUCraw <- AUC(dat[1:length(dat[[timeColumn]]),1], dat[1:length(dat[[controlColumn]]),controlColumn])
AUCraw <- AUC(dat[1:j,1], dat[1:j,controlColumn])
PI <- 0
IscZ <- 0
```

```
storage.vector_figZ <- data.frame( "mumax"= mumax, "K"= K, "lambda"= lambda, "UpperBound" = upperbound,


for (i in a:b) {
if(i != controlColumn) {
figZ <- gcFitSpline(dat$time, dat[[i]], gcID = "spline444",
            control = grofit.control())
#plot(figZ)

lambda <- as.numeric(figZ$parameters$lambda)
mumax <- as.numeric(figZ$parameters$mu)
K <- as.numeric(figZ$parameters$A)
upperbound <- as.numeric(dat[j,1])

  AUCraw <- AUC(dat[1:j,1], dat[1:j,i])
  if(i == controlColumn) {
    PI <- 0
  } else {
    PI <- (1 - (AUCraw/storage.vector_figZ$AUC[1])) * 100
  }

  IscZ <-  (1 - sqrt((AUCraw*K)/(storage.vector_figZ$AUC[1]*storage.vector_figZ$K[1]))) * 100


storage.vector_figZ <- rbind(storage.vector_figZ, c( mumax, K, lambda, upperbound, as.numeric(AUCraw), I


}
}
#output <- rbind
if(firstRun == TRUE) {
  #something
  firstRun <- FALSE
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- storage.vector_figZ
} else {
  rownames(storage.vector_figZ) <- colnames(dat[-timeColumn])
  FinalOutPutfigZ <- rbind(FinalOutPutfigZ, storage.vector_figZ)
}

}
#storage.vector_figZ <- storage.vector_figZ[-c(1), ]

#storage.vector_figZ
#Below code to get rid of scientific notation:
options(scipen=999)
#here is how to get back to scientific notation: options(scipen=0)



library(data.table)
setDT(FinalOutPutfigZ, keep.rownames = TRUE)
#here I'm fixing the group names to not say replicates
```

```r
# FinalOutPutfigZ$rn <- c(str_replace_all(string=FinalOutPutfigZ$rn,pattern="\\l.*$",replacement="l"))
FinalOutPutfigZ$rn <- substr(FinalOutPutfigZ$rn, 1, trim)
# df$col1 <- strtrim(df$col, 1, 1)

IscOrderedOutputFigZ <- FinalOutPutfigZ %>%
  arrange((Isc))



if (exists("maxIsc") == TRUE) {
  save(FinalOutPutfigZ,file="FinalOutPutfigZ.Rda")
} else {
maxIsc <- IscOrderedOutputFigZ[1,1]
}
# }

# for writing a data.frame or list of data.frames to an xlsx file
#write.xlsx(FinalOutPutfigZ, 'Isc_figZISC.xlsx')
#FinalOutPUtfigZsave <- FinalOutPutfigZ

IscOrderedOutputFigZ
```

```
##                  rn      mumax        K     lambda UpperBound        AUC
##  1: 000000S150SSV9C 0.03432187 1.330917 -3.5211076          14  5.0260000
##  2: 000000S150SSV9C 0.03432187 1.330917 -3.5211076           2  0.3160000
##  3: 000000S150SSV9C 0.03432187 1.330917 -3.5211076           8  2.0290000
##  4: 000000S150SSV9C 0.03432187 1.330917 -3.5211076          20  9.2500000
##  5: 000000S150SSV9C 0.03432187 1.330917 -3.5211076          26 14.0920000
##  6: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835           2  0.2520000
##  7: 00000S150CTLAVG 0.05130124 1.345000  6.2419032           8  1.6760000
##  8: 000000S150SSV9A 0.06947241 1.428000  8.3718667           2  0.2350000
##  9: 00000S150CTLAVG 0.05130124 1.345000  6.2419032          20  7.6040000
## 10: 00000S150CTLAVG 0.05130124 1.345000  6.2419032          14  3.8690000
## 11: 000000S150SSV9C 0.03432187 1.330917 -3.5211076          32 20.1520000
## 12: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835          20  7.3970000
## 13: 00000S150CTLAVG 0.05130124 1.345000  6.2419032          26 12.8750000
## 14: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835          14  3.7010000
## 15: 000000S150SSV9B 0.04893593 1.411000  5.9612343          54 46.9670000
## 16: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835          26 12.3900000
## 17: 000000S150SSV9A 0.06947241 1.428000  8.3718667          26 11.8240000
## 18: 000000S150SSV9B 0.04893593 1.411000  5.9612343          60 54.4820000
## 19: 000000S150SSV9B 0.04893593 1.411000  5.9612343          72 67.2710000
## 20: 00000S150CTLAVG 0.05130124 1.345000  6.2419032          32 19.2830000
## 21: 000000S150SSV9B 0.04893593 1.411000  5.9612343          66 61.1780000
## 22: 000000S150SSV9B 0.04893593 1.411000  5.9612343          48 38.5880000
## 23: 000000S150SSV9A 0.06947241 1.428000  8.3718667          72 66.1920000
## 24: 0000000S150CTL3 0.03554674 1.321285  2.5723313           2  0.2370000
## 25: 000000S150SSV9A 0.06947241 1.428000  8.3718667          60 53.2830000
## 26: 000000S150SSV9A 0.06947241 1.428000  8.3718667          54 45.4860000
## 27: 000000S150SSV9A 0.06947241 1.428000  8.3718667          66 59.8260000
## 28: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835           8  1.5220000
## 29: 000000S150SSV9C 0.03432187 1.330917 -3.5211076          72 69.7440000
## 30: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835          72 67.7356667
```

```
## 31: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835        32 18.5170000
## 32: 000000S150SSV9A 0.06947241 1.428000  8.3718667        32 17.7070000
## 33: 000000S150SSV9C 0.03432187 1.330917 -3.5211076        66 63.2430000
## 34: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835        54 46.6906667
## 35: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835        60 54.5226667
## 36: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835        66 61.4156667
## 37: 000000S150SSV9A 0.06947241 1.428000  8.3718667        48 37.0590000
## 38: 000000S150SSV9C 0.03432187 1.330917 -3.5211076        60 55.8030000
## 39: 000000S150SSV9C 0.03432187 1.330917 -3.5211076        48 39.6240000
## 40: 000000S150SSV9C 0.03432187 1.330917 -3.5211076        54 47.6190000
## 41: 000000S150SSV9B 0.04893593 1.411000  5.9612343        32 17.6920000
## 42: 000000S150SSV9A 0.06947241 1.428000  8.3718667        20  6.5830000
## 43: 00000S150CTLAVG 0.05130124 1.345000  6.2419032        48 39.0510000
## 44: 0000S150SSV9AVG 0.03704368 1.366112  0.5566835        48 38.4236667
## 45: 00000S150CTLAVG 0.05130124 1.345000  6.2419032        72 67.4940000
## 46: 00000S150CTLAVG 0.05130124 1.345000  6.2419032        66 61.4940000
## 47: 00000S150CTLAVG 0.05130124 1.345000  6.2419032        54 46.5540000
## 48: 00000S150CTLAVG 0.05130124 1.345000  6.2419032        60 54.3570000
## 49: 000000S150SSV9B 0.04893593 1.411000  5.9612343        26 11.2540000
## 50: 0000000S150CTL2 0.06753609 1.311000  8.9836671        48 38.9780000
## 51: 0000000S150CTL2 0.06753609 1.311000  8.9836671        32 18.4500000
## 52: 0000000S150CTL2 0.06753609 1.311000  8.9836671        72 67.2890000
## 53: 0000000S150CTL2 0.06753609 1.311000  8.9836671        54 46.6130000
## 54: 0000000S150CTL2 0.06753609 1.311000  8.9836671        66 61.1690000
## 55: 0000000S150CTL2 0.06753609 1.311000  8.9836671        60 54.1490000
## 56: 000000S150SSV9B 0.04893593 1.411000  5.9612343         2  0.2050000
## 57: 00000S150CTLAVG 0.05130124 1.345000  6.2419032         2  0.2150000
## 58: 000000S150SSV9B 0.04893593 1.411000  5.9612343        20  6.3580000
## 59: 0000000S150CTL3 0.03554674 1.321285  2.5723313        14  3.4620000
## 60: 0000000S150CTL1 0.04586753 1.308333  4.5715098         2  0.2186667
## 61: 0000000S150CTL1 0.04586753 1.308333  4.5715098         8  1.4666667
## 62: 0000000S150CTL1 0.04586753 1.308333  4.5715098        14  3.4636667
## 63: 0000000S150CTL1 0.04586753 1.308333  4.5715098        20  6.7896667
## 64: 0000000S150CTL1 0.04586753 1.308333  4.5715098        26 11.6926667
## 65: 0000000S150CTL1 0.04586753 1.308333  4.5715098        32 18.0196667
## 66: 0000000S150CTL1 0.04586753 1.308333  4.5715098        48 37.9450000
## 67: 0000000S150CTL1 0.04586753 1.308333  4.5715098        54 45.6480000
## 68: 0000000S150CTL1 0.04586753 1.308333  4.5715098        60 53.3200000
## 69: 0000000S150CTL1 0.04586753 1.308333  4.5715098        66 60.1410000
## 70: 0000000S150CTL1 0.04586753 1.308333  4.5715098        72 65.8950000
## 71: 0000000S150CTL2 0.06753609 1.311000  8.9836671        26 11.5890000
## 72: 000000S150SSV9B 0.04893593 1.411000  5.9612343        14  3.1390000
## 73: 0000000S150CTL3 0.03554674 1.321285  2.5723313        60 51.4540000
## 74: 0000000S150CTL3 0.03554674 1.321285  2.5723313        66 57.7600000
## 75: 0000000S150CTL3 0.03554674 1.321285  2.5723313        54 43.7770000
## 76: 0000000S150CTL3 0.03554674 1.321285  2.5723313        20  6.5040000
## 77: 0000000S150CTL3 0.03554674 1.321285  2.5723313        72 62.9020000
## 78: 0000000S150CTL3 0.03554674 1.321285  2.5723313        48 35.8060000
## 79: 000000S150SSV9B 0.04893593 1.411000  5.9612343         8  1.2850000
## 80: 0000000S150CTL3 0.03554674 1.321285  2.5723313         8  1.3650000
## 81: 0000000S150CTL2 0.06753609 1.311000  8.9836671         2  0.2040000
## 82: 000000S150SSV9A 0.06947241 1.428000  8.3718667         8  1.2520000
## 83: 0000000S150CTL2 0.06753609 1.311000  8.9836671         8  1.3590000
## 84: 000000S150SSV9A 0.06947241 1.428000  8.3718667        14  2.9380000
```

```
## 85: 0000000S150CTL2 0.06753609 1.311000  8.9836671         20  6.2610000
## 86: 0000000S150CTL3 0.03554674 1.321285  2.5723313         26 10.6140000
## 87: 0000000S150CTL3 0.03554674 1.321285  2.5723313         32 16.3260000
## 88: 0000000S150CTL2 0.06753609 1.311000  8.9836671         14  3.0600000
##                  rn      mumax        K     lambda UpperBound        AUC
##               PI         Isc
##  1: -45.10634203 -21.4953162
##  2: -44.51219512 -21.2463260
##  3: -38.34090909 -18.6292136
##  4: -36.23643772 -17.7234493
##  5: -20.51998404 -10.7250465
##  6: -15.24390244  -9.6965449
##  7: -14.27272727  -8.3860116
##  8:  -7.46951220  -8.3047588
##  9: -11.99371594  -7.2997639
## 10: -11.70243480  -7.1601367
## 11: -11.83336725  -6.6601130
## 12:  -8.94496539  -6.6565553
## 13: -10.11175096  -6.3944009
## 14:  -6.85208353  -5.6271272
## 15:  -2.88950228  -5.3391566
## 16:  -5.96385199  -5.1871866
## 17:  -1.12321113  -5.0582897
## 18:  -2.17929482  -4.9749688
## 19:  -2.08817057  -4.9281496
## 20:  -7.01085851  -4.8855991
## 21:  -1.72428127  -4.7409766
## 22:  -1.69455791  -4.7256730
## 23:  -0.45071705  -4.7083754
## 24:  -8.38414634  -4.6217345
## 25:   0.06939235  -4.4369463
## 26:   0.35488959  -4.2876538
## 27:   0.52376914  -4.1992424
## 28:  -3.77272727  -4.0939733
## 29:  -5.84111086  -3.7632422
## 30:  -2.79333283  -3.6015956
## 31:  -2.75994746  -3.5847703
## 32:   1.73514123  -3.5628558
## 33:  -5.15787898  -3.4277906
## 34:  -2.28414534  -3.3446815
## 35:  -2.25556389  -3.3302416
## 36:  -2.11946370  -3.2614534
## 37:   2.33495849  -3.2462936
## 38:  -4.65678920  -3.1810735
## 39:  -4.42482541  -3.0666636
## 40:  -4.31782334  -3.0138449
## 41:   1.81838362  -2.9009520
## 42:   3.04384113  -2.8709147
## 43:  -2.91474503  -2.8586311
## 44:  -1.26147494  -2.8267466
## 45:  -2.42658775  -2.6143956
## 46:  -2.24971317  -2.5257579
## 47:  -1.98475289  -2.3928339
## 48:  -1.94486122  -2.3728063
```

```
## 49:   3.75163920  -1.8828241
## 50:  -2.72236131  -1.4552767
## 51:  -2.38813149  -1.2900887
## 52:  -2.11548676  -1.1551384
## 53:  -2.11400280  -1.1544034
## 54:  -1.70931644  -0.9537627
## 55:  -1.55476369  -0.8770312
## 56:   6.25000000  -0.5518214
## 57:   1.67682927  -0.5379155
## 58:   6.35770043  -0.4940476
## 59:   0.04811856  -0.4695649
## 60:   0.00000000   0.0000000
## 61:   0.00000000   0.0000000
## 62:   0.00000000   0.0000000
## 63:   0.00000000   0.0000000
## 64:   0.00000000   0.0000000
## 65:   0.00000000   0.0000000
## 66:   0.00000000   0.0000000
## 67:   0.00000000   0.0000000
## 68:   0.00000000   0.0000000
## 69:   0.00000000   0.0000000
## 70:   0.00000000   0.0000000
## 71:   0.88659559   0.3428783
## 72:   9.37349629   1.1374253
## 73:   3.49962491   1.2803662
## 74:   3.95902961   1.5156315
## 75:   4.09875570   1.5872980
## 76:   4.20737395   1.6430451
## 77:   4.54207451   1.8150257
## 78:   5.63710634   2.3798074
## 79:  12.38636364   2.7946526
## 80:   6.93181818   3.0518224
## 81:   6.70731707   3.3134789
## 82:  14.63636364   3.4746569
## 83:   7.34090909   3.6423588
## 84:  15.17659513   3.7805763
## 85:   7.78634199   3.8742441
## 86:   9.22515537   4.2537459
## 87:   9.39898999   4.3454673
## 88:  11.65431624   5.9118757
##               PI         Isc
```

```
gdIsc <- FinalOutPutfigZ %>%
        group_by(rn, UpperBound) %>%
        #group_by(UpperBound) %>%
        summarise(Isc = mean(Isc))
```

```
## `summarise()` regrouping output by 'rn' (override with `.groups` argument)
```

```
gdVr <- gdIsc %>%
  group_by(rn) %>%
  summarize(Vr = auc(UpperBound, Isc, type = "spline")/(max(UpperBound) - min(UpperBound)))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
gdPI <- FinalOutPutfigZ %>%
        group_by(rn) %>%
        summarise(PI = mean(PI))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
AUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == max(FinalOutPutfigZ$UpperBound)) %>%
    mutate(rAUC = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select(c(-1,-4, -5, -6, -7, -8))
AUCfigZ
```

```
##         mumax        K      rAUC
## 1: 0.04586753 1.308333  94.48125
## 2: 0.06753609 1.311000  96.47998
## 3: 0.03554674 1.321285  90.18984
## 4: 0.06947241 1.428000  94.90709
## 5: 0.04893593 1.411000  96.45418
## 6: 0.03432187 1.330917 100.00000
## 7: 0.05130124 1.345000  96.77392
## 8: 0.03704368 1.366112  97.12042
```

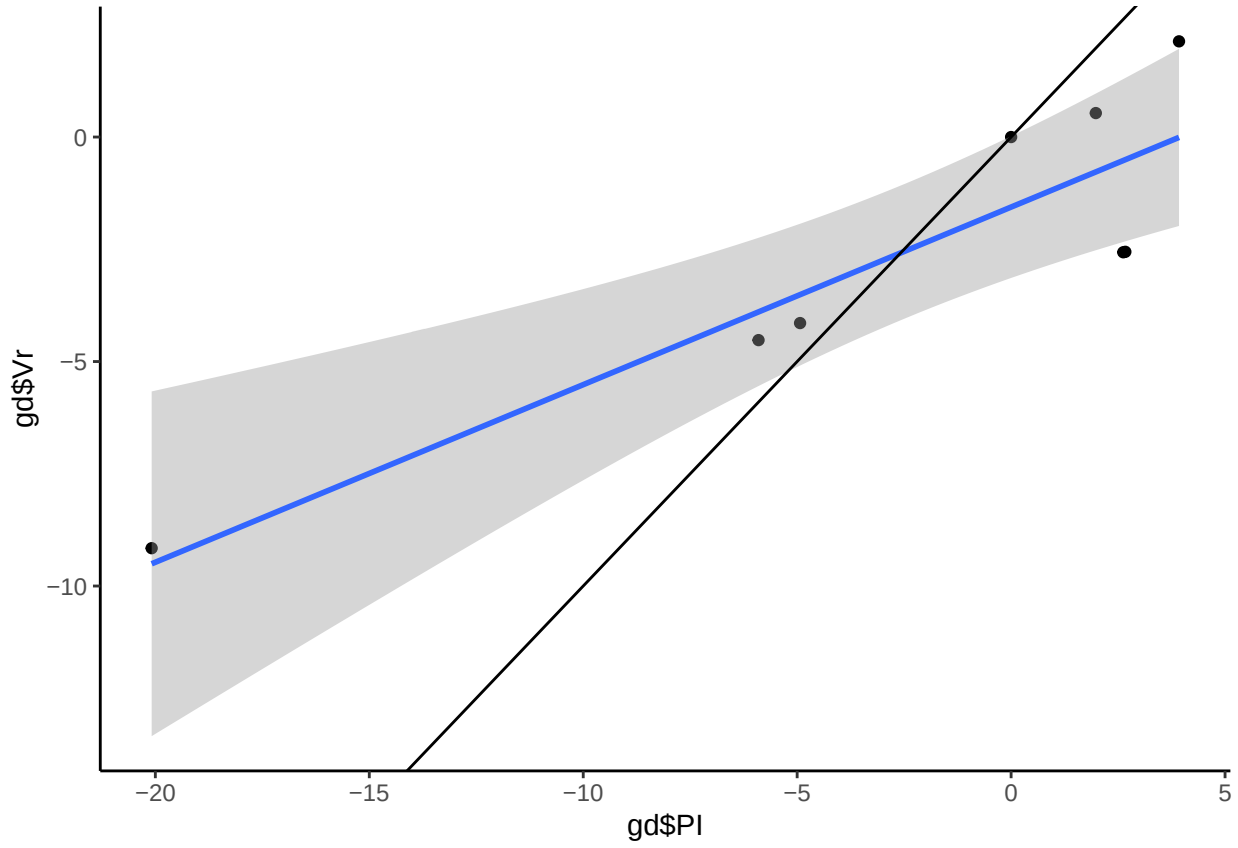
```
trimAUCfigZ <- FinalOutPutfigZ %>%
    filter(UpperBound == t_stationary) %>%
    mutate(AUCtrim = 100*(AUC/max(AUC))) %>%
    arrange(rn) %>%
    select("AUCtrim")


gd <- cbind(gdVr, gdPI, AUCfigZ, trimAUCfigZ)
gd <- gd[-3]
gd
```

```
##                rn         Vr         PI      mumax        K      rAUC   AUCtrim
## 1 0000000S150CTL1  0.0000000   0.000000 0.04586753 1.308333  94.48125  82.97379
## 2 0000000S150CTL2  0.5338219   1.979220 0.06753609 1.311000  96.47998  82.23815
## 3 0000000S150CTL3  2.1312889   3.923991 0.03554674 1.321285  90.18984  75.31933
## 4 000000S150SSV9A -2.5668192   2.621046 0.06947241 1.428000  94.90709  83.90576
## 5 000000S150SSV9B -2.5562872   2.669252 0.04893593 1.411000  96.45418  79.86091
## 6 000000S150SSV9C -9.1572436 -20.086151 0.03432187 1.330917 100.00000 100.00000
## 7 00000S150CTLAVG -4.5228601  -5.903211 0.05130124 1.345000  96.77392  91.36389
## 8 0000S150SSV9AVG -4.1435730  -4.931951 0.03704368 1.366112  97.12042  87.92223
```

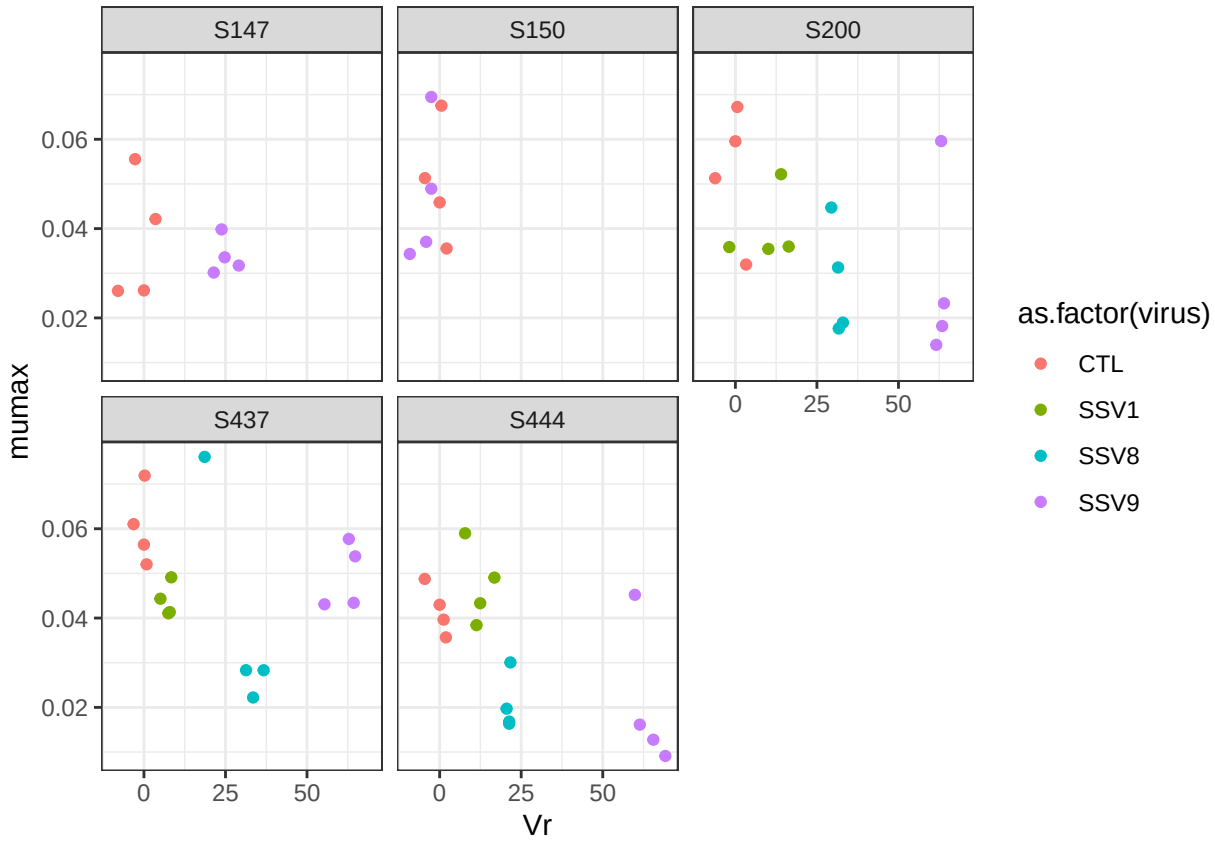
```
gd$virus <- c(
    rep("CTL", 3),
    rep("SSV9", 3),
    rep("CTL", 1),
    rep("SSV9", 1)
)
```

```r
# gd$virus <- c(
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("asnC", 12),
#     rep("ura3_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("ura3_2", 12),
#     rep("idr1_1", 12),
#     rep("idr2_1", 12),
#     rep("sirR_1", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
#     rep("rosR", 12),
#     rep("ura3_3", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12),
#     rep("trmB", 12),
#     rep("VNG1179", 12),
#     rep("ura3_4", 12)
# )

gd$host <- rep("S150", 8)



gd150 <- gd


ggplot(data = NULL, aes(x = gd$PI, y = gd$Vr)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_abline() +
  theme_classic()
```

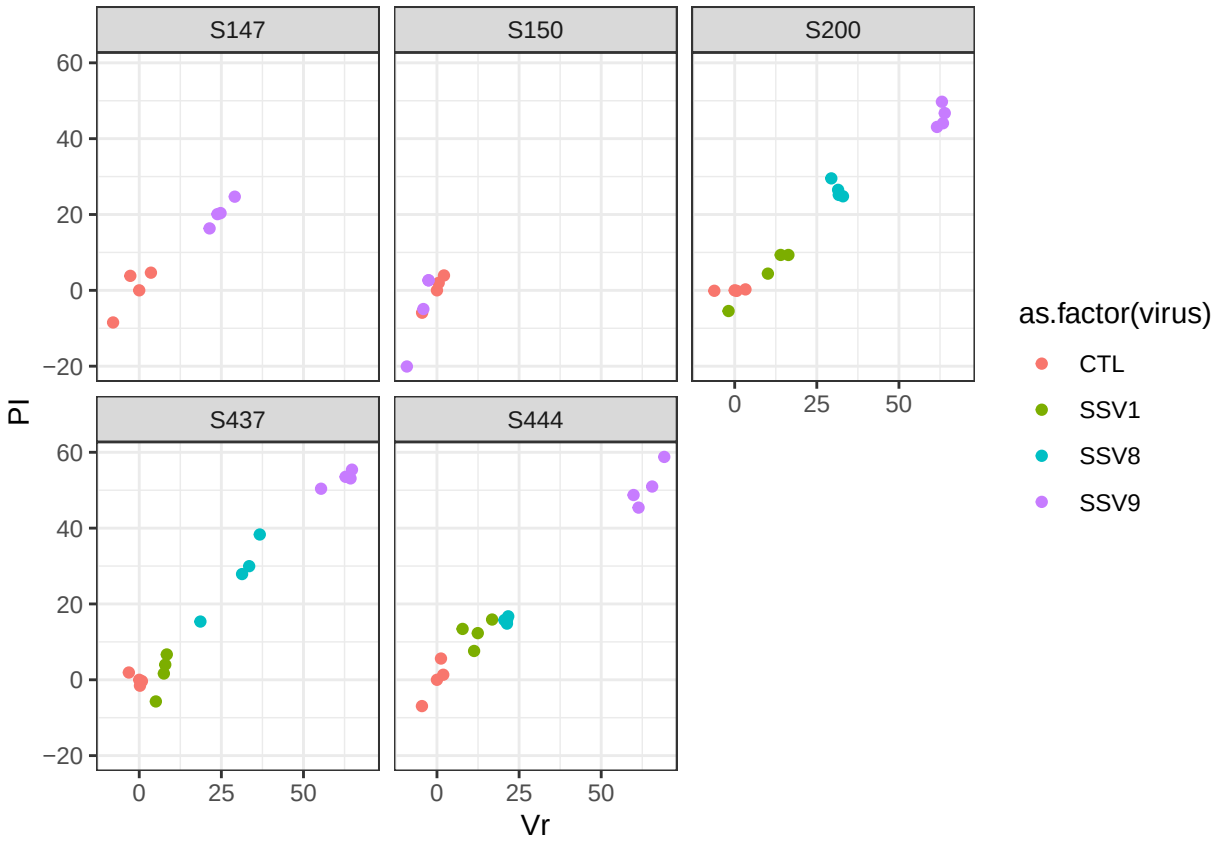
```
## `geom_smooth()` using formula 'y ~ x'
```



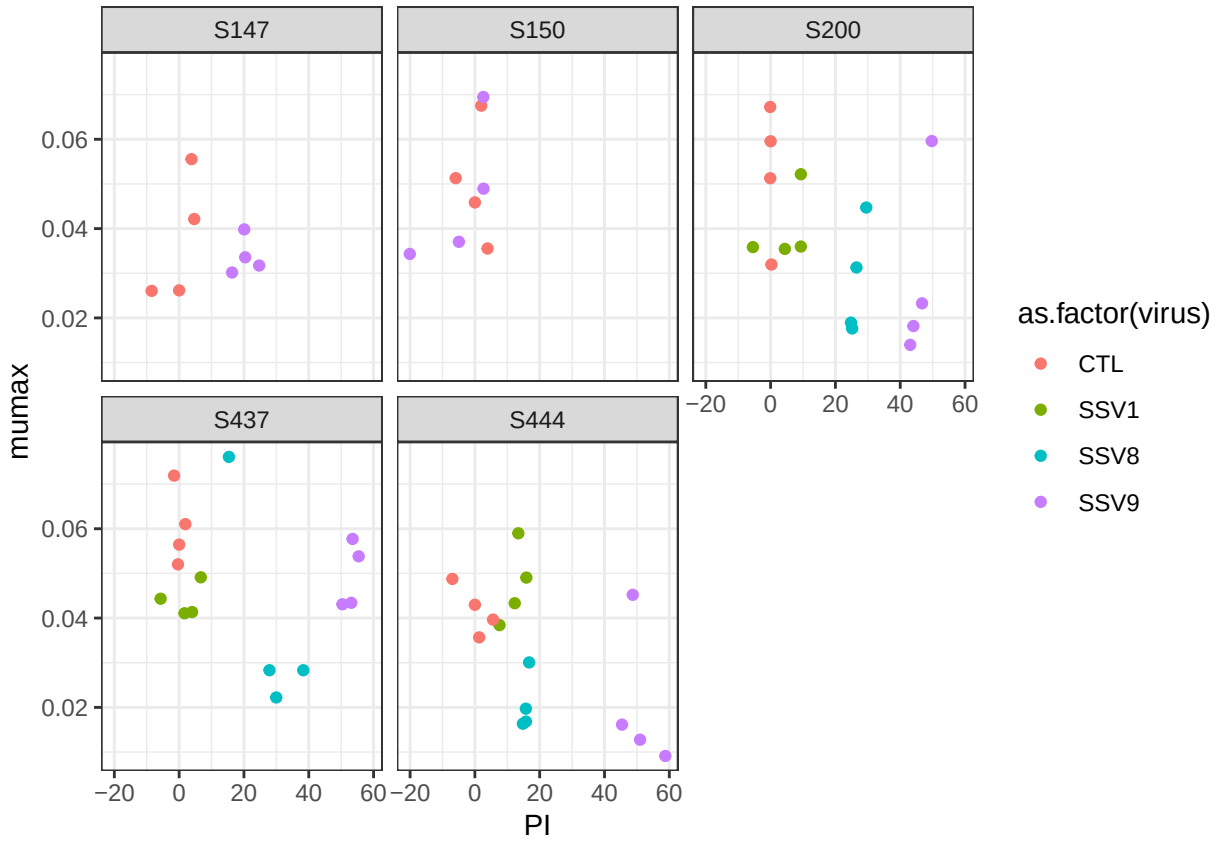

```
gd <- rbind(gd437, gd444, gd200, gd147, gd150)

ggplot(data = gd, aes(x = Vr, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



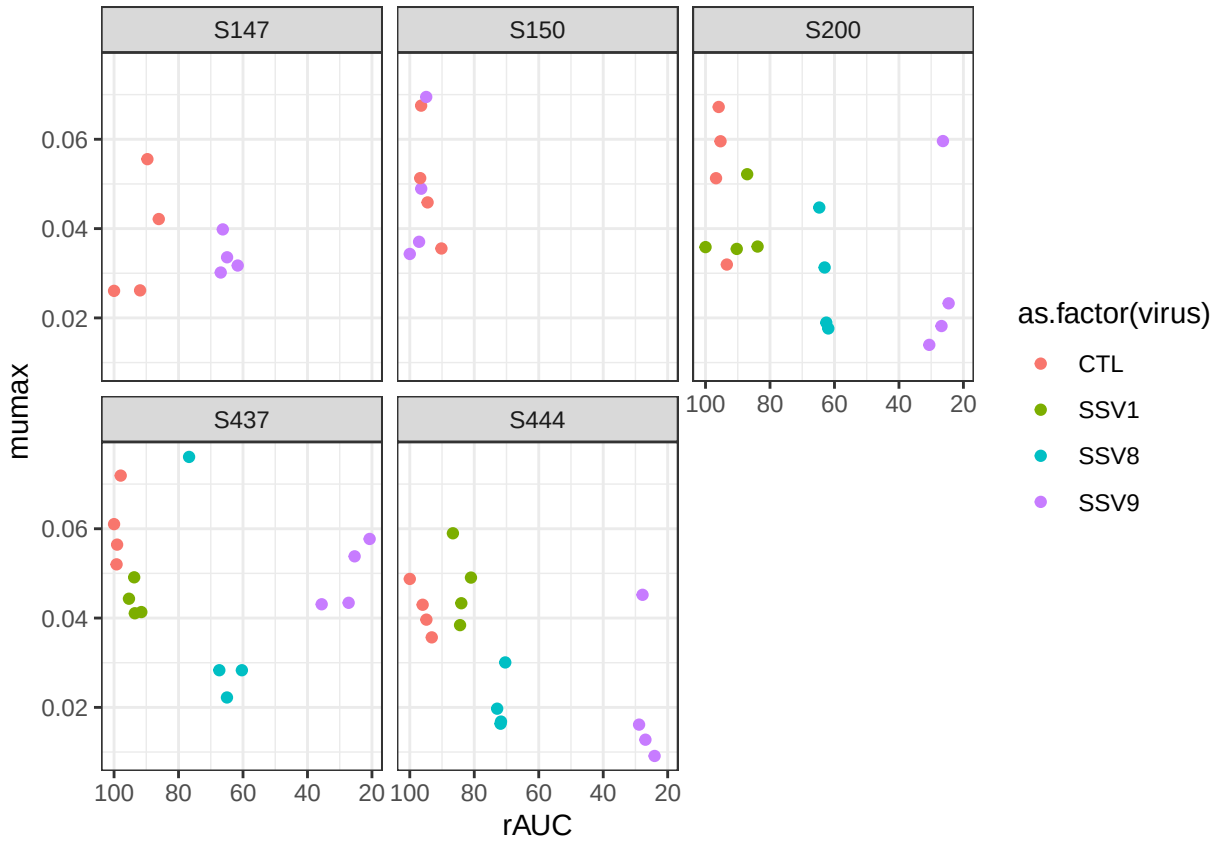

```
ggplot(data = gd, aes(y = PI, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



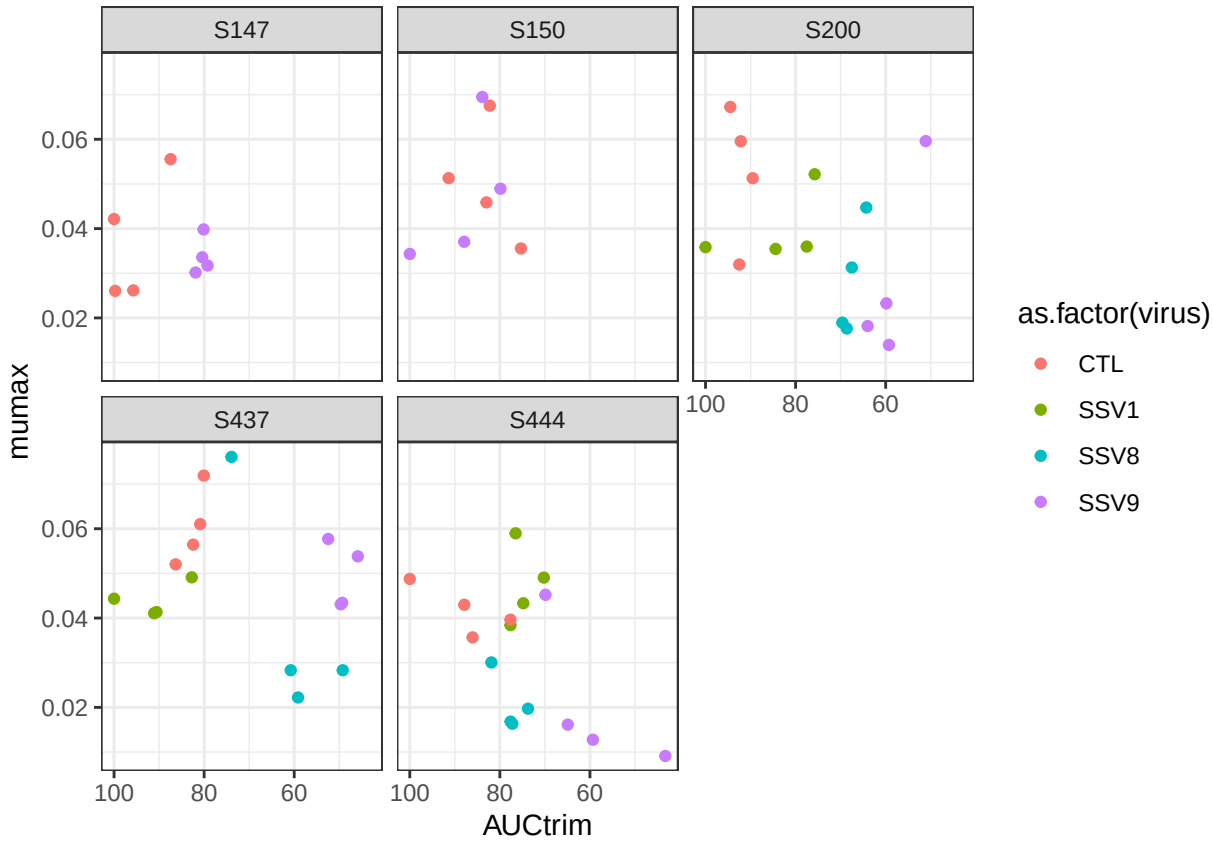

```
ggplot(data = gd, aes(y = mumax, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



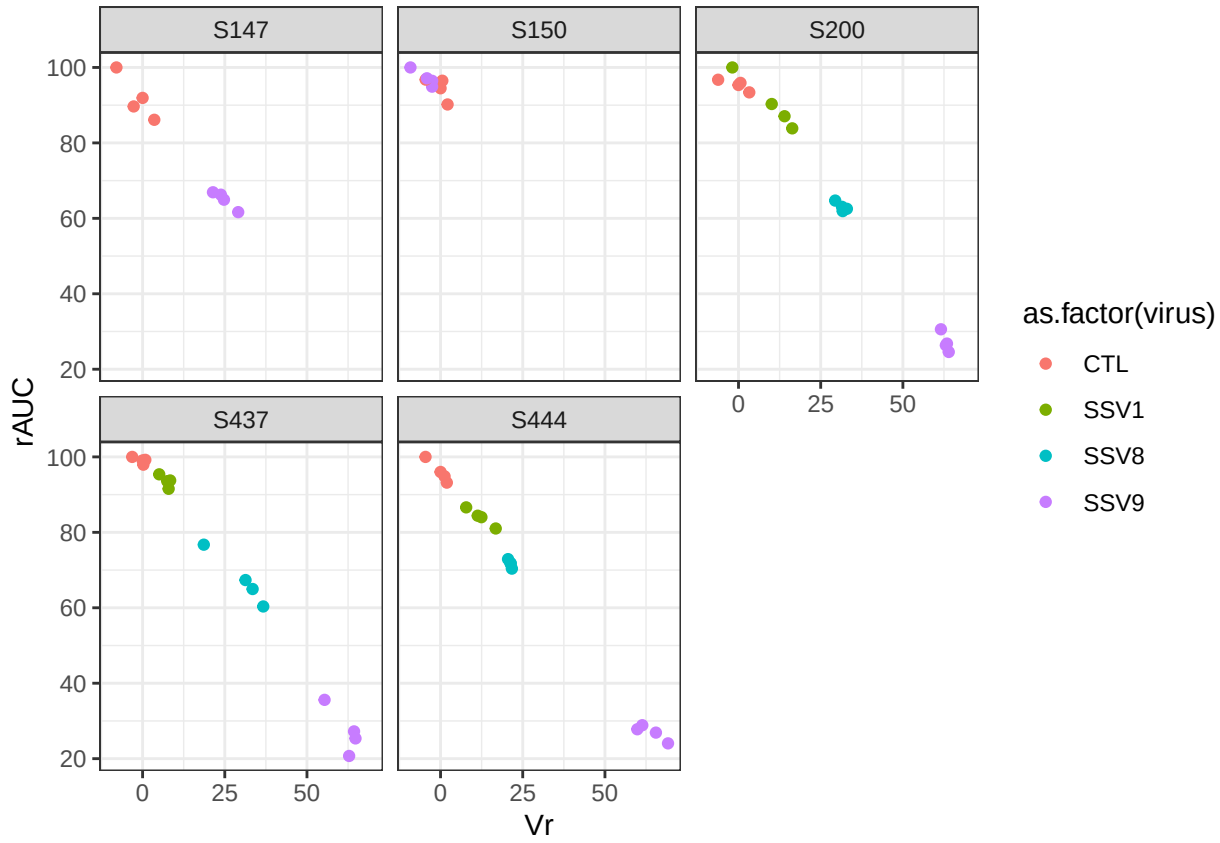

```
ggplot(data = gd, aes(y = mumax, x = rAUC, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



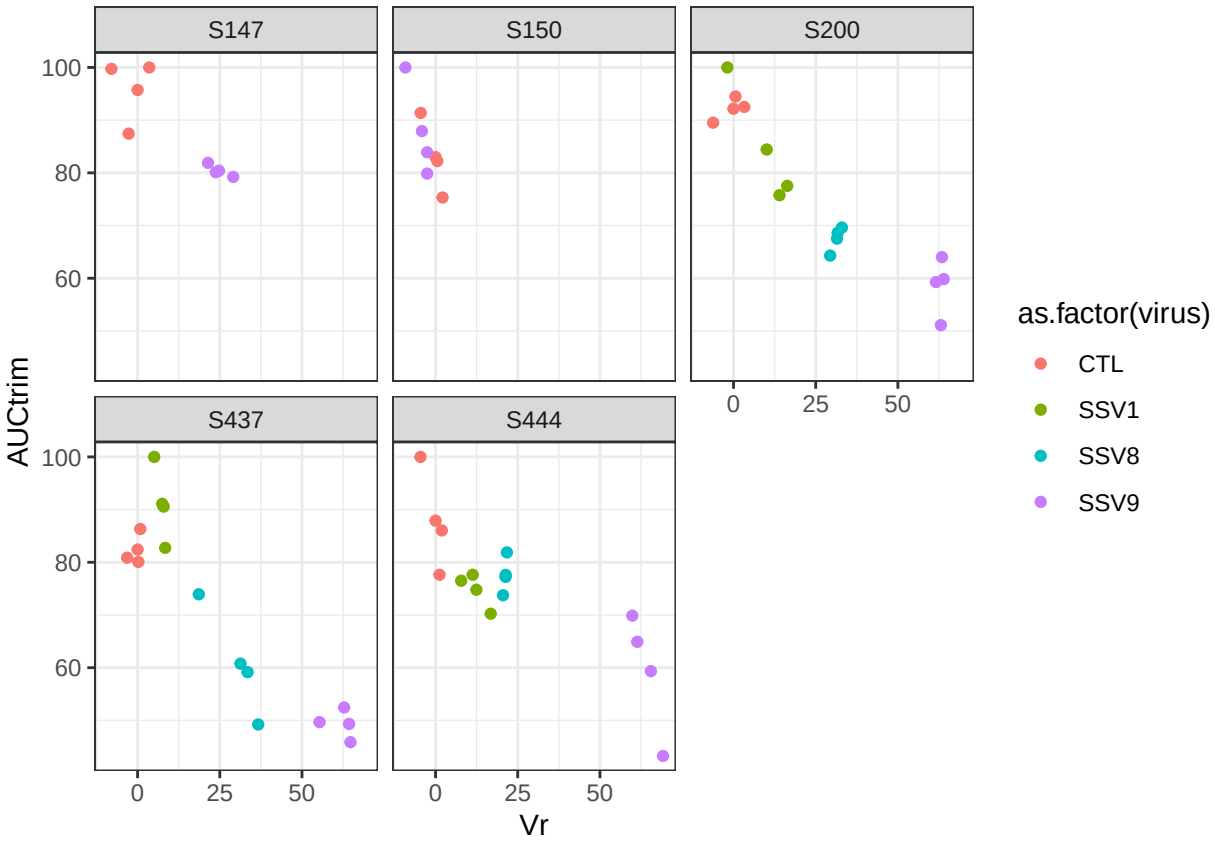

```
ggplot(data = gd, aes(y = mumax, x = AUCtrim, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```



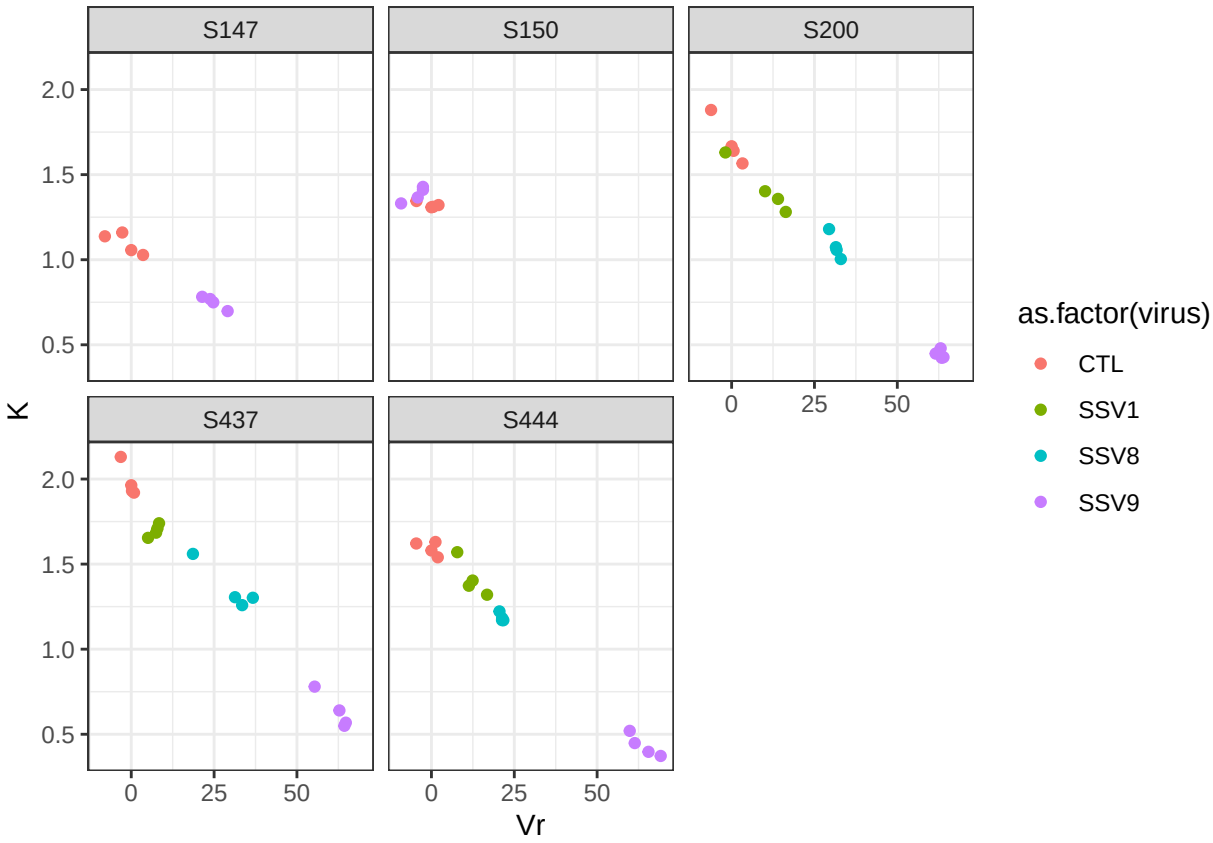

```
ggplot(data = gd, aes(y = rAUC, x = Vr, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



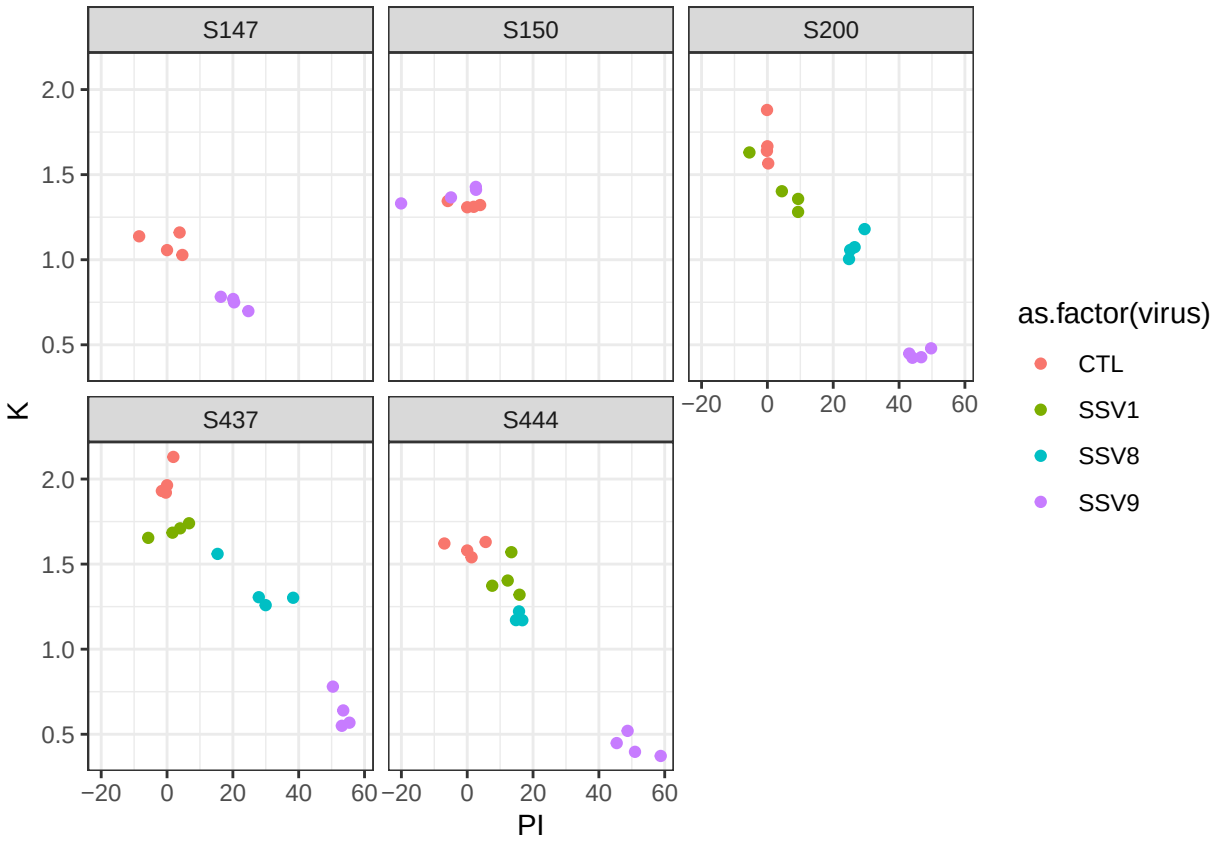

```
ggplot(data = gd, aes(y = AUCtrim, x = Vr, color =  as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



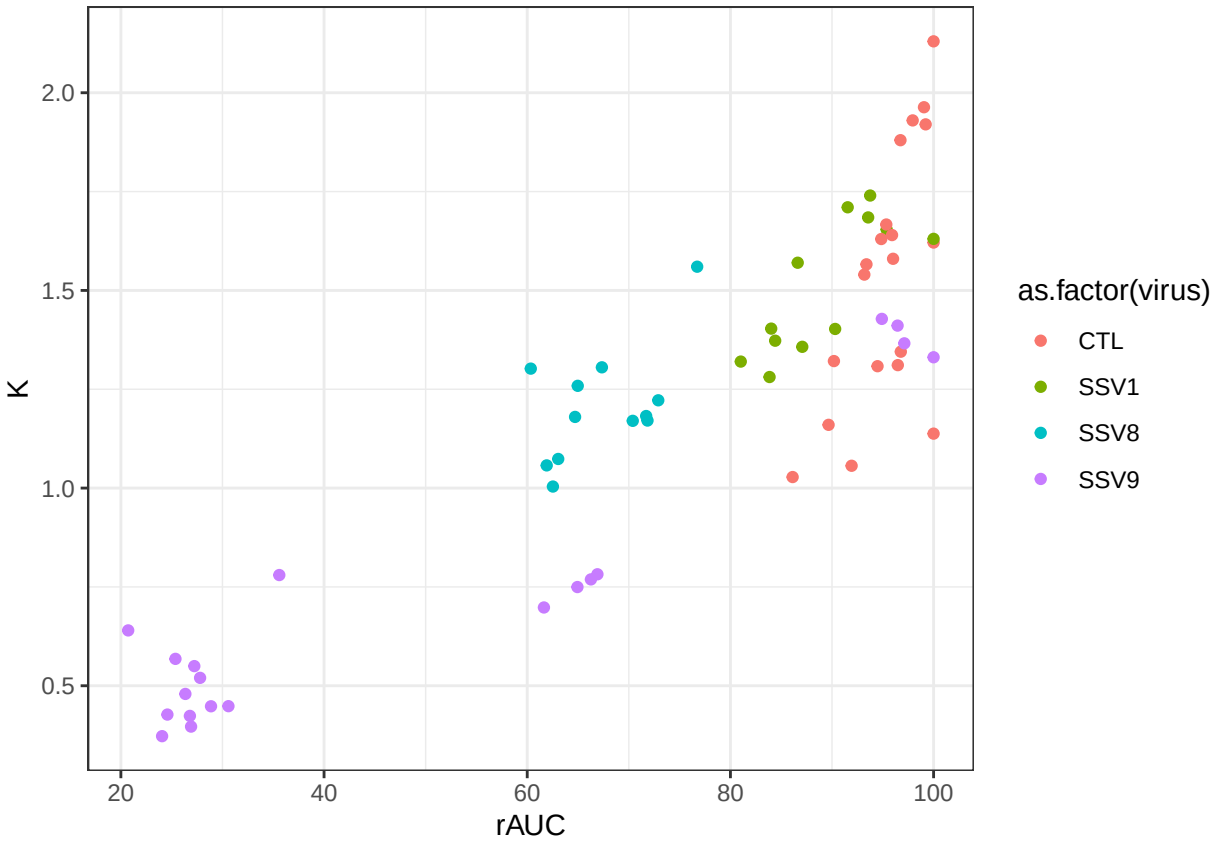

```
ggplot(data = gd, aes(y = K, x = Vr, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



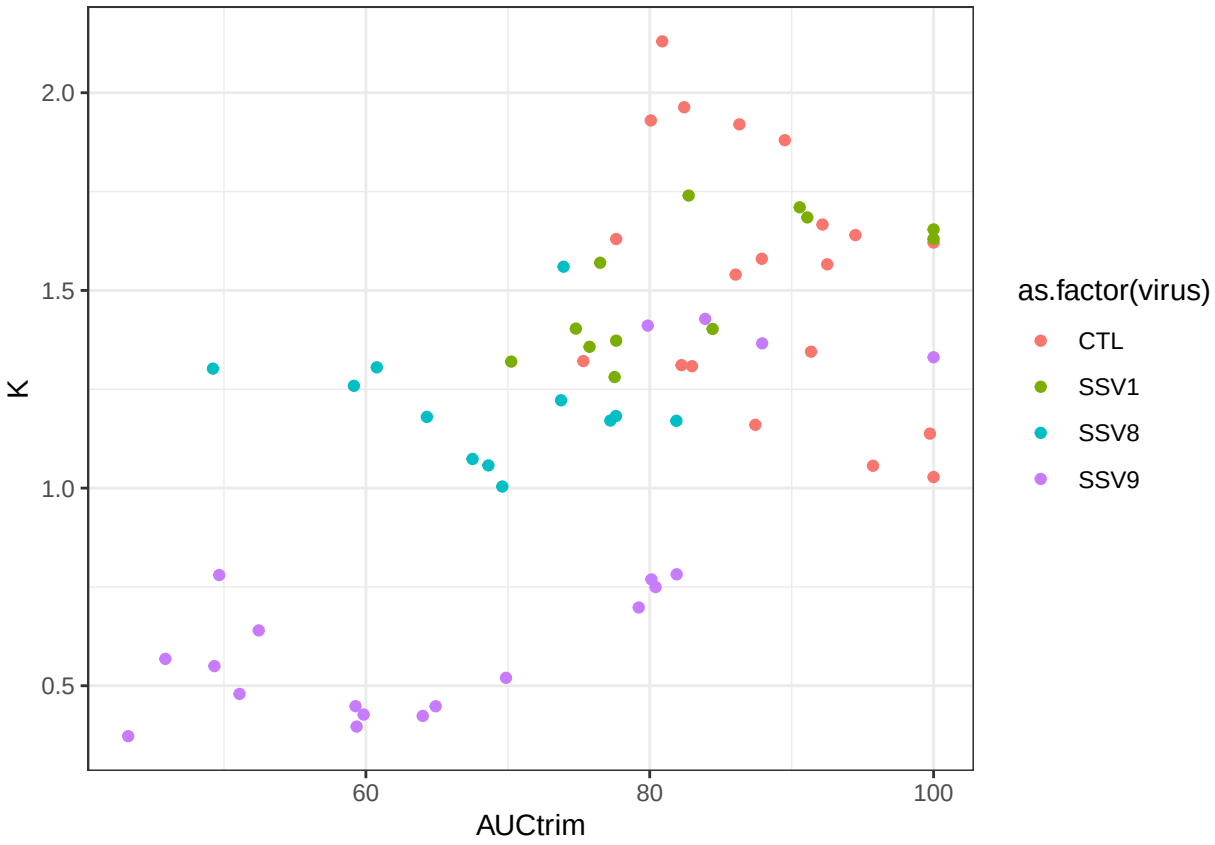

```
ggplot(data = gd, aes(y = K, x = PI, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw()
```



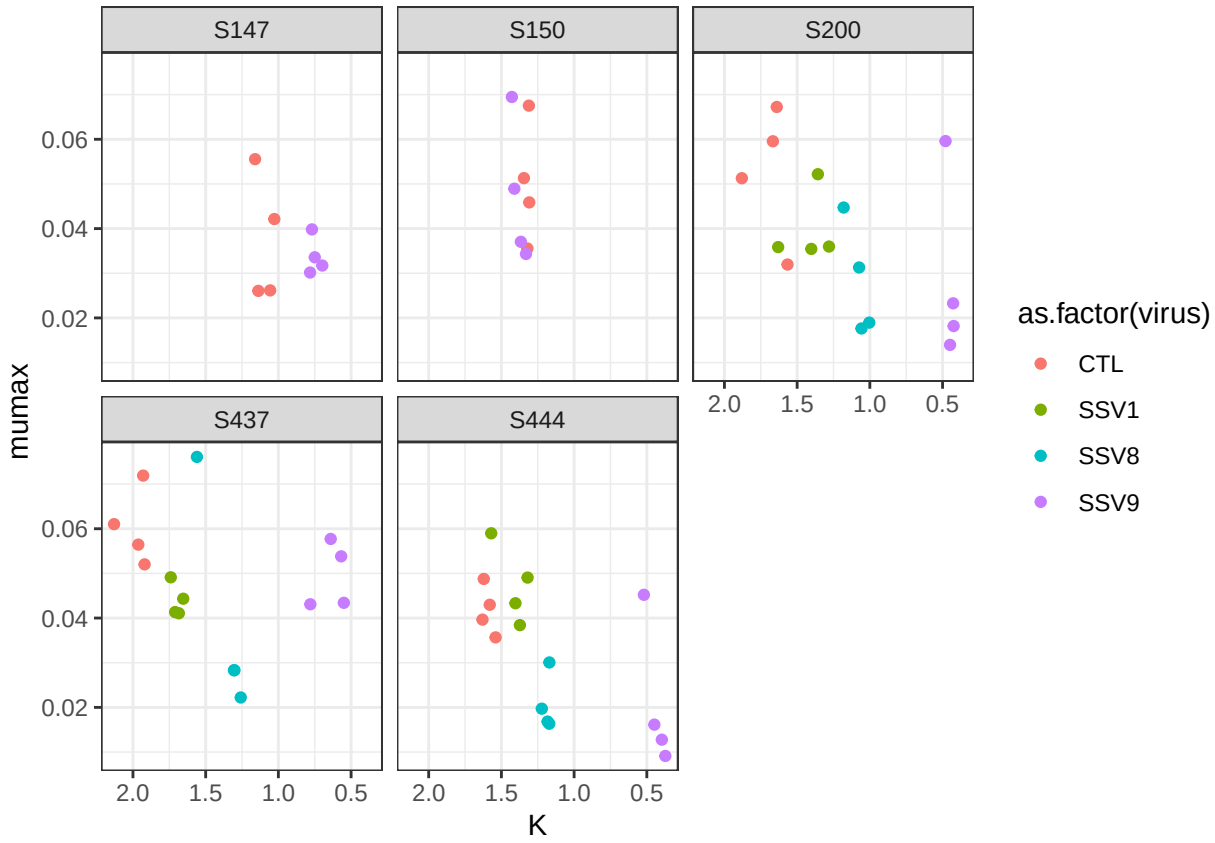

```
ggplot(data = gd, aes(y = K, x = rAUC, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```




```
ggplot(data = gd, aes(y = K, x = AUCtrim, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  # facet_wrap(~host) +
  theme_bw()
```

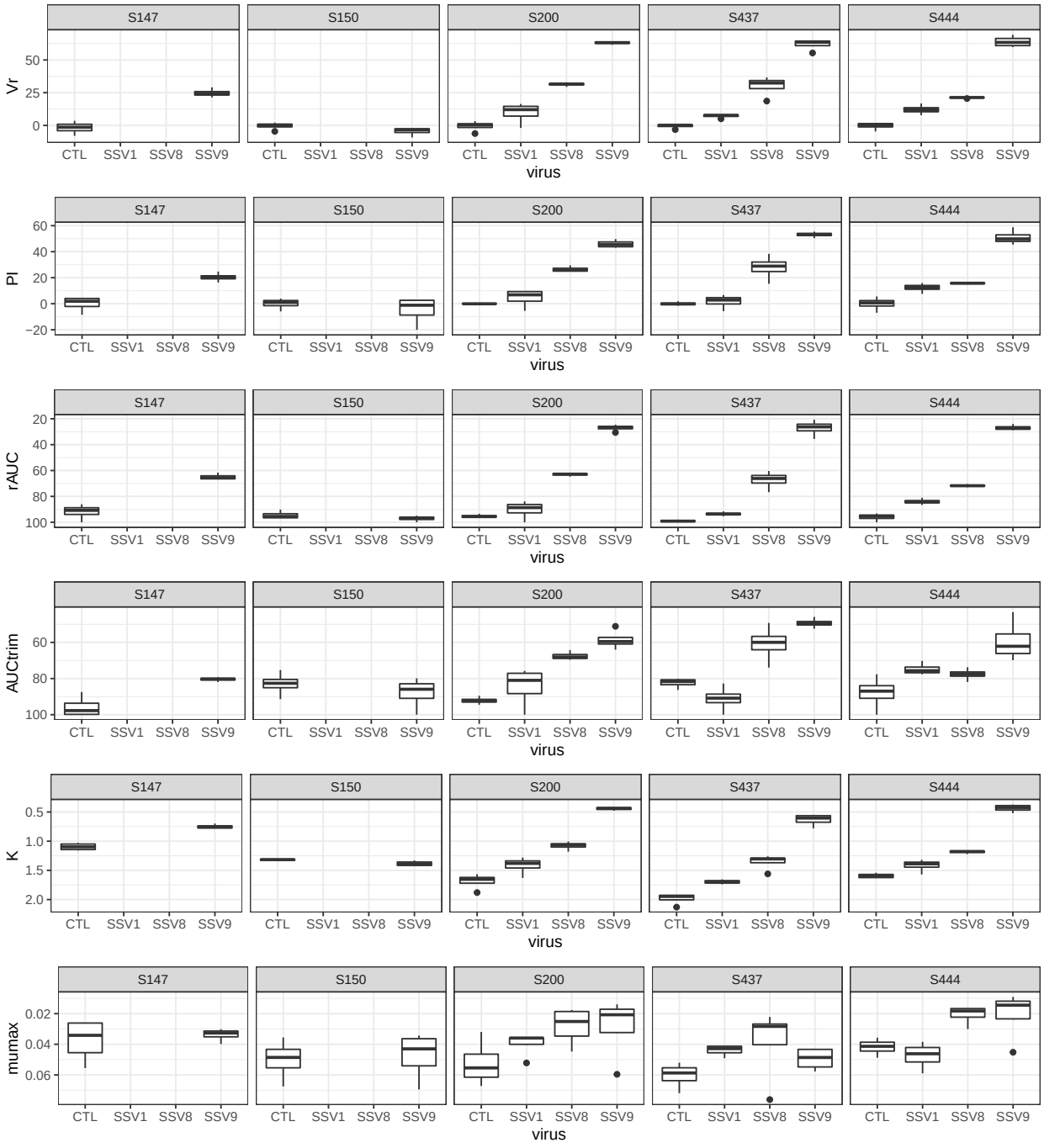



```
ggplot(data = gd, aes(x = K, y = mumax, color = as.factor(virus))) +
  geom_point() +
  #geom_smooth(method = "lm") +
  #scale_color_gradientn(colours = rainbow(6)) +
  facet_wrap(~host) +
  theme_bw() +
    scale_x_reverse()
```




```
#boxplot comparisons:

ggVr <- ggplot(data=gd, aes(y = Vr, x = virus)) +
    geom_boxplot() +
    labs(fill="") +
    facet_wrap(~host, nrow = 1) +
  theme_bw()


ggPI <- ggplot(data=gd, aes(y = PI, x = virus)) +
    geom_boxplot() +
    labs(fill="") +
    facet_wrap(~host, nrow = 1) +
  theme_bw()


ggmu <- ggplot(data=gd, aes(y = mumax, x = virus)) +
    geom_boxplot() +
    facet_wrap(~host, nrow = 1)  +
    scale_y_reverse() +
  theme_bw()


ggK <- ggplot(data=gd, aes(y = K, x = virus)) +
    geom_boxplot() +
    facet_wrap(~host, nrow = 1)  +
```

```
    scale_y_reverse() +
  theme_bw()


ggrAUC <- ggplot(data=gd, aes(y = rAUC, x = virus)) +
    geom_boxplot() +
    facet_wrap(~host, nrow = 1)  +
    scale_y_reverse() +
  theme_bw()


ggAUCt <- ggplot(data=gd, aes(y = AUCtrim, x = virus)) +
    geom_boxplot() +
    facet_wrap(~host, nrow = 1)  +
    scale_y_reverse() +
  theme_bw()


ggarrange(ggVr, ggPI, ggrAUC, ggAUCt, ggK, ggmu, ncol = 1)
```


S147
S150
S200
S437
S444
Vr
PI
rAUC
AUCtrim
K
mumax
CTL
SSV1
SSV8
SSV9
virus