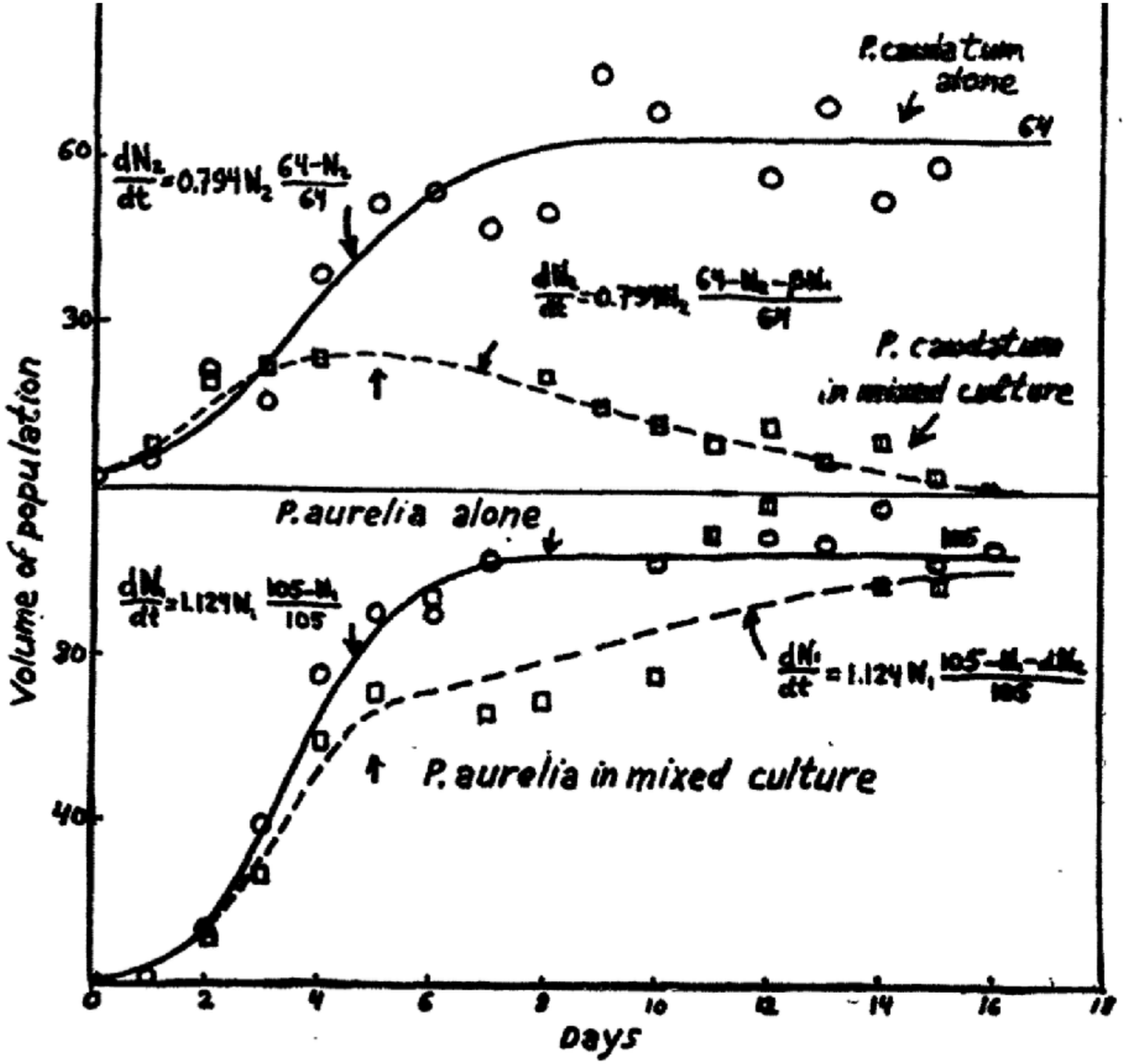
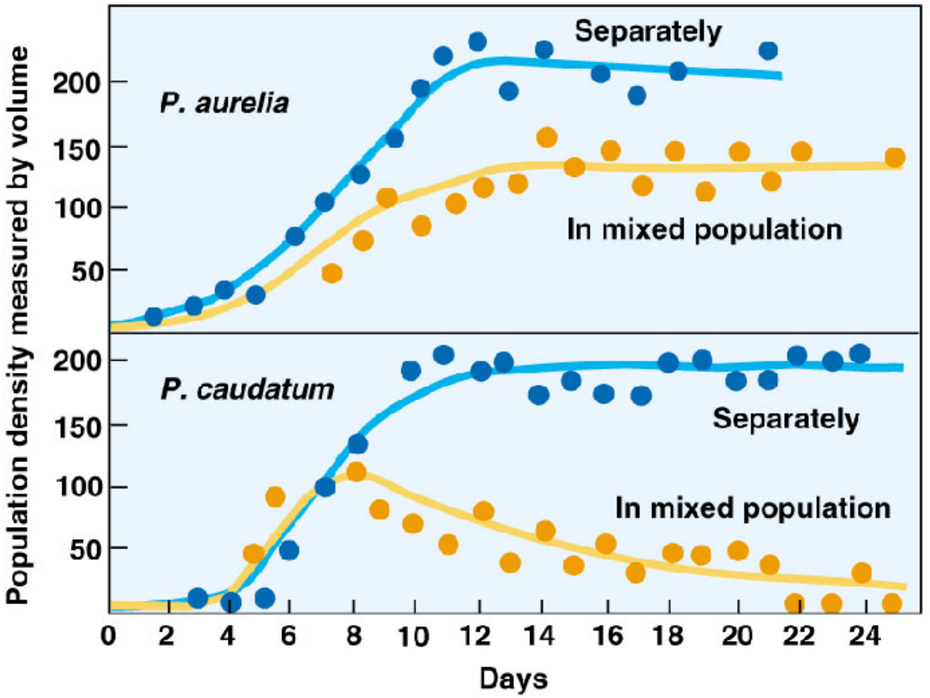
Fitting the competitive exclusion data for Gause [1934]



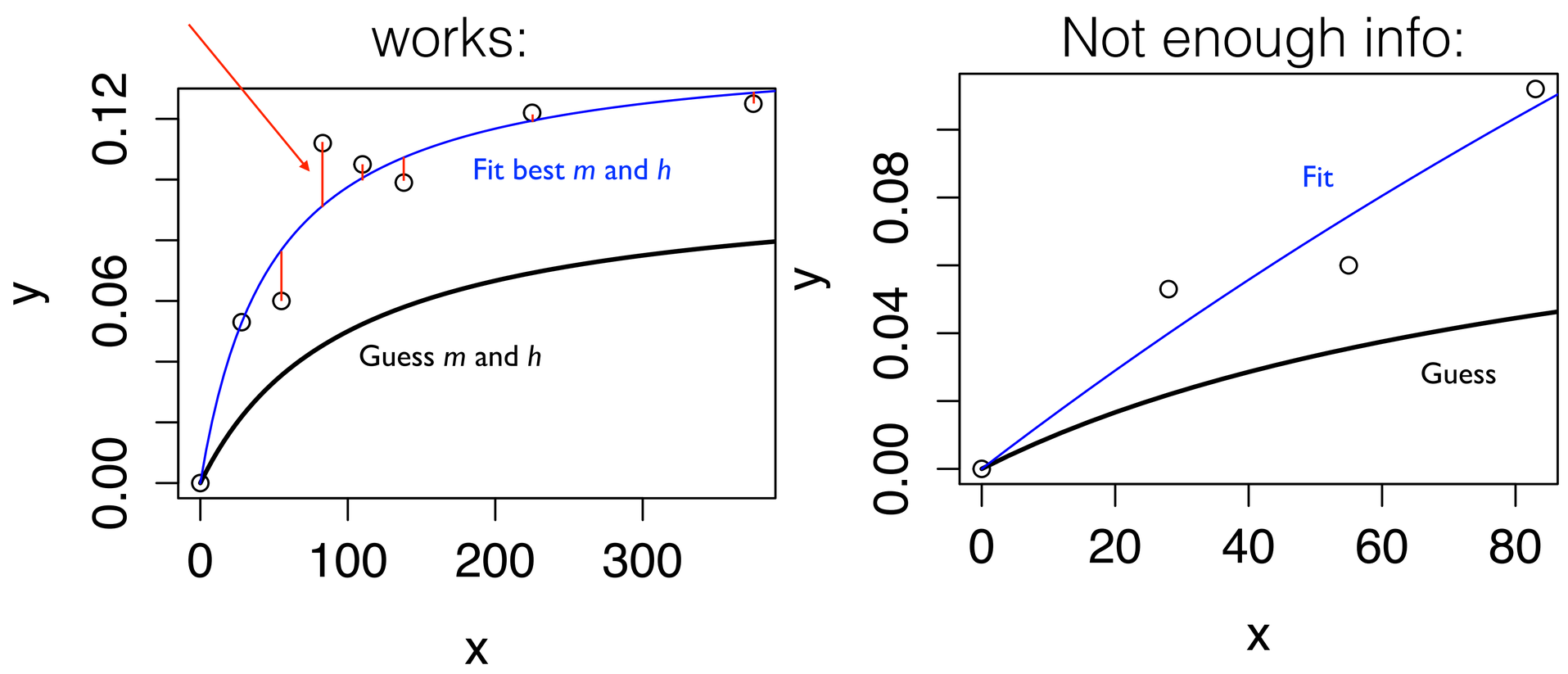
From: Gause 1934



Non linear parameter optimization (Appendix 14.7)

Define a cost function based upon distance of model prediction to the data:

$$y = \frac{mx}{h + x}$$



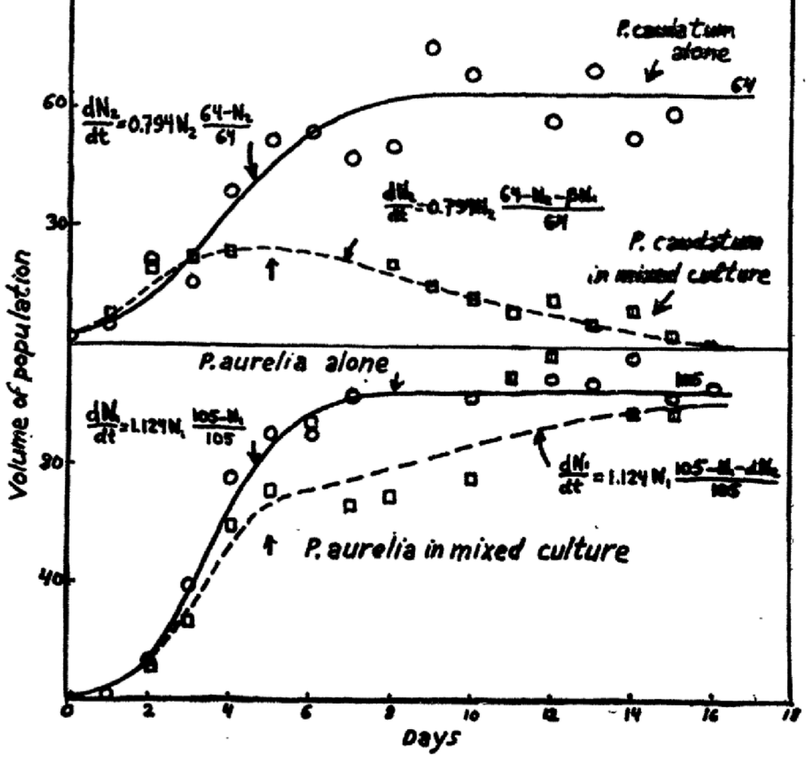
Non linear parameter optimization (Appendix 14.7)

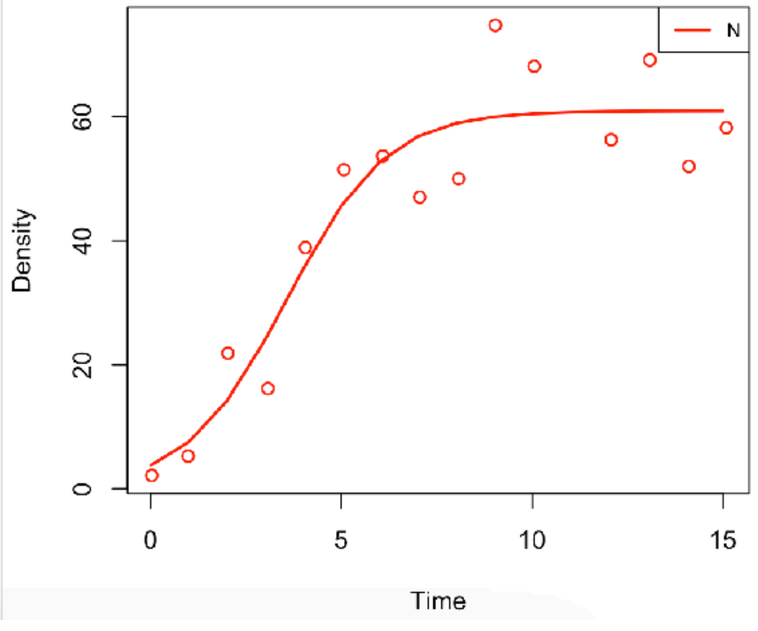
Function fit() in Grind minimizes the Summed Squared Residuals (SSR)

User needs to provide an initial guess for all parameters.
Grind gradually changes the free parameters.
(Gradient descent model: use partial derivatives)

Fitting stops at (local) optimum.

```r
model <- function(t, state, parms) {
  with(as.list(c(state,parms)), {
    dtN <- r*N*(1 - N/K)
    return(list(c(dtN)))
  })
}

p <- c(r=1,K=100)
s <- c(N=2)
run(18)

aurelia <- read.table("data/aurelia1.txt", header=TRUE)
caudatum <- read.table("data/caudatum1.txt", header=TRUE)

names(aurelia) <- c("time", "N")
names(caudatum) <- c("time", "N")

free <- c("r","K","N")
fA <- fit(aurelia,free=free)
```





```r
model <- function(t, state, parms) {
  with(as.list(c(state,parms)), {
    dtA <- rA*A*(1 - (A+alpha*C)/kA)
    dtC <- rC*C*(1 - (C+beta*A)/kC)
    return(list(c(dtA,dtC)))
  })
}

p <- c(rA=1.1,rC=0.8,kA=105,kC=64,alpha=0,beta=0)
s <- c(A=2,C=2)

free <- c("rA","kA","rC","kC")
f1 <- fit(list(aurelia1,caudatum1),free=free,add=TRUE)
p[free] <- f1$par;p
```