Appendix 1. R code for *j*-factor experiments.

The following code can be copied into a text file and renamed “jfactor.Rnw”. This script requires a space delimited input file (the example “jf-211mer-75nM-ligase100UmL.txt” is included in Appendix 3). Execution of this script requires running the R command: `R CMD Sweave jfactor.Rnw`, which performs the fitting and generates the output plots of the fits as “.eps” files, followed by running the latex command: `latex jfactor.tex`, which generates the final output “jfactor.dvi” which can be saved as a “.pdf” file.

```
\documentclass[11pt]{amsart}
\usepackage[margin=1in]{geometry}
\geometry{letterpaper,landscape}
\usepackage{graphicx,amssymb,epstopdf}
\usepackage{/Library/Frameworks/R.framework/Resources/share/texmf/Sweave}
\pagestyle{empty}
\begin{document}
\thispagestyle{empty}

<<results=hide,echo=FALSE>>=

###### to run script :::  Sweave("jfactor.Rnw") then will need to run latex on jfactor.tex  ##############

library("deSolve")
library("xtable")
library("numDeriv")

###### labels and file names ################################################################

label <- "jf-211mer-75nM-ligase100UmL"
file <- "jf-211mer-75nM-ligase100UmL.txt"
### "file.txt" contains the following columns:  time, M, C, W, D, T, Q, P (where the first row is time zero)
full <- paste(label,"-full.eps",sep="")
zoom <- paste(label,"-zoom.eps",sep="")

###### starting values for optimization ####################################################

start <- c(3.8e-03, 6.2e-02, 2.7e-02, 1.3e-02, 3.1e-09, 3.0, 75)

###### reading in data #####################################################################

dat <- NULL
dat <- read.table(file=file, header=FALSE)
dat <- as.data.frame(dat)

n <- dim(dat)[1]
time <- dat[,1]                          ### timepoints in minutes
timesh <- dat[2:n,1]
conc <- dat[1,2:8]
M0 <- as.numeric(conc[1])                ### initial concentration in nM
M <- dat[2:n,2]
C <- dat[2:n,3]
W <- dat[2:n,4]
D <- dat[2:n,5]
T <- dat[2:n,6]
```

```
Q <- dat[2:n,7]
P <- dat[2:n,8]

###### FUNCTIONS ################################################################
###### ODE function ##############################################################

odeFUN <- function(t,y,pars){
        with(as.list(c(y,pars)),{
                kc1=pars[1]
                kc2=pars[2]
                kc3=pars[3]
                kc4=pars[4]
                kc5=pars[5]
                j1=pars[6]
                kd <- kc1/j1

# rates of changes
dM      = -4*kd*M^2 -4*kd*M*D -4*kd*M*T -4*kd*M*Q -kc1*M -4*kd*M*P
dD      = 2*kd*M^2 -4*kd*M*D -4*kd*D^2 -4*kd*D*T -kc2*D -4*kd*D*Q -4*kd*D*P
dT      = 4*kd*M*D -4*kd*M*T -4*kd*D*T -kc3*T -4*kd*T^2 -4*kd*T*Q -4*kd*T*P
dQ      = 2*kd*D^2 +4*kd*M*T -4*kd*M*Q -kc4*Q -4*kd*D*Q -4*kd*T*Q -4*kd*Q^2 -4*kd*Q*P
dP      = 4*kd*M*Q +4*kd*D*T -kc5*P -4*kd*M*P -4*kd*D*P -4*kd*T*P -4*kd*Q*P -4*kd*P^2
dCm     = kc1*M
dCd     = kc2*D
dCt     = kc3*T
dCq     = kc4*Q
dCp     = kc5*P

                return(list(dy=c(dM,dD,dT,dQ,dP,dCm,dCd,dCt,dCq,dCp)))})}

###### optimization function #####################################################

optFUN <- function(names,start,time,n,M0,M,C,D,T,Q,P,W){

        ###### model function ####################################################

                modelFUN<-function(s){

                        parms <- s[1:6]

                        M0par <- s[7]
                        times <- time

                        y <- c(M=M0par, D=0, T=0, Q=0, P=0, Cm=0, Cd=0, Ct=0, Cq=0, Cp=0)
                        ODE <- as.data.frame(lsoda(y=y,times=times,func=odeFUN,parms=parms))

                        ### least squares
                        lsse = sum((M-ODE$M[2:n])^2) +sum((C-ODE$Cm[2:n])^2) +sum((D-
2*ODE$D[2:n])^2) +sum((T-3*ODE$T[2:n])^2) +sum((Q-4*ODE$Q[2:n])^2) +sum((P-
5*ODE$P[2:n])^2) +sum((W-(2*ODE$Cd[2:n]+3*ODE$Ct[2:n]+4*ODE$Cq[2:n]+5*ODE$Cp[2:n]))^2)
                        if(is.na(lsse)){lsse <- 1e6}

                        lsse}

        ###### output results into csv file and LaTeX table ######################################
```

```
	nlminbFit <- nlminb(start,modelFUN,lower=0,upper=Inf)
	out <- NULL
	out <- c(nlminbFit$par,nlminbFit$objective)
	out <- as.data.frame(out)
	hessian <- hessian(modelFUN,nlminbFit$par)
	FIM <- solve(2*hessian)
	se <- sqrt(abs(diag(FIM)))
	outnames <- c("cost function","kc1","kc2","kc3","kc4","kc5","j1", "M0")
	out <- cbind(outnames,out,c(NA,se))
	names(out) <- c("Parameter","Estimate","Std.Error")
	csv <- paste(label,"-summary.csv",sep="")
	write.table(out,file=csv,sep=",",row.names=FALSE,col.names=TRUE,append=FALSE)
	xtab <- xtable(out,digits=3)

	###### generating best fit data for plots ###############################################

	parmsfit <- nlminbFit$par
	maxtime <- max(time)
	times2 <- seq(0,maxtime,length=1000)
	y2 <- c(M=parmsfit[7], D=0, T=0, Q=0, P=0, Cm=0, Cd=0, Ct=0, Cq=0, Cp=0)
	ODEfit <- as.data.frame(lsoda(y=y2,times=times2,func=odeFUN,parms=parmsfit[1:6]))

	##################################################################################

	return <- NULL
	return[[1]] <- xtab
	return[[2]] <- ODEfit
	return[[3]] <- out

	return}

###### function for generating eps plots ####################################################

epsFUN <- function(full,zoom,timesh,M0,M,C,D,T,Q,P,W,ODEfit){

	cexvar <- 1.75
	mexvar <- 1.1
	lwdv <- 3
	lwdv2 <- 2

	xmax <- max(timesh)
	ymax1 <- round(M0*1.06)
	ymax2 <- round(ymax1*0.25)

	postscript(file=full, paper="letter", horizontal=TRUE, family="sans")

	par(cex.axis=cexvar, cex.lab=cexvar, mex=mexvar, font.axis=2, font.lab=2, mar=c(5, 7, 4, 2) +
0.7, tck=0.02)

	plot(timesh,M,xlim=c(0,xmax),ylim=c(0,ymax1),xlab="time (minutes)",ylab="concentration
(nM)", pch=16, col="tan3", cex=cexvar,axes=FALSE)

	box(lwd=lwdv)
	axis(1,at=seq(0,xmax,by=round(xmax*0.25)),labels=TRUE,lwd=lwdv,las=1)
	axis(2,at=seq(0,ymax1,by=round(ymax1*0.25)),labels=TRUE,lwd=lwdv,las=1)
	axis(3,at=seq(0,xmax,by=round(xmax*0.25)),labels=FALSE,lwd=lwdv)
	axis(4,at=seq(0,ymax1,by=round(ymax1*0.25)),labels=FALSE,lwd=lwdv)

	points(ODEfit$time, ODEfit$M, pch=16, type="l",col="tan3", lwd=lwdv2)
	points(timesh,C,pch=16,cex=cexvar,col="orangered")
	points(ODEfit$time, ODEfit$Cm, pch=16, col="orangered", type="l", lwd=lwdv2)
	points(timesh,D,pch=16,cex=cexvar,col="darkblue")
	points(ODEfit$time, 2*ODEfit$D, pch=16, col="darkblue", type="l", lwd=lwdv2)
	points(timesh,W,pch=16,cex=cexvar,col="deeppink")
	points(ODEfit$time, 2*ODEfit$Cd+3*ODEfit$Ct+4*ODEfit$Cq+5*ODEfit$Cp, pch=16,
col="deeppink", type="l", lwd=lwdv2)

	legend("topright",c("linear monomer", "linear dimer", "linear trimer", "linear tetramer","linear
pentamer", "circular monomer","circular multimers"), col=c("tan3","darkblue","springgreen","gold",
"gray", "orangered","deeppink"), lty=1, bty="n", cex=cexvar, lwd=lwdv2)

	dev.off()

	#################################################################

	postscript(file=zoom, paper="letter", horizontal=TRUE, family="sans")

	par(cex.axis=cexvar, cex.lab=cexvar, mex=mexvar, font.axis=2, font.lab=2, mar=c(5, 7, 4, 2) +
0.7, tck=0.02)

	plot(timesh,M,xlim=c(0,xmax),ylim=c(0,ymax2),xlab="time (minutes)",ylab="concentration
(nM)", pch=16, col="tan3", cex=cexvar, axes=FALSE, type="n")

	box(lwd=lwdv)
	axis(1,at=seq(0,xmax,by=round(xmax*0.25)),labels=TRUE,lwd=lwdv,las=1)
	axis(2,at=seq(0,ymax2,by=round(ymax2*0.25)),labels=TRUE,lwd=lwdv,las=1)
	axis(3,at=seq(0,xmax,by=round(xmax*0.25)),labels=FALSE,lwd=lwdv)
	axis(4,at=seq(0,ymax2,by=round(ymax2*0.25)),labels=FALSE,lwd=lwdv)

	points(timesh,C,pch=16,cex=cexvar,col="orangered")
	points(ODEfit$time, ODEfit$Cm, pch=16, col="orangered", type="l", lwd=lwdv2)
	points(timesh,T,pch=16,cex=cexvar,col="springgreen")
	points(ODEfit$time, 3*ODEfit$T, pch=16, col="springgreen", type="l", lwd=lwdv2)
	points(timesh,Q,pch=16,cex=cexvar,col="gold")
	points(ODEfit$time, 4*ODEfit$Q, pch=16, col="gold", type="l", lwd=lwdv2)
	points(timesh,P,pch=16,cex=cexvar,col="gray")
	points(ODEfit$time, 5*ODEfit$P, pch=16, col="gray", type="l", lwd=lwdv2)

	legend("topright",c("linear monomer", "linear dimer", "linear trimer", "linear tetramer","linear
pentamer", "circular monomer","circular multimers"), col=c("tan3","darkblue","springgreen","gold",
"gray", "orangered","deeppink"), lty=1, bty="n", cex=cexvar, lwd=lwdv2)

	dev.off()}

###### performing optimization and generating plots ###########################################

optimize <- optFUN(label,start,time,n,M0,M,C,D,T,Q,P,W)
tab <- optimize[[1]]
ODEfit <- optimize[[2]]
epsFUN(full,zoom,timesh,M0,M,C,D,T,Q,P,W,ODEfit)
graphics <- paste("%\r\r\\begin{center}\r\\includegraphics[angle=270,scale=0.45]{",full,"}
\\includegraphics[angle=270,scale=0.45]{",zoom,"}\r\\end{center}", sep="")
```

```
@
<<results=tex,echo=FALSE>>=
cat("\\newpage")
cat(" \\ ",label,sep="")
print(tab,include.rownames=FALSE)
cat(graphics)
@

\end{document}
```

Appendix 2. R code for *lac* looping experiments.

The following code can be copied into a text file and renamed “lac.Rnw”. This script requires two comma delimited input files (the examples “E-IPTG.csv” and “E+IPTG.csv” are included in Appendix 3). Execution of this script requires running the R command: `R CMD Sweave lac.Rnw`, which performs the fitting and generates the output plots of the fits as “.eps” files, followed by running the latex command: `latex lac.tex`, which generates the final output “lac.dvi” which can be saved as a “.pdf” file.

```
\documentclass[11pt]{amsart}
\usepackage[margin=1in]{geometry}
\geometry{letterpaper}
\usepackage{graphicx,amssymb,epstopdf}
\usepackage{/Library/Frameworks/R.framework/Resources/share/texmf/Sweave}
\pagestyle{empty}
\begin{document}
\thispagestyle{empty}

<<results=hide,echo=FALSE>>=

###### to run script :::  Sweave("lac.Rnw") then will need to run latex on lac.tex  #################

library("numDeriv")
library("xtable")

###### reading in data ########################################################################

identifier <- "wild-type"

### data files contain 3 columns: sp, expression, and standard deviation of expression
eu_dat <- read.table("E-IPTG.csv",header=TRUE,sep=",")
eu_dat <- as.data.frame(eu_dat)
ei_dat <- read.table("E+IPTG.csv",header=TRUE,sep=",")
ei_dat <- as.data.frame(ei_dat)

### last row is O2 alone strain
max <- dim(eu_dat)[1]
O2_U_exp <- eu_dat[max,2]
O2_U_exp_sd <- eu_dat[max,3]
O2_I_exp <- ei_dat[max,2]
O2_I_exp_sd <- ei_dat[max,3]

sp <- eu_dat[1:(max-1),1]
E_U_exp <- eu_dat[1:(max-1),2]
E_U_exp_sd <- eu_dat[1:(max-1),3]
E_I_exp <- ei_dat[1:(max-1),2]
E_I_exp_sd <- ei_dat[1:(max-1),3]

### calculating Eprime from E
Eprime_U_exp <- E_U_exp/O2_U_exp
Eprime_I_exp <- E_I_exp/O2_I_exp

### propagation of error in Eprime data
error_Eprime_U <- E_U_exp/O2_U_exp*sqrt((E_U_exp_sd/E_U_exp)^2 + (O2_U_exp_sd/O2_U_exp)^2)
```

```
error_Eprime_I <- E_I_exp/O2_I_exp*sqrt((E_I_exp_sd/E_I_exp)^2 + (O2_I_exp_sd/O2_I_exp)^2)

### experimental fbound (activity is normalized by the maximum induced activity)
### fbound_exp <- (max induced activity - observed activity)/(max induced activity)

max_U_exp <- max(E_U_exp,na.rm=TRUE)
max_I_exp <- max(E_I_exp,na.rm=TRUE)

fbound_r_exp <- (max_U_exp - E_U_exp)/(max_U_exp)
fbound_i_exp <- (max_I_exp - E_I_exp)/(max_I_exp)

### calculating repression ratio
### RR = E(+IPTG)/E(-IPTG)
RR_exp <- E_I_exp/E_U_exp
### propagation of error in RR data
error_RR <- E_I_exp/E_U_exp*sqrt((E_I_exp_sd/E_I_exp)^2 + (E_U_exp_sd/E_U_exp)^2)

### calculating KO2
K_O2_U <- (1-O2_U_exp/max_I_exp)/(1-(1-O2_U_exp/max_I_exp))
K_O2_I <- (1-O2_I_exp/max_I_exp)/(1-(1-O2_I_exp/max_I_exp))

##########################################################################################

start <- c(c(11.6,0.69,211,20.2,78.7,0),c(11,0.89,2.5,0,79.3,238))

###### global parameters ###################################################################

sp_U <- sp
sp_I <- sp
sp_mean <- mean(sp,na.rm=TRUE)
# l = separation between base pairs (units = cm)
l <- 3.4e-8
# k = Boltzmann's constant (units = erg/K)
k <- 1.3806503e-16
# T = absolute temperature (units = K)
T <- 293

###### FUNCTIONS ###########################################################################
###### adding error bars to plots ##########################################################

errAdd<-function(x,y,upper,lower,col){

        for(i in 1:length(x)){
                if(!is.na(y[i])){
                        if(lower[i]==Inf){
                                lower[i] <- 5
                                upper[i] <- 5}
                points(c(x[i],x[i]),c((y[i]-lower[i]),(y[i]+upper[i])),type="l",col=col)
                d<-0.01*(diff(range(x,na.rm=TRUE)))
                points(c((x[i]-d),(x[i]+d)),c((y[i]-lower[i]),(y[i]-lower[i])),type="l",col=col)
                points(c((x[i]-d),(x[i]+d)),c((y[i]+upper[i]),(y[i]+upper[i])),type="l",col=col)}}}

###### model function ######################################################################

modelFUN<-function(s,sp){
```

```
### following the method described in Becker et al. 2005 and 2008
# DNA length - actual spacing between operators (center of operator)
sp_U <- sp
sp_I <- sp

# DNA helical repeat
hr_U <- s[1]

# apparent torsional modulus for the DNA loop
C_app_U <- s[2]
### for comparision accepted torsional rigidity: C = 2.4e-19 (units = erg*cm)

# equilibrium constant for formation of a loop with perfect phasing
K_max_U <- s[3]

# equilibrium constant for non-specific loop formation
K_NSL_U <- s[4]

# optimal spacing between (center of) operators
sp_optimal_U <- s[5]

# P_app
P_app_U <- s[6]

# modifed K_max
K_maxp_U <- K_max_U*exp(P_app_U*(1/sp_mean - 1/sp_U))

# standard deviation of twist per base-pair
sigma_bp_U <- hr_U/(2*pi)*sqrt(l*k*T/(C_app_U*1e-19))

# standard deviation of torsion angle between sites (considering thermal fluctuations)
sigma_TW_U <- sqrt(sp_U*sigma_bp_U^2)

# phasing-dependent equilibrium constant for specific loop formation
K_SL_U <- 0
for(i in seq(-5,5,by=1)){
        K_SL_U <- K_SL_U + K_maxp_U*exp(-(sp_U-sp_optimal_U+i*hr_U)^2/(2*sigma_TW_U^2))}

### fbound <- (K_SL + K_NSL + K_O2)/(1 + K_SL + K_NSL + K_O2)
fbound_U <- (K_SL_U + K_NSL_U + K_O2_U)/(K_SL_U + K_NSL_U + K_O2_U + 1)

hr_I <- s[7]
C_app_I <- s[8]
K_max_I <- s[9]
K_NSL_I <- s[10]
sp_optimal_I <- s[11]
P_app_I <- s[12]
K_maxp_I <- K_max_I*exp(P_app_I*(1/sp_mean - 1/sp_I))
sigma_bp_I <- hr_I/(2*pi)*sqrt(l*k*T/(C_app_I*1e-19))
sigma_TW_I <- sqrt(sp_I*sigma_bp_I^2)
K_SL_I <- 0
for(i in seq(-5,5,by=1)){
        K_SL_I <- K_SL_I + K_maxp_I*exp(-(sp_I-sp_optimal_I + i*hr_I)^2/(2*sigma_TW_I^2))}
fbound_I <- (K_SL_I + K_NSL_I + K_O2_I)/(K_SL_I + K_NSL_I + K_O2_I + 1)

max_activity <- max(max_U_exp,max_I_exp)
```

```
E_U_theory <- (max_activity)*(1 - fbound_U)
E_I_theory <- (max_activity)*(1 - fbound_I)
Ep_U <- E_U_theory/O2_U_exp
Ep_I <- E_I_theory/O2_I_exp

### repression ratio
### RR = E(+IPTG)/E(-IPTG)
RR_theory <- E_I_theory/E_U_theory

out <- NULL

out[[1]] <- Ep_U
out[[2]] <- Ep_I
out[[3]] <- RR_theory

out}

###### model function 2 #################################################################

modelFUN2<-
function(hr_U,C_app_U,K_max_U,K_NSL_U,sp_optimal_U,P_app_U,hr_I,C_app_I,K_max_I,K_NSL_I,s
p_optimal_I,P_app_I,sp){

sp_U <- sp
sp_I <- sp

K_maxp_U <- K_max_U*exp(P_app_U*(1/sp_mean - 1/sp_U))
sigma_bp_U <- hr_U/(2*pi)*sqrt(l*k*T/(C_app_U*1e-19))
sigma_TW_U <- sqrt(sp_U*sigma_bp_U^2)
K_SL_U <- 0
for(i in seq(-5,5,by=1)){
        K_SL_U <- K_SL_U + K_maxp_U*exp(-(sp_U-sp_optimal_U+i*hr_U)^2/(2*sigma_TW_U^2))}
fbound_U <- (K_SL_U + K_NSL_U + K_O2_U)/(K_SL_U + K_NSL_U + K_O2_U + 1)

K_maxp_I <- K_max_I*exp(P_app_I*(1/sp_mean - 1/sp_I))
sigma_bp_I <- hr_I/(2*pi)*sqrt(l*k*T/(C_app_I*1e-19))
sigma_TW_I <- sqrt(sp_I*sigma_bp_I^2)
K_SL_I <- 0
for(i in seq(-5,5,by=1)){
        K_SL_I <- K_SL_I + K_maxp_I*exp(-(sp_I-sp_optimal_I + i*hr_I)^2/(2*sigma_TW_I^2))}
fbound_I <- (K_SL_I + K_NSL_I + K_O2_I)/(K_SL_I + K_NSL_I + K_O2_I + 1)

max_activity <- max(max_U_exp,max_I_exp)
E_U_theory <- (max_activity)*(1 - fbound_U)
E_I_theory <- (max_activity)*(1 - fbound_I)
RR_theory <- E_I_theory/E_U_theory

out <- NULL

out[[1]] <- E_U_theory
out[[2]] <- E_I_theory
out[[3]] <- RR_theory

out}

###### fit function ######################################################################
```

```
fitFUN <- function(s){

hr_U <- s[1]
C_app_U <- s[2]
K_max_U <- s[3]
K_NSL_U <- s[4]
sp_optimal_U <- s[5]
P_app_U <- s[6]

K_maxp_U <- K_max_U*exp(P_app_U*(1/sp_mean - 1/sp_U))
sigma_bp_U <- hr_U/(2*pi)*sqrt(l*k*T/(C_app_U*1e-19))
sigma_TW_U <- sqrt(sp_U*sigma_bp_U^2)
K_SL_U <- 0
for(i in seq(-5,5,by=1)){
            K_SL_U <- K_SL_U + K_maxp_U*exp(-(sp_U-sp_optimal_U+i*hr_U)^2/(2*sigma_TW_U^2))}
fbound_U <- (K_SL_U + K_NSL_U + K_O2_U)/(K_SL_U + K_NSL_U + K_O2_U + 1)

hr_I <- s[7]
C_app_I <- s[8]
K_max_I <- s[9]
K_NSL_I <- s[10]
sp_optimal_I <- s[11]
P_app_I <- s[12]

K_maxp_I <- K_max_I*exp(P_app_I*(1/sp_mean - 1/sp_I))
sigma_bp_I <- hr_I/(2*pi)*sqrt(l*k*T/(C_app_I*1e-19))
sigma_TW_I <- sqrt(sp_I*sigma_bp_I^2)
K_SL_I <- 0
for(i in seq(-5,5,by=1)){
            K_SL_I <- K_SL_I + K_maxp_I*exp(-(sp_I-sp_optimal_I + i*hr_I)^2/(2*sigma_TW_I^2))}
fbound_I <- (K_SL_I + K_NSL_I + K_O2_I)/(K_SL_I + K_NSL_I + K_O2_I + 1)

max_activity <- max(max_U_exp,max_I_exp)
E_U_theory <- (max_activity)*(1 - fbound_U)
E_I_theory <- (max_activity)*(1 - fbound_I)
RR_theory <- E_I_theory/E_U_theory

### non-linear least squares
lsse <- sum((E_U_theory - E_U_exp)^2/E_U_exp_sd^2,na.rm=TRUE) + sum((E_I_theory -
E_I_exp)^2/E_I_exp_sd^2,na.rm=TRUE)

lsse}

##########################################################################################

### initializing output
dat1 <- NULL
dat2 <- NULL
dat3 <- NULL
dat4 <- NULL
dat5 <- NULL

### first fitting
nlminbFit <- nlminb(start,fitFUN,lower=c(8,0.1,0,0,50,0),upper=c(15,3,300,100,90,500))
dat1$Parameter <- c("hr","C_app","K_max","K_NSL","sp_optimal","P_app")
```

```
fit1 <- nlminbFit$par
dat1$Estimate=nlminbFit$par
dat2$Cost.Function=nlminbFit$objective
hessian <- hessian(fitFUN,fit1)
FIM <- solve(2*hessian)
se <- sqrt(abs(diag(FIM)))
cl <- sqrt(qchisq(0.95,2*length(sp)-12))*se
cl1 <- cl
dat1$Confidence.Limits=cl

dat1 <- as.data.frame(dat1)
dat2 <- as.data.frame(dat2)
xtab1 <- xtable(dat1)
xtab2 <- xtable(dat2)
csv1 <- paste(identifier,"-parameters1.csv",sep="")
csv2 <- paste(identifier,"-costfuction1.csv",sep="")
write.table(dat1,file=csv1,sep=",",row.names=FALSE)
write.table(dat2,file=csv2,sep=",",row.names=FALSE)

### generating data points for plotting first fit
parfit <- nlminbFit$par
sp_seq <- seq(40,95,length=300)
test <- modelFUN(parfit,sp_seq)

##########################################################################################

parFUN <- function(s){

hr_U <- s[1]
C_app_U <- s[2]
K_max_U <- s[3]
K_NSL_U <- s[4]
sp_optimal_U <- s[5]
P_app_U <- s[6]
hr_I <- s[7]
C_app_I <- s[8]
K_max_I <- s[9]
K_NSL_I <- s[10]
sp_optimal_I <- s[11]
P_app_I <- s[12]

out <- NULL

out[[1]] <- c(hr_U,C_app_U,sp_optimal_U,hr_I,C_app_I,sp_optimal_I)
out[[2]] <- c(K_max_U,K_NSL_U,P_app_U,K_max_I,K_NSL_I,P_app_I)

out}

##########################################################################################

### second starting values
new_par <- parFUN(parfit)
start2 <- new_par[[1]]
fixed <- new_par[[2]]

###### fit function 2 ####################################################################
```

```
fitFUN2 <- function(s){

hr_U <- s[1]
C_app_U <- s[2]
K_max_U <- fixed[1]
K_NSL_U <- fixed[2]
sp_optimal_U <- s[3]
P_app_U <- fixed[3]

hr_I <- s[4]
C_app_I <- s[5]
K_max_I <- fixed[4]
K_NSL_I <- fixed[5]
sp_optimal_I <- s[6]
P_app_I <- fixed[6]

K_maxp_U <- K_max_U*exp(P_app_U*(1/sp_mean - 1/sp_U))
sigma_bp_U <- hr_U/(2*pi)*sqrt(l*k*T/(C_app_U*1e-19))
sigma_TW_U <- sqrt(sp_U*sigma_bp_U^2)
K_SL_U <- 0
for(i in seq(-5,5,by=1)){
            K_SL_U <- K_SL_U + K_maxp_U*exp(-(sp_U-sp_optimal_U+i*hr_U)^2/(2*sigma_TW_U^2))}
fbound_U <- (K_SL_U + K_NSL_U + K_O2_U)/(K_SL_U + K_NSL_U + K_O2_U + 1)

K_maxp_I <- K_max_I*exp(P_app_I*(1/sp_mean - 1/sp_I))
sigma_bp_I <- hr_I/(2*pi)*sqrt(l*k*T/(C_app_I*1e-19))
sigma_TW_I <- sqrt(sp_I*sigma_bp_I^2)
K_SL_I <- 0
for(i in seq(-5,5,by=1)){
            K_SL_I <- K_SL_I + K_maxp_I*exp(-(sp_I-sp_optimal_I + i*hr_I)^2/(2*sigma_TW_I^2))}
fbound_I <- (K_SL_I + K_NSL_I + K_O2_I)/(K_SL_I + K_NSL_I + K_O2_I + 1)

max_activity <- max(max_U_exp,max_I_exp)
E_U_theory <- (max_activity)*(1 - fbound_U)
E_I_theory <- (max_activity)*(1 - fbound_I)
RR_theory <- E_I_theory/E_U_theory

### non-linear least squares
lsse <- sum((E_U_theory - E_U_exp)^2/E_U_exp_sd^2,na.rm=TRUE) + sum((E_I_theory -
E_I_exp)^2/E_I_exp_sd^2,na.rm=TRUE) + sum((RR_theory - RR_exp)^2/error_RR^2,na.rm=TRUE)

lsse}

#########################################################################################

### second fitting

nlminbFit <- nlminb(start2,fitFUN2,lower=0,upper=1000)
fit2 <- nlminbFit$par
dat3$Parameter <- c("hr","C_app","sp_optimal")
dat5$Parameter <- c("K_max","K_NSL","P_app")#,"K_max","K_NSL","P_app")
dat5$Estimate <- fixed
dat5$Confidence.Limits <- parFUN(cl)[[2]]
dat3$Estimate=nlminbFit$par
dat4$Cost.Function=nlminbFit$objective
```

```
hessian <- hessian(fitFUN2,fit2)
FIM <- solve(2*hessian)
se <- sqrt(abs(diag(FIM)))
cl <- sqrt(qchisq(0.95,3*length(sp)-6))*se
cl2 <- cl
dat3$Confidence.Limits=cl
dat3 <- as.data.frame(dat3)
dat4 <- as.data.frame(dat4)
dat5 <- as.data.frame(dat5)
dat5 <- cbind(dat3,dat5)
xtab3 <- xtable(dat5)
xtab4 <- xtable(dat4)
csv3 <- paste(identifier,"-parameters2.csv",sep="")
csv4 <- paste(identifier,"-costfunction2.csv",sep="")
write.table(dat3,file=csv3,sep=",",row.names=FALSE)
write.table(dat4,file=csv4,sep=",",row.names=FALSE)

########################################################################################

parFUN2 <- function(s,s2){

out <- c(s[1],s[2],s2[1],s2[2],s[3],s2[3],s[4],s[5],s2[4],s2[5],s[6],s2[6])

out}

########################################################################################

### generating data points for plotting second fit
parfit2 <- nlminbFit$par
parfit2full <- parFUN2(parfit2,fixed)
test2 <- modelFUN(parfit2full,sp_seq)
datfit <- parFUN2(dat5[[2]],dat5[[5]])
clfit <- parFUN2(dat5[[3]],dat5[[6]])
rindex <- rep(c(1,2,0,0,1,0),2)
datfitr <- NULL
clfitr <- NULL
comr <- NULL
for(k in 1:12){
        if(k<7){
                if(k==3|k==4){
                        datfit[k] <- datfit[k]/K_O2_U
                        clfit[k] <- clfit[k]/K_O2_U}
                datfitr[2*k-1] <- round(datfit[k],digits=rindex[k])
                clfitr[2*k-1] <- round(clfit[k],digits=rindex[k])
                if(clfit[k]/datfit[k] > 1){
                        datfitr[2*k-1] <- paste("(",datfitr[2*k-1],")",sep="")}}
        if(k>6){
                if(k==9|k==10){
                        datfit[k] <- datfit[k]/K_O2_I
                        clfit[k] <- clfit[k]/K_O2_I}
                datfitr[2*(k-6)] <- round(datfit[k],digits=rindex[k])
                clfitr[2*(k-6)] <- round(clfit[k],digits=rindex[k])
                if(clfit[k]/datfit[k] > 1){
                        datfitr[2*(k-6)] <- paste("(",datfitr[2*(k-6)],")",sep="")}}}

comr <- paste(datfitr," $\\pm$ ",clfitr,sep="")
comr <- rbind(cbind(comr[1:4]),round(K_O2_U,1),round(K_O2_I,1),cbind(comr[5:12]))
mat <- matrix(comr,nrow=2)
mat <- as.data.frame(mat)
row.names(mat) <- c("$-$IPTG","$+$IPTG")
names(mat) <- c("hr (bp/turn)","C$_{app}$ ($\\times 10^{-19}$ erg-
cm)","K$_{O2}$","K$^o_{max}$","K$^o_{NSL}$","sp$_{optimal}$ (bp)","P$_{app}$ (bp)")
xtab5 <- xtable(mat)
names(mat) <- c("hr (bp/turn)","Capp (x10-19 erg-cm)","KO2","Kmax","KNSL","spoptimal (bp)","Papp
(bp)")
csv5 <- paste(identifier,"-summary.csv",sep="")
write.table(mat,file=csv5,sep=",",append=FALSE)

##########################################################################################

plot1 <- paste(identifier,"-Eprime.eps",sep="")
plot2 <- paste(identifier,"-RR.eps",sep="")

lwdv <- 3
cexvar <- 2.25
mexvar <- 1.1
cexv <- 1.5

postscript(file=plot1, paper="letter", horizontal=TRUE, family="sans")

par(cex.axis=cexvar, cex.lab=cexvar,mex=mexvar,mar=c(5, 7, 4, 2) + 0.7,font.axis=2, font.lab=2,tck=0.02)

plot(sp_U,Eprime_U_exp,pch=16,log="y",xlim=c(45,90),ylim=c(0.001,10),xlab="",ylab="",type="n",axes
=FALSE)

mtext(expression(paste(bold(Operator)," ",bold(Spacing)," ",bold((bp)),sep="")),side=1,line=4,cex=cexvar)
mtext(expression(paste(bold(Reporter)," ",bold(Activity),sep="")),side=2,line=6,cex=cexvar)
mtext(expression(paste(bold((E*minute)),sep="")),side=2,line=4,cex=cexvar)

polygon(c(sp_seq,sp_seq[length(sp_seq):1]),c(test2[[1]],test2[[2]][length(sp_seq):1]),col="gray",border=N
A)
points(sp_seq,test2[[1]],type="l",lty=1,col=2,lwd=lwdv,cex=cexv)
points(sp_seq,test2[[2]],type="l",lty=2,col=2,lwd=lwdv,cex=cexv)
points(sp_U,Eprime_U_exp,pch=16,cex=cexv)
errAdd(sp_U,Eprime_U_exp,error_Eprime_U,error_Eprime_U,col=1)
points(sp_I,Eprime_I_exp,cex=cexv)
errAdd(sp_I,Eprime_I_exp,error_Eprime_I,error_Eprime_I,col=1)
points(sp_I,Eprime_I_exp,pch=16,col="white",cex=cexv)
points(sp_I,Eprime_I_exp,cex=cexv)

box(lwd=lwdv)
axis(1,at=seq(45,90,by=5),labels=seq(45,90,by=5),las=1,lwd=lwdv)
axis(2,at=c(0.001,0.01,0.1,1,10),labels=c(0.001,0.01,0.1,1,10),las=1,lwd=lwdv)
axis(3,at=seq(45,90,by=5),labels=FALSE,lwd=lwdv)
axis(4,at=c(0.001,0.01,0.1,1,10),labels=FALSE,lwd=lwdv)

legend("topright",c(expression(bold(+IPTG)),expression(bold(-IPTG))), col=c("red","red"), lty=c(2,1),
bty="n", inset=0.05,cex=cexv,lwd=lwdv)

dev.off()

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

```
postscript(file=plot2, paper="letter", horizontal=TRUE, family="sans")

par(cex.axis=cexvar, cex.lab=cexvar,mex=mexvar,mar=c(5, 7, 4, 2) + 0.7,font.axis=2, font.lab=2,tck=0.02)

plot(sp_U,RR_exp,pch=16,xlim=c(45,90),ylim=c(0,180),xlab="",ylab="",type="n",axes=FALSE)

mtext(expression(paste(bold(Operator)," ",bold(Spacing)," ",bold((bp)),sep="")),side=1,line=4,cex=cexvar)
mtext(expression(paste(bold(Repression)," ",bold(Ratio),sep="")),side=2,line=6,cex=cexvar)
mtext(expression(paste(bold((+IPTG/-IPTG)),sep="")),side=2,line=4,cex=cexvar)

points(sp_seq,test2[[3]],type="l",lwd=lwdv,col="gray",cex=cexv)
points(sp_seq,test2[[3]],type="l",lwd=lwdv,col=2,cex=cexv)
points(sp_U,RR_exp,pch=16,cex=cexv)
errAdd(sp_U,RR_exp,error_RR,error_RR,col=1)

box(lwd=lwdv)
axis(1,at=seq(45,90,by=5),las=1,lwd=lwdv)
axis(2,at=seq(0,180,30),las=1,lwd=lwdv)
axis(3,at=seq(45,90,by=5),labels=FALSE,lwd=lwdv)
axis(4,at=seq(0,180,20),labels=FALSE,lwd=lwdv)

dev.off()
@
<<results=tex,echo=FALSE>>=

        cat("%\n\n\\newpage")
        cat("%\n\n",identifier,"\n\n%",sep="")
        cat("%\n\n%")
        cat("%\n\n","*** uninduced (-IPTG) parameters are given first","\n\n%",sep="")
        cat("%\n\n%")
        cat("%\n\n","fit to E data (+/- IPTG)","\n\n%",sep="")
        print(xtab2,include.rownames=FALSE,floating=TRUE)
        print(xtab1,include.rownames=FALSE,floating=TRUE)
        cat("%\n\n%")
        cat("%\n\n","simultaneous fit to E data (+/- IPTG) and RR","\n\n%",sep="")
        print(xtab4,include.rownames=FALSE,floating=TRUE)
        print(xtab3,include.rownames=FALSE,floating=TRUE)
        cat("%\n\n%")
        cat("%\n\n","summary","\n\n%",sep="")
        cat("%\n\n","*** parentheses indicate parameters not well determined","\n\n%",sep="")
        print(xtab5,floating=FALSE,sanitize.text.function=function(x){x})
        cat("%\n\n\\newpage")
        cat("%\n\n",identifier,"\n\n%",sep="")
        graphics <- paste("%\n\n\\begin{center}\n\\includegraphics[angle=270,scale=0.75]{",plot1,"}
\\includegraphics[angle=270,scale=0.75]{",plot2,"}\n\\end{center}", sep="")
        cat(graphics)

@
\end{document}
```

Appendix 3. Required files for R scripts

jf-211mer-75nM-ligase100UmL.txt
```
0 75 0 0 0 0 0 0
2 45.080055326137 3.13754834457182 4.52876619762908 12.4953524807161 5.85426352044715
2.57495147095558 1.32906265954329
4 33.2914310979581 4.72815611179549 6.90455253480098 16.7073389035882 7.94323178631252
3.5307552097175 1.89453435582711
6 27.8558990883505 5.51944962022432 8.23865362979177 17.0354732696023 9.44777091565846
4.59285289006137 2.30990058631128
8 25.0083855626908 6.14174562868144 9.30618026546597 16.3757667623754 10.1833197734334
5.28744752830441 2.69715447904854
```

E-IPTG.csv
```
E-IPTG,WT,WT-sd
63.5,16.113,6.525
64.5,6.118,1.026
65.5,6.776,0.81
66.5,5.911,0.577
67.5,4.344,0.641
70.5,16.895,2.489
71.5,17.092,3.022
72.5,19.596,4.24
73.5,32.972,8.475
74.5,24.414,7.842
75.5,12.227,1.621
76.5,6.179,1.054
77.5,4.586,0.579
78.5,4.385,0.452
79.5,5.395,0.574
80.5,8.299,1.528
81.5,10.145,1.603
82.5,12.257,2.145
83.5,21.979,3.04
84.5,31.112,7.855
85.5,16.982,3.687
86.5,8.355,1.82
87.5,5.304,0.476
88.5,7.154,1.058
89.5,4.62,0.776
90.5,5.787,1.207
,262.716,40.24
```

E+IPTG.csv
```
E+IPTGc,WT,WT-sd
63.5,905.081,130.881
64.5,593.962,85.541
65.5,671.263,144.889
66.5,367.261,84.026
67.5,155.75,37.047
70.5,522.824,74.554
71.5,553.266,72.633
72.5,568.988,124.858
73.5,734.275,86.648
74.5,635.396,97.21
```

```
75.5,660.628,67.778
76.5,414.782,65.594
77.5,326.426,62.478
78.5,222.558,46.877
79.5,236.466,43.836
80.5,345.038,53.205
81.5,396.226,72.239
82.5,441.869,65.91
83.5,476.839,62.281
84.5,464.427,55.665
85.5,456.611,29.975
86.5,421.753,73.024
87.5,286.891,51.174
88.5,168.242,24.018
89.5,157.051,25.593
90.5,180.356,29.883
,798.207,85.917
```