

Supplementary text

Predicting mutational routes to new adaptive phenotypes

Peter A. Lind, Eric Libby, Jenny Herzog and Paul B. Rainey

Analysis of mutations and molecular effects

Wsp pathway: Mutations were identified in five genes of the seven-gene pathway all of which were predicted by null model IV (Figure 5 – figure supplement 1). The most commonly mutated gene was *wspA* (PFLU1219), with ten of 15 mutations (Figure 6) being amino acid substitutions (six unique) clustered in the region 352-420 at the stalk of the signalling domain. This region has been implicated in trimer-of-dimer formation for the WspA homologue in *Pseudomonas aeruginosa* (O'Connor, et al. 2012) which is critical for self-assembly and localization of Wsp clusters in the membrane. It is possible that these mutations stabilize trimer of dimer formation, change the subcellular location of the Wsp complex, or affect interaction with WspD (putative interface 383-420 in WspA) (Griswold, et al. 2002) and thus affecting relay of signal to WspE. These effects we interpreted as enabling mutations increasing r_3 in Figure 4A. The four additional mutations were in frame deletions in a separate region of the transducer domain ($\Delta T293 - E299$, $\Delta A281-A308$). Although it is possible that these mutations could also affect trimer-of-dimer formation, there are predicted methylation sites in the region (Rice and Dahlquist 1991) that regulate the activity of the protein via methyltransferase WspC and methylesterase WspF. Given that disabling mutations are more common than enabling mutations it is likely that these mutations decrease r_2 in Figure 4A by disrupting the interaction with WspF. We also identified a single mutation that fused the open reading frame of WspC, the

methyltransferase that positively regulates WspA activity, to WspD, resulting in a chimeric protein (Figure 6, Figure 6 – source data). This mutation is likely to be a rare enabling mutation that increases the activity of WspC (increasing r_1 in Figure 4A) by physically tethering it to the WspABD complex thus allowing it to more effectively counteract the negative regulator WspF. Alternatively, the tethering may physically block the interaction with WspF (decrease of r_2 in Figure 4A).

The second most commonly mutated gene in the *wsp* operon was *wspE* (PFLU1223) (Figure 6). Four amino acids were repeatedly mutated in the response regulatory domain of WspE and all cluster closely in a structural homology model made with Phyre2 (Kelley, et al. 2015). All mutated residues surround the active site of the phosphorylated D682 and it is likely that they disrupt feedback regulation by decreasing phosphorylation of the negative regulator WspF (decreasing r_6) rather than increasing activation of WspR (r_5 in Figure 4A).

Twelve mutations were detected in *wspF* (PFLU1224). These are distributed throughout the gene and include amino acid substitutions, in-frame deletions as well as a frame-shift and a stop codon (Figure 6). The pattern of mutations is consistent with both the role of WspF as a negative regulator of WspA activity and the well-characterised effect of loss-of-function mutations in this gene (Bantinaki, et al. 2007; McDonald, et al. 2009). The mutations are interpreted as decreasing r_2 in Figure 4A. Five mutations were found in WspR (PFLU1225), the DGC output response regulator that produces c-di-GMP and activates expression of cellulose (Figure 6). All mutations were located in the linker region between the response regulator and DGC domains. Mutations in this region are known to generate constitutively active *wspR*

alleles by relieving the requirement for phosphorylation (Goymer, et al. 2006). They may additionally affect subcellular clustering of WspR (Huangyutitham, et al. 2013) or shift the equilibrium between the dimeric form of WspR, with low basal activity, towards a tetrameric activated form (De, et al. 2009). In our model these increase reaction r_5 .

Aws pathway: Mutations were identified in all three genes of the Aws pathway – all of which were predicted by the null model. In the Aws pathway, mutations were most commonly found in *awsX* (25 out of 41 mutations (Figure 6). The above-mentioned mutational hotspot produced in-frame deletions likely mediated by 6 bp direct repeats (Figure 6 – source data 1). The deletions are consistent with a loss of function and a decrease in r_3 (Figure 4B) that would leave the partially overlapping open reading frame of the downstream gene (*awsR*) unaffected.

The DGC AwsR, was mutated in 14 cases with an apparent mutational hot spot at T27P (9 mutants) in a predicted transmembrane helix (amino acids 19-41). The remaining mutations were amino acid substitutions in the HAMP linker and in the PAS-like periplasmic domain between the two transmembrane helices. These amino acid substitutions are distant to the output DGC domain (Figure 6) and their effects are difficult to interpret, but they could cause changes in dimerization (Malone, et al. 2012) or the packing of HAMP domains, which could, in turn, alter transmission of conformational changes in the periplasmic PAS-like domain to the DGC domain causing constitutive activation (Parkinson 2010). Such effects would increase r_4 in Figure 4B. Mutations in the N-terminal part of the protein are easier to interpret based on the existing functional model (Malone, et al. 2012) and most likely disrupt

interactions with the periplasmic negative regulator AwsX resulting in a decrease in r_3 in Figure 4B.

Two mutations were found in the outer membrane lipoprotein protein AwsO between the signal peptide and the OmpA domain (Figure 6). Both mutations were glutamine to proline substitutions (Q34P, Q40P), which together with a previously reported G42V mutation (McDonald, et al. 2009) suggest that multiple changes in this small region can cause a WS phenotype. This is also supported by data from *Pseudomonas aeruginosa* in which mutations in nine different positions in this region lead to a small colony variant phenotype similar to WS (Malone, et al. 2012). A functional model based on the YfiBNR in *P. aeruginosa* (Malone, et al. 2012; Xu, et al. 2016), suggest that AwsO sequesters AwsX at the outer membrane and that mutations in the N-terminal part of the protein lead to constitutive activation and increased binding of AwsX. This would correspond to an increase in r_2 in Figure 4B, which would relieve negative regulation of AwsR.

Mws pathway: The MwsR pathway (comprising just a single gene) harbours mutations in both DGC and phosphodiesterase (PDE) domains. Only mutations in the C-terminal phosphodiesterase (PDE) domain were predicted (Figure 4C). Eleven of 18 mutations were identical in-frame deletions (Δ R1024-E1026) in the PDE domain, mediated by 8 bp direct repeats (Figure 6, Figure 6 – source data 1). It has been shown previously that deletion of the entire PDE domain generates the WS phenotype (McDonald, et al. 2009), suggesting a negative regulatory role that causes a decrease of r_2 in the model in Figure 4C. One additional mutation was found in the PDE domain (E1083K) located close to R1024 in a structural homology model made

with Phyre2 (Kelley, et al. 2015), but distant to the active site residues (E1059-L1061). Previously reported mutations (A1018T, ins1089DV) (McDonald, et al. 2009) are also distant from the active site and cluster in the same region in a structural homology model. This suggests that loss of phosphodiesterase activity may not be the mechanism leading to the WS phenotype. This is also supported by the high solvent accessibility of the mutated residues, which indicates that major stability-disrupting mutations are unlikely and changes in interactions between domains or dimerization are more probable. Thus, it is likely that the WS phenotype resulting from a deletion in the PDE domain is caused by disruption of domain interactions or dimerization rather than loss of phosphodiesterase activity.

The remaining mutations within *mwsR* are amino acid substitutions in the GGDEF domain, close to the DGC active site (927-931) with the exception of a duplication of I978-G985. While it is possible that these mutations directly increase the catalytic activity of the DGC, increasing r_1 in Figure 4C, such enabling mutations are considered to be rare. An alternative hypothesis is that these mutations either interfere with c-di-GMP feedback regulation or produce larger conformational changes that change inter-domain or inter-dimer interactions, similar to the mutations in the PDE domain. Based on these data we reject the current model of Mws function, which predicted mutations decreasing r_2 (Figure 4C) through mutations inactivating the PDE domain. We instead suggest that the mutations are likely to disrupt the conformational dynamics between the domains and could be seen either as activating mutations causing constitutive activation or disabling mutations with much reduced mutational target size that must specifically disrupt the interaction surface between the domains.

In both cases the previous model leads to an overestimation of the rate to WS for the Mws pathway.

References

- Bantinaki E, Kassen R, Knight CG, Robinson Z, Spiers AJ, Rainey PB. 2007. Adaptive divergence in experimental populations of *Pseudomonas fluorescens*. III. Mutational origins of wrinkly spreader diversity. *Genetics* 176:441-453. <http://dx.doi.org/10.1534/genetics.106.069906>
- De N, Navarro MV, Raghavan RV, Sondermann H. 2009. Determinants for the activation and autoinhibition of the diguanylate cyclase response regulator WspR. *J Mol Biol* 393:619-633. <http://dx.doi.org/10.1016/j.jmb.2009.08.030>
- Goymer P, Kahn SG, Malone JG, Gehrig SM, Spiers AJ, Rainey PB. 2006. Adaptive divergence in experimental populations of *Pseudomonas fluorescens*. II. Role of the GGDEF regulator WspR in evolution and development of the wrinkly spreader phenotype. *Genetics* 173:515-526. <http://dx.doi.org/10.1534/genetics.106.055863>
- Griswold IJ, Zhou HJ, Matison M, Swanson RV, McIntosh LP, Simon MI, Dahlquist FW. 2002. The solution structure and interactions of CheW from *Thermotoga maritima*. *Nature Structural Biology* 9:121-125. <http://dx.doi.org/DOI 10.1038/nsb753>
- Huangyutitham V, Guvener ZT, Harwood CS. 2013. Subcellular clustering of the phosphorylated WspR response regulator protein stimulates its diguanylate cyclase activity. *MBio* 4:e00242-00213. <http://dx.doi.org/10.1128/mBio.00242-13>
- Kelley LA, Mezulis S, Yates CM, Wass MN, Sternberg MJ. 2015. The Phyre2 web portal for protein modeling, prediction and analysis. *Nat Protoc* 10:845-858. <http://dx.doi.org/10.1038/nprot.2015.053>
- Malone JG, Jaeger T, Manfredi P, Dotsch A, Blanka A, Bos R, Cornelis GR, Haussler S, Jenal U. 2012. The YfiBNR signal transduction mechanism reveals novel targets

for the evolution of persistent *Pseudomonas aeruginosa* in cystic fibrosis airways. *PLoS Pathog* 8:e1002760. <http://dx.doi.org/10.1371/journal.ppat.1002760>

McDonald MJ, Gehrig SM, Meintjes PL, Zhang XX, Rainey PB. 2009. Adaptive divergence in experimental populations of *Pseudomonas fluorescens*. IV. Genetic constraints guide evolutionary trajectories in a parallel adaptive radiation. *Genetics* 183:1041-1053. <http://dx.doi.org/10.1534/genetics.109.107110>

Parkinson JS. 2010. Signaling mechanisms of HAMP domains in chemoreceptors and sensor kinases. *Annu Rev Microbiol* 64:101-122. <http://dx.doi.org/10.1146/annurev.micro.112408.134215>

Rice MS, Dahlquist FW. 1991. Sites of deamidation and methylation in Tsr, a bacterial chemotaxis sensory transducer. *J Biol Chem* 266:9746-9753.

Xu M, Yang X, Yang XA, Zhou L, Liu TZ, Fan Z, Jiang T. 2016. Structural insights into the regulatory mechanism of the *Pseudomonas aeruginosa* YfiBNR system. *Protein Cell* 7:403-416. <http://dx.doi.org/10.1007/s13238-016-0264-7>

Differential equations for WSP, AWS, & MWS pathways

WSP

$$\begin{aligned}
 \frac{d[ABD]}{dt} &= r_2 F^*[ABDm] - r_1 [ABD][Cm] \\
 \frac{d[ABDm]}{dt} &= r_1 [ABD][Cm] - r_2 F^*[ABDm] - r_3 S[ABDm] + r_4 E[ABDm^*] \\
 \frac{d[ABDm^*]}{dt} &= -r_4 E[ABDm^*] + r_3 S[ABDm] \\
 \frac{dE^*}{dt} &= r_4 E[ABDm^*] - r_5 E^* R - r_6 E^* F \\
 \frac{dR^*}{dt} &= r_5 R E^* \\
 \frac{dE}{dt} &= r_6 E^* F - r_4 E[ABDm^*] + r_5 R E^* \\
 \frac{dF}{dt} &= -r_6 E^* F + r_2 F^*[ABDm] \\
 \frac{dF^*}{dt} &= r_6 E^* F - r_2 F^*[ABDm] \\
 \frac{dR}{dt} &= -r_5 R E^* - .01 R
 \end{aligned}$$

AWS

$$\begin{aligned}
 \frac{dX}{dt} &= -r_2 X O^* - r_3 X R \\
 \frac{d[XR]}{dt} &= r_3 X R \\
 \frac{dO}{dt} &= -r_1 S O \\
 \frac{dO^*}{dt} &= r_1 S O - r_2 X O^* \\
 \frac{d[OX]}{dt} &= r_2 O^* X \\
 \frac{dR}{dt} &= -r_3 X R - r_4 R R \\
 \frac{d[RR]}{dt} &= r_4 R R
 \end{aligned}$$

MWS

$$\begin{aligned}\frac{dG}{dt} &= -r_1GS \\ \frac{dG^*}{dt} &= r_1GS - r_2G^*E - r_3G^*C \\ \frac{dE}{dt} &= -r_2G^*E \\ \frac{d[GE]}{dt} &= r_2G^*E \\ \frac{dC}{dt} &= -r_3G^*C \\ \frac{d[GC]}{dt} &= r_3G^*C\end{aligned}$$

Julia code for WSP, AWS, and MWS differential equations

The following three functions implement the differential equation model (ODE model) for the WSP, AWS, and MWS pathways.

```
using ODE,StatsBase,MAT,HDF5,JLD

# code for WSP differential equations
function lindodeWSP(t,y)
    # convert y to reactants for ease of reading
    ABD=y[1];
    ABDm=y[2];
    ABDmp=y[3];
    Ep=y[4];
    Rp=y[5];
    E=y[6];
    F=y[7];
    Fp=y[8];
    R=y[9];
    # pull reaction rates from rs variable
    r1=rs[1];r2=rs[2];r3=rs[3];r4=rs[4];r5=rs[5];r6=rs[6];
    # compute derivatives, i.e. y'
    yp=zeros(size(y));
    yp[1]=r2*Fp*ABDm-r1*ABD*Cm; # dABD/dt
    yp[2]=r1*ABD*Cm-r2*Fp*ABDm-r3*S*ABDm+r4*E*ABDmp; # dABDm/dt
    yp[3]=-r4*E*ABDmp + r3*S*ABDm ; # dABDmp/dt
    yp[4]=r4*E*ABDmp - r5*Ep*R -r6*Ep*F ; # dEp/dt
    yp[5]=r5*R*Ep ; # dRp/dt
    yp[6]=r6*Ep*F-r4*E*ABDmp+r5*R*Ep ; # dE/dt
    yp[7]=-r6*Ep*F +r2*Fp*ABDm ; # dF/dt
    yp[8]=r6*Ep*F -r2*Fp*ABDm ; # dFp/dt
    yp[9]=-r5*R*Ep-.01*R; # dR/dt
    return yp
end

# code for AWS differential equations
```

```

function lindodeAWS(t,y)
    # convert y to reactants for ease of reading
    X=y[1];
    XR=y[2];
    O=y[3];
    Op=y[4];
    OX=y[5];
    R=y[6];
    RR=y[7];
    # pull reaction rates from rs variable
    r1=rs[1];r2=rs[2];r3=rs[3];r4=rs[4];
    # compute derivatives, i.e. y'
    yp=zeros(size(y));
    yp[1]=-r2*X*Op-r3*X*R; # dX/dt
    yp[2]=r3*X*R; # dXR/dt
    yp[3]=-r1*S*O; # dO/dt
    yp[4]=r1*S*O-r2*X*Op; # dOp/dt
    yp[5]=r2*Op*X; # dOX/dt
    yp[6]=-r3*X*R-r4*R*R; # dR/dt
    yp[7]=r4*R*R; # dRR/dt
    return yp;
end

# code for MWS differential equations
function lindodeMWS(t,y)
    # convert y to reactants for ease of reading
    G=y[1];
    Gp=y[2];
    E=y[3];
    GE=y[4];
    C=y[5];
    GC=y[6];
    # pull reaction rates from rs variable
    r1=rs[1];r2=rs[2];r3=rs[3];
    # compute derivatives, i.e. y'
    yp=zeros(size(y));
    yp[1]=-r1*G*S; # dG/dt
    yp[2]=r1*G*S-r2*Gp*E-r3*Gp*C; # dGp/dt
    yp[3]= -r2*Gp*E; # dE/dt
    yp[4]= r2*Gp*E; # dGE/dt
    yp[5]=-r3*Gp*C; # dC/dt
    yp[6]=r3*Gp*C; # dGC/dt
    return yp;
end

```

Julia code for running differential equation solvers

We include the code for implementing our Bayesian sampling method. The colored sections correspond to statements that make it specific to WSP (blue), AWS (red), or MWS (green). It was run in julia version 0.4.3.

```

# Variables for reactions
numrxns=6; # number of reaction rates for WSP

```

```

numrxns=4; # number of reaction rates for AWS
numrxns=3; # number of reaction rates for MWS
rs=rand(numrxns); # establish variable scope, will be reaction rates later
rs_save=copy(rs); # establish variable scope, will be a saved version of reaction rates later
totnumruns=3.*length(rs); # all possible combinations for reaction rates (down,nothing, up)
v=zeros(length(rs)); # establish variable scope (used to alter reaction rates)
S=0;Cm=0; # initialize constants used in differential equations
indexWS=5; # reactant corresponding to WS in WSP
indexWS=7; # reactant corresponding to WS in AWS
indexWS=6; # reactant corresponding to WS in MWS

# Variables for running ODE solver
testnums=1000; # number of runs
yorig=zeros(testnums); # storage for baseline WS production
yout=0; # establish variable scope
tf=1.0; # establish variable scope
tftimes=1.0; # establish variable scope

numreactants=9; # number of reactants
numreactants=7; # number of reactants
numreactants=6; # number of reactants
init=10*rand(numreactants); # establish variable scope

# Variables for storing data
res=-1*ones(totnumruns,testnums); # storage for altered WS production
numfinished=1; # counter for runs completed

# Code for Bayesian sampling method
while numfinished<=testnums
    done=0;
    try
        println(numfinished) # keeps track of how many sims have been done
        # Sample concentrations and rates to establish a baseline amount of WS production
        rs=10.^(4*rand(numrxns)-2); # sample reaction rates from [.01,100]
        rs_save=copy(rs); # saved copy as a reference when altering later
        init=10*rand(numreactants); # sample initial concentrations for reactants from [0,10]
        S=10*rand(); # sample initial concentration for constant reactant of signal (S) from [0,10]
        Cm=10*rand(); # sample initial concentration for constant reactant Cm from [0,10]

        # ODE solver for baseline
        tf=1.0; # initial time for ode solver
        dst=100; # initial distance, used to determine solution converged
        tol=10^(-8.0); # tolerance for ODE solver
        while dst>tol
            tout, yout = ode45(lindodeWSP, init, [0.0 ,tf]);
            tout, yout = ode45(lindodeAWS, init, [0.0 ,tf]);
            tout, yout = ode45(lindodeMWS, init, [0.0 ,tf]);
            dst=sum((yout[end-1]-yout[end]).^2); # Euclidean distance in final step of solution
            yorig[numfinished]=yout[end][indexWS]; # reactant corresponding to WS
            tftimes=tout[end]; # final time
            tf*=2;
        end
        done=1; # successful completion of try loop
    end
    if done==1 # baseline WS production is established, now sample changes/mutations
        i0=1; # counter for completed changes
        cct=0; # counter for total number of attempts
    end
end

```

```

while i0<=totnumruns;
    cct+=1;
    println([numfinished cct]) # report status for tracking progress
    # Alter reaction rates
    num=i0-1; # used for determining which rates change down/none/up
    for i1=length(rs)-1:-1:0;
        v[i1+1]=floor(num/(3^i1)); # v is num into base 3 number
        num=num-v[i1+1]*3^i1;
    end
    v+=1;
    facs=[10.^(-2*rand()), 1, 10.^(2*rand())]; # factors to alter rxn rates [.01,1] down, 1 none, [1,100] up
    for i1=1:length(rs)
        rs[i1]=facs[v[i1]]*rs_save[i1]; # alter reaction rates
    end
    try
        tout, yout = ode45(lindodeWSP, init, [0.0,tftimes]);
        tout, yout = ode45(lindodeAWS, init, [0.0,tftimes]);
        tout, yout = ode45(lindodeMWS, init, [0.0,tftimes]);
        if abs(tout[end]-tftimes)<.01 # ODE solver finished
            res[i0,numfinished]=yout[end][indexWS]; # store amount of WS produced
            i0+=1
        end
    end
    if cct>10000
        i0=2*totnumruns; # baseline and sampling occurred in space with poorly conditioned ODEs, try again
    end
end
if i0<2*totnumruns # successful
    numfinished+=1;
    # Save data to a file for checking in MATLAB
    file=matopen("pathway_results_temp.mat","w")
    write(file,"res",res); altered WS production
    write(file,"yorig",yorig); baseline WS production
    close(file);
end
end

# Save data to a file for processing in MATLAB
file=matopen("pathway_results_complete_WSP.mat","w")
file=matopen("pathway_results_complete_AWS.mat","w")
file=matopen("pathway_results_complete_MWS.mat","w")
write(file,"res",res); altered WS production
write(file,"yorig",yorig); baseline WS production
close(file);

```

MATLAB code for interpreting saved data WSP vs MWS

This code shows how the data from the julia code is analyzed and transformed into the contour plots shown in the paper.

```

% create record of how parameters change down, none, up for WSP
rs1=rand(1,6);

```

```

totnumruns=3.^length(rs1);
paramsWSP=zeros(totnumruns,length(rs1));
v=zeros(size(rs1));
i0=1;
while i0<=totnumruns;
    num=i0-1;
    for il=length(rs1)-1:-1:0;
        v(il+1)=floor(num/(3^il));
        num=num-v(il+1)*3^il;
    end
    v=v+1;
    paramsWSP(i0,:)=v;
    i0=i0+1;
end
% create record of how parameters change down, none, up for MWS
rs1=rand(1,3);
totnumruns=3.^length(rs1);
paramsMWS=zeros(totnumruns,length(rs1));
v=zeros(size(rs1));
i0=1;
while i0<=totnumruns;
    num=i0-1;
    for il=length(rs1)-1:-1:0;
        v(il+1)=floor(num/(3^il));
        num=num-v(il+1)*3^il;
    end
    v=v+1;
    paramsMWS(i0,:)=v;
    i0=i0+1;
end

% Load data
load pathway_results_complete_MWS.mat
resMWS=res;
yorigMWS=yorig';
clear res yorig
load pathway_results_complete_WSP.mat
resWSP=res;
yorigWSP=yorig';

% Variables and data storage to compare likelihood of pathways
numsampWSP=size(resWSP,2); % in case want to use fewer samples
numsampMWS=size(resMWS,2); % in case want to use fewer samples

perange=10.^[-7:.5:-1]; % range for probability enabling mutations
pdrange=10.^[-7:.5:-1]; % range for probability disabling mutations
pemat=zeros(length(perange),length(pdrange)); % matrix for plotting data and reference
pdmat=zeros(size(pemat)); % matrix for plotting data and reference
psummatWSP=zeros(size(pemat)); % matrix for probability WSP used
psummatMWS=zeros(size(pemat)); % matrix for probability MWS used
tol=0;
% Code to compare likelihood of pathways
for i0=1:length(perange)
    for j0=1:length(pdrange)
        % Retrieve probabilities
        pe=perange(i0);
        pd=pdrange(j0);
        pemat(i0,j0)=pe;
    end
end

```

```

    pdmat(i0,j0)=pd;
    % WSP computation
    pmatforr=ones(6,1)*[pd 1-pe-pd pe];
    psum=0; % total sum of (prob of rxn changes) X (number of times WS produced)
    for il=1:size(resWSP,1)
        probevent=1; % initialize, probability to get combination of down, none, up for rxns
        for jl=1:6;
            probevent=probevent*pmatforr(jl,paramsWSP(il,jl)); % multiply by prob of each change
        end
        psum=psum+probevent*sum(resWSP(il,1:numsampWSP)>yorigWSP(1:numsampWSP)+tol)/numsampWSP;
    end
    psummatWSP(i0,j0)=psum;

    % MWS computation
    pmatforr=ones(3,1)*[pd 1-pe-pd pe];
    psum=0; % total sum of (prob of rxn changes) X (number of times WS produced)
    for il=1:size(resMWS,1)
        probevent=1; % initialize, probability to get combination of down, none, up for rxns
        for jl=1:3;
            probevent=probevent*pmatforr(jl,paramsMWS(il,jl)); % multiply by prob of each change
        end
        psum=psum+probevent*sum(resMWS(il,1:numsampMWS)>yorigMWS(1:numsampMWS)+tol)/numsampMWS;
    end
    psummatMWS(i0,j0)=psum;
end
end

% Plot data
close all
figure
contourf(pemat,pdmat,log2(psummatWSP./psummatMWS),400,'LineStyle','None')
set(gca,'xScale','log','yScale','log','TickLength',[.025 .025],'LineWidth',3);
set(gca,'FontSize',18,'xTick',10.^[-7:1:-1],'yTick',10.^[-7:1:-1]);
xlabel('Probability of enabling change','FontSize',24);
ylabel('Probability of disabling change','FontSize',24);
c=colorbar('FontSize',18);
c.Label.String='log2 ratio probability WSP/MWS';
colormap jet;
axis square
eval(['print -fl -depsc -r300 WSP.vs.MWS.contour.eps']);

```

MATLAB code for interpreting saved data WSP vs AWS

```

% create record of how parameters change down, none, up for WSP
rs1=rand(1,6);
totnumruns=3.^length(rs1);
paramsWSP=zeros(totnumruns,length(rs1));
v=zeros(size(rs1));
i0=1;
while i0<=totnumruns;
    num=i0-1;
    for il=length(rs1)-1:-1:0;
        v(il+1)=floor(num/(3^il));
    end
    i0=i0+1;
end

```

```

        num=num-v(i1+1)*3^i1;
    end
    v=v+1;
    paramsWSP(i0,:)=v;
    i0=i0+1;
end
% create record of how parameters change down, none, up for AWS
rs1=rand(1,4);
totnumruns=3.^length(rs1);
paramsAWS=zeros(totnumruns,length(rs1));
v=zeros(size(rs1));
i0=1;
while i0<=totnumruns;
    num=i0-1;
    for i1=length(rs1)-1:-1:0;
        v(i1+1)=floor(num/(3^i1));
        num=num-v(i1+1)*3^i1;
    end
    v=v+1;
    paramsAWS(i0,:)=v;
    i0=i0+1;
end

% Load data
load pathway_results_complete_AWS.mat
resAWS=res;
yorigAWS=yorig';
clear res yorig
load pathway_results_complete_WSP.mat
resWSP=res;
yorigWSP=yorig';

% Variables and data storage to compare likelihood of pathways
numsampWSP=size(resWSP,2); % in case want to use fewer samples
numsampAWS=size(resAWS,2); % in case want to use fewer samples

perange=10.^[-7:.5:-1]; % range for probability enabling mutations
pdrange=10.^[-7:.5:-1]; % range for probability disabling mutations
pemat=zeros(length(perange),length(pdrange)); % matrix for plotting data and reference
pdmat=zeros(size(pemat)); % matrix for plotting data and reference
psummatWSP=zeros(size(pemat)); % matrix for probability WSP used
psummatAWS=zeros(size(pemat)); % matrix for probability MWS used
tol=0;
fac=5; % factor increase of mutation because of hotspot

% Code to compare likelihood of pathways
for i0=1:length(perange)
    for j0=1:length(pdrange)
        % Retrieve probabilities
        pe=perange(i0);
        pd=pdrange(j0);
        pemat(i0,j0)=pe;
        pdmat(i0,j0)=pd;
        % WSP computation
        pmatfor=ones(6,1)*[pd 1-pe-pd pe];
        psum=0; % total sum of (prob of rxn changes) X (number of times WS produced)
        for i1=1:size(resWSP,1)
            probevent=1; % initialize, probability to get combination of down, none, up for rxns

```

```

        for j1=1:6;
            probevent=probevent*pmatforr(j1,paramsWSP(i1,j1)); % multiply by prob of each change
        end
        psum=psum+probevent*sum(resWSP(i1,1:numsampWSP)>yorigWSP(1:numsampWSP)+tol)/numsampWSP;
    end
    psummatWSP(i0,j0)=psum;

    % AWS computation
    pmatforr=ones(4,1)*[pd 1-pe-pd pe];
    pmatforr(4,:)=[.5*pd 1-.5*pd-.5*pe .5*pe]; % because only reactant in dimerization
    pmatforr(3,:)=[fac*pd 1-fac*pd-fac*pe fac*pe]; % effect of hotspot
    pmatforr(2,:)=[fac*pd 1-fac*pd-fac*pe fac*pe]; % effect of hotspot
    psum=0; % total sum of (prob of rxn changes) X (number of times WS produced)
    for i1=1:size(resAWS,1)
        probevent=1; % initialize, probability to get combination of down, none, up for rxns
        for j1=1:4;
            probevent=probevent*pmatforr(j1,paramsAWS(i1,j1)); % multiply by prob of each change
        end
        psum=psum+probevent*sum(resAWS(i1,1:numsampAWS)>yorigAWS(1:numsampAWS)+tol)/numsampAWS;
    end
    psummatAWS(i0,j0)=psum;
end

% Plot data
close all
figure
contourf(pemat,pdmat,log2(psummatWSP./psummatAWS),400,'LineStyle','None')
set(gca,'xScale','log','yScale','log','TickLength',[.025 .025],'LineWidth',3);
set(gca,'FontSize',18,'xTick',10.^[-7:1:-1],'yTick',10.^[-7:1:-1]);
xlabel('Probability of enabling change','FontSize',24);
ylabel('Probability of disabling change','FontSize',24);
c=colorbar('FontSize',18);
c.Label.String='log2 ratio probability WSP/AWS';
colormap jet;
axis square
eval(['print -fl -depsc -r300 WSP-vs-AWS-contour.eps']);

```

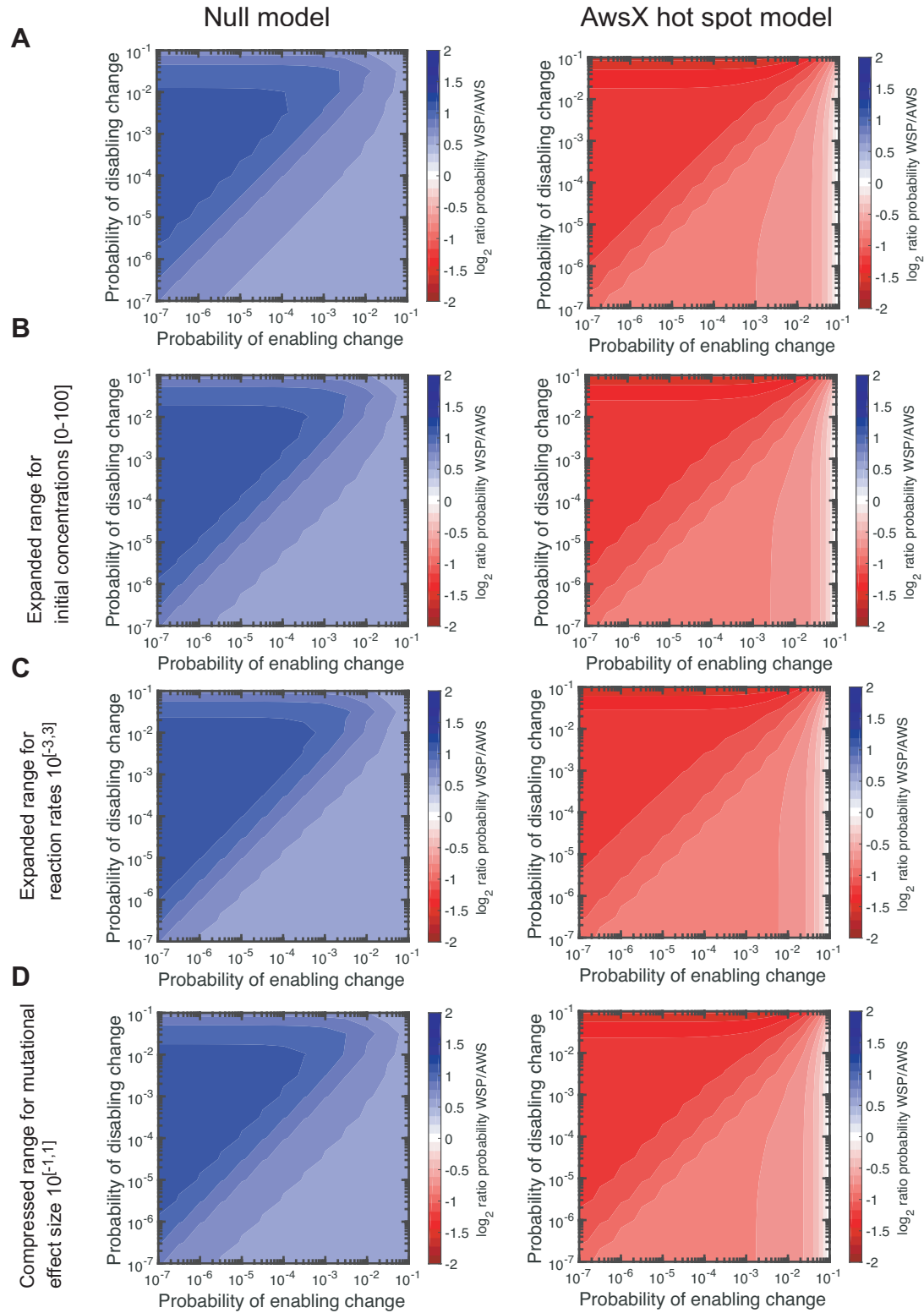


Figure S1. Parameter sensitivity analysis. To assess the effect of the chosen parameter (A) ranges on our results, we redid our sampling procedure for WSP for three different parameter regimes: (B) an expanded range for initial concentrations [0-100], (C) an expanded range for reaction rates 10^[-3,3], (D) a compressed range for mutational effect size 10^[-1,1]. We found that our qualitative results are robust to these changes.