This *Notebook* can be used to restore the results of figure

*"**Dependence of the optimal plasmid copy number on the antibiotic concentration.**" (Panel C, E, F)*

Clear Mathematica environment

In[1582]:=

```
ClearAll["Global`*"]

(*Skipped Antibiotic Concentrations. *)
skippedAB = 1;(*Use 4 for testing (~10s)
 and use 1 to obtain results from paper (~5min)*)
```

Set parameters

In[1584]:=

```
(*Set N0*u*)
N0u=.2;
(*Set death rate (constant)*)
μ=1.;
(*Set parameters for fitness curve*)
{Smax,Smin,cMIC,κ}={.5,-.2,1.,8.};
(*Set parameters for function of available enzymes*)
{η,ω}={1.3,0.2};
(*Set parameter for diffusion coefficient*)
γ=1.;
(*Set parameter for plasmid cost*)
c=0.01;
(*Set range of plasmid copy numbers n*)
nRange={1,2,5,10};
(*Set range of antibiotic concentrations outside the cell*)
coutRange=Range[cMIC,2.4,.02][[1;;-1;;skippedAB]];
```

Define parameters and functions for "degradation types", etc.

In[1592]:=

```
(*Define function for the fitness S*)
S[cin_]:=Smax-((Smax-Smin)(cin/cMIC)^κ)/((cin/cMIC)^κ-Smin/Smax);

(*Define function for the amount of available enzymes*)
                 η*i/n  StringContainsQ[type,"Relative"]
f[i_,n_,type_]:={ ω*i    StringContainsQ[type,"Absolute"] ;
                 Null   True
(*Define function for the internal concentration*)
cin[cout_,i_,n_,type_]:=cout/(f[i,n,type]/γ+1.);
```

In[1595]:=

```
(*Helper function returning the birth rate for a given fitness*)
λ1[S_]=μ+S;

(*Helper function for death rate if necessary including costs*)
μ1[n_,type_]:={ μ+c*n  StringContainsQ[type,"Cost"]
                μ      True

(*Helper function returning the birth probability for a given birth rate*)
p1[λ_,μ_]:=λ/(λ+μ);
```

In[1598]:=

```
(*Functions for birth rate, death rate and birth probability respectively*)
λ[n_,type_,cout_]:=Function[i,λ1[S[cin[cout,i,n,type]]]]
p[n_,type_,cout_]:=Function[i,p1[λ1[S[cin[cout,i,n,type]]],μ1[n,type]]]
```

Define functions for calculation of establishment and rescue probability

In[1600]:=

```
(*Define a module to calculate the establishment probability
  from the fixed point of the probability generating function
given the plasmid copy number n and the function
  returning the probabilities of birth p[n]:
*)

Pest[n_, p_Function] := Module[{Pest, fi, f, q},
If[p[n] <= 0.5, (*Exclude the trivial case where establishment is impossible*)
Pest = 0,
(*Define vector components of probability generating function (PGF)*)
fi[i_Integer, x_List] =
    1 - p[i] (1 - Sum[PDF[HypergeometricDistribution[n, 2 i, 2 n], l] * Indexed[x, l + 1] *
          Indexed[x, 2 i - l + 1], {l, Max[0, 2 i - n], Min[n, 2 i]}]);
(*Define multitype probability generating function*)
f[x_List] = Table[Apply[fi, {i, x}], {i, 0, n}];
(*Calculation of the fixed point by numerically iterating the PGF*)
q = FixedPoint[f, ConstantArray[0, n + 1]];
(*Calculate the establishment
     probability from the component Subscript[Q, 1]=q[[2]]
since q=(Subscript[Q, 0],Subscript[Q, 1],...)*)
Pest = 1 - q[[2]];
];
(*Return the establishment probability*)
Pest
]
(*Define a function for the de-novo rescue probability given the PCN n,
establishment
 probability Pest,
exclude the trivial case λ0≥μ where wild-type population does not
   decline*)
Pres1[n_, Pest_, λ0_, μ0_] := 1 - Exp[-N0u * λ0 / (μ0 - λ0) * n * Pest];
Pres[n_, type_, cout_] :=
 { 1.                                          λ[n, type, cout][0] ≥ μ1[n, type]
   Pres1[n, Pest[n, p[n, type, cout]],         True
    λ[n, type, cout][0], μ1[n, type]]
```

Calculate rescue probabilities
for the three types
Panel C – *Relative* (Relative number of mutated plasmids determine the degredation rate),
Panel E – *Absolute* (Absolute number of mutated plasmids determine the degredation rate),
Panel F – *AbsoluteCost* (like Absolute and fitness is deteriorating with

increasing PCN n)
for range of antibiotic concentrations c_{out} (columns)
and for various plasmid copy numbers n (rows)

In[1603]:=

```
(*Helper function to calculate data for ranges*)
CalculateRescueProbabilities[type_] :=
 Table[{cout, Pres[n, type, cout]}, {n, nRange}, {cout, coutRange}]
(*Helper function to show result table as a grid*)
GridData[t_] :=
 Grid[MapThread[Prepend, {Prepend[t[[All, 1 ;; -1 ;; Ceiling[Length[t[[1]]] / 10], 2]]
     , coutRange[[1 ;; -1 ;; Ceiling[Length[t[[1]]] / 10]]]], Prepend[nRange, ""]}]]
(*Helper function fr plots*)
ListPlotData[t_, type_] := ListPlot[t, PlotLabel → type,
  FrameLabel → {"Antibiotic concentration cout", "Rescue probability"
}, PlotTheme → "Scientific", PlotRange → {0, 1},
  PlotLegends → PointLegend[nRange, LegendLabel → "Plasmid copy \nnumber n"]]
```

In[1606]:=

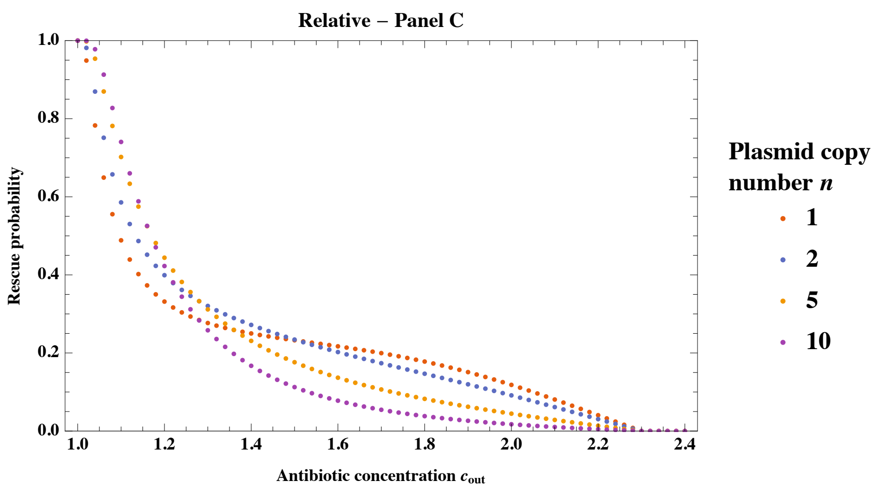
```
type="Relative - Panel C";

t=CalculateRescueProbabilities[type];
GridData[t]
ListPlotData[t,type]
```

Out[1608]=

	1.	1.16	1.32	1.48	1.64	1.8	1.96	2.12	2.28
1	1.	0.373062	0.270002	0.235751	0.21044	0.178014	0.131999	0.0728973	0.00800886
2	1.	0.451719	0.30926	0.241318	0.190601	0.146823	0.102938	0.055315	0.00601854
5	1.	0.524793	0.292558	0.185786	0.123721	0.0824521	0.0514308	0.0254862	0.00264904
10	1.	0.52549	0.235672	0.121536	0.0674237	0.0382158	0.0207279	0.00917493	0.000876751

Out[1609]=



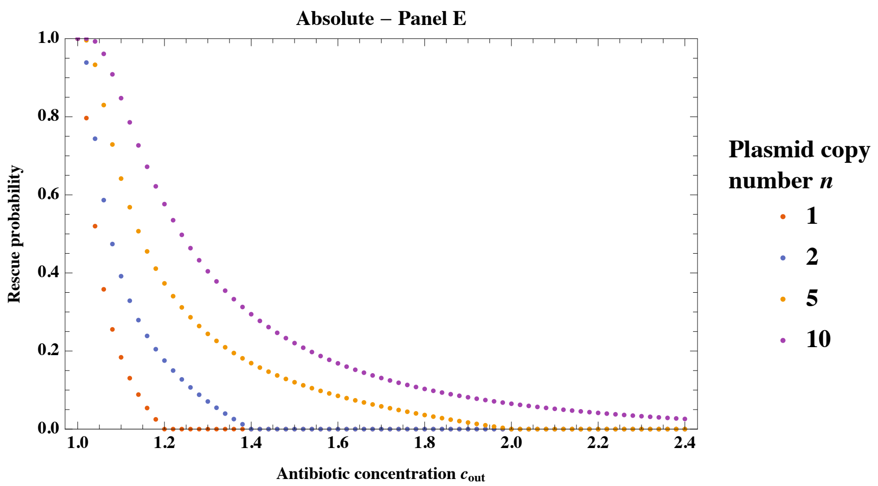
In[1610]:=
```
type="Absolute - Panel E";

t=CalculateRescueProbabilities[type];
GridData[t]
ListPlotData[t,type]
```

Out[1612]=

	1.	1.16	1.32	1.48	1.64	1.8	1.96	2.12	2.28
1	1.	0.0540246	0.	0.	0.	0.	0.	0.	0.
2	1.	0.238689	0.0547335	0.	0.	0.	0.	0.	0.
5	1.	0.455145	0.225918	0.128467	0.0736891	0.0360827	0.0065019	0.	0.
10	1.	0.671943	0.378353	0.23303	0.152042	0.102874	0.07115	0.0497087	0.0346207

Out[1613]=



In[1614]:=
```
type="AbsoluteCost - Panel F";

t=CalculateRescueProbabilities[type];
GridData[t]
ListPlotData[t,type]
```

Out[1616]=

	1.	1.16	1.32	1.48	1.64	1.8	1.96	2.12	2.28
1	0.9763⋱ 48	0.0380⋱ 724	0.	0.	0.	0.	0.	0.	0.
2	0.9531⋱ 65	0.1910⋱ 71	0.0352⋱ 53	0.	0.	0.	0.	0.	0.
5	0.8786⋱ 72	0.2923⋱ 89	0.1418⋱ 54	0.0769⋱ 704	0.0404⋱ 814	0.0152⋱ 813	0.	0.	0.
10	0.7791⋱ 25	0.31226	0.15717	0.0885⋱ 821	0.0533⋱ 444	0.0334⋱ 503	0.0214⋱ 323	0.0137⋱ 749	0.0086⋱ 4951

Out[1617]=