Analytics

Deriving the model dynamics

Stationary population sizes of a wild - type only population

In old - habitat patches

In old-habitat patches, we assume that the wild-type local populations are at carrying capacity K_{old}

In[•]:=
```
Nhatold = Kold;
```

In new - habitat patches

The stationary population size is found recursively, by considering the different events in the life cycle, and assuming here that carrying capacity is not reached, so that there is no need for density regulation.

Note : for simplicity, we write f for f_{old}

In[•]:=
```
Solve[x == (1 - m + m (1 - f)/(1 - f + ℯ^(π ω) f)) ωWnew x + m (1 - f)/(1 - f + ℯ^(π ω) f) f/(1 - f) ωWnew Kold, x] //
  FullSimplify
solNhatnew = %[[1]];
```

Out[•]=
$$\left\{\left\{x \to \frac{f\, m\, K_{old}\, \omega W_{new}}{1 + \left(-1 + e^{\pi \omega}\right) f + \left(-1 + f + e^{\pi \omega} f\, (-1 + m)\right) \omega W_{new}}\right\}\right\}$$

By definition, this population size cannot exceed K_{new}:

In[•]:=
```
Nhatnew = Min[Knew, x /. solNhatnew];
```

Population sizes after dispersal, wild - type only population

In old - habitat patches

In[•]:=
```
Ntildeold =
  (1 - m + m ℯ^(π ω) f/(1 - f + ℯ^(π ω) f)) Nhatold + m ℯ^(π ω) f/(1 - f + ℯ^(π ω) f) (1 - f)/f Nhatnew // FullSimplify
```

Out[•]=
$$\frac{-e^{\pi \omega}\left(-1 + f\right) m\, \text{Min}\left[K_{new}, \frac{f\, m\, K_{old}\, \omega W_{new}}{1 + \left(-1 + e^{\pi \omega}\right) f + \left(-1 + f + e^{\pi \omega} f\, (-1 + m)\right) \omega W_{new}}\right] + \left(1 - m + f\left(-1 + e^{\pi \omega} + m\right)\right) K_{old}}{1 + \left(-1 + e^{\pi \omega}\right) f}$$

In new - habitat patches

```
In[•]:= Ntildenew =
  (1 - m + m (1 - f)/(1 - f + e^(πw) f)) Nhatnew + m (1 - f)/(1 - f + e^(πw) f) f/(1 - f) Nhatold // FullSimplify
```

$$\text{Out[•]=}\quad \frac{-\left(-1+f+e^{\pi w} f\,(-1+m)\right)\,\text{Min}\left[K_{new},\ \frac{f\,m\,K_{old}\,\omega w_{new}}{1+\left(-1+e^{\pi w}\right) f+\left(-1+f+e^{\pi w} f\,(-1+m)\right)\,\omega w_{new}}\right]+f\,m\,K_{old}}{1+\left(-1+e^{\pi w}\right) f}$$

Per capita growth rate of mutants

In old - habitat patches

```
In[•]:= a_old = ωm_old K_old/(ωw_old Ntildeold) - 1;
```

In new - habitat patches

Since we assume that there is no density regulation in new - habitat patches because population size is below carrying capacity,

```
In[•]:= a_new = ωm_new - 1;
```

Establishment probability

System to be numerically solved

```
Eq_old =
  {1 - ϕ_old == Exp[-(1 - m (1 - f)/(1 - f + e^(πm) f)) (1 + a_old) ϕ_old - m (1 - f)/(1 - f + e^(πm) f) (1 + a_new) ϕ_new]};
Eq_new = {1 - ϕ_new == Exp[-m e^(πm) f/(1 - f + e^(πm) f) (1 + a_old) ϕ_old - (1 - m e^(πm) f/(1 - f + e^(πm) f)) (1 + a_new) ϕ_new]};
```

Approximation of the establishment probability

Definition of the Reproduction matrix

To avoid replacing $a_{old}$and $a_{new}$by their expressions, we write them differently, as s_{old} and s_{new}
The matrix contain the expected number of successful offspring over the whole life cycle (including dispersal), i.e. the λ terms defined in the main text.

To avoid too lengthy expressions,

we use μ_{new} and μ_{old} for the probabilites that a disperser goes to a new or old patch

```
In[•]:= reproMatrix =
          {{(1 - m μnew) (1 + sold), m μnew (1 + snew)}, {m μold (1 + sold), (1 - m μold) (1 + snew)}};
        reproMatrix // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} (1+s_{old})\,(1-m\,\mu_{new}) & m\,(1+s_{new})\,\mu_{new} \\ m\,(1+s_{old})\,\mu_{old} & (1+s_{new})\,(1-m\,\mu_{old}) \end{pmatrix}$$

Largest eigenvalue of the reproduction matrix and Taylor - expansion

We search for the largest eigenvalue of this reproduction matrix
(the denominator is positive, so we pick the one with a + in front of the square root)

```
In[•]:= Eigenvalues[reproMatrix]
        ρ = %[[2]] // FullSimplify;
```

Out[•]=

$$\left\{\frac{1}{2}\left(2+s_{new}+s_{old}-m\,\mu_{new}-m\,s_{old}\,\mu_{new}-m\,\mu_{old}-m\,s_{new}\,\mu_{old}-\sqrt{\left(\left(-2-s_{new}-s_{old}+m\,\mu_{new}+m\,s_{old}\,\mu_{new}+m\,\mu_{old}+m\,s_{new}\,\mu_{old}\right)^2-4\,(1+s_{new}+s_{old}+s_{new}\,s_{old}-m\,\mu_{new}-m\,s_{new}\,\mu_{new}-m\,s_{old}\,\mu_{new}-m\,s_{new}\,s_{old}\,\mu_{new}-m\,\mu_{old}-m\,s_{new}\,\mu_{old}-m\,s_{old}\,\mu_{old}-m\,s_{new}\,s_{old}\,\mu_{old})\right)}\right),\right.$$
$$\left.\frac{1}{2}\left(2+s_{new}+s_{old}-m\,\mu_{new}-m\,s_{old}\,\mu_{new}-m\,\mu_{old}-m\,s_{new}\,\mu_{old}+\sqrt{\left(\left(-2-s_{new}-s_{old}+m\,\mu_{new}+m\,s_{old}\,\mu_{new}+m\,\mu_{old}+m\,s_{new}\,\mu_{old}\right)^2-4\,(1+s_{new}+s_{old}+s_{new}\,s_{old}-m\,\mu_{new}-m\,s_{new}\,\mu_{new}-m\,s_{old}\,\mu_{new}-m\,s_{new}\,s_{old}\,\mu_{new}-m\,\mu_{old}-m\,s_{new}\,\mu_{old}-m\,s_{old}\,\mu_{old}-m\,s_{new}\,s_{old}\,\mu_{old})\right)}\right)\right\}$$

Define assumptions on the parameters to help Mathematical simplify solutions
(note that both $\epsilon>0$ or $\epsilon<0$ work. But one needs to assume one for Mathematica to solve the square roots)

```
In[•]:= assumptions = {m > 0 && m < 1 && πm ∈ Reals &&
            πw ∈ Reals && ϵ > 0 && snew > 0 && ξ > 0 && f > 0 && f < 1 && snew > sold};
```

Define scaling to rescale parameters as function of ϵ, assumed to be small

```
In[•]:= scaling = {m → ϵ mm, sold → ϵ ssold, snew → ϵ ssnew};
```

Taylor - expand the eigenvalue, at the first order in ϵ (after rescaling), and simplify

```
In[•]:= Assuming[assumptions, Series[ρ /. scaling, {ϵ, 0, 1}] // Normal // FullSimplify]
```

Out[•]=

$$1+\frac{1}{2}\epsilon\left(ss_{new}+ss_{old}-mm\,(\mu_{new}+\mu_{old})+\sqrt{(ss_{new}-ss_{old}+mm\,\mu_{new})^2+2\,mm\,(-ss_{new}+ss_{old}+mm\,\mu_{new})\,\mu_{old}+mm^2\,\mu_{old}^2}\right)$$

Eigenvectors

Left eigenvector associated to the largest eigenvalue (second one in the order of vectors in the output)

```
In[•]:= u = Eigenvectors[Transpose[reproMatrix]][[2]] // FullSimplify;
```

Right eigenvector

```
In[•]:= v = Eigenvectors[reproMatrix][[2]] // FullSimplify;
```

Normalize the eigenvectors

Find the coefficients satisfying the following constraints (lev = left eigenvector, rev = right eigenvector, the numbers refer to the elements)

```
In[•]:= Solve[{a (lev1 + lev2) == 1, a lev1 b rev1 + a lev2 b rev2 == 1}, {a, b}]
```

$$\text{Out[•]= } \left\{\left\{a \to \frac{1}{\text{lev1} + \text{lev2}},\ b \to \frac{\text{lev1} + \text{lev2}}{\text{lev1 rev1} + \text{lev2 rev2}}\right\}\right\}$$

a is the sum of the elements of the left eigenvector, b' s denominator is the product of the vectors. The normalized vectors (n) are therefore

```
In[•]:= un = 1/Total[u] u;
        vn = Total[u]/u.v v;
```

Approximate establishment probability

Directly applying Theorem 5.6 from Haccou et al.(2005)

We define

```
In[•]:= B = Sum[un[[i]] (Sum[vn[[j]] reproMatrix[[i, j]], {j, 2}]), {i, 2}] + ρ (1 - ρ) (Sum[un[[j]] vn[[j]]^2, {j, 2}]);
```

and according to the theorem, the approximate establishment probabilities are then

```
In[•]:= exactϕ = 2 (ρ - 1) vn / B;
```

which we Taylor - Expand as (this takes a few seconds/minutes)

```
In[•]:= solutions = Assuming[assumptions,
          Assuming[assumptions, Normal[Series[exactϕ /. scaling, {ϵ, 0, 1}]]]];
```

and we recover the original parameters (this also takes a few seconds/minutes)

```
In[•]:= backscaling = {mm → m / ϵ, ss_old → s_old / ϵ, ss_new → s_new / ϵ};
        backsolutions = Assuming[assumptions, FullSimplify[solutions /. backscaling]]
```

$$\text{Out[•]= } \left\{\left(-s_{new}\left(s_{old} - 2\,m\,\mu_{new}\right) + s_{old}\left(s_{old} - m\,\mu_{new} + m\,\mu_{old} + \sqrt{\left(s_{new} - s_{old} + m\,\mu_{new}\right)^2 + 2\,m\left(-s_{new} + s_{old} + m\,\mu_{new}\right)\mu_{old} + m^2\,\mu_{old}^2}\right)\right) \Big/ \left(\sqrt{\left(s_{new} - s_{old} + m\,\mu_{new}\right)^2 + 2\,m\left(-s_{new} + s_{old} + m\,\mu_{new}\right)\mu_{old} + m^2\,\mu_{old}^2}\right),\ \left(s_{new}^2 + 2\,m\,s_{old}\,\mu_{old} + s_{new}\left(-s_{old} + m\,\mu_{new} - m\,\mu_{old} + \sqrt{\left(s_{new} - s_{old} + m\,\mu_{new}\right)^2 + 2\,m\left(-s_{new} + s_{old} + m\,\mu_{new}\right)\mu_{old} + m^2\,\mu_{old}^2}\right)\right) \Big/ \left(\sqrt{\left(s_{new} - s_{old} + m\,\mu_{new}\right)^2 + 2\,m\left(-s_{new} + s_{old} + m\,\mu_{new}\right)\mu_{old} + m^2\,\mu_{old}^2}\right)\right\}$$

Extract the two establishment probabilities, and recover the model parameters

```
In[•]:= recoverparams = {μ_new → (1 - f)/(1 - f + e^(π m) f), μ_old → e^(π m) f/(1 - f + e^(π m) f), s_old → a_old, s_new → a_new};
```

```
In[•]:= ϕapprox_old = backsolutions[[1]] /. recoverparams;
        ϕapprox_new = backsolutions[[2]] /. recoverparams;
```

Figures

Initializations

Parameters

```
In[•]:= constantParam = {K_old → 1000, K_new → 500, ωw_old → 1.5, ωw_new → 0.75, ωm_new → 1.02};
```

The other default parameters are
f = 0.5,
m = 0.06,
ωm_{old} = 1.45 or 1.35

```
In[•]:= mdefault = 0.06;
        fdefault = 0.5;
        ωmdefault1_old = 1.35;
        ωmdefault2_old = 1.45;
```

```
In[•]:= π0 = {πw → 0, πm → 0};
        πOO = {πw → 1 / 2, πm → 1 / 2};
        πON = {πw → 1 / 2, πm → -1 / 2};
        πNO = {πw → -1 / 2, πm → 1 / 2};
        πNN = {πw → -1 / 2, πm → -1 / 2};
```

Emigration probability m

```
In[•]:= mmin = 10^-3; (* minimum value *)
        mmax = 1; (* max value *)
        nm = 101; (* number of values *)
        logmvals = Table[Log10[mmin] + (Log10[mmax] - Log10[mmin]) (i - 1) / (nm - 1), {i, 1, nm}];
        mvals = Table[10^logmvals[[i]], {i, 1, nm}];
```

Plotting parameters

```
In[•]:= col0 = Black;
        colOO = Blue;
        colON = Orange;
        colNO = Red;
        colNN = Green;
```

```
In[•]:= mark0 = ●;
        markOO = ▲;
        markON = ■;
        markNO = ◆;
        markNN = ▼;
```

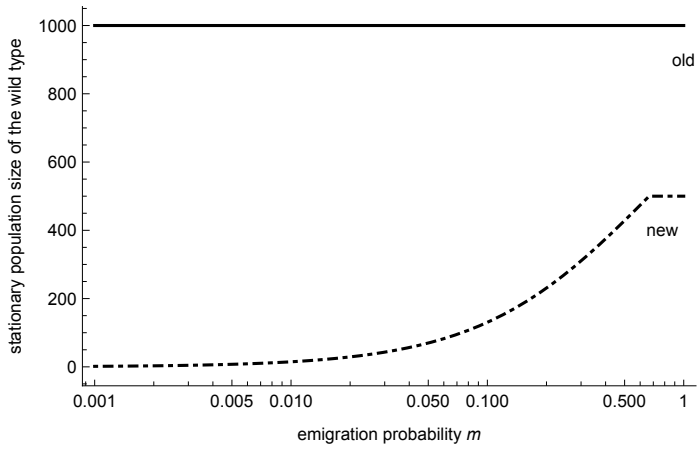
Population variables

Nhatold and Nhatnew, the population densities of the wild-type population at the beginning of a generation

Only for unbiaised dispersal

In[•]:=
```
Show[LogLinearPlot[
  {Nhatold /. constantParam, Nhatnew /. constantParam /. {f → fdefault} /. π0},
  {m, mmin, mmax}, PlotStyle → {{col0, Full}, {col0, DotDashed}},
  AxesOrigin → {mmin, 0}, Frame → {True, True, False, False},
  FrameLabel → {"emigration probability m",
    "stationary population size of the wild type"}],
 Graphics[Text["old", {0, 900}]], Graphics[Text["new", {-0.25, 400}]]]
```

Out[•]=

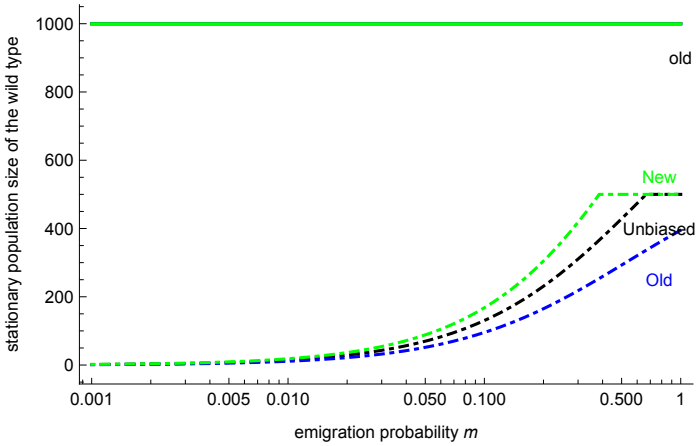


For all WT dispersal biases

(The dispersal bias of the mutant does not affect the population size of the wild - type population)

```
In[•]:= Show[LogLinearPlot[
          {Nhatold /. constantParam, Nhatnew /. constantParam /. {f → fdefault} /. π0},
          {m, mmin, mmax}, PlotStyle → {{col0, Full}, {col0, DotDashed}},
          AxesOrigin → {mmin, 0}, Frame → {True, True, False, False},
          FrameLabel → {"emigration probability m",
            "stationary population size of the wild type"}],
        LogLinearPlot[{Nhatold /. constantParam,
          Nhatnew /. constantParam /. {f → fdefault} /. πOO},
          {m, mmin, mmax}, PlotStyle → {{colOO, Full}, {colOO, DotDashed}}],
        LogLinearPlot[{Nhatold /. constantParam,
          Nhatnew /. constantParam /. {f → fdefault} /. πNN},
          {m, mmin, mmax}, PlotStyle → {{colNN, Full}, {colNN, DotDashed}}],
        Graphics[Text["old", {0, 900}]],
        Graphics[Text[Style["Old", colOO], {-0.25, 250}]],
        Graphics[Text[Style["New", colNN], {-0.25, 550}]],
        Graphics[Text[Style["Unbiased", Small], {-0.25, 400}]]]
```

Out[•]=



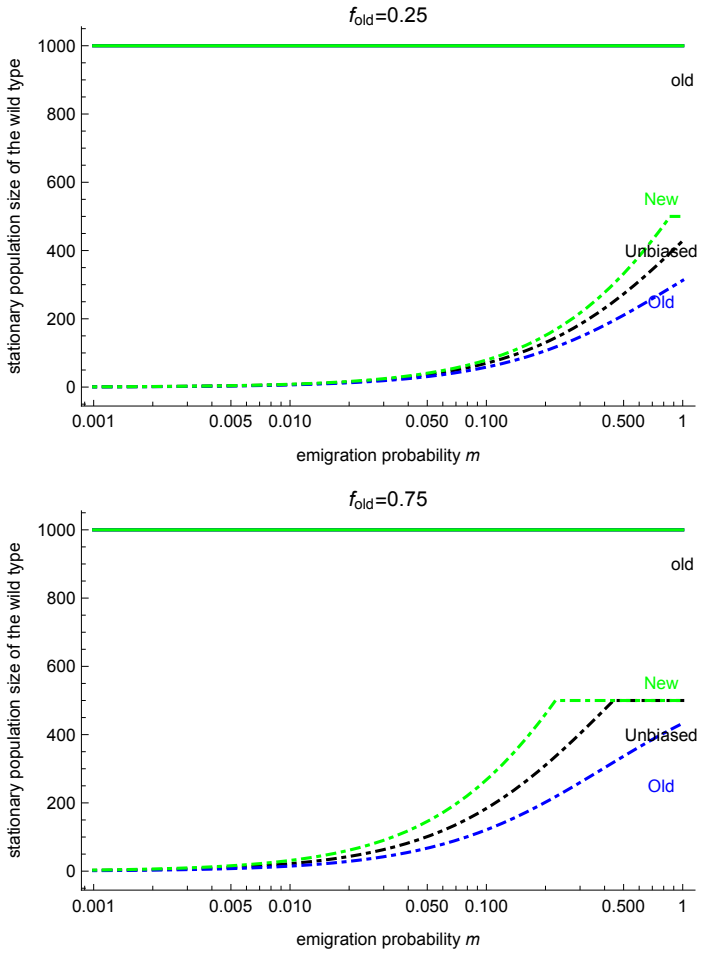
With other values of f_{old}

```
In[•]:= Do[Pthef = Show[LogLinearPlot[
            {Nhatold /. constantParam, Nhatnew /. constantParam /. {f → thef} /. π0},
            {m, mmin, mmax}, PlotStyle → {{col0, Full}, {col0, DotDashed}},
            AxesOrigin → {mmin, 0}, Frame → {True, True, False, False},
            FrameLabel → {"emigration probability m",
              "stationary population size of the wild type"}],
          LogLinearPlot[{Nhatold /. constantParam,
            Nhatnew /. constantParam /. {f → thef} /. πOO},
            {m, mmin, mmax}, PlotStyle → {{colOO, Full}, {colOO, DotDashed}}],
          LogLinearPlot[{Nhatold /. constantParam,
            Nhatnew /. constantParam /. {f → thef} /. πNN}, {m, mmin, mmax},
            PlotStyle → {{colNN, Full}, {colNN, DotDashed}}], Graphics[
            Text["old", {0, 900}]], Graphics[Text[Style["Old", colOO], {-0.25, 250}]],
          Graphics[Text[Style["New", colNN], {-0.25, 550}]],
          Graphics[Text[Style["Unbiased", Small], {-0.25, 400}]],
          PlotLabel → "fold=" <> ToString[thef]]
        , {thef, {0.25, 0.75}}]
```

In[•]:= $P_{0.25}$

$P_{0.75}$

Out[•]=

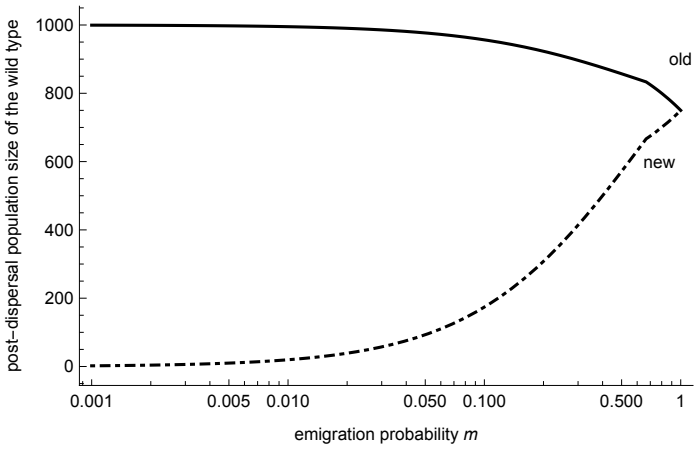


Ntildeold and Ntildenew, the population densities of the wild - type population right after dispersal

Only for unbiaised dispersal

In[•]:=
```
Show[LogLinearPlot[{Ntildeold /. constantParam /. {f → fdefault} /. π0,
    Ntildenew /. constantParam /. {f → fdefault} /. π0}, {m, mmin, mmax},
  PlotStyle → {{col0, Full}, {col0, DotDashed}}, AxesOrigin → {mmin, 0},
  Frame → {True, True, False, False}, FrameLabel → {"emigration probability m",
    "post-dispersal population size of the wild type"}],
 Graphics[Text["old", {0, 900}]], Graphics[Text["new", {-0.25, 600}]]]
```

Out[•]=

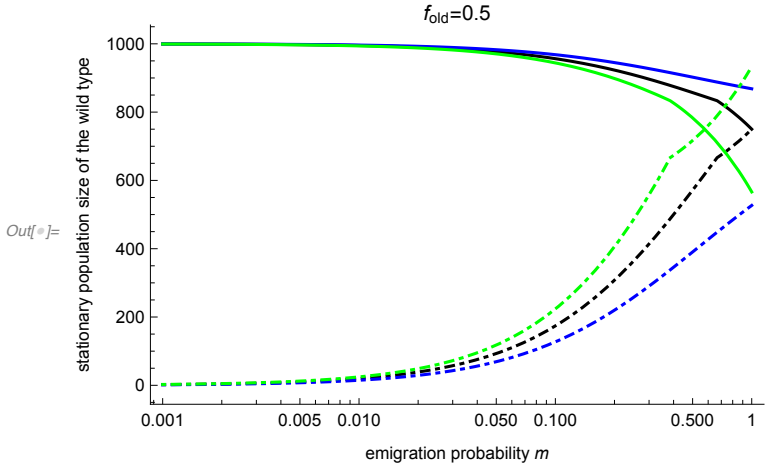


For all WT dispersal biases

Again, dispersal bias of the mutant has no influence

In[•]:=
```
Show[LogLinearPlot[{Ntildeold /. constantParam /. {f → fdefault} /. π0,
    Ntildenew /. constantParam /. {f → fdefault} /. π0}, {m, mmin, mmax},
  PlotStyle → {{col0, Full}, {col0, DotDashed}}, AxesOrigin → {mmin, 0},
  Frame → {True, True, False, False}, FrameLabel → {"emigration probability m",
    "stationary population size of the wild type"}],
 LogLinearPlot[{Ntildeold /. constantParam /. {f → fdefault} /. π00,
    Ntildenew /. constantParam /. {f → fdefault} /. π00},
  {m, mmin, mmax}, PlotStyle → {{col00, Full}, {col00, DotDashed}}],
 LogLinearPlot[{Ntildeold /. constantParam /. {f → fdefault} /. πNN,
    Ntildenew /. constantParam /. {f → fdefault} /. πNN},
  {m, mmin, mmax}, PlotStyle → {{colNN, Full}, {colNN, DotDashed}}],
 PlotLabel → "f_old=" <> ToString[fdefault]]
```

Out[•]=

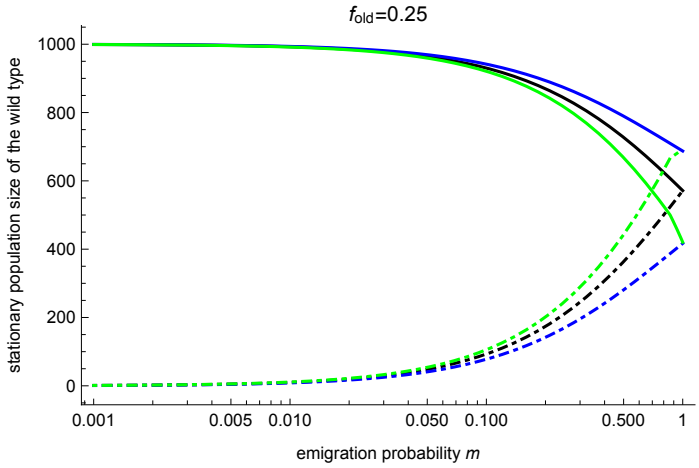


For other values of f_{old}

In[•]:=
```
Do[P_thef =
  Show[LogLinearPlot[{Ntildeold /. constantParam /. {f → thef} /. π0,
      Ntildenew /. constantParam /. {f → thef} /. π0},
    {m, mmin, mmax}, PlotStyle → {{col0, Full}, {col0, DotDashed}},
    AxesOrigin → {mmin, 0}, Frame → {True, True, False, False},
    FrameLabel → {"emigration probability m",
      "stationary population size of the wild type"}],
   LogLinearPlot[{Ntildeold /. constantParam /. {f → thef} /. π00,
      Ntildenew /. constantParam /. {f → thef} /. π00},
    {m, mmin, mmax}, PlotStyle → {{col00, Full}, {col00, DotDashed}}],
   LogLinearPlot[{Ntildeold /. constantParam /. {f → thef} /. πNN,
      Ntildenew /. constantParam /. {f → thef} /. πNN},
    {m, mmin, mmax}, PlotStyle → {{colNN, Full}, {colNN, DotDashed}}],
   PlotLabel → "f_old=" <> ToString[thef]], {thef, {0.25, 0.75}}]
```

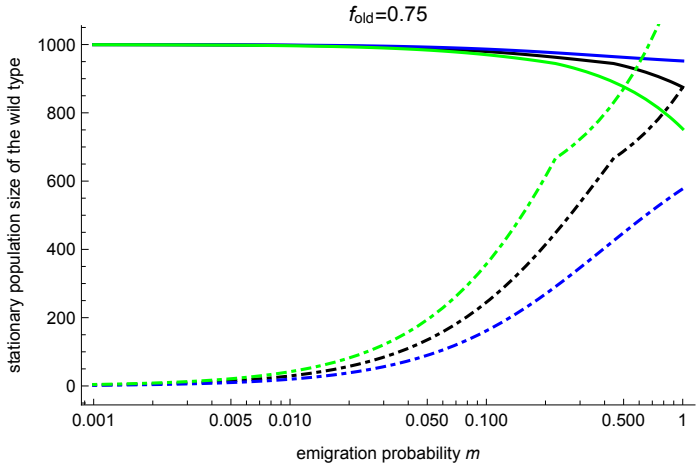
In[•]:= **P0.25**

Out[•]=



In[•]:= **P0.75**

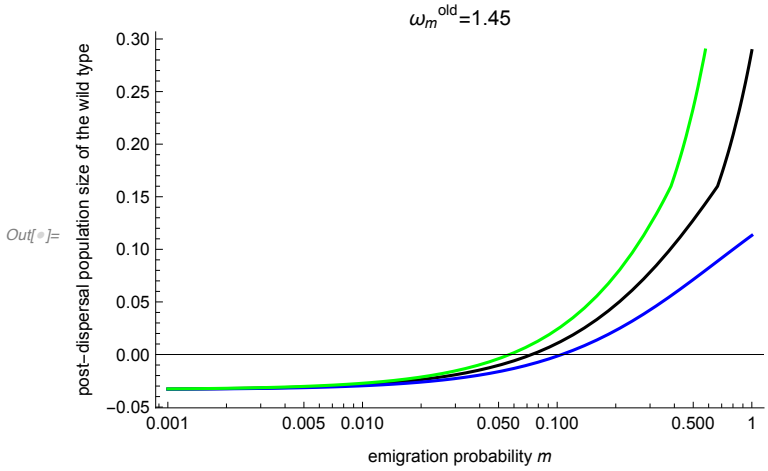
Out[•]=



a_{old}, the per capita local growth rate of the mutant

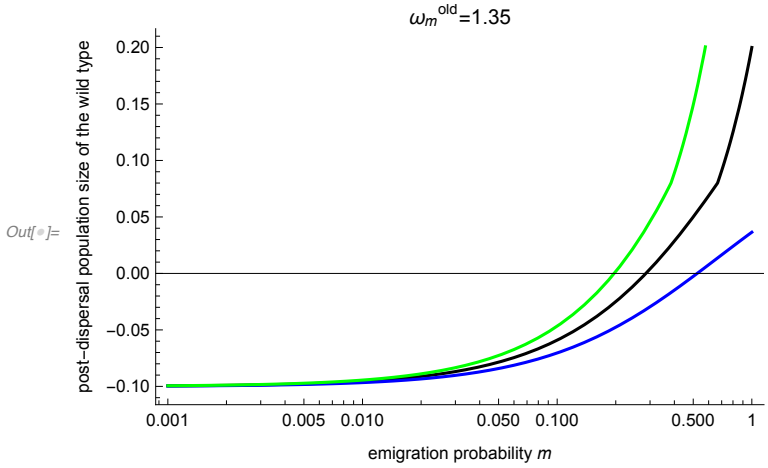
In[•]:=
```
Show[
 LogLinearPlot[aold /. constantParam /. {f → fdefault, ωmold → ωmdefault2old} /. π0,
  {m, mmin, mmax}, PlotStyle → {{col0, Full}, {col0, DotDashed}},
  AxesOrigin → {mmin, 0}, Frame → {True, True, False, False},
  FrameLabel → {"emigration probability m",
    "post-dispersal population size of the wild type"}],
 LogLinearPlot[aold /. constantParam /. {f → fdefault, ωmold → ωmdefault2old} /. πOO,
  {m, mmin, mmax}, PlotStyle → {{colOO, Full}, {colOO, DotDashed}}],
 LogLinearPlot[aold /. constantParam /. {f → fdefault, ωmold → ωmdefault2old} /. πNN,
  {m, mmin, mmax}, PlotStyle → {{colNN, Full}, {colNN, DotDashed}}],
 Graphics[Text["old", {0, 900}]], Graphics[Text["new", {-0.25, 600}]],
 PlotLabel → "ωm^old=" <> ToString[ωmdefault2old]]
```

Out[•]=



In[•]:=
```
Show[
 LogLinearPlot[a_old /. constantParam /. {f → fdefault, ωm_old → ωmdefault1_old} /. π0,
  {m, mmin, mmax}, PlotStyle → {{col0, Full}, {col0, DotDashed}},
  AxesOrigin → {mmin, 0}, Frame → {True, True, False, False},
  FrameLabel → {"emigration probability m",
    "post-dispersal population size of the wild type"}],
 LogLinearPlot[a_old /. constantParam /. {f → fdefault, ωm_old → ωmdefault1_old} /. π00,
  {m, mmin, mmax}, PlotStyle → {{col00, Full}, {col00, DotDashed}}],
 LogLinearPlot[a_old /. constantParam /. {f → fdefault, ωm_old → ωmdefault1_old} /. πNN,
  {m, mmin, mmax}, PlotStyle → {{colNN, Full}, {colNN, DotDashed}}],
 Graphics[Text["old", {0, 900}]], Graphics[Text["new", {-0.25, 600}]],
 PlotLabel → "ω_m^old=" <> ToString[ωmdefault1_old]]
```

Out[•]=



In[•]:=

In[•]:=

Tests -- Sandbox

Numerically solving the establishment probabilities

In[•]:=
```
FindRoot[Evaluate[{Eq_old, Eq_new} /. constantParam /. {f → fdefault,
      m → mdefault, ωm_old → ωmdefault1_old} /. π0], {{ϕ_old, 0.5}, {ϕ_new, 0.5}}]
```

Out[•]= $\{\phi_{old} \to 0.114666, \phi_{new} \to 0.0743409\}$

Plotting simulation data

In[•]:=
```
SetDirectory[
 "/Users/flo/Documents/Work/Projects/1_EnCours/2018_RescuePete/CodesPete"]
```

Out[•]= /Users/flo/Documents/Work/Projects/1_EnCours/2018_RescuePete/CodesPete

In[]:=
```
aa = Import[
  "../evolutionary_rescue_and_dispersal/Fig2/a/vary_mu_phi1_pi1_0_pi2_0.csv"]
```

Out[]=
```
{{0.00229, 0.02}, {0.00135, 0.008}, {0.00156, 0.01}, {0, 0.06},
 {0.00017, 0.001}, {0.00039, 0.003}, {0.00032, 0.0015}, {0, 0.15},
 {0.00044, 0.002}, {0.00067, 0.004}, {0, 0.08}, {0.00087, 0.005},
 {0.00098, 0.006}, {0, 0.1}, {0.00177, 0.015}, {0.00188, 0.03},
 {0.0013, 0.04}, {0.00019, 0.05}, {0.09898, 1}, {0.09258, 0.8},
 {0.00007, 0.2}, {0.0237, 0.3}, {0.04701, 0.4}, {0.08215, 0.6}, {0.06763, 0.5}}
```

In[]:=
```
ListLogLinearPlot[Table[{aa[[i, 2]], aa[[i, 1]]}, {i, 1, Length[aa]}]]
```

Out[]=