Trait-Based Model

This model examines how the initial distribution of traits in a microbial population impacts diversity in a chemostat as dilution rate increases. Two scenarios are examined, one where the trait governing bacteria specific growth rate, ϵ_i is uniformly distributed, such that there are as many bacteria that have a specific growth rate near the upper growth rate limit as for any other specific growth rate. The other scenario examines our hypothesis that as you approach the upper specific growth rate limit for bacteria, there are many fewer ASVs capable of growing at that boundary. We randomly select ϵ_i from a beta probability distribution to explore the second scenario. Experimental details and data the model simulates can be found here http://ecosystems.mbl.edu/MEP-FoodWeb/Experiments/Exp1/index.html

---

Notebook setup

Set the working directory to where this notebook is saved

```
In[◦]:= SetDirectory[NotebookDirectory[]]
Out[◦]=
C:\Users\jvallino\OneDrive - Marine Biological
  Laboratory\Documents\AshelyMegSESpaper\Supplemental\Model
```

Turn off plot highlighting as it slows the front end down when plots have a lot of elements

```
In[◦]:= Charting`$InteractiveHighlighting = False;
SetOptions[Plot, PlotHighlighting → None];
SetOptions[ListPlot, PlotHighlighting → None];
```

Plot sometimes needs more recursion than the default, so set it higher here

```
In[◦]:= $RecursionLimit = 2000;
```

Experimental On-Line Data

Dissolved oxygen, DO (μM), and partial pressure of oxygen in the chemostat gas headspace (%) from MC1 and MC2 during the course of the experiment.

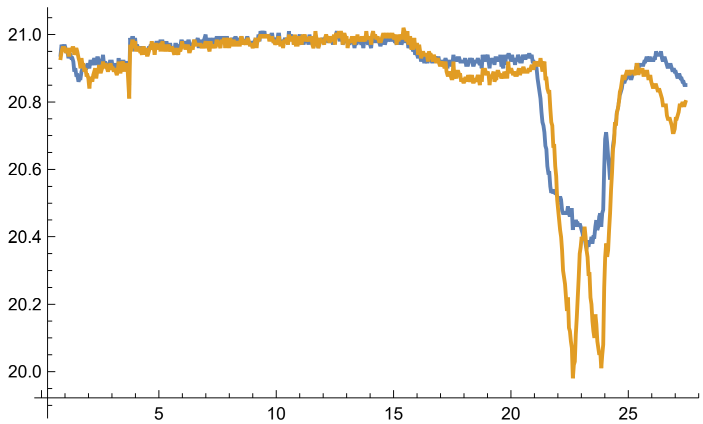
Import Exp1.csv that has the O2, CO2, DO and pH for the two chemostats. From Exp1.csv, just use the O2 data, as the DO can be obtained from the Exp1_DOpH.csv file, which is of higher resolution because it did not rely on an A2D converted. In both cases drop the header from the csv data.

```
In[◦]:= o2Data = {Drop[Import["../Data/Exp1.csv"], 1][[All, {1, 2}]],
    Drop[Import["../Data/Exp1.csv"], 1][[All, {6, 7}]]};
```

In[ ]:=
```
Dimensions[o2Data]
```
Out[ ]=
```
{2, 639, 2}
```

Plot the pO2 data

In[ ]:=
```
ListLinePlot[o2Data, PlotRange → All]
```
Out[ ]=



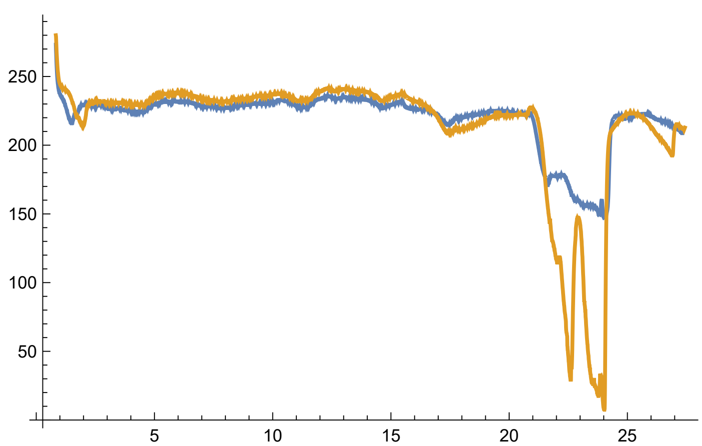
In[ ]:=
```
doData = {Drop[Import["../Data/Exp1_DOpH.csv"], 1]⟦All, {3, 4}⟧,
   Drop[Import["../Data/Exp1_DOpH.csv"], 1]⟦All, {7, 8}⟧};
```

Now convert mg/L to uM O2

In[ ]:=
```
doData = {doData⟦1⟧ /. {x_, y_} → {x, 1000 y / 32}, doData⟦2⟧ /. {x_, y_} → {x, 1000 y / 32}};
```

Plot the DO data

In[ ]:=
```
ListLinePlot[doData, PlotRange → All]
```
Out[ ]=



Dissolved Oxygen Data for k_L estimation

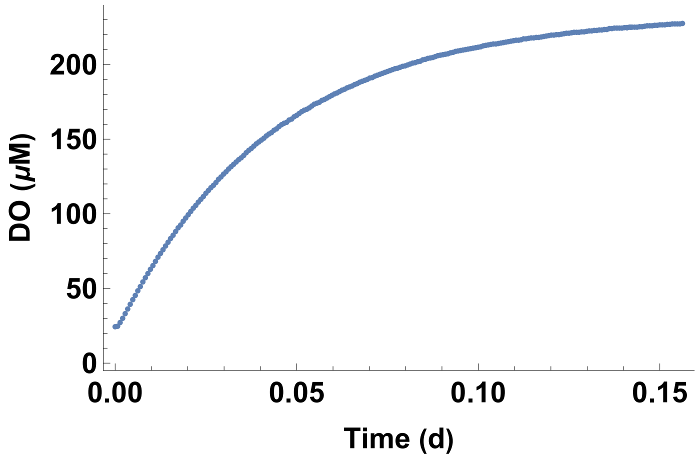
This data is used to determine the liquid-side oxygen mass transfer velocity, k_L (see below). DO data was collected following sparging of the chemostat with N_2 gas. After sparging with N2, the gas flow was changed to air, and the chemostat headspace was quickly equilibrated with air using a small fan for a short time so that the gas overlying the water would be at atmospheric pressures (i.e., fixed boundary condition). Import the DO data:

```
In[]:= o2DataN2 = Import["../Data/DO_N2toAir.dat"];
```

Plot the DO observations.

```
In[]:= ListPlot[o2DataN2, FrameLabel → {"Time (d)", "DO (μM)"},
         Frame → {{True, False}, {True, False}}, BaseStyle → {FontSize → 15, FontWeight → Bold}]
```

Out[]=



Estimate k_L a from data for transition from N_2 to Air

Preliminary model testing (no need to evaluate this subsection)

Estimate kL based on DO observations Only

Since we quickly equilibrated the headspace with air after driving the system anoxic with N_2, the governing ODE only needs to track dissolved oxygen (DO), as the gas phase remains constant as it is in equilibrium with air. Here is the governing equation for DO.

```
In[]:= odesDO[input_List] := {
          c'[t] == (fL (cF - c[t]) + kL a (p[t] h - c[t])) / vL,
          c[0] == cF} /. input
```

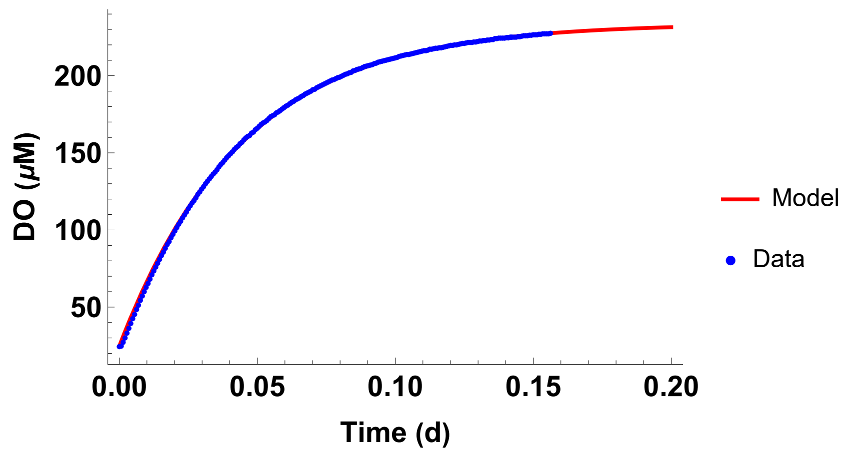
The Henry's constant as a function of temperature (K) for oxygen (see https://henrys-law.org/henry/-casrn/7782-44-7) in μM atm^{-1} is:

In[]:= hO2[tk_] := $0.00131722\, e^{\left(1700\left(\frac{1}{tk}-\frac{1}{298.15}\right)\right)}\, 10^6$

Below solves the ODE and plots the solution versus the DO observations. The only parameters adjusted here are the oxygen piston velocity, kL (m d^{-1}), which was the objective of the experiment and the oxygen concentration in the atmosphere, p[t] (atm). Note, it should not have been necessary to change p[t], as it should be 0.21 atm; however, our DO probe may not be perfectly calibrated (or perhaps the headspace still contained some of the N2 gas used for purging) and is reading a bit low; consequently, the O_2 concentration was decreased to 0.177 atm in order to get a good fit. **The value found for the piston velocity is 1.65 m d^{-1}, which is pretty close to the estimated value of 1.3**.

In[∘]:=
```
Show[Plot[
  Evaluate[c[t] /. NDSolve[odesDO[{fL → 0, cF → 24.7, kL → 1.65, a → π 0.114², h → hO2[298],
        vL → 3 / 1000, p[t] → 0.177}], {c[t]}, {t, 0, 10}][[1]]], {t, 0, 0.2},
  PlotRange → All, PlotStyle → Red, FrameLabel → {"Time (d)", "DO (μM)"},
  Frame → {{True, False}, {True, False}}, PlotLegends → {"Model"}],
 ListPlot[o2DataN2, PlotStyle → Blue, PlotLegends → {"Data"}],
 BaseStyle → {FontSize → 15, FontWeight → Bold}]
```

Out[∘]=



Governing Equations for Trait-Based Model

Adaptive Monod Kinetics

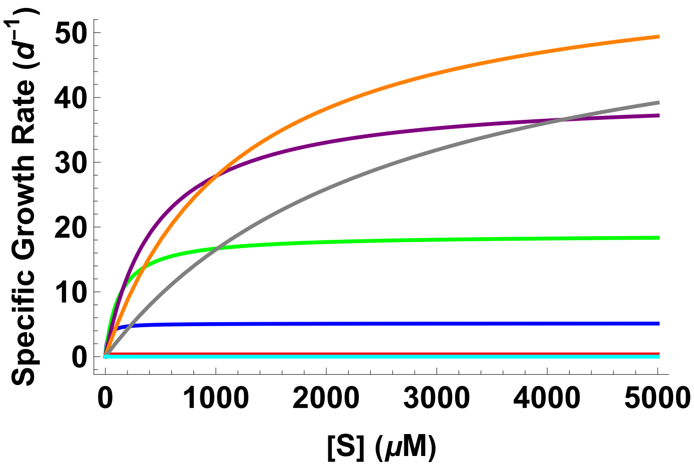
The adaptive Monod equation, written as the specific substrate uptake rate, v, not the specific growth rate; that is, $\mu = \epsilon v$. Standard parameters are used here. This function chooses along a continuum from gleaners ($\epsilon \to 0$) to opportunists ($\epsilon \to 0.7$). Note, as ϵ exceeds 0.77, uptake rate decreases due to loss of thermodynamic drive (i.e., growth reaction approaching equilibrium). Also see Vallino & Huber (2018) doi: 10.3389/fenvs.2018.001000

In[∘]:=
$$v[s\_, \epsilon\_] := vM\, \epsilon^2 \frac{s}{s + kM\, \epsilon^4} \left(1 - \epsilon^2\right)$$

The adaptive Monod equation gives the following family of curves for different values of ϵ for the standard values of vM (350 d^{-1}) and kM (5000 μM).

In[ ]:=
```
Plot[Evaluate[Table[ Tooltip[ϵ v[s, ϵ], ϵ], {ϵ, 0.1, 1, 0.15}] /. {vM → 350, kM → 5000}],
  {s, 0, 5000}, PlotStyle → {{Red, Thick}, {Blue, Thick}, {Green, Thick},
    {Purple, Thick}, {Orange, Thick}, {Gray, Thick}, {Cyan, Thick}},
  Frame → True, FrameStyle → {{Black, White}, {Black, White}},
  FrameLabel → {"[S]  (μM)", "Specific Growth Rate  (d^-1)"},
  LabelStyle → {Bold, FontFamily → "Arial", FontSize → 16}]
```

Out[ ]=



redefine the function with the standard values for the parameters set

In[ ]:=

$$v[s\_, \epsilon\_] := 350\, \epsilon^2 \frac{s}{s + 5000\, \epsilon^4} \left(1 - \epsilon^2\right)$$

Saturated Oxygen in saline water

This function, satO2, returns the oxygen concentration (in μM) for a given temperature (K) and salinity (PSU) in equilibrium with air in seawater . Reference Weiss1970 : (in deep - sea research, 17 : 721 - 735) . However, the modified version here, hO2, allows for different partial pressures of oxygen, and changes the temperature unit to K .

In[ ]:=
```
satO2[tK_, s_] := Module[{a1 = -173.4292, a2 = 249.6339, a3 = 143.3483, a4 = -21.8492,
    b1 = -0.033096, b2 = 0.014259, b3 = -0.0017, mlToumole = 44.62, t100},
t100 = tK/100;
mlToumole Exp[a1 + a2/t100 + a3 Log[t100] + a4 t100 + s (b1 + b2 t100 + b3 t100^2)]
]
```

In[ ]:=
```
satO2[293.15, 0]
```

Out[ ]=
```
283.405
```

This can be used for different partial pressures of oxygen (i.e., not just atmospheric). Also, if variables are passed to satO2, then the Exp term in satO2 is simplified, but this then leads to problems because of large and small number multiplication. By ensuring the arguments are numbers, Exp will not be simplified which keeps this problem form occurring.

```
In[•]:= hO2[s_?NumberQ, t_?NumberQ] := satO2[t, s] / 0.20946
```

```
In[•]:= hO2[3, 298] 0.20946
Out[•]=
253.757
```

Setup governing equations where the number of bacteria is specified and the ϵ trait is randomly selected from a specified distribution.

The function *generateODEs* generates a set of ODEs based on the length of ranSample, where the latter is a list of ϵ_i values chosen from an appropriate probability distribution (that is, ranSample is given). The initial conditions for x[i][0] is set to 1/n, where n is the number of ASVs used and x[i] biomass of ASV i and has the units of μM C.

```
In[•]:= generateODEs[params_List, ranSample_List] :=
  Flatten[{Table[{x[i]'[t] == 𝒟 (0 - x[i][t]) + ϵ[i] × v[s[t], ϵ[i]] × x[i][t]},
       {i, 1, Length[ranSample]}],
      s'[t] == 𝒟 (s0 - s[t]) - Sum[v[s[t], ϵ[i]] × x[i][t], {i, 1, Length[ranSample]}],
      Table[x[i][0] == 1 / Length[ranSample], {i, 1, Length[ranSample]}],
      s[0] == 0,
      o2'[t] == 𝒟 (o2f - o2[t]) + kLO2 a (pO2[t] × hO2[sal, tK] - o2[t]) / vL -
        Sum[(1 - ϵ[i]) v[s[t], ϵ[i]] × x[i][t], {i, 1, Length[ranSample]}],
      pO2'[t] == (fG (pO2f - pO2[t]) + kLO2 a rGas tK (o2[t] - pO2[t] × hO2[sal, tK])) / vG,
      o2[0] == o2f,
      pO2[0] == pO2f} /. Table[ϵ[i] → ranSample⟦i⟧, {i, Length[ranSample]}] /. params]
```

Example, generate a set of ODEs for 3 ASVs with ϵ selected from a flat distribution where ϵ can vary between 0 and 0.7

In[∘]:=
```
generateODEs[{}, RandomReal[{0, .7}, 3]] // TableForm
```

Out[∘]//TableForm=

$$x[1]'[t] == -\mathcal{D}\, x[1][t] + \frac{7.12915\, s[t] \times x[1][t]}{31.0297 + s[t]}$$

$$x[2]'[t] == -\mathcal{D}\, x[2][t] + \frac{49.3075\, s[t] \times x[2][t]}{674.937 + s[t]}$$

$$x[3]'[t] == -\mathcal{D}\, x[3][t] + \frac{12.46\, s[t] \times x[3][t]}{69.1889 + s[t]}$$

$$s'[t] == \mathcal{D}\, (s0 - s[t]) - \frac{25.4001\, s[t] \times x[1][t]}{31.0297 + s[t]} - \frac{81.3466\, s[t] \times x[2][t]}{674.937 + s[t]} - \frac{36.3287\, s[t] \times x[3][t]}{69.1889 + s[t]}$$

$$x[1][0] == \frac{1}{3}$$

$$x[2][0] == \frac{1}{3}$$

$$x[3][0] == \frac{1}{3}$$

$$s[0] == 0$$

$$o2'[t] == \mathcal{D}\, (o2f - o2[t]) + \frac{a\, kLO2\, (-o2[t] + hO2[sal, tK] \times pO2[t])}{vL} - \frac{18.271\, s[t] \times x[1][t]}{31.0297 + s[t]} - \frac{32.0391\, s[t] \times x[2][t]}{674.937 + s[t]} - \frac{23.8688\, s}{69.18}$$

$$pO2'[t] == \frac{fG\, (pO2f - pO2[t]) + a\, kLO2\, rGas\, tK\, (o2[t] - hO2[sal, tK] \times pO2[t])}{vG}$$

$$o2[0] == o2f$$

$$pO2[0] == pO2f$$

Define a function that numerically solves the generated set of ODEs over time from t0 to tf. Note, experimental time is based on 00:00 30-Oct-2018. The simulation “begins” when water was collected and placed into the collection container at about 17:00, or at t0 of 0.71 d.

In[∘]:=
```
solveODEs[t0_, tf_, params_List, ranSample_List] :=
 NDSolve[Evaluate[generateODEs[params, ranSample]], Evaluate[
    Flatten[{Table[x[i], {i, 1, Length[ranSample]}], s, o2, pO2}]], {t, t0, tf}][[1]]
```

Define function to capture the shifts in dilution rate to match Exp1

First define a smooth and continuous step function, where the steepness of the function is given by σ. As σ approaches ∞, the step function approaches a perfect step, which produces discontinuities at the steps that we wish to avoid for numerical reasons. A little trial and error was used to figure out what a reasonable value of σ should be.

In[∘]:=

$$\text{stepFcn}[t\_, tm\_, \sigma\_] := \frac{1}{1 + \text{Exp}[-\sigma\, (t - tm)]}$$

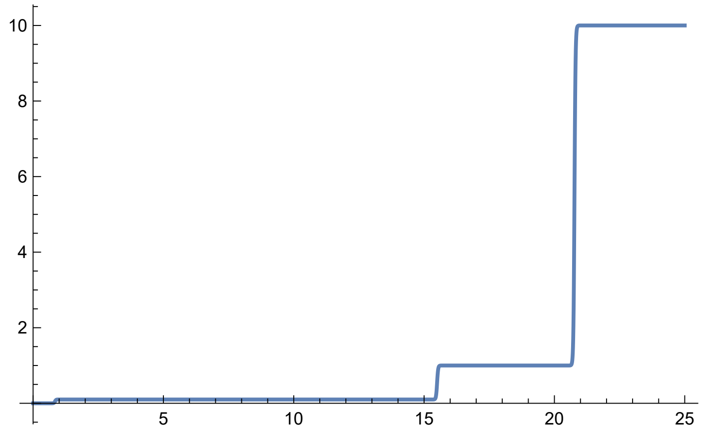
Then add these together to get the three dilution rates at the experimental step times of 0.84, 15.5 and 20.77 days.

In[∘]:=

$$\text{expDil}[t\_, \sigma\_] := \text{stepFcn}[t, 0.84, \sigma] / 10 + \frac{9}{10}\, \text{stepFcn}[t, 15.5, \sigma] + 9\, \text{stepFcn}[t, 20.77, \sigma]$$

Here' s what that function looks like over 25 days.

In[ ]:=
```
Plot[expDil[t, 50], {t, 0, 25}, PlotRange → All]
```
Out[ ]=



So a value of 50 for σ works well.

Uniform Distribution Solution for MS

Look at 500 ASVs with ϵ_i chosen from a uniform distribution. Note, each time this notebook is run, a new distribution will be generated. If you want to use the same distribution, make sure to save it (the one used for the manuscript is given below).

Parameters use for simulations

Notes from Exp 1 state: "Air feed rate to MC1 and MC2 reduced from 20 sccm to 10 sccm (0 °C, 1 atm) (15 : 30 2 - Nov - 2018, t = 3.65 d)", but just run the gas feed rate constant at 10 sccm for the simulations as the 20 sccm was only run for a short time.
Also, the feed O2 was measured at 20.97 for the first 3.73 days, then either a new air tank was installed or the old one remeasured (probably the former), but that increased the O2 in the feed gas to 21.06, so just use the later for the whole simulation.

In[ ]:=
```
params = {𝒟 → expDil[t, 50] (*d^-1*),
   s0 → 600, (*mmol m^-3*)
   pO2f → .2106, (*atm*)
   sal → 3, (*PSU*)
   tK → 298.15, (*K*)
   o2f → hO2[3, 298.15] .21, (*mmol m^-3*)
   kLO2 → 1.65, (*m d^-1*)
   a → π 0.114^2, (*m^2*)
   rGas → 0.082057338 × 10^-6, (*atm m^3 mmol^-1 K^-1*)
   vL → 3 / 1000, (*m^3*)
   vG → 4.74 / 1000, (*m^3*)
   fG → 15.72 / 1000 (*m^3 d^-1*)
  };
```

```
In[◦]:= uniform1 = RandomReal[{0, .7}, 500];
```

This is the uniform distribution used in the manuscript:

```
In[◦]:= uniform1 = {0.2065482419257525`, 0.4172611296271571`, 0.35183185854095655`,
    0.4366451828429523`, 0.3560395316384306`, 0.42449820124996496`, 0.6809594877225937`,
    0.3884279434152404 5`, 0.5847126988356506`, 0.12897170837409266`, 0.6597463035814124`,
    0.4014891766117026`, 0.23870072093247252`, 0.691996463899059`, 0.014317448635199903`,
    0.652940269807988`, 0.2984156979074397`, 0.4664411076580335`, 0.3921983780638971`,
    0.6771372386680696`, 0.17839566115559657`, 0.35206258907344345`, 0.5171330447617475`,
    0.41650325004106237`, 0.29624707718079657`, 0.45497483718317033`, 0.392919401893258`,
    0.3732743676998975`, 0.08501767658279824`, 0.598904272697989`, 0.2320248755931488 8`,
    0.21349869826588597`, 0.667393800437001`, 0.3067727707769514`, 0.6166236165635428`,
    0.6314711452881445`, 0.3269875087285028`, 0.05652320530437438`, 0.2386818464093351`,
    0.23618223486855672`, 0.42154032217211923`, 0.5062047536523835`, 0.23721474795391917`,
    0.13196347011561338`, 0.22088640475114218`, 0.699243864658845`, 0.3116951761916804`,
    0.431017207214462`, 0.3002163870388015`, 0.250338211615442`, 0.29806288030076`,
    0.32310060134129626`, 0.33340928184110874`, 0.44892330441803985`, 0.5570765748832738`,
    0.6653657004231293`, 0.5762303034727962`, 0.4839664379219819`, 0.32493975268700726`,
    0.5213334157672369`, 0.5152970522232245`, 0.6598936468281722`, 0.538656435600180 2`,
    0.5232076771678291`, 0.544446484685907`, 0.27026510317930574`, 0.1833846234772416`,
    0.5736104296623166`, 0.6006094731759439`, 0.2672580718518641`, 0.04709393805897244`,
    0.05941964602570604`, 0.5622078444826373`, 0.2274815658721403`, 0.5007093025156355`,
    0.25133611580942705`, 0.6105546481148267`, 0.1979532672525618`, 0.05000807220941472`,
    0.552066823233587`, 0.2590003935727958`, 0.237457443753463`, 0.016898439574316693`,
    0.5721126107501573`, 0.06465811590082526`, 0.6363583574943912`, 0.19032981350339662`,
    0.559154482555692`, 0.26915324214085745`, 0.6896679493978137`, 0.632896679704996`,
    0.5150649385804618`, 0.43121230981447023`, 0.4059060326582662`, 0.6318075104634733`,
    0.2651502889830646`, 0.41725198184995405`, 0.6893854894405991`, 0.041624697186729565`,
    0.019746996267215988`, 0.14415802609371964`, 0.10793432544039216`, 0.362822618693736`,
    0.05922100283814591`, 0.3421355433833069`, 0.07993191200692384`, 0.6226334616255764`,
    0.520752176474091 2`, 0.5929204690868779`, 0.08851949859515662`, 0.024743815530322788`,
    0.4215260923680446`, 0.692383172946867`, 0.2844521825106663`, 0.39357078535235956`,
    0.04313401864321609`, 0.04664460703269424`, 0.37557069136499965`, 0.5191583634036083`,
    0.13789813647016524`, 0.26735140531088375`, 0.00789442336616597`,
    0.006493044553240734`, 0.11834930490153739`, 0.4754075910671882`,
    0.52006360986820 3`, 0.6758308528502595`, 0.11776190213599891`, 0.40489996300538067`,
    0.5151682312491013`, 0.5265715848238735`, 0.42585993926787546`, 0.6595932369253765`,
    0.5001001019038422`, 0.10277948543833249`, 0.19363619115146724`, 0.625862692797022 3`,
    0.10015463157328586`, 0.44941615196498774`, 0.6639580030649792`, 0.24589710316420454`,
    0.296662824322695`, 0.3222794353065439`, 0.632920585713636`, 0.4710608030494874`,
    0.3073407065991345`, 0.5361889598429055`, 0.3154393501050501`, 0.4450246792831589`,
    0.22390945967960085`, 0.22850025250821604`, 0.2732681802864583`, 0.06829244841718873`,
    0.3031393426845157`, 0.05019573461237692`, 0.03554777864214709`, 0.3122836938365021`,
    0.003771339155015485`, 0.3369145107495395`, 0.4552134222451223`, 0.11775377493499473`,
    0.5175914026439326`, 0.5237186397082338`, 0.05420197407621552`, 0.5688008019027067`,
    0.4996714669896485`, 0.4564187989826092`, 0.27038034952758283`, 0.01442339627684297`,
```

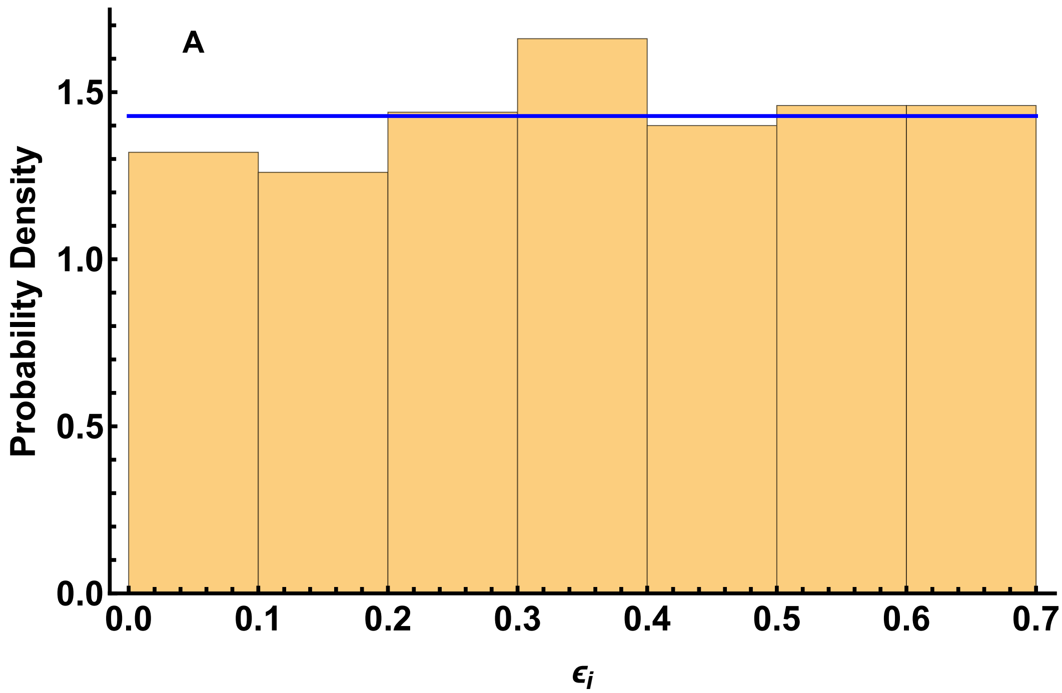
```
0.41882803811838465`, 0.4137107641086115`, 0.6087810281810224`, 0.10070729583572058`,
0.27684253948583837`, 0.3756644167406118`, 0.009527840235670992`, 0.3731076200296297`,
0.3225264273249502`, 0.43326904786272125`, 0.014147089358764187`,
0.13270432978998092`, 0.0655469802774602`, 0.28043454840314586`, 0.3080004140569841`,
0.03656213543141551`, 0.65832884512829`, 0.26509051825202956`, 0.3300361486394825`,
0.4502018712692031`, 0.04773952093543998`, 0.262175760481590 1`, 0.688980247647696 5`,
0.629855884807345`, 0.1398463154268665 4`, 0.01757923964267427`, 0.610493740258150 5`,
0.09934303877249051`, 0.169071230845382`, 0.45717267734061284`, 0.6504508010188645`,
0.2522272379556365`, 0.5022273971169731`, 0.3765684623791232`, 0.514910328496084 8`,
0.4247873460092446`, 0.04822777488911689`, 0.22102623881592098`, 0.14559108428134138`,
0.4055362957450488`, 0.36902889555685725`, 0.43307663966311116`, 0.5696344426145243`,
0.3080624291319367`, 0.2701104715656921`, 0.6117185734001893`, 0.0093902602305459`,
0.06278858891336392`, 0.5838417162319471`, 0.0290078055168333 5`, 0.3471502734734766`,
0.1893819695608724`, 0.03422406201783146`, 0.6694259146116552`, 0.14061068774787966`,
0.4968517825663361`, 0.5890820727151236`, 0.42706637558872274`, 0.10359572387306759`,
0.6597840002623023`, 0.4709404884580477`, 0.1397027612185489 7`, 0.1967209163978387 3`,
0.5599146198962612`, 0.07630275549233345`, 0.10684108574568352`, 0.517352115183371 5`,
0.6103963511901886`, 0.4135177575553539`, 0.20769641660087623`, 0.3789065398505702`,
0.46778882513647213`, 0.042822853000587524`, 0.07409679433592253`,
0.32427759401300893`, 0.2540107391691374`, 0.27928681476832284`, 0.292269082680959 5`,
0.46492661047249584`, 0.6814345942230915`, 0.14685698824737725`, 0.36716847668310026`,
0.23877521091896214`, 0.5399763882201027`, 0.4050280510382773`, 0.30254017129046673`,
0.1444924602273546 2`, 0.63453056953147`, 0.17384426833101085`, 0.4287762953332446`,
0.3755860451070956`, 0.28291452467275524`, 0.5899259142489273`, 0.2479852512740359 2`,
0.34024573256340473`, 0.11382921457231265`, 0.1499386193208816`, 0.4485868723048414`,
0.4555787169377248`, 0.13305545599992508`, 0.3296476973223019`, 0.16217458133313722`,
0.3649241587876635`, 0.24665428324586935`, 0.3251145789644134`, 0.381160493178047 7`,
0.26777481529216807`, 0.3859233829913593`, 0.39843041442399274`, 0.21367901211309348`,
0.3368844187090161`, 0.5356207438138778`, 0.19991123084250617`, 0.08527272409912157`,
0.370173623904956`, 0.12751757830050825`, 0.40558520071674353`, 0.5723067926812806`,
0.6695618615760031`, 0.19040052824921672`, 0.49440562752954387`, 0.6809292090875803`,
0.4994083356666488`, 0.4032815134252552`, 0.3309663577180897`, 0.341373161045812 4`,
0.6351540088058383`, 0.605694035552284`, 0.11327180124522707`, 0.5469156660817527`,
0.027047039765383474`, 0.2638053244903391`, 0.22977235607880309`,
0.5954757029191076`, 0.5094477977916994`, 0.4387785400065811`, 0.2179903592467023`,
0.0368866574851715`, 0.5310169159067182`, 0.6707172118145324`, 0.21345597592247656`,
0.41863771171505215`, 0.3815895732995318`, 0.4248102247655767`, 0.1941248104398629 6`,
0.0808361503612387`, 0.5004663989257818`, 0.49574593818404367`, 0.3523246024814899`,
0.5436841577233462`, 0.1642127892550807 6`, 0.537256442191552`, 0.22213798017575914`,
0.29791918320031807`, 0.2598702747058729`, 0.34861726597628717`, 0.6214245108224481`,
0.41264465422080265`, 0.13274394601523398`, 0.46639540530449985`,
0.2011993651726387`, 0.29339618979577287`, 0.17565861330516108`, 0.6488071870992942`,
0.48255110726410777`, 0.3578696701881823`, 0.48357732694294286`, 0.5097080668089093`,
0.29783127068269544`, 0.5421039541466608`, 0.5067617238851734`, 0.578102112951764 5`,
0.105674828699027`, 0.2418580921410678`, 0.6359859147360984`, 0.21614322002293906`,
0.6514983971431387`, 0.39478548904559463`, 0.3844683155403963`, 0.30969947260076713`,
0.32649147360537367`, 0.4947863773287675`, 0.010521962139906194`, 0.6494553238693299`,
```

```
0.4294003996078899`, 0.1929944572220853`, 0.15469519642795904`, 0.039917320657550825`,
0.526098811395663`, 0.23579130471925858`, 0.3371519879451188`, 0.6290213594015168`,
0.3281784693720078`, 0.5858753959171139`, 0.0205872734250806`, 0.3221316850911165`,
0.3464977067174497`, 0.613194480452915`, 0.2821644470030129`, 0.0676624457125063`,
0.5409083213601404`, 0.3588434777950602`, 0.08244302955417204`, 0.351451971628621`,
0.0714344987434734`, 0.4979492668409198`, 0.1162155963026190`, 0.21473308996235463`,
0.6634217560065465`, 0.6489452642437117`, 0.019836432709910534`, 0.136406482283564`,
0.37530637999389316`, 0.6774720266396821`, 0.5255601276399882`, 0.447229231764219`,
0.09499037882820016`, 0.5422013043958789`, 0.3953419482650642`, 0.216632251457404`,
0.3281954751520617`, 0.6623971077752824`, 0.18101907262714523`, 0.624155202345908`,
0.016511817410057916`, 0.5391156020265957`, 0.6662903734546595`, 0.6561237479173125`,
0.3068165327549526`, 0.5552057467067735`, 0.0898565845336392`, 0.41748334971300083`,
0.6295216478605794`, 0.45415946628305504`, 0.3254019880735495`, 0.627535167697875`,
0.4008693425916874`, 0.6280778318971867`, 0.1305289119835934`, 0.3577954974155293`,
0.03815430590973501`, 0.2717405099220672`, 0.6634969529024755`, 0.3615148752261521`,
0.6842371281532844`, 0.536276579173385`, 0.6833934401359436`, 0.6905416089972782`,
0.06317887158481994`, 0.6433864303628316`, 0.2082159859106154`, 0.6738000597945417`,
0.6555720881238061`, 0.5554346153246026`, 0.6176525267784809`, 0.29729638414525184`,
0.37108743414729295`, 0.40091978301839504`, 0.0857563355669333`, 0.39704447609159343`,
0.16543233487499953`, 0.09218986791140704`, 0.5698681076233858`, 0.695676793328194`,
0.41950240939788275`, 0.2952400204149369`, 0.08067700407842804`, 0.25445234120507754`,
0.36144733726060085`, 0.3957455661682834`, 0.21194023467288614`, 0.5895714586824707`,
0.105130076108367`, 0.4130576892931277`, 0.15773724076328866`, 0.0348287500414356`,
0.13923808962973494`, 0.5763757494959347`, 0.3081722320227287`, 0.10879844961608343`,
0.063858091223713`, 0.22018399730578853`, 0.3330353253304428`, 0.345312610169551`,
0.1919241414515721`, 0.2412064425045216`, 0.5305119840694461`, 0.5279098494207628`,
0.09222150904942761`, 0.6242624195065067`, 0.5902858683907959`, 0.11540739636530517`,
0.5043633698671388`, 0.23482345851686148`, 0.2223792664108203`, 0.3334528599286597`,
0.061433857369995404`, 0.0747635812348807`, 0.4563892275506414`, 0.41160558793918733`,
0.542439113807941`, 0.4796261046982988`, 0.4077866867246682`, 0.2591494587638222`,
0.10554280829901075`, 0.33913813127207737`, 0.1270218904668956`, 0.1916306449708426`,
0.671447992469268`, 0.37673456646458847`, 0.15856388976844382`, 0.5261913306831196`,
0.52860682110935`, 0.582104699603283`, 0.13973327283704529`, 0.5873707991867072`,
0.3189383131140051`, 0.16079910882618154`, 0.099264140210233`, 0.3617999954507791`,
0.14617602634649052`, 0.01762801396450997`, 0.6849049180681488`, 0.6698191461374112`,
0.6668137282252296`, 0.44924603983737343`, 0.006476528919752256`,
0.17729932817205374`, 0.49145878512771213`, 0.5208475936058174`, 0.334942734026195`};
```

Plot the realized probability density function based on uniform1 sample as well as the theoretical one.

In[ ]:=
```
uniformDistributionPlot = Show[{Histogram[uniform1, Automatic, "ProbabilityDensity",
    Frame → {{True, False}, {True, False}}, FrameLabel → {"ϵi", "Probability Density"},
    LabelStyle → {Bold, FontFamily → "Arial", FontSize → 18, FontColor → Black},
    FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
    Epilog → {Text[Style["A", Bold, FontFamily → "Arial", FontSize → 16], {0.05, 1.65}]},
    ImageSize → 550], Plot[PDF[UniformDistribution[{0, .7}], x],
    {x, 0, .7}, PlotStyle → {Blue, Thick}]}]
```

Out[ ]=



In[ ]:=
```
Export["uniformDistributionPlot.svg", uniformDistributionPlot]
```

Out[ ]=

uniformDistributionPlot.svg

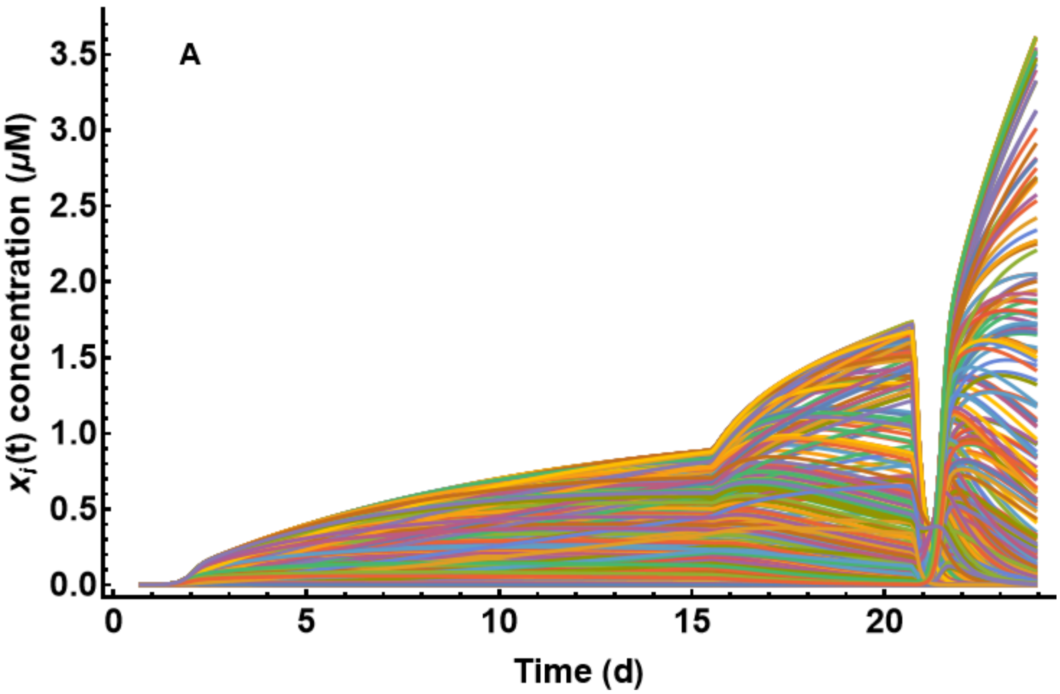
Solve the 500 ASV ode equations for the uniform distribution

In[ ]:=
```
soln500 = solveODEs[0.71, 24, params, uniform1];
```

500 ASVs for the uniform distribution case in absolute concentrations (μM C).

In[ ]:=
```
xiUniformPlot =
  ListLinePlot[Evaluate[Table[{t, x[i][t]}, {i, 1, 500}, {t, 0.71, 24, 0.1}] /. soln500],
   PlotRange → All, Frame → {{True, False}, {True, False}},
   FrameLabel → {"Time (d)", "xi(t) concentration (μM)"}, LabelStyle →
    {Bold, FontFamily → "Arial", FontSize → 18, FontColor → Black}, PlotStyle → Thick,
   FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
   Epilog → {Text[Style["A", Bold, FontFamily → "Arial", FontSize → 16], {2, 3.5}]},
   ImageSize → 550]
```

Out[ ]=



Save this plot for the manuscript

In[ ]:=
```
Export["xiUniformPlot.svg", xiUniformPlot]
```

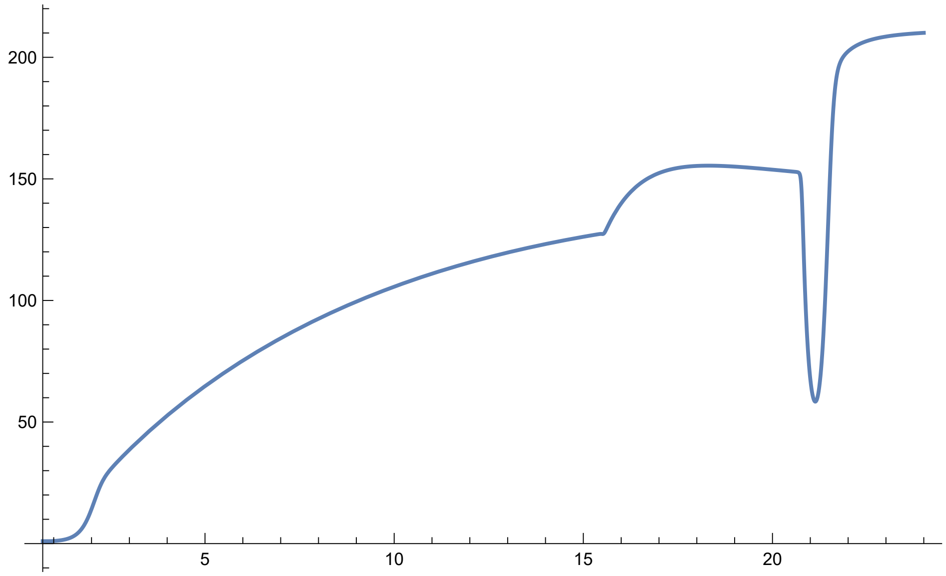
Out[ ]=
```
xiUniformPlot.svg
```

The Manipulate function below allows you to slide the slider below to look at profile for individual ASVs (μM C). However, this can cause issues with the front end, so it’s not evaluated below

In[ ]:=
```
Manipulate[Plot[Evaluate[x[i][t] /. soln500],
  {t, 0.71, 24}, PlotLabel → "ASV: " <> ToString[i]], {i, 1, 500, 1}]
```

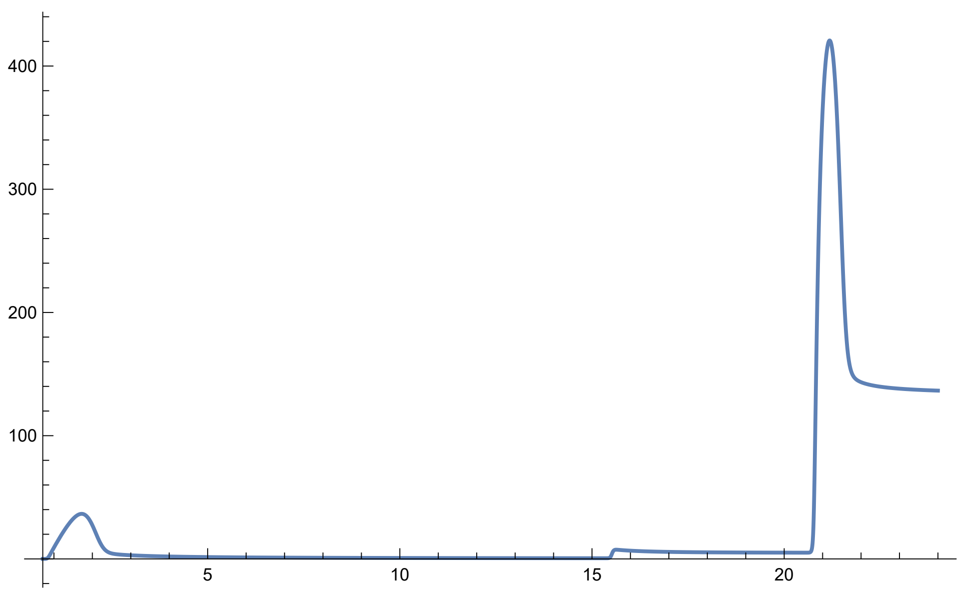
Plot the sum of all ASVs (μM C).

In[ ]:=
```
Plot[Evaluate[Sum[x[i][t], {i, 1, 500}] /. soln500], {t, 0.71, 24}, PlotRange → All]
```
Out[ ]=



Plot the substrate concentration (μM C).

In[ ]:=
```
Plot[Evaluate[s[t] /. soln500], {t, 0.71, 24}, PlotRange → All]
```
Out[ ]=

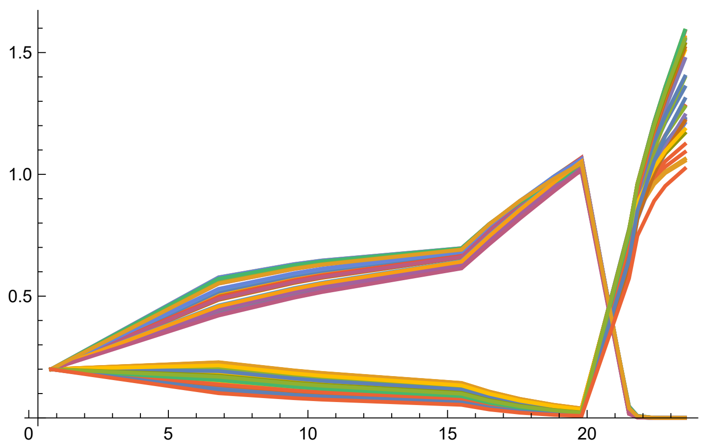


Look at diversity using relative abundance, and sample at the same times points as the MC's were. Note, because of the high diversity, the min ASV relative abundance must be set rather low, as no ASV reaches high rel abundance. Here, using 1% (that is, include any ASV that at any times exceeds 1% in abundance)

In[ ]:=
```
times = {0.8, 6.8, 9.5, 10.5, 15.5, 16.5, 17.6, 18.8, 19.8, 21.5, 21.8, 22.4, 22.8, 23.5};
uniformASVs =
  Prepend[Evaluate[Table[Table[x[i][t], {i, 1, 500}] /. soln500 /. t → times[[j]],
      {j, 1, Length[times]}]], Evaluate[Table["ASV_" <> ToString[i], {i, 500}]]]ᵀ;
uniformASVsRel = Prepend[Table[100 uniformASVs[[1 ;;, i]] / Total[uniformASVs[[1 ;;, i]]],
     {i, 2, Dimensions[uniformASVs][[2]]}], uniformASVs[[1 ;;, 1]]]ᵀ;
```

In[ ]:=
```
asvUniformRelGT1 = Select[uniformASVsRel, (Max[#[[2 ;;]]] > 1) &];
ListLinePlot[Table[{times, asvUniformRelGT1[[i, 2 ;;]]}ᵀ,
  {i, 1, Length[asvUniformRelGT1]}], PlotRange → All]
```

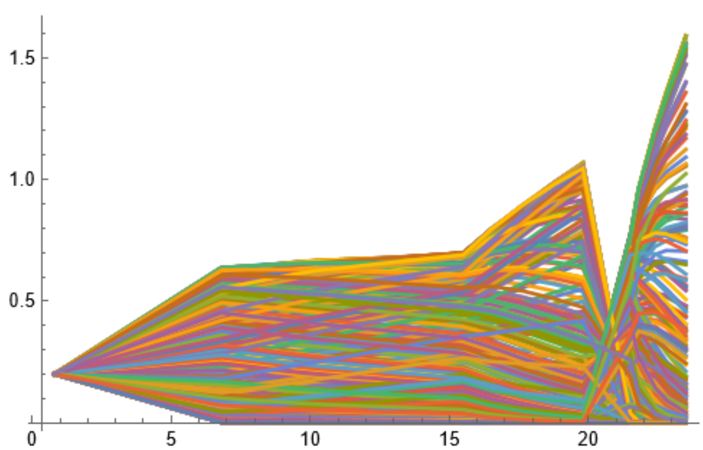
Out[ ]=



Look at including all ASVs

In[ ]:=
```
asvUniformRelGT0 = Select[uniformASVsRel, (Max[#[[2 ;;]]] > 0) &];
ListLinePlot[Table[{times, asvUniformRelGT0[[i, 2 ;;]]}ᵀ,
  {i, 1, Length[asvUniformRelGT0]}], PlotRange → All]
```

Out[ ]=

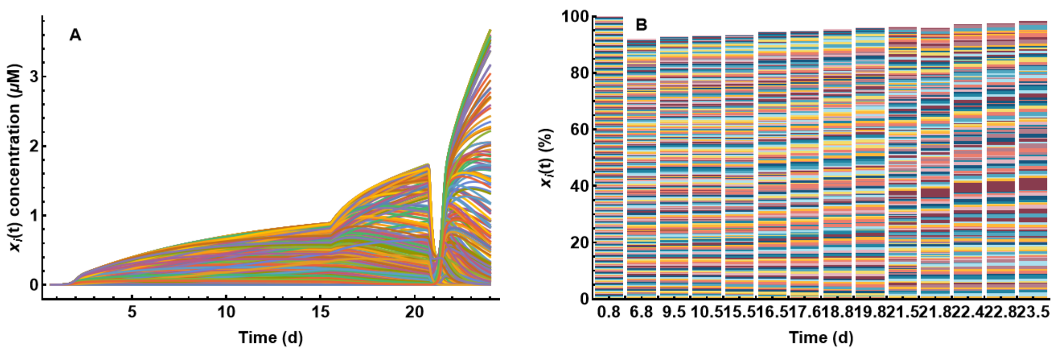


Plot absolute ASV concentrations and relative abundance at the experimental sample times as a bar plot at the 0.1% relative abundance or greater.

In[•]:=
```
asvMinAbunRel = 0.1;
asvUniformMin = Chop[N[uniformASVsRel], asvMinAbunRel];
(* this sets any ASV that is < asvMinAbunRel to 0 *)
```

In[•]:=
```
GraphicsRow[{Plot[Evaluate[Table[x[i][t], {i, 1, 500}] /. soln500],
   {t, 0.71, 24}, PlotRange → All, Frame → {{True, False}, {True, False}},
   FrameLabel → {"Time (d)", "xᵢ(t) concentration (μM)"},
   LabelStyle → {Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black}, FrameStyle →
    {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}}, PlotStyle → Thick,
   Epilog → {Text[Style["A", Bold, FontFamily → "Arial", FontSize → 16], {2, 3.6}]}],
  BarChart[asvUniformMin[[1 ;;, 2 ;;]]ᵀ, ChartLayout → "Stacked",
   ChartLabels → {{0.8, 6.8, 9.5, 10.5, 15.5, 16.5, 17.6, 18.8, 19.8, 21.5, 21.8,
      22.4, 22.8, 23.5}, Table["", {14}]}, PlotRange → {{0.5, 14.5}, {0, 100}},
   LabelStyle → {Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black},
   FrameLabel → {"Time (d)", "xᵢ(t) (%)"}, Frame → {{True, False}, {True, False}},
   FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
   ChartStyle → 24, Epilog → {Text[
      Style["B", Bold, FontFamily → "Arial", FontSize → 16], {2, 97}]}]}, ImageSize → 1024]
```

Out[•]=



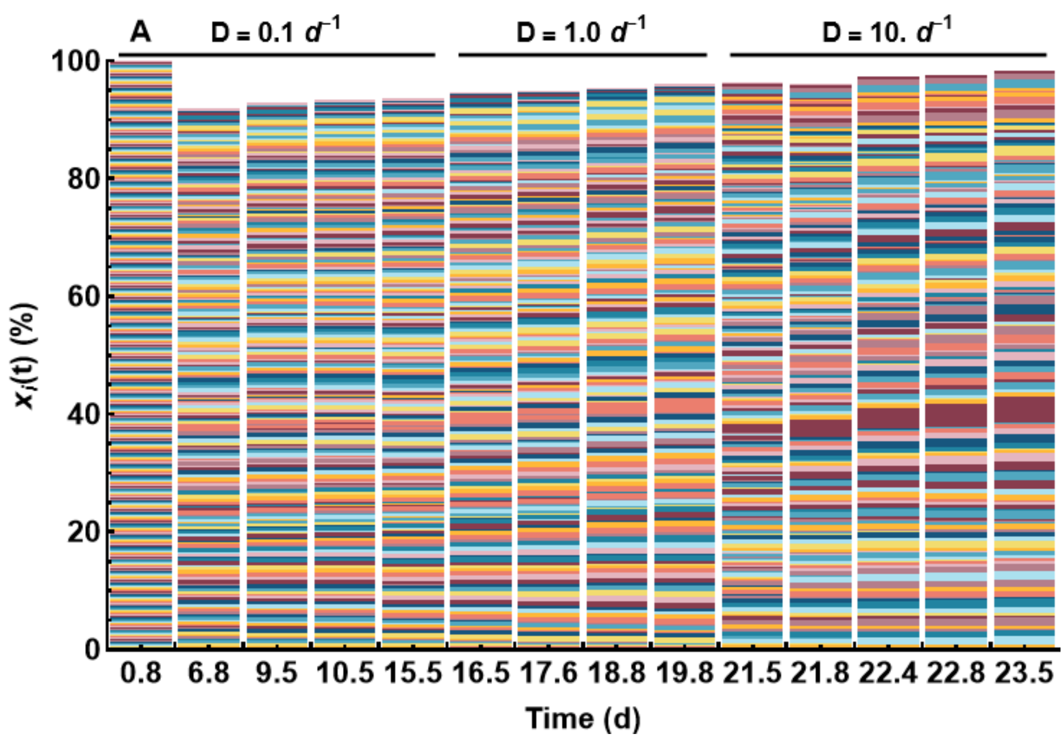
Bar Plot alone

```
ImagePadding → {{All, All}, {All, 30}}, ImageSize → 550,
Epilog → {Thick, Text[Style["D = 0.1 d⁻¹", Bold, 14], {(dBars[[1]] + dBars[[2]]) / 2, 105}],
  Line[{{dBars[[1]] - 0.3, 101}, {dBars[[2]] + 0.3, 101}}],
  Text[Style["D = 1.0 d⁻¹", Bold, 14], {(dBars[[3]] + dBars[[4]]) / 2, 105}],
  Text[Style["D = 10. d⁻¹", Bold, 14], {(dBars[[5]] + dBars[[6]]) / 2, 105}],
  Line[{{dBars[[3]] - 0.3, 101}, {dBars[[4]] + 0.3, 101}}],
  Line[{{dBars[[5]] - 0.3, 101}, {dBars[[6]] + 0.3, 101}}]}
```

In[∘]:=
```
xiBarChart[xiRel_List, labels_List, dBars_List, fig_] :=
 BarChart[xiRel, ChartLayout → "Stacked", ChartLabels → labels,
  PlotRange → {{0.5, 14.5}, {0, 100}}, LabelStyle →
   {Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black}, ChartStyle → 24,
  FrameLabel → {"Time (d)", "xi(t) (%)"}, Frame → {{True, False}, {True, False}},
  FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
  Epilog → {Text[Style[fig, Bold, FontFamily → "Arial", FontSize → 16], {1, 105}],
    Thick, Text[Style["D = 0.1 d^-1", Bold, 14], {(dBars[[1]] + dBars[[2]]) / 2, 105}],
    Line[{{dBars[[1]] - 0.3, 101}, {dBars[[2]] + 0.3, 101}}],
    Text[Style["D = 1.0 d^-1", Bold, 14], {(dBars[[3]] + dBars[[4]]) / 2, 105}],
    Text[Style["D = 10. d^-1", Bold, 14], {(dBars[[5]] + dBars[[6]]) / 2, 105}],
    Line[{{dBars[[3]] - 0.3, 101}, {dBars[[4]] + 0.3, 101}}],
    Line[{{dBars[[5]] - 0.3, 101}, {dBars[[6]] + 0.3, 101}}]},
  ImagePadding → {{All, All}, {All, 30}}, ImageSize → 550]

xiBarUniformPlot = xiBarChart[asvUniformMin[[1 ;;, 2 ;;]]^T,
  {{0.8, 6.8, 9.5, 10.5, 15.5, 16.5, 17.6, 18.8, 19.8, 21.5, 21.8, 22.4, 22.8, 23.5},
   Table["", {14}]}, {1, 5, 6, 9, 10, 14}, "A"]
```

Out[∘]=



Save this plot for the manuscript

In[∘]:=
```
Export["xiBarUniformPlot.svg", xiBarUniformPlot]
```

Out[∘]=
```
xiBarUniformPlot.svg
```

Export the ASVs table for the time points that correspond to all sample times

```
In[ ]:= Export["uniformASVs.csv", Prepend[uniformASVs, Flatten[{"ASVs", times}]]]
Out[ ]=
uniformASVs.csv
```

Export the relative abundances with no cutoff

```
In[ ]:= Export["uniformASVsRelative.csv", Prepend[uniformASVsRel, Flatten[{"ASVs", times}]]]
Out[ ]=
uniformASVsRelative.csv
```

Beta distribution function Solution for MS

This is basically the same as above, except ϵ is drawn from a beta probability distribution that is skewed towards oligotrophic growth characteristics.

Run with $\alpha = 1.1$, $\beta = 13.3$

```
In[ ]:= params = {𝒟 → expDil[t, 50] (*d^-1*),
    s0 → 600, (*mmol m^-3*)
    pO2f → .2106, (*atm*)
    sal → 3, (*PSU*)
    tK → 298.15, (*K*)
    o2f → hO2[3, 298.15] .21, (*mmol m^-3*)
    kLO2 → 1.65, (*m d^-1*)
    a → π 0.114^2, (*m^2*)
    rGas → 0.082057338 × 10^-6, (*atm m^3 mmol^-1 K^-1*)
    vL → 3 / 1000, (*m^3*)
    vG → 4.74 / 1000, (*m^3*)
    fG → 15.72 / 1000, (*m^3 d^-1*)
    α → 1.1,
    β → 13.3};
```

Look at 500 samples taken from the beta distribution for $\alpha = 1.1$ and $\beta = 13.3$

```
In[ ]:= betaSample = RandomVariate[BetaDistribution[α, β] /. params, 500];
```

This is the beta distribution used in the manuscript

```
In[ ]:= betaSample = {0.04871417003885394`, 0.03017318186219966`, 0.012564993680192262`,
    0.13259384803132962`, 0.021620267469453208`, 0.2344049750022628`,
    0.15149911540580235`, 0.004684454408583026`, 0.004601369086191348`,
    0.042992808450216524`, 0.025453219010936827`, 0.09417271597016663`,
    0.045696091326077035`, 0.13284625263752917`, 0.0266933889076973`, 0.2629338141682852`,
    0.059354581681200495`, 0.07931635152117568`, 0.2205508867126184`,
    0.02118909371323286`, 0.08205366876494401`, 0.010931389024451262`,
```

```
0.1496050188706156`, 0.05525010558659861`, 0.15288468109095826`,
0.015915678507397803`, 0.031122272663232253`, 0.0803226560606708`,
0.04314709139378374`, 0.14315676215797007`, 0.058359206495349786`,
0.18170250048485312`, 0.03360900445255381`, 0.02942456554931353`, 0.2087416396838662`,
0.0054045809746471685`, 0.0013462079542849493`, 0.07212716572067226`,
0.10450967555532742`, 0.08744099507163389`, 0.09493157153736054`, 0.0784467555946978`,
0.05166651168990486`, 0.00133414934878874`, 0.07578982056455534`,
0.03715395761693538 6`, 0.005309644300830824`, 0.15411445948443941`,
0.009751849308989518`, 0.07225647802771631`, 0.06936608679819813`,
0.06120727001345328`, 0.12843678844929338`, 0.042131323662609736`,
0.01672018869895678`, 0.015215508943382137`, 0.11225009904324758`,
0.04919497886917243`, 0.0188384047140116`, 0.037266167674637106`,
0.11093762041353886`, 0.12789992411848625`, 0.12052621965819525`,
0.010627490500548538`, 0.01826746581890762`, 0.023365705979300037`,
0.02368827336416486`, 0.0739020305574193`, 0.011295655622854098`,
0.05408094424198332`, 0.1490046520058161`, 0.12578634491357937`,
0.004869659200823763`, 0.26630301156497094`, 0.003849534540199007`,
0.06905215114004311`, 0.06859481953405061`, 0.028748724748098115`,
0.14847369252819873`, 0.2174523572636497`, 0.193708893081989`, 0.07714866769087032`,
0.022871419185523347`, 0.06341184944387644`, 0.12244641637062309`,
0.11624046267496087`, 0.07164536542764394`, 0.08039840001044087`,
0.052431646380330536`, 0.10668348177075203`, 0.09154832908079456`,
0.03257046394954066`, 0.09198072965736782`, 0.12734788662242694`, 0.1615990690664287`,
0.0007023097483443249`, 0.04599757673754196`, 0.1420387979824347`,
0.02609236076796676`, 0.05483319833189066`, 0.0034541546335561378`,
0.05979826807602424 4`, 0.022552148777120484`, 0.023829340082174116`,
0.03753009055321549`, 0.15590155990015542`, 0.13292412426629985`,
0.0031347163603508955`, 0.005809975013229053`, 0.05347731920469983`,
0.1195926517599274`, 0.024169207539693684`, 0.033868128972670254`,
0.0047255961787532365`, 0.009001096613049143`, 0.05456996503276114`,
0.015376698457814768`, 0.03673534349017801`, 0.0442168643374372`,
0.004980503399767064`, 0.06866586347795299`, 0.039909944942066865`,
0.024100042679365354`, 0.06122708284578606`, 0.12067197707201006`,
0.22428778681067332`, 0.10321455171062971`, 0.061810528166035536`,
0.03429125097271478`, 0.009354432062220638`, 0.13554475676696812`,
0.05585587463340101`, 0.0011726470568291143`, 0.01148917385931319`,
0.0035962648649721933`, 0.03181949730803888`, 0.2267838072655365`,
0.16117554712647378`, 0.02698462446839613`, 0.010845142280690185`,
0.023816314095584358`, 0.13052320860442276`, 0.024311528150703785`,
0.027374071069603177`, 0.002366588015184559`, 0.024849074414056854`,
0.2439961188452165`, 0.0235668780209888`, 0.021101438545775366`,
0.003727398656108021`, 0.22215983371715115`, 0.06126381635770991`,
0.03857094915832731`, 0.010191563790865634`, 0.09609645624578363`,
0.02155436426391609`, 0.0497287884915788`, 0.027378315530424253`,
0.10126542602303192`, 0.11074720582468918`, 0.059013521119175315`,
0.0800919148967481 5`, 0.026025990574873595`, 0.0516453688165229`, 0.436635733572928`,
0.09505766406257224`, 0.18151396744044626`, 0.004026497589513533`,
```

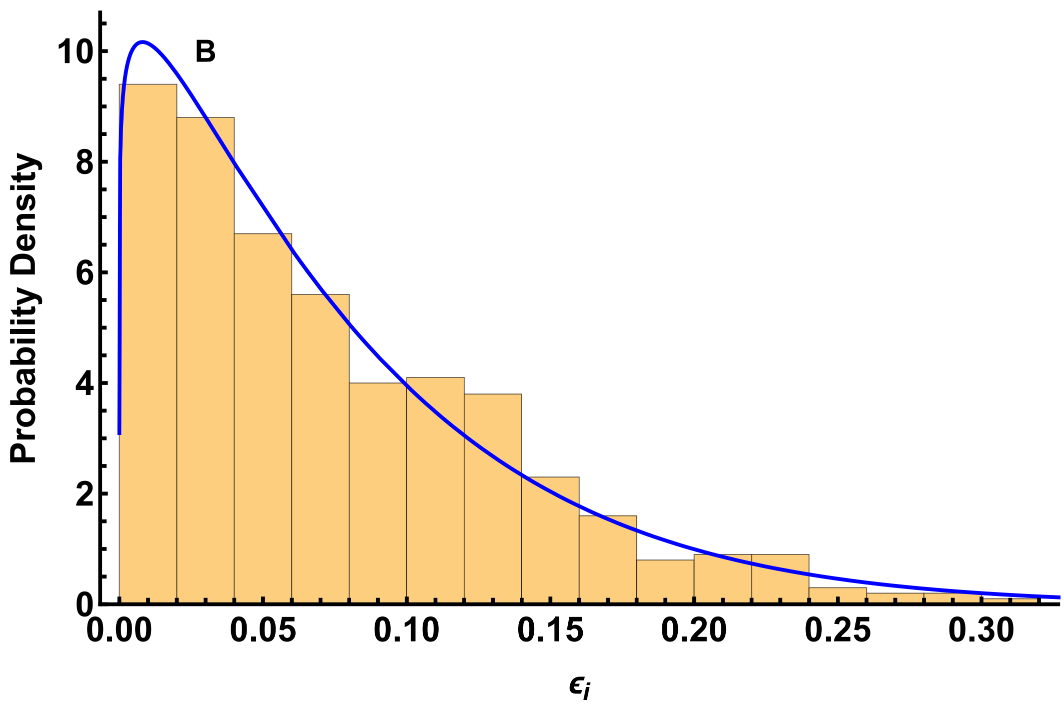
```
0.08694990562535915`, 0.010689776034261059`, 0.06093180898944036`,
0.14815681455394025`, 0.017389728855135546`, 0.15376068053083183`,
0.10547790711750345`, 0.053319059254476144`, 0.007828492425429658`,
0.11547561707596206`, 0.001962946563101294`, 0.020742429167826728`,
0.09263697203414759`, 0.11065848773597486`, 0.06350342699675426`,
0.22296694310636778`, 0.08771449197198464`, 0.04811220738693807`,
0.03711701562767455`, 0.037113529737442814`, 0.1275420670922201`,
0.10890010804702391`, 0.0010086477888882534`, 0.12210310077445455`,
0.032606956528330545`, 0.03526221608701037`, 0.17583830800943565`,
0.06714135752884476`, 0.09828443656338597`, 0.0009216775278314409`,
0.1705419780457287`, 0.03811325289180692`, 0.047644995061664995`,
0.021183078103196543`, 0.13362095999997362`, 0.01524195729847747`,
0.012765560839104026`, 0.012902106693768548`, 0.09892142768556995`,
0.03809981023599537`, 0.09142479321216293`, 0.02117021052803575`,
0.01790428904971133`, 0.023014618759056738`, 0.041208699630478025`,
0.1604234626667358`, 0.13624906440479095`, 0.2172044212009787`, 0.01472299488027583`,
0.10320904273889435`, 0.03538900540536`, 0.15383534367786517`, 0.0831765127288058`,
0.2891594570145767`, 0.012784100076480495`, 0.058137248081689574`,
0.07989725697937083`, 0.03750493565627171`, 0.030127139394336866`,
0.0692569612013841`, 0.1057124791160422`, 0.07636299363027825`, 0.02805523454207417`,
0.09803811080087738`, 0.0040968904614844814`, 0.18555557437046988`,
0.13923635889454775`, 0.054409267476525254`, 0.07170089091396022`,
0.17785360980939177`, 0.08703292497589263`, 0.11066124950446958`,
0.01849525339465957`, 0.01125351097552394`, 0.02617265456166756`,
0.028703125286146262`, 0.02435163110964077`, 0.14764131913850376`,
0.04282363862056955`, 0.016388741616495523`, 0.0024044815732956557`,
0.06621756332129385`, 0.08482442792532743`, 0.0972863022943594`,
0.06252463091101566`, 0.006323773999152156`, 0.07362815024186246`,
0.0611440977466474`, 0.07972642169305205`, 0.1447556892689672`, 0.09928994351387667`,
0.020794784747151036`, 0.050520000331074226`, 0.12141743932292878`,
0.036350985169579256`, 0.01753499671893606`, 0.09981444140966093`,
0.3040912124150068`, 0.21577960371639454`, 0.12282298401480057`,
0.005609713311279573`, 0.014979100908650154`, 0.03170198434798988`,
0.008771309668806833`, 0.12799598858285297`, 0.18253831146701313`,
0.22604220035805841`, 0.022101791111936452`, 0.17276189858838992`,
0.08861263428291366`, 0.04812472719891856`, 0.046674493469162186`,
0.021464146507729518`, 0.16044991188722343`, 0.0466983600698937`,
0.06957677647088273`, 0.09043801855017315`, 0.1644442606878382`,
0.12545534607706213`, 0.08260742429007133`, 0.08309558671739452`,
0.014564491167656008`, 0.07667963494845564`, 0.051622435206747475`,
0.12374328132302406`, 0.002498968309387258`, 0.013030313862055545`,
0.006428129906045408`, 0.05001626077167433`, 0.026443982143654638`,
0.0539879735120826`, 0.1312999823623269`, 0.11512505634597918`, 0.21686950178922165`,
0.13585031425223854`, 0.07210004355656577`, 0.04793868805193183`,
0.09283559178732992`, 0.08969893726785652`, 0.1127165179660344`, 0.10775185052769168`,
0.082934653731241`, 0.06823372191103245`, 0.06587945352658324`, 0.2328410568252529`,
0.29667855682759264`, 0.2417366751952173`, 0.02268594766727997`, 0.10536656975095389`,
```

```
0.041595829142700776`, 0.022853557801980416`, 0.14210978074465158`,
0.0832995335530726`, 0.042037241378529315`, 0.1356399267952631`, 0.1019334755670714`,
0.017516502335782242`, 0.09479937599913722`, 0.10588947742985351`,
0.027886623944396186`, 0.04829511408747073`, 0.05377671868006312`,
0.027548383900668754`, 0.10032500675730449`, 0.0025943474034798727`,
0.0002914995586228621`, 0.11763744700827708`, 0.1030882775938096`,
0.07394200450453615`, 0.10622159011543529`, 0.014848515802661616`,
0.25550171739791566`, 0.2053131811836873`, 0.0017261960196748958`,
0.06294153015556311`, 0.01920130174043932`, 0.03554558881839803`,
0.0791365044139423`, 0.026971779170970305`, 0.07574165069700076`,
0.1421689759972292`, 0.09329687854788936`, 0.05024255981023416`, 0.0490125971519095`,
0.022976297480557004`, 0.029544565269354898`, 0.15065479884821742`,
0.008209166237822623`, 0.17455831112154385`, 0.013752289171807503`,
0.07422293537549755`, 0.1639564288054622`, 0.14087312121450782`,
0.014014135664685779`, 0.0027830560397054402`, 0.16666839002916034`,
0.026505413568927255`, 0.11446821433343855`, 0.11050762219668844`,
0.00541900015390286`, 0.037520865133616896`, 0.030772458132362467`,
0.05280786810800987`, 0.03407556710718542`, 0.03084800257306437`,
0.10102930634113923`, 0.13044375778720677`, 0.030600156418795557`,
0.010512867983209667`, 0.1274287373380895`, 0.17745371321519915`,
0.11057254754925154`, 0.007516639911862173`, 0.026979097585571113`,
0.002611592186387166`, 0.0337798156974438`, 0.05597223108797968`,
0.0491196252297267`, 0.13053360462565602`, 0.12067387539058357`,
0.11878864049784978`, 0.1250216585092904`, 0.12074100130465794`, 0.20558704294754135`,
0.14246463035757395`, 0.07402279649358105`, 0.07786475965990819`, 0.0340581506060693`,
0.08041523444578938`, 0.01558213710139311`, 0.16397534159102004`,
0.1177929195267288`, 0.017105792256956742`, 0.08357290011776101`,
0.23607846408267072`, 0.040883668439963376`, 0.03321036888035268`,
0.04703692179272006`, 0.1166662308048623`, 0.003812621377562659`,
0.07725972617027421`, 0.01032278551667974`, 0.13436662312586525`,
0.36570938724633445`, 0.10770468048277729`, 0.01296953141933141`,
0.04731132700667836`, 0.044446624328923344`, 0.00618235122746949`,
0.05300837236083615`, 0.10050303763689357`, 0.190210728684964`, 0.0513335253832927`,
0.11537750376189984`, 0.4878235997476536`, 0.009631465023873832`,
0.16066235202414658`, 0.17448920272235646`, 0.0816645835667271`, 0.21018687411435585`,
0.06529908683101382`, 0.14954184309941784`, 0.03624690107933765`,
0.02128298021358151`, 0.025108762686227807`, 0.10885757617356026`,
0.015409928243411416`, 0.05263634391372248`, 0.013831751270294623`,
0.1400540494420741`, 0.00406843759633261`, 0.07963999651514335`, 0.07330640592822639`,
0.07304679090028407`, 0.005040331979276583`, 0.05734420056467882`,
0.021502024380563593`, 0.03827259182580616`, 0.10814138756654423`,
0.12808460159445015`, 0.06217340387635631`, 0.031154479356549615`,
0.12116323038579138`, 0.010500654548535904`, 0.06516840129529362`,
0.03301747044044952`, 0.07547029436558676`, 0.04258003424194382`,
0.04384589188197849`, 0.03184852282369526`, 0.0780225475565889`, 0.01872239907846052`,
0.07086199931284655`, 0.010902596630949998`, 0.13966318078624584`,
0.04083229677657463 5`, 0.03782425997615499`, 0.06320106847160721`,
```

```
0.052262142440238066`, 0.04638169643741573`, 0.04276469913748643`,
0.12672730542857397`, 0.0456168335292967`, 0.05412109296934094`, 0.11581652833622333`,
0.078930530472329`, 0.04601024282871352`, 0.1506702363443959`, 0.07005305360127806`,
0.04711101142771468`, 0.017198257945144423`, 0.111219962811526`, 0.205405329278806`,
0.09596661055086143`, 0.0031196112401487386`, 0.049203736550980134`,
0.188870226685046`, 0.09812922801643555`, 0.04157488368849383`, 0.1878757151311634`,
0.06781604975098905`, 0.13737459056442122`, 0.0017052641579928189`,
0.01147809467931195`, 0.10381208418397085`, 0.09294383557583803`,
0.025770371736442615`, 0.0766033200309779`, 0.04359861555511431`,
0.0015258878654216925`, 0.130528588016478`, 0.15164736597095563`};
```

In[∘]:=
```
betaDistributionPlot = Show[Histogram[betaSample, Automatic,
  "ProbabilityDensity", Frame → {{True, False}, {True, False}},
  FrameLabel → {{"Probability Density", ""}, {"ϵi", ""}},
  LabelStyle → {Bold, FontFamily → "Arial", FontSize → 18},
  FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
  Epilog → {Text[Style["B", Bold, FontFamily → "Arial", FontSize → 16], {0.03, 10}]},
  ImageSize → 550], Plot[PDF[BetaDistribution[α, β] /. params, x],
  {x, 0, 1}, PlotStyle → {Thick, Blue}, PlotRange → {{0, 0.7}, All}]]
```

Out[∘]=



In[∘]:=
```
Export["betaDistributionPlot.svg", betaDistributionPlot]
```

Out[∘]=

```
betaDistributionPlot.svg
```

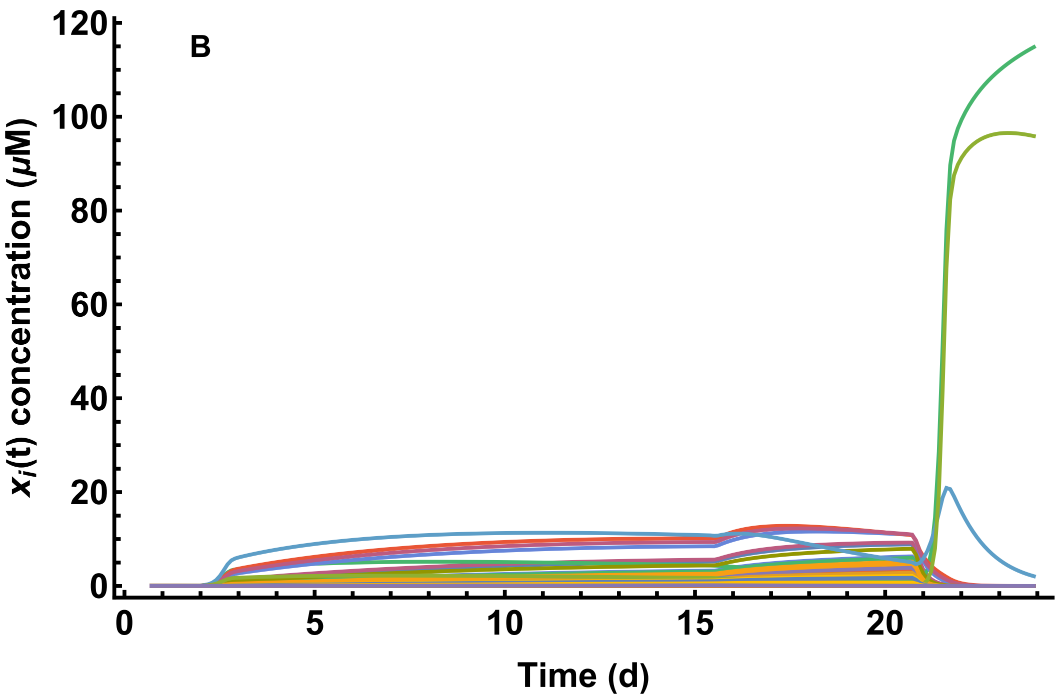
Solve ODEs for 500 ASVs again, but using the sampled beta distribution.

In[∘]:=
```
soln500beta = solveODEs[0.71, 24, params, betaSample];
```

Plot the absolute concentrations over time

In[∘]:=
```
xiBetaPlot = ListLinePlot[
  Evaluate[Table[{t, x[i][t]}, {i, 1, 500}, {t, 0.71, 24, 0.1}] /. soln500beta],
  PlotRange → All, Frame → {{True, False}, {True, False}},
  FrameLabel → {"Time (d)", "xi(t) concentration (μM)"}, LabelStyle →
   {Bold, FontFamily → "Arial", FontSize → 18, FontColor → Black}, PlotStyle → Thick,
  FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
  Epilog → {Text[Style["B", Bold, FontFamily → "Arial", FontSize → 16], {2, 115}]},
  ImageSize → 550]
```

Out[∘]=



In[∘]:=
```
Export["xiBetaPlot.svg", xiBetaPlot]
```

Out[∘]=

xiBetaPlot.svg

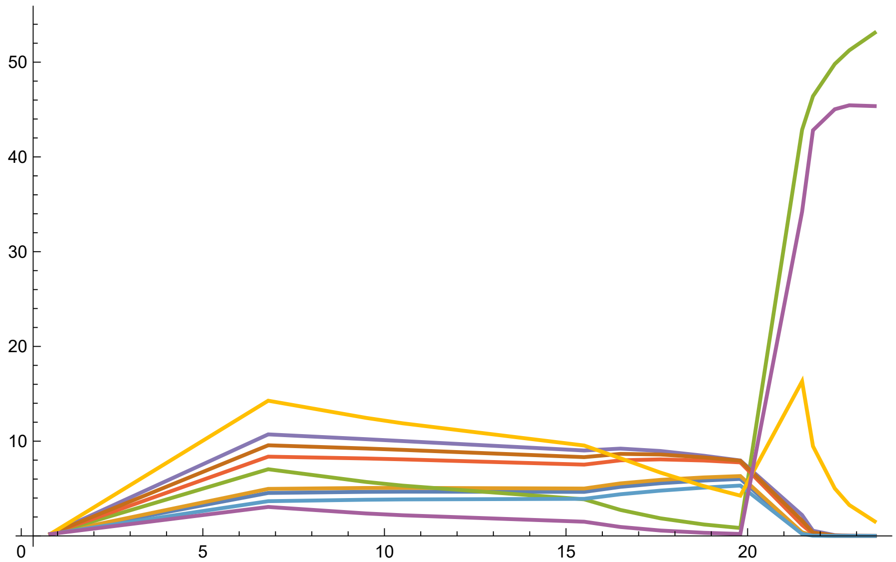
Look at relative abundances

In[∘]:=
```
times = {0.8, 6.8, 9.5, 10.5, 15.5, 16.5, 17.6, 18.8, 19.8, 21.5, 21.8, 22.4, 22.8, 23.5};
betaASVs =
  Prepend[Evaluate[Table[Table[x[i][t], {i, 1, 500}] /. soln500beta /. t → times[[j]],
      {j, 1, Length[times]}]], Evaluate[Table["ASV_" <> ToString[i], {i, 500}]]]ᵀ;
betaASVsRel = Prepend[Table[100 betaASVs[[1 ;;, i]] / Total[betaASVs[[1 ;;, i]]],
     {i, 2, Dimensions[betaASVs][[2]]}], betaASVs[[1 ;;, 1]]]ᵀ;
```

Plot ASVs that have a relative abundance > 5 %

In[∘]:=
```
asvBetaRelGT5 = Select[betaASVsRel, (Max[#[[2 ;;]]] > 5) &];
ListLinePlot[
 Table[{times, asvBetaRelGT5[[i, 2 ;;]]}ᵀ, {i, 1, Length[asvBetaRelGT5]}], PlotRange → All]
```

Out[∘]=

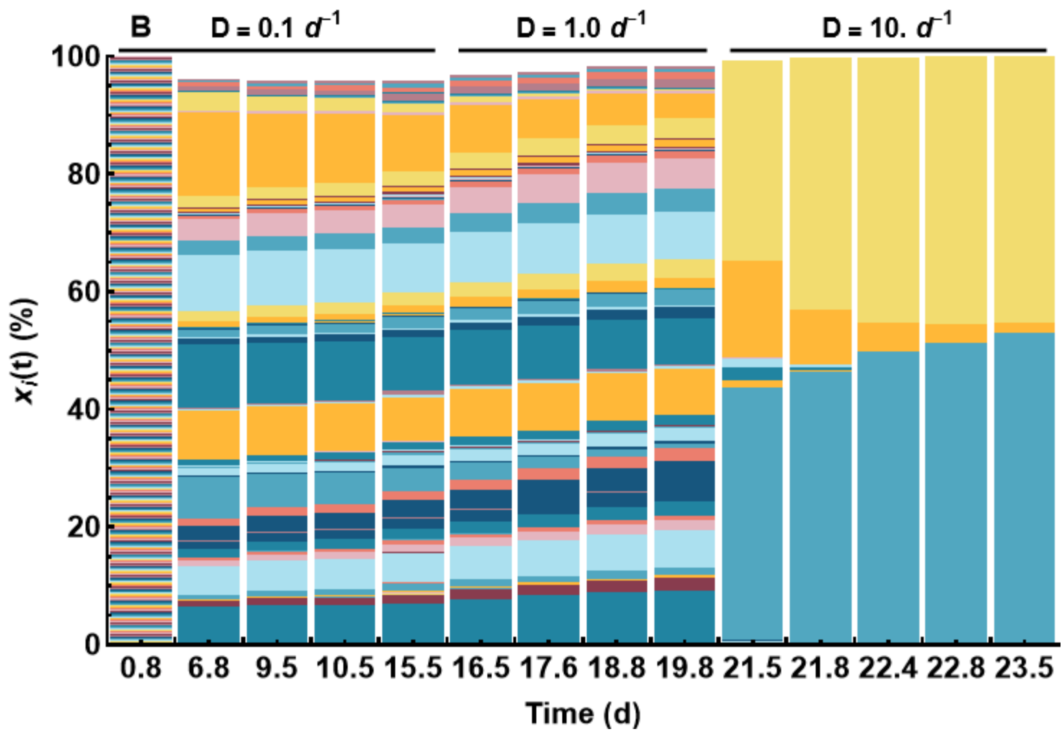


Look at bar plot with relative abundances > 0.1%.

In[∘]:=
```
asvMinAbunRel = 0.1;
asvBetaMin = Chop[N[betaASVsRel], asvMinAbunRel];
(* this sets any ASV that is < asvMinAbunRel to 0 *)
```

In[ ]:=
```
xiBarBetaPlot = xiBarChart[asvBetaMin[[1 ;;, 2 ;;]]ᵀ,
  {{0.8, 6.8, 9.5, 10.5, 15.5, 16.5, 17.6, 18.8, 19.8, 21.5, 21.8, 22.4, 22.8, 23.5},
   Table["", {14}]}, {1, 5, 6, 9, 10, 14}, "B"]
```

Out[ ]=



In[ ]:=
```
Export["xiBarBetaPlot.svg", xiBarBetaPlot]
```

Out[ ]=

xiBarBetaPlot.svg

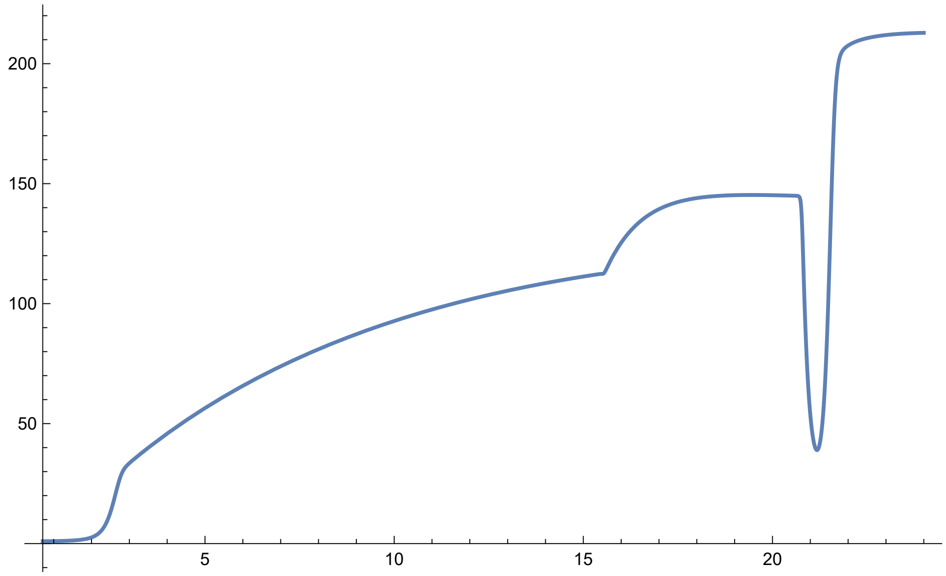
For a single ASVs (this evaluation has been removed from the notebook)

In[ ]:=
```
Manipulate[Plot[Evaluate[x[i][t] /. soln500beta],
  {t, 0.71, 24}, PlotLabel → "ASV: " <> ToString[i]], {i, 1, 500, 1}]
```

Sum of all ASVs (absolute concentration in μM C)

In[∘]:=
```
Plot[Evaluate[Sum[x[i][t], {i, 1, 500}] /. soln500beta], {t, 0.71, 24}, PlotRange → All]
```

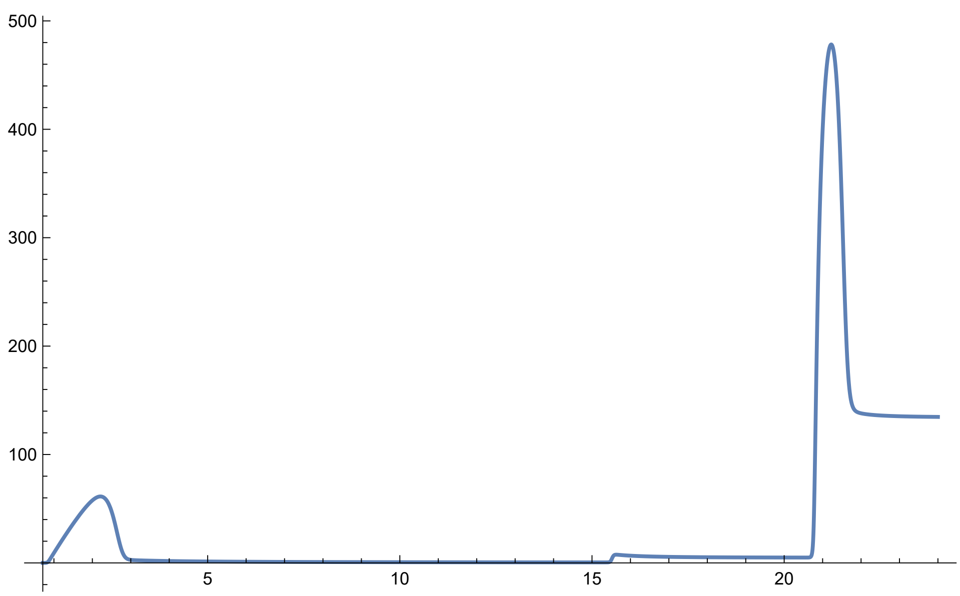
Out[∘]=



Substrate concentration (μM C)

In[∘]:=
```
Plot[Evaluate[s[t] /. soln500beta], {t, 0.71, 24}, PlotRange → All]
```

Out[∘]=



Export the ASVs for the beta distribution simulation

In[∘]:=
```
Export["betaASVs.csv", Prepend[betaASVs, Flatten[{"ASVs", times}]]]
```

Out[∘]=

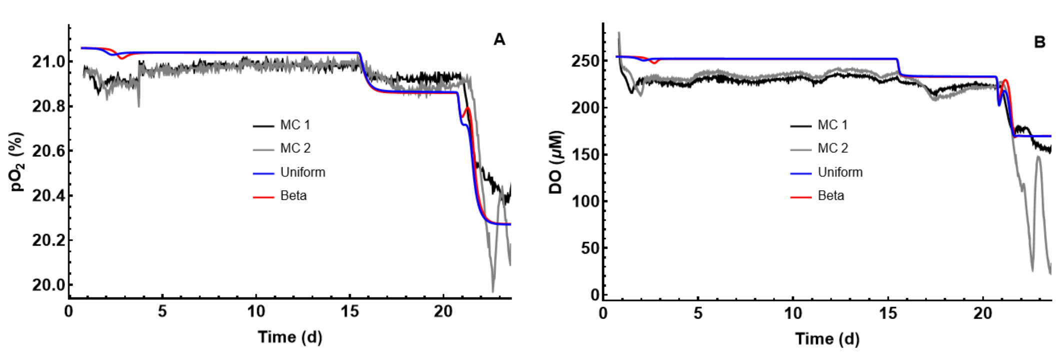
```
betaASVs.csv
```

In[•]:=
```
Export["betaASVsRelative.csv", Prepend[betaASVsRel, Flatten[{"ASVs", times}]]]
```
Out[•]=
```
betaASVsRelative.csv
```

Simulation comparison to observed pO_2 and DO

Plot the simulated vs observed and export fig.

In[•]:=
```
pO2andDOFig = GraphicsRow[
  {Show[ListLinePlot[o2Data, PlotRange → {{0, 23.6}, All}, PlotStyle → {Black, Gray},
     PlotLegends → Placed[LineLegend[{Black, Gray, Blue, Red},
        {"MC 1", "MC 2", "Uniform", "Beta"}], Center], Epilog → {Text[Style["A",
         Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black], {23, 21.1}]}],
    Plot[Evaluate[100 pO2[t] /. soln500beta], {t, 0.71, 25}, PlotStyle → Red,
     PlotRange → All], Plot[Evaluate[100 pO2[t] /. soln500],
     {t, 0.71, 24}, PlotStyle → Blue, PlotRange → All],
    Frame → {{True, False}, {True, False}}, FrameLabel → {"Time (d)", "pO2 (%)"},
    FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
    LabelStyle → {Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black},
    PlotRange → {{0, 23.6}, {20, 21.1}}, PlotLegends → Placed[LineLegend[
       {Black, Gray, Blue, Red}, {"MC 1", "MC 2", "Uniform", "Beta"}], Center]],
   Show[ListLinePlot[doData, PlotRange → {{0, 23.6}, All},
     PlotStyle → {Black, Gray}, PlotLegends → Placed[
       LineLegend[{Black, Gray, Blue, Red}, {"MC 1", "MC 2", "Uniform", "Beta"}], Center],
     Epilog → {Text[Style["B", Bold, FontFamily → "Arial",
         FontSize → 16, FontColor → Black], {23, 270}]}],
    Plot[Evaluate[o2[t] /. soln500beta], {t, 0.71, 24}, PlotStyle → Red, PlotRange → All],
    Plot[Evaluate[o2[t] /. soln500], {t, 0.71, 24}, PlotStyle → Blue, PlotRange → All],
    Frame → {{True, False}, {True, False}},
    FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
    FrameLabel → {"Time (d)", "DO (μM)"},
    LabelStyle → {Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black},
    PlotRange → {{0, 23.6}, {0, 275}}]}, ImageSize → 1024]
```
Out[•]=



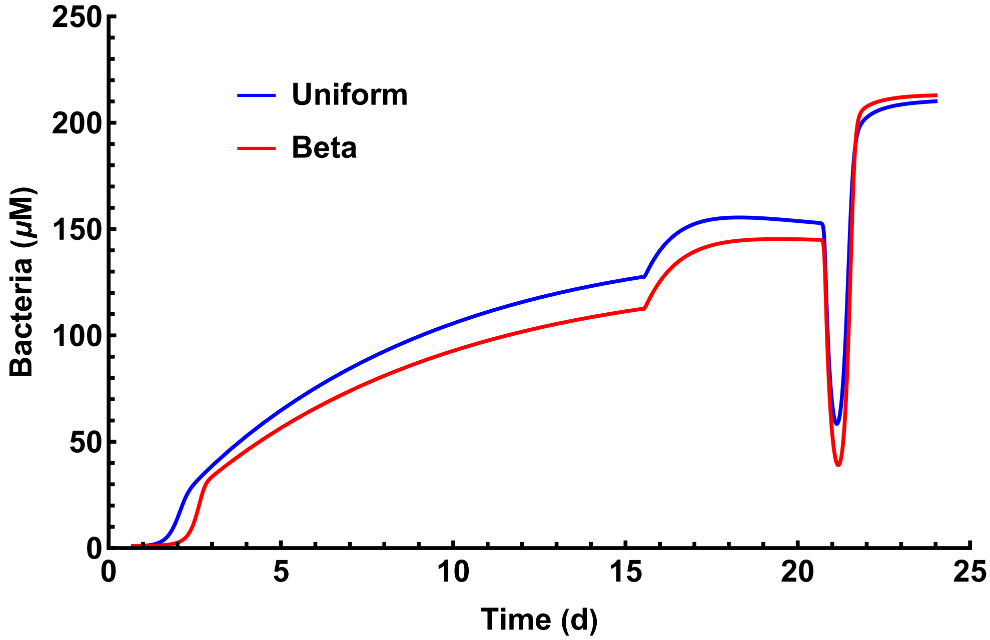
In[∘]:=
```
Export["pO2andDOFig.svg", pO2andDOFig]
```

Out[∘]=
```
pO2andDOFig.svg
```

Total simulated bacterial concentrations for both distributions

In[∘]:=
```
modelBacteriaConcentrationFig = Plot[{Evaluate[Sum[x[i][t], {i, 1, 500}] /. soln500],
   Evaluate[Sum[x[i][t], {i, 1, 500}] /. soln500beta]}, {t, 0.71, 24},
  Frame → {{True, False}, {True, False}}, FrameLabel → {"Time (d)", "Bacteria (μM)"},
  LabelStyle → {Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black},
  PlotRange → {{0, 25}, {0, 250}}, ImageSize → 512,
  PlotStyle → {{Thick, Blue}, {Thick, Red}},
  FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
  PlotLegends → Placed[LineLegend[{Blue, Red}, {"Uniform", "Beta"}], {0.25, 0.8}]]
```

Out[∘]=



In[∘]:=
```
Export["modelBacteriaConcentrationFig.svg", modelBacteriaConcentrationFig]
```

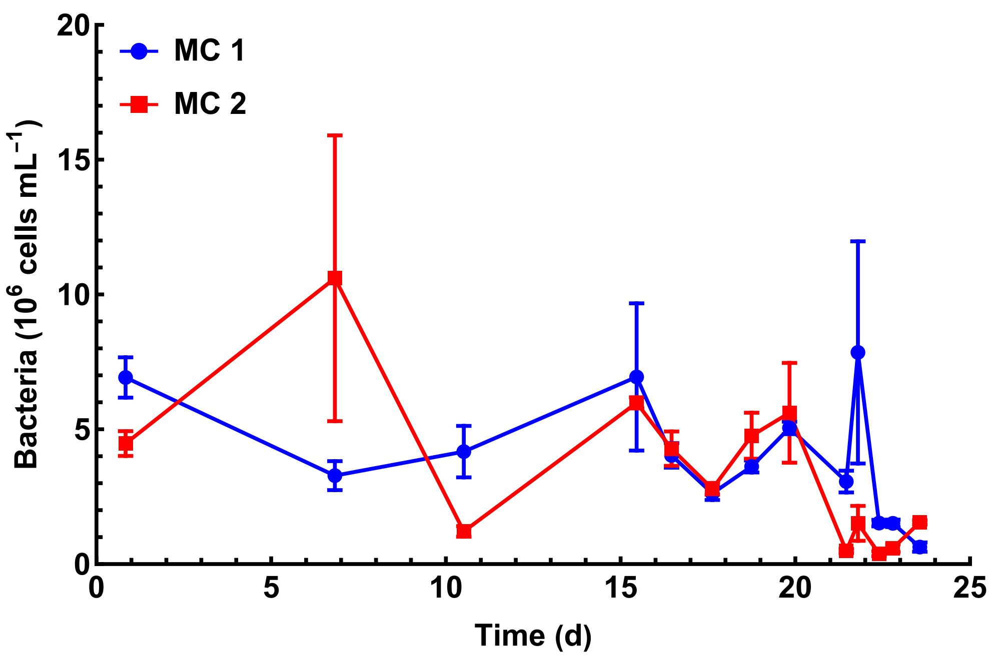
Out[∘]=
```
modelBacteriaConcentrationFig.svg
```

Import bacterial cell counts

In[∘]:=
```
counts = Import["..\\Data\\BacterialCellCounts.csv"];
```

In[∘]:=
```
bacterialCounts =
 ListLinePlot[{counts[[2 ;;, {1, 2, 3}]] /. {x_, y_, z_} → {x, Around[10^-6 y, 10^-6 z]},
   counts[[2 ;;, {1, 4, 5}]] /. {x_, y_, z_} → {x, Around[10^-6 y, 10^-6 z]}},
  PlotMarkers → Automatic, Frame → {{True, False}, {True, False}},
  FrameLabel → {"Time (d)", "Bacteria (10^6 cells mL^-1)"},
  LabelStyle → {Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black},
  PlotRange → {{0, 25}, {0, 20}}, ImageSize → 512,
  PlotStyle → {{Thick, Blue}, {Thick, Red}},
  FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
  PlotLegends → Placed[LineLegend[{Blue, Red}, {"MC 1", "MC 2"}], {0.1, 0.9}]]
```

Out[∘]=



In[∘]:=
```
Export["BacterialCounts.svg", bacterialCounts]
```

Out[∘]=
```
BacterialCounts.svg
```

Observed ASV Data Plots

Import all of the ASV data for MC1 and MC2, including taxa names and ASV data for Siders Pond which is in the file Table_S1_ASVs.csv in the Data subdirectory. Drop the header.

In[∘]:=
```
asvData = Drop[Import["..\\Data\\TableS1_ASVs.csv"], 1];
```

In[∘]:=
```
Dimensions[asvData]
```

Out[∘]=
```
{1420, 34}
```

Get the ASV taxon names

```
In[ ]:= taxaNames = asvData⟦ ;; , {29, 30, 31, 32, 33, 34}⟧;
```

Add Siders ASV data as time "Siders" to each dataset and specify when samples were collected in days, which are slightly different for each MC.

```
In[ ]:= timesMC1 =
          {"Siders", 0.8, 10.5, 15.5, 16.5, 17.6, 18.8, 19.8, 21.5, 21.8, 22.4, 22.8, 23.5};
        timesMC2 =
          {"Siders", 0.8, 6.8, 9.5, 10.5, 15.5, 16.5, 17.6, 18.8, 19.8, 21.5, 21.8, 22.4, 23.5};
        timesMC12 = {"Siders", 0.8, 6.8, 9.5, 10.5,
           15.5, 16.5, 17.6, 18.8, 19.8, 21.5, 21.8, 22.4, 22.8, 23.5};
```

```
In[ ]:= asvMC1 = Table[0, Dimensions[asvData]⟦1⟧, Length[timesMC12] + 1];
        (* initialize these to 0 *)
        asvMC2 = Table[0, Dimensions[asvData]⟦1⟧, Length[timesMC12] + 1];
```

Copy in the asvData, but add in Siders' ASVs and keep columns 0 for where samples did not sequence.

```
In[ ]:= asvMC1⟦ ;; , Flatten[{1, 2, 3, Range[6, 16]}]⟧ = asvData⟦ ;; , Range[1, 14]⟧;
        asvMC2⟦ ;; , Flatten[{Range[1, 14], 16}]⟧ = asvData⟦ ;; , Flatten[{1, 2, Range[15, 27]}]⟧;
```

Define function to name and ASV based on columns in taxaNames above

```
In[ ]:= nameMake[n1_String, n2_String] := n1 <> " " <> n2
```

Define function for generating bar charts with a percentage cutoff. This version of plotASVobs sets ASVs with relative abundances less than minPer to zero instead of removing them. That way, ASV color indexes stay the same. maxLedgend specifies how many of the top ASV taxon names to display, and colorIndex is based on the Mathematica's Indexed Color Palette Collections

```
In[•]:= plotASVobs[asvData_, timePts_List, minPer_, maxLedgend_, colorIndex_, dBars_List, fig_] :=
  Module[{asvDataRel, asvDataRelmin, tf, colors},
   (*Relative abundance of an ASV in each sample time, gen in percentage*)
   asvDataRel = Prepend[Table[
       100 asvData[[ ;; , i]] / If[Total[asvData[[ ;; , i]]] ≠ 0, Total[asvData[[ ;; , i]]], 1],
       {i, 2, Dimensions[asvData][[2]]}], asvData[[ ;; , 1]]]ᵀ;
   (*Any ASV less than minPer is set to zero. Note,
   Chop only works on reals, not integers or rational numbers*)
   asvDataRelmin = Chop[N[asvDataRel], minPer];
   (*Get a vector of colors based on colorIndex and add it as the first column *)
   colors = Table[ColorData[colorIndex][i], {i, Length[asvData]}];
   asvDataRelmin = Join[Partition[colors, 1], asvDataRelmin, 2];
   (* sort asvDataRelmin so the largest are first *)
   asvDataRelmin = ReverseSortBy[asvDataRelmin, Max[#[[3 ;;]]] &];
   tf[p_] := TableForm[Partition[Flatten[p], UpTo[10]], TableSpacing → {1, 1}];
   TableForm[{BarChart[asvDataRelmin[[ ;; , 3 ;;]]ᵀ, ChartLayout → "Stacked",
      ChartLabels → {timePts, Table["", {Length[asvDataRelmin]}]},
      PlotRange → {{0.5, Dimensions[asvDataRelmin][[2]] - 1.5}, {0, 100}},
      LabelStyle → {Bold, FontFamily → "Arial", FontSize → 16, FontColor → Black},
      ChartStyle → asvDataRelmin[[ ;; , 1]], FrameLabel →
       {"Time (d)", "ASV Rel. Abundance (%)"}, Frame → {{True, False}, {True, False}},
      FrameStyle → {{Directive[Thick, Black], None}, {Directive[Thick, Black], None}},
      ImagePadding → {{All, All}, {All, 30}}, ImageSize → 550,
      Epilog → {Text[Style[fig, Bold, FontFamily → "Arial", FontSize → 16], {1, 105}],
        Thick, Text[Style["D = 0.1 d^-1", Bold, 14], {(dBars[[1]] + dBars[[2]]) / 2, 105}],
        Line[{{dBars[[1]] - 0.3, 101}, {dBars[[2]] + 0.3, 101}}],
        Text[Style["D = 1.0 d^-1", Bold, 14], {(dBars[[3]] + dBars[[4]]) / 2, 105}],
        Text[Style["D = 10. d^-1", Bold, 14], {(dBars[[5]] + dBars[[6]]) / 2, 105}],
        Line[{{dBars[[3]] - 0.3, 101}, {dBars[[4]] + 0.3, 101}}],
        Line[{{dBars[[5]] - 0.3, 101}, {dBars[[6]] + 0.3, 101}}]}],
     SwatchLegend[asvDataRelmin[[ ;; maxLedgend, 1]], asvDataRelmin[[ ;; maxLedgend, 2]],
      LegendMarkerSize → 10, LabelStyle → {FontSize → 8}, LegendLayout → tf]}]
  ];
```

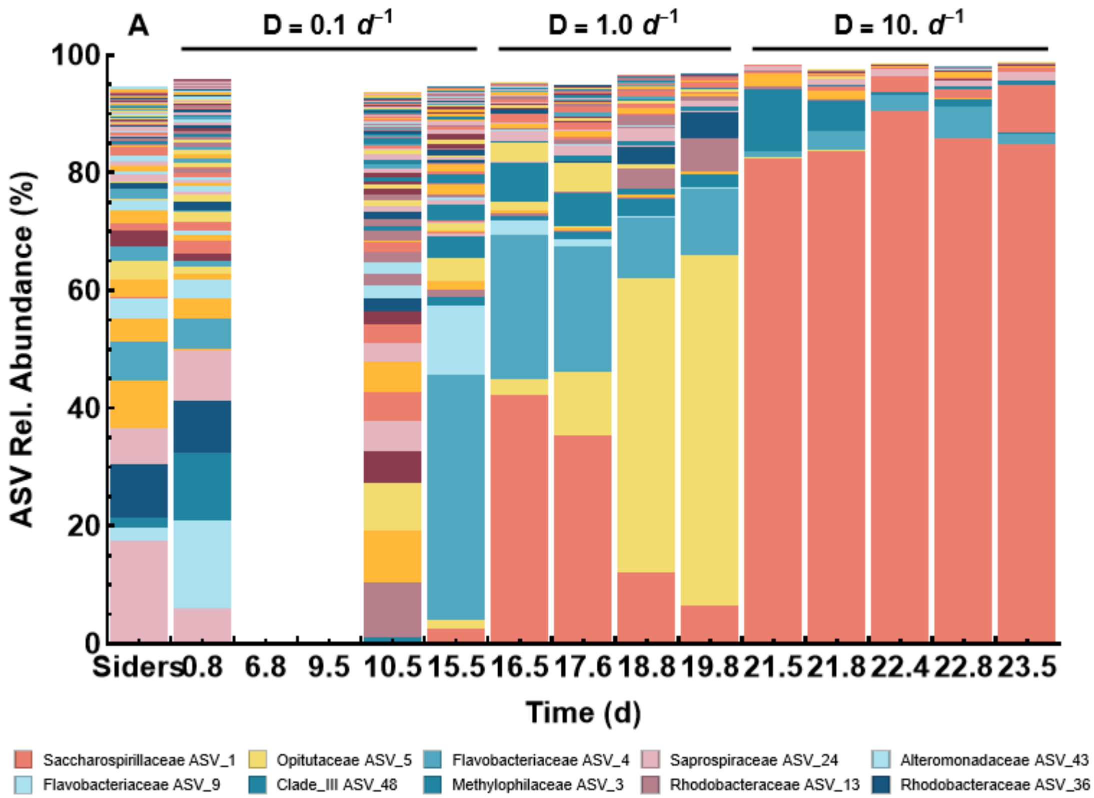
Plot with names from Family

```
In[•]:= familyMC1 = asvMC1;
familyMC1[[ ;; , 1]] = Thread[nameMake[taxaNames[[ ;; , 5]], asvMC1[[ ;; , 1]]]];
familyMC2 = asvMC2;
familyMC2[[ ;; , 1]] = Thread[nameMake[taxaNames[[ ;; , 5]], asvMC2[[ ;; , 1]]]];
```

Plot ASVs who’s relative abundance > 0.1%

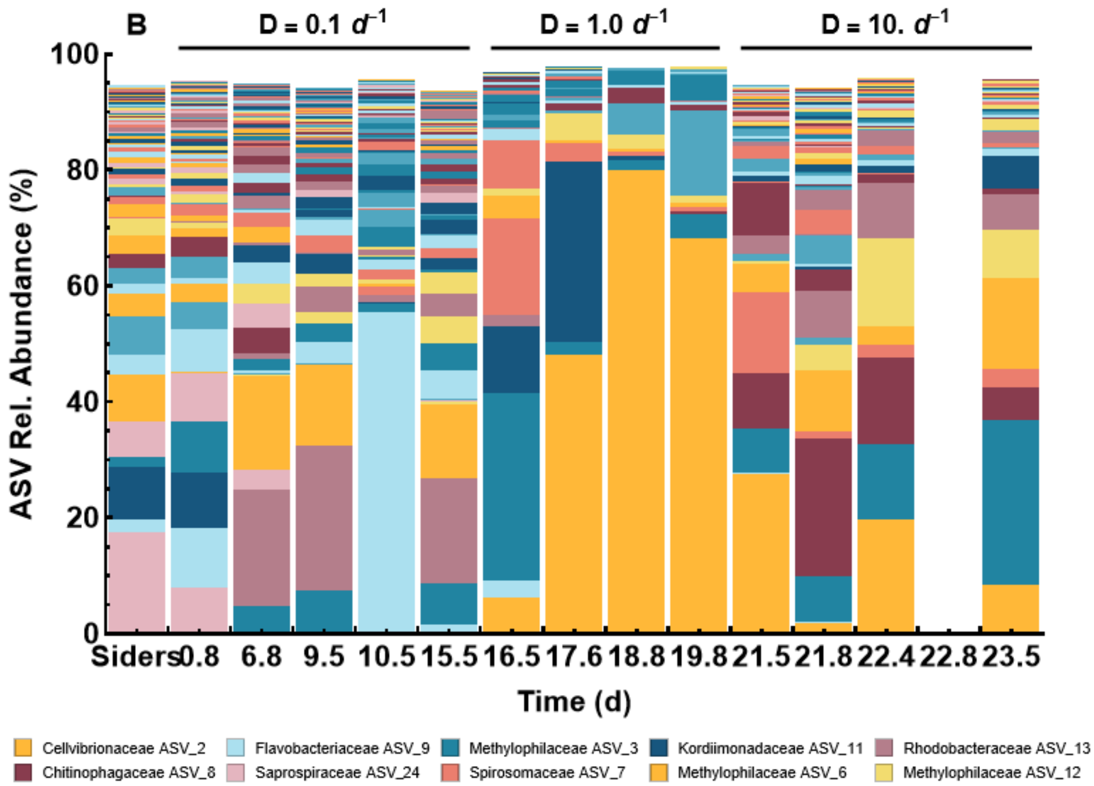
```
In[•]:= mc1family = plotASVobs[familyMC1, timesMC12, 0.1, 10, 24, {2, 6, 7, 10, 11, 15}, "A"]
```

Out[•]//TableForm=



In[•]:=
```
mc2family = plotASVobs[familyMC2, timesMC12, 0.1, 10, 24, {2, 6, 7, 10, 11, 15}, "B"]
```

Out[•]//TableForm=



In[•]:=
```
Export["familyMC1_24.svg", mc1family];
Export["familyMC2_24.svg", mc2family];
```

Generate a legend based on the top maxLedgend based on both MCs

In[∘]:=
```
legendASVobs[asvData_, minPer_, maxLedgend_, colorIndex_, fontSize_] :=
 Module[{asvDataRel, asvDataRelmin, tf, colors},
  (*Relative abundance of an ASV in each sample time, gen in percentage*)
  asvDataRel = Prepend[
     Table[100 asvData⟦ ;; , i⟧ / If[Total[asvData⟦ ;; , i⟧] ≠ 0, Total[asvData⟦ ;; , i⟧], 1],
      {i, 2, Dimensions[asvData]⟦2⟧}], asvData⟦ ;; , 1⟧]ᵀ;
  (*Any ASV less than minPer is set to zero. Note,
  Chop only works on reals, not integers or rational numbers*)
  asvDataRelmin = Chop[N[asvDataRel], minPer];
  (*Get a vector of colors based on colorIndex and add it as the first column *)
  colors = Table[ColorData[colorIndex][i], {i, Length[asvData]}];
  asvDataRelmin = Join[Partition[colors, 1], asvDataRelmin, 2];
  (* sort asvDataRelmin so the largest are first *)
  asvDataRelmin = ReverseSortBy[asvDataRelmin, Max[#⟦3 ;;⟧] &];
  tf[p_] := TableForm[Partition[Flatten[p], UpTo[10]], TableSpacing → {1, 1}];
  SwatchLegend[asvDataRelmin⟦ ;; maxLedgend, 1⟧, asvDataRelmin⟦ ;; maxLedgend, 2⟧,
   LegendMarkerSize → 10, LabelStyle → {FontSize → fontSize}, LegendLayout → tf]
 ]
```

Top ASVs for both MCs

In[∘]:=
```
legendFamilyTop30 =
 legendASVobs[Join[familyMC1, familyMC2⟦ ;; , Range[3, 16]⟧, 2], 0.1, 30, 24, 10]
```

Out[∘]=

Saccharospirillaceae ASV_1	Cellvibrionaceae ASV_2	Opitutaceae ASV_5	Flavobacteriaceae ASV_9	Flavo
Methylophilaceae ASV_3	Kordiimonadaceae ASV_11	Rhodobacteraceae ASV_13	Chitinophagaceae ASV_8	Sapro
Spirosomaceae ASV_7	Methylophilaceae ASV_6	Methylophilaceae ASV_12	Alteromonadaceae ASV_43	Sphir
Clade_III ASV_48	Rhodobacteraceae ASV_36	Caulobacteraceae ASV_14	Comamonadaceae ASV_17	Marin
Cyclobacteriaceae ASV_21	Saprospiraceae ASV_67	Burkholderiaceae ASV_29	Ilumatobacteraceae ASV_58	Rhod
Rhizobiaceae ASV_19	Saprospiraceae ASV_30	Opitutaceae ASV_32	Methylophilaceae ASV_94	Methy

In[∘]:=
```
Export["legendFamilyTop30.svg", legendFamilyTop30]
```

Out[∘]=
```
legendFamilyTop30.svg
```

Plot with names from Order

In[∘]:=
```
orderMC1 = asvMC1;
orderMC1⟦ ;; , 1⟧ = Thread[nameMake[taxaNames⟦ ;; , 4⟧, asvMC1⟦ ;; , 1⟧]];
orderMC2 = asvMC2;
orderMC2⟦ ;; , 1⟧ = Thread[nameMake[taxaNames⟦ ;; , 4⟧, asvMC2⟦ ;; , 1⟧]];
```

Plot ASVs who’s relative abundance > 0.1%

In[∘]:=
```
Export["orderMC1_24b.svg", mc1Order];
Export["orderMC2_24b.svg", mc2Order];
```

Conclusions

We see a decrease in α diversity with increasing dilution rate that matches the chemostat data only if the community maximum specific growth rate is skewed like the beta distribution; that is, the natural community appears to be skewed towards gleaners (oligotrophs) over opportunists (copiotrophs).