



Lectures 15-16: Carbonates

1. Carbon cycle

A. Carbonate Factories

- a. T vs C vs M
- b. Examples of each
- c. Sedimentation rates and growth potential of each

B. Geometry of Carbonate Accumulations

C. Modeling Carbonate Stratigraphy

We acknowledge and respect the lək̓ʷəŋən peoples on whose traditional territory the university stands and the Songhees, Esquimalt and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

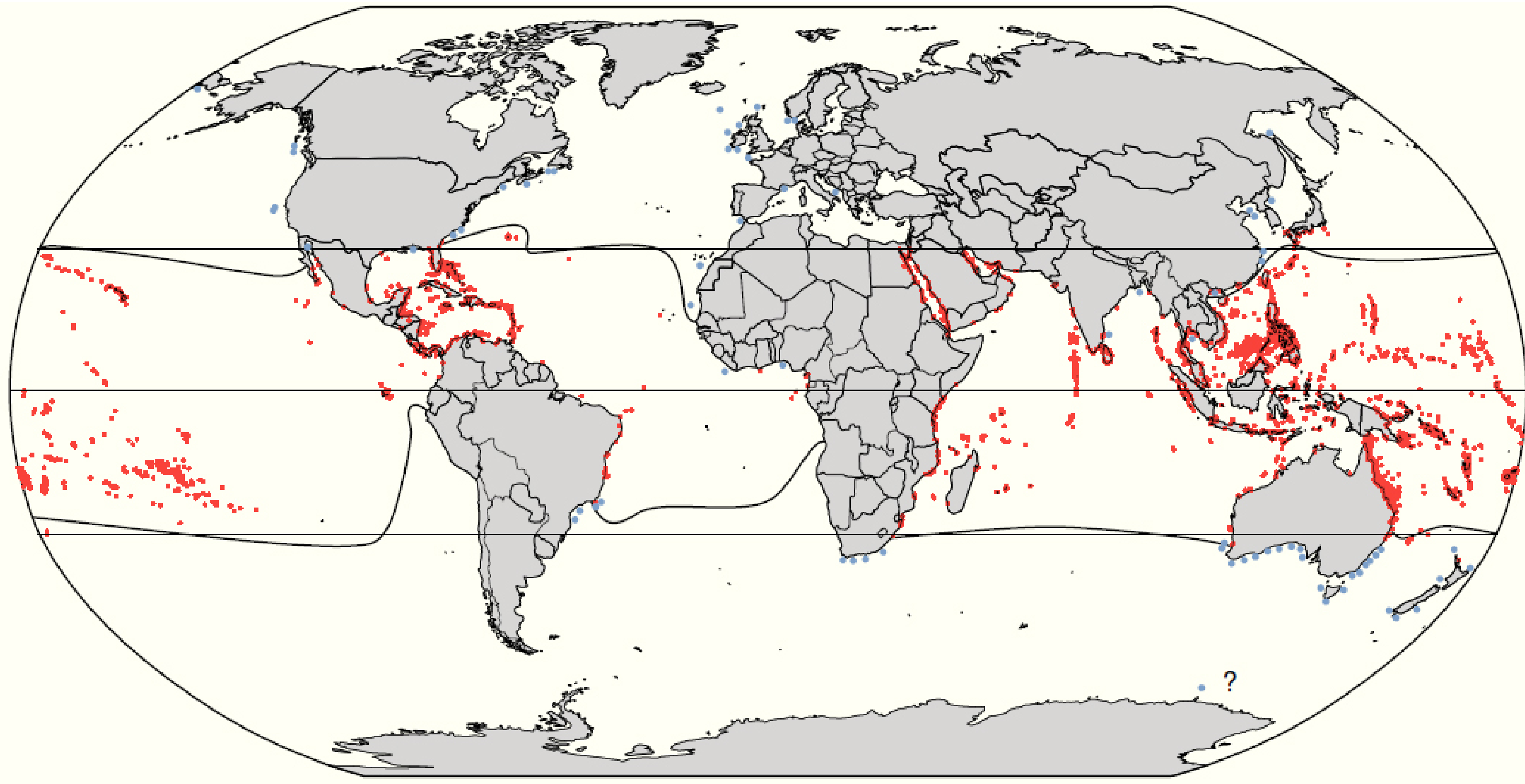


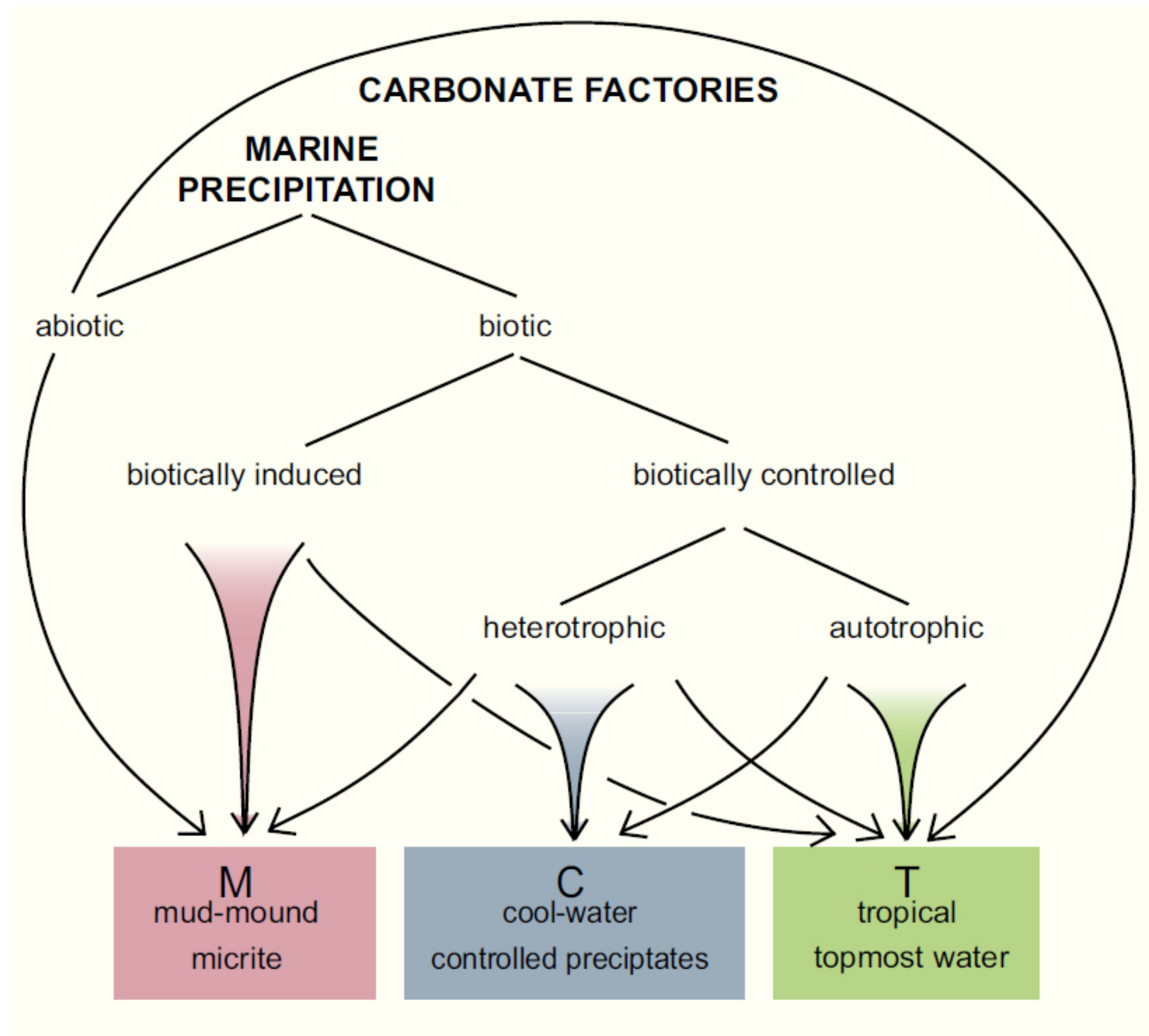
Introduction to final assignment concept.
(drawing on board)

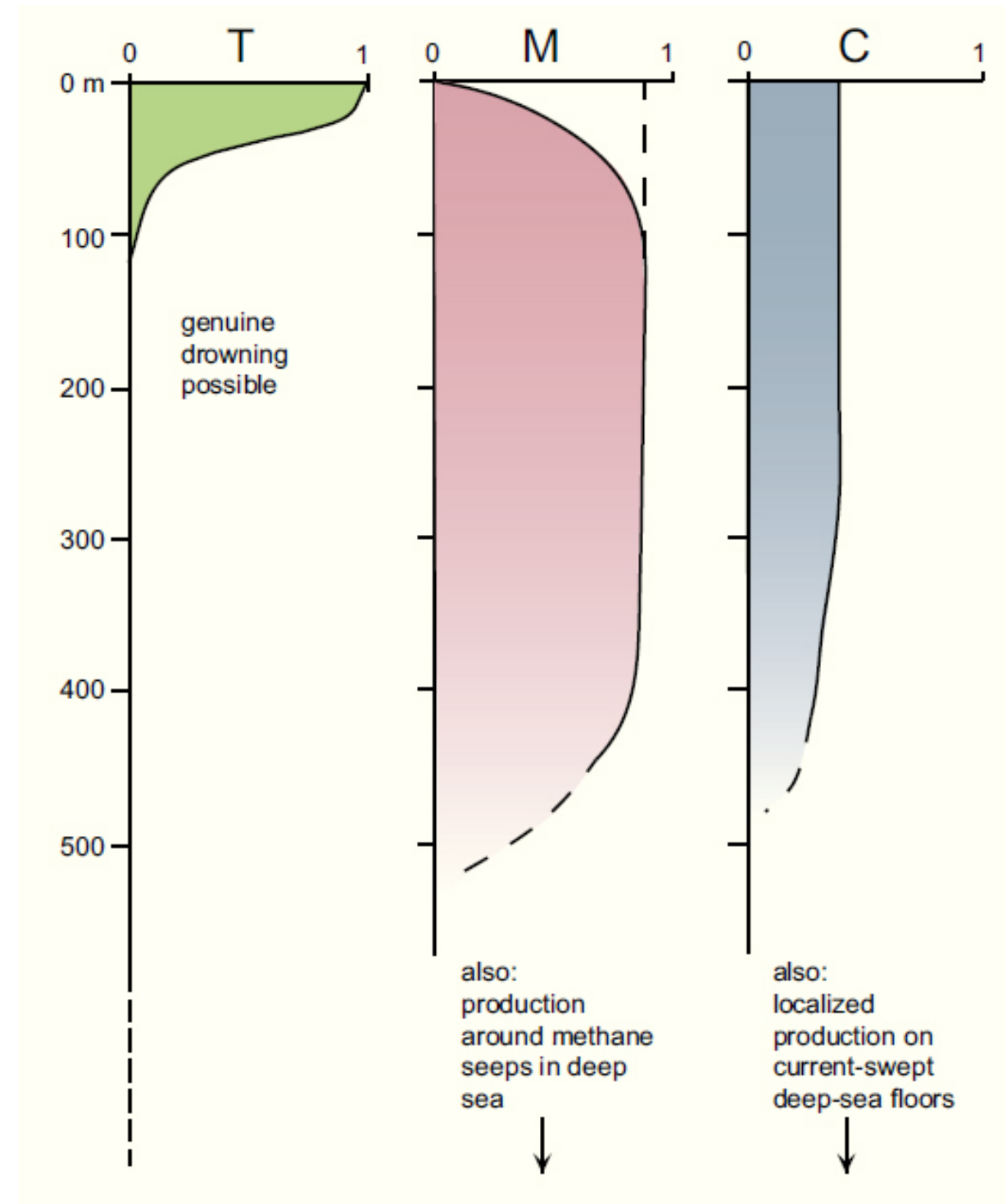


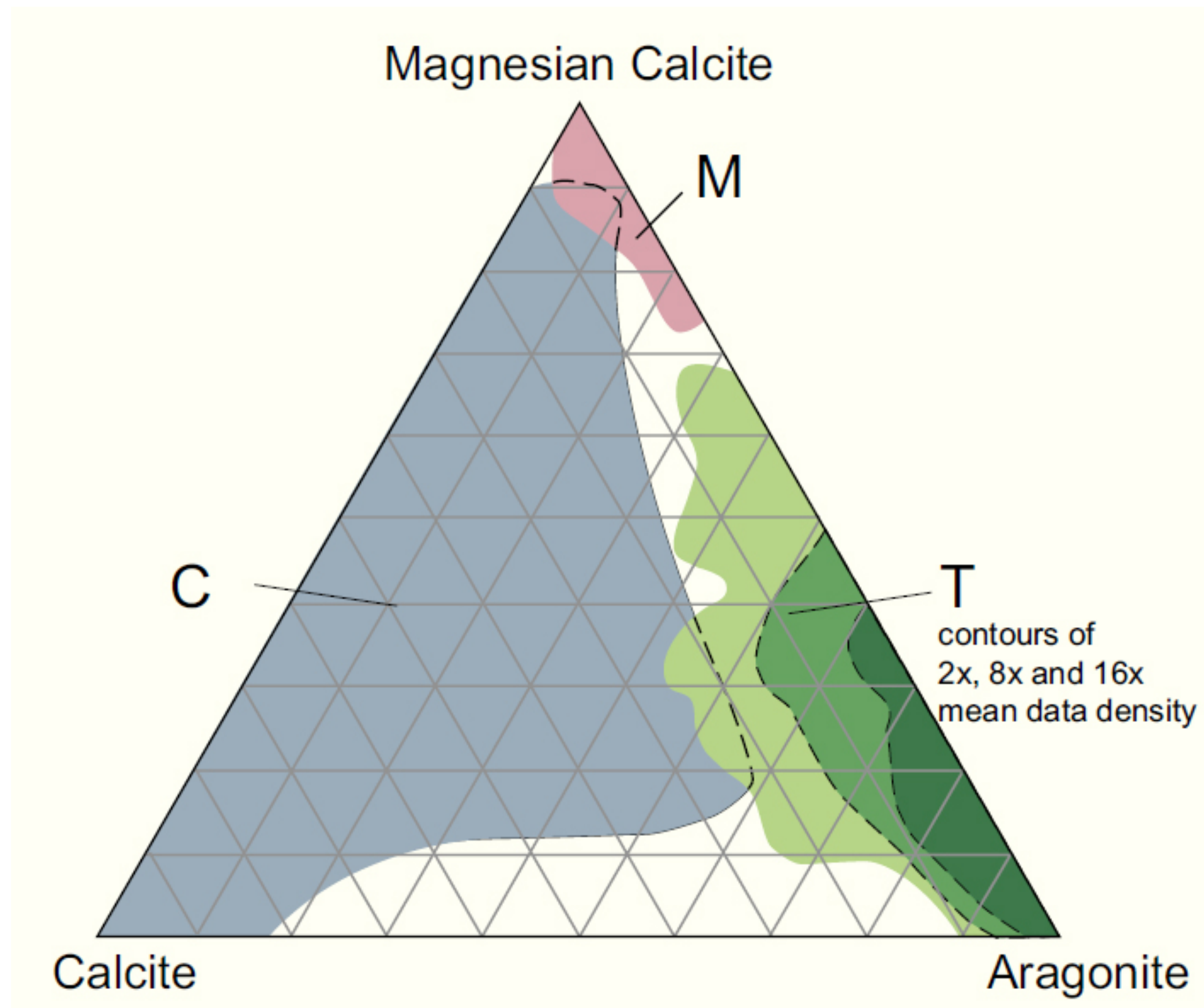
Carbonates, the big picture:
(drawing on board)











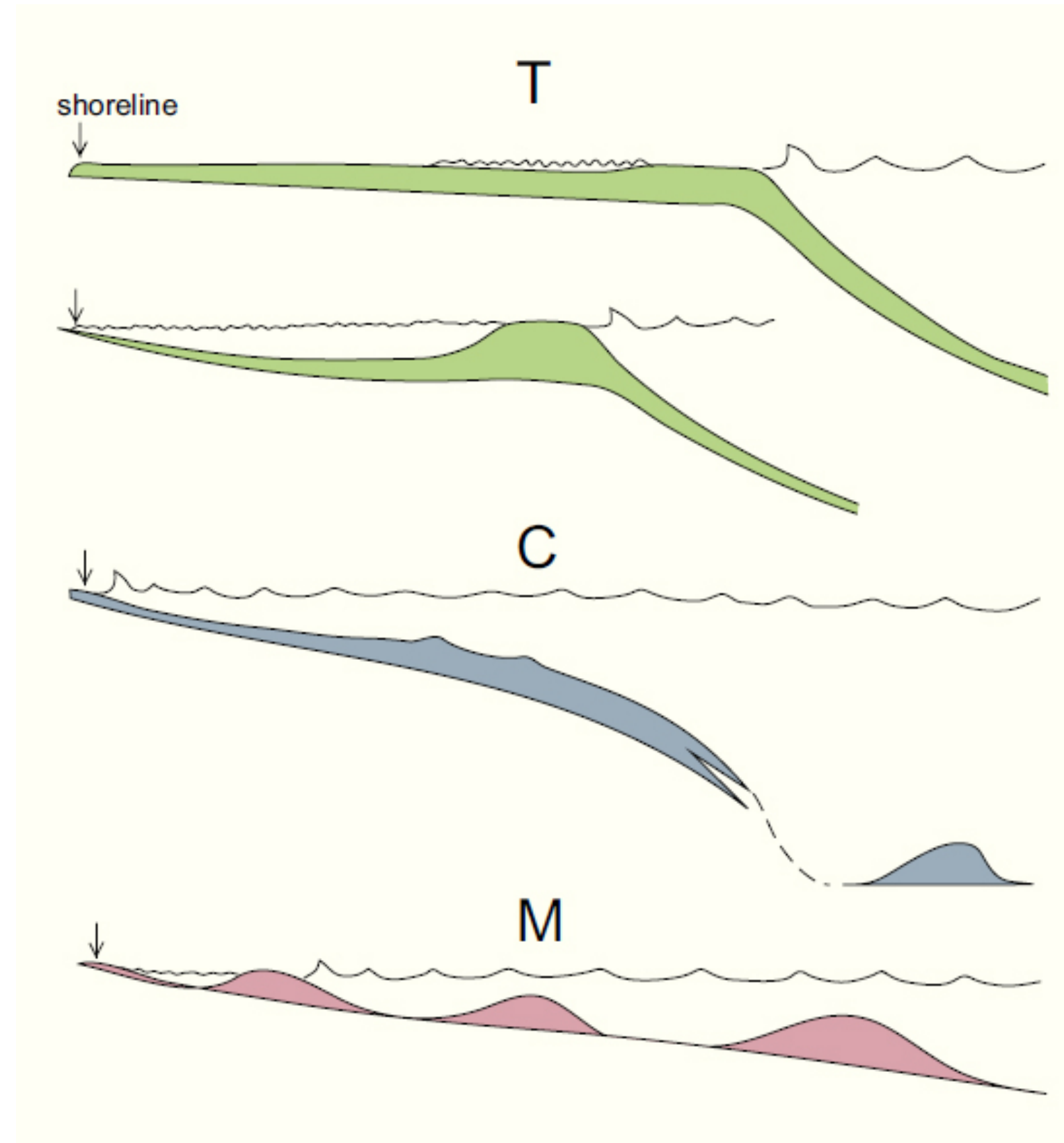
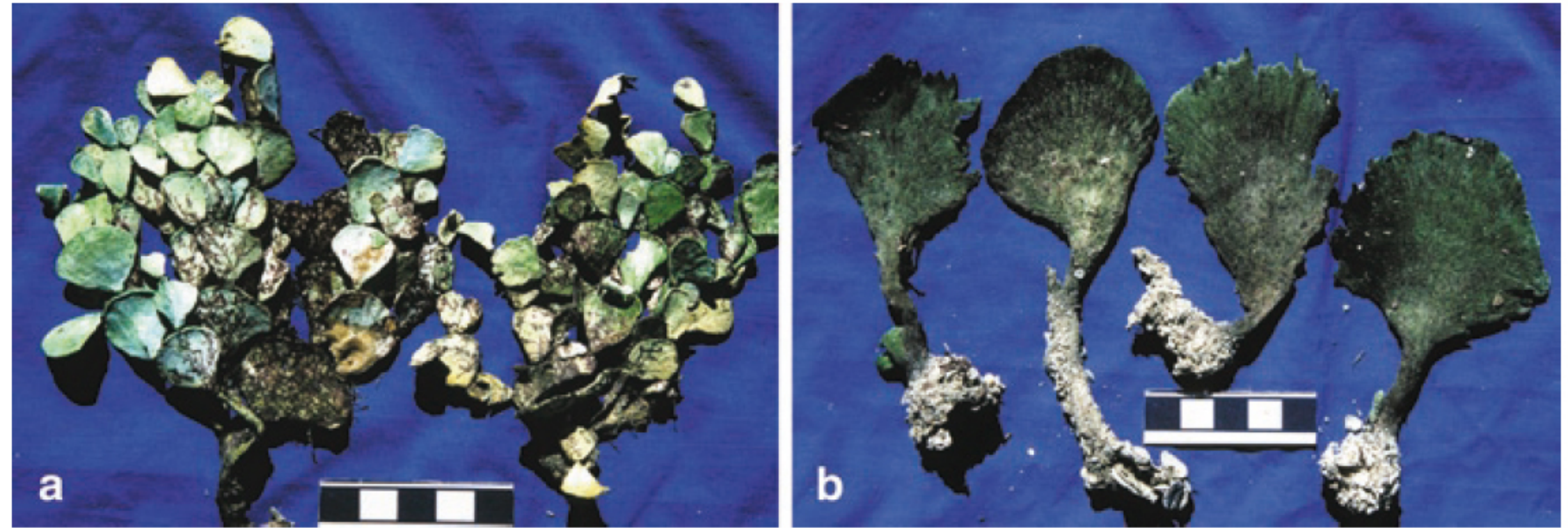






Fig. 4.11 Calcareous green algae that are non-calci-fied. **(a)** *Halimeda cuncata*, ~2 mwd. Recherche Archi-pelago, Albany Shelf, **(b)** *Rhipiliopsis peltata* ~2 mwd. Recherche Archipelago, Albany shelf





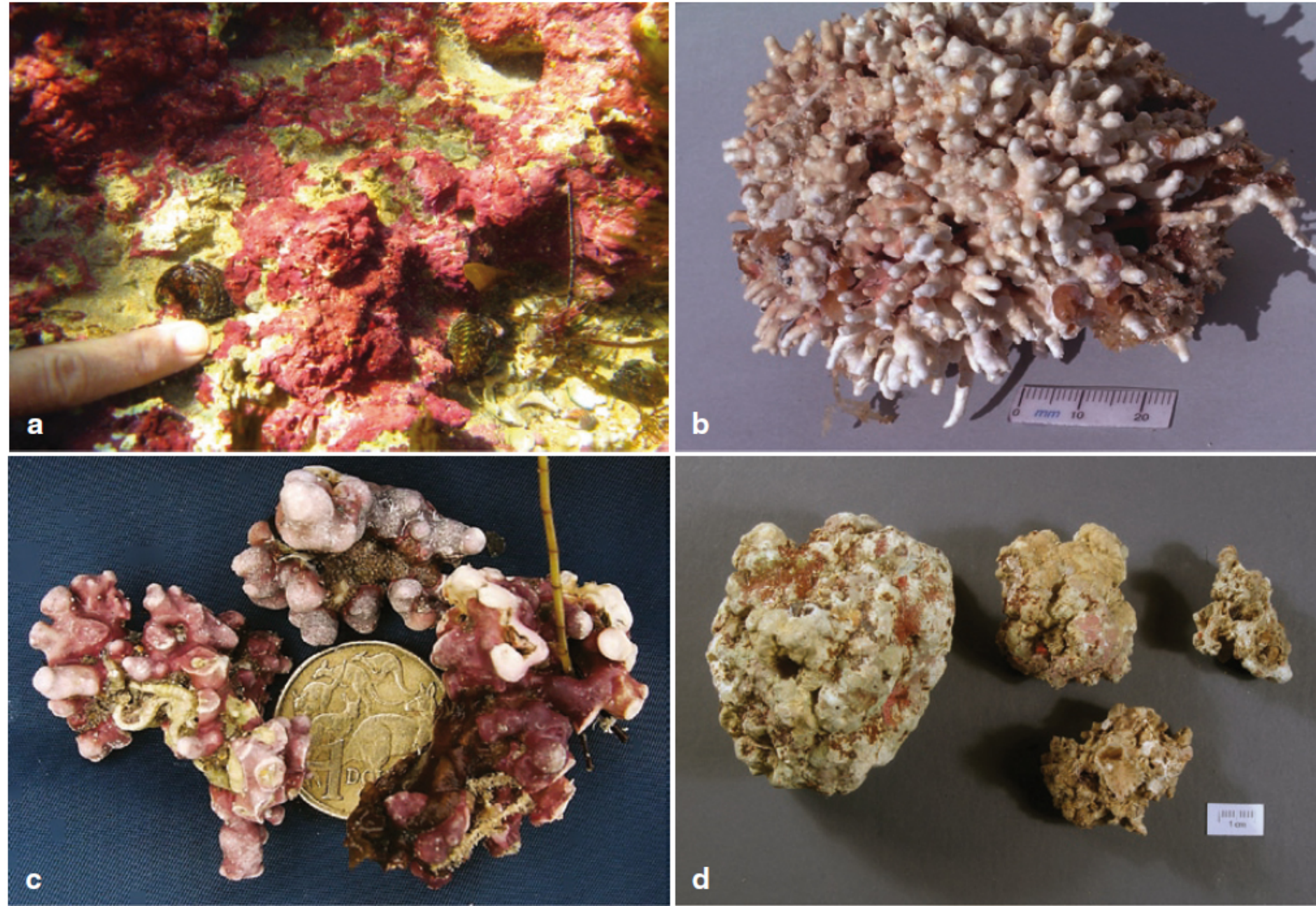
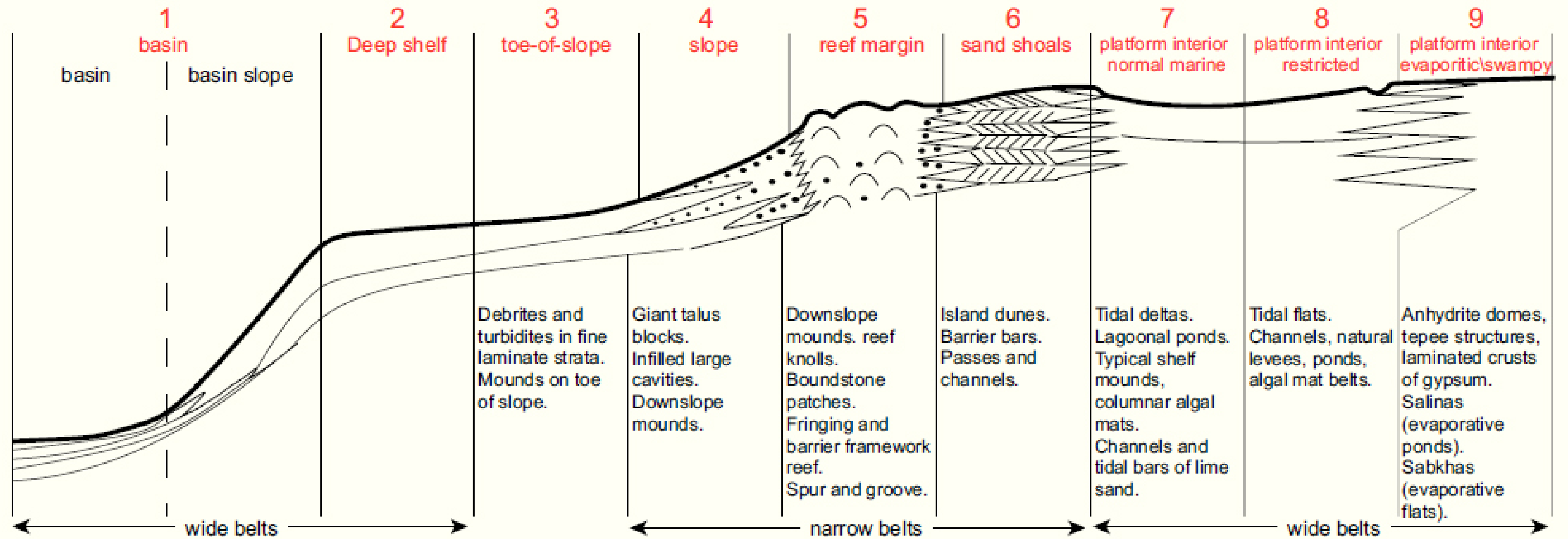


Fig. 4.9 Non-geniculate calcareous red algae. (a) Encrusting Pleistocene aeolianite, Robe, S.A. Depth 1 m, finger for scale, (b) fruticose rhodolith, Spencer Gulf, 2 mwd, (c) fruticose

rhodolith, attached to *Amphibolis* stem (coin scale 2 cm in diameter, Spencer Gulf), (d) rhodoliths with a smooth nodular growth form; cm scale, Great Australian Bight, Baxter Sector, 46 mwd

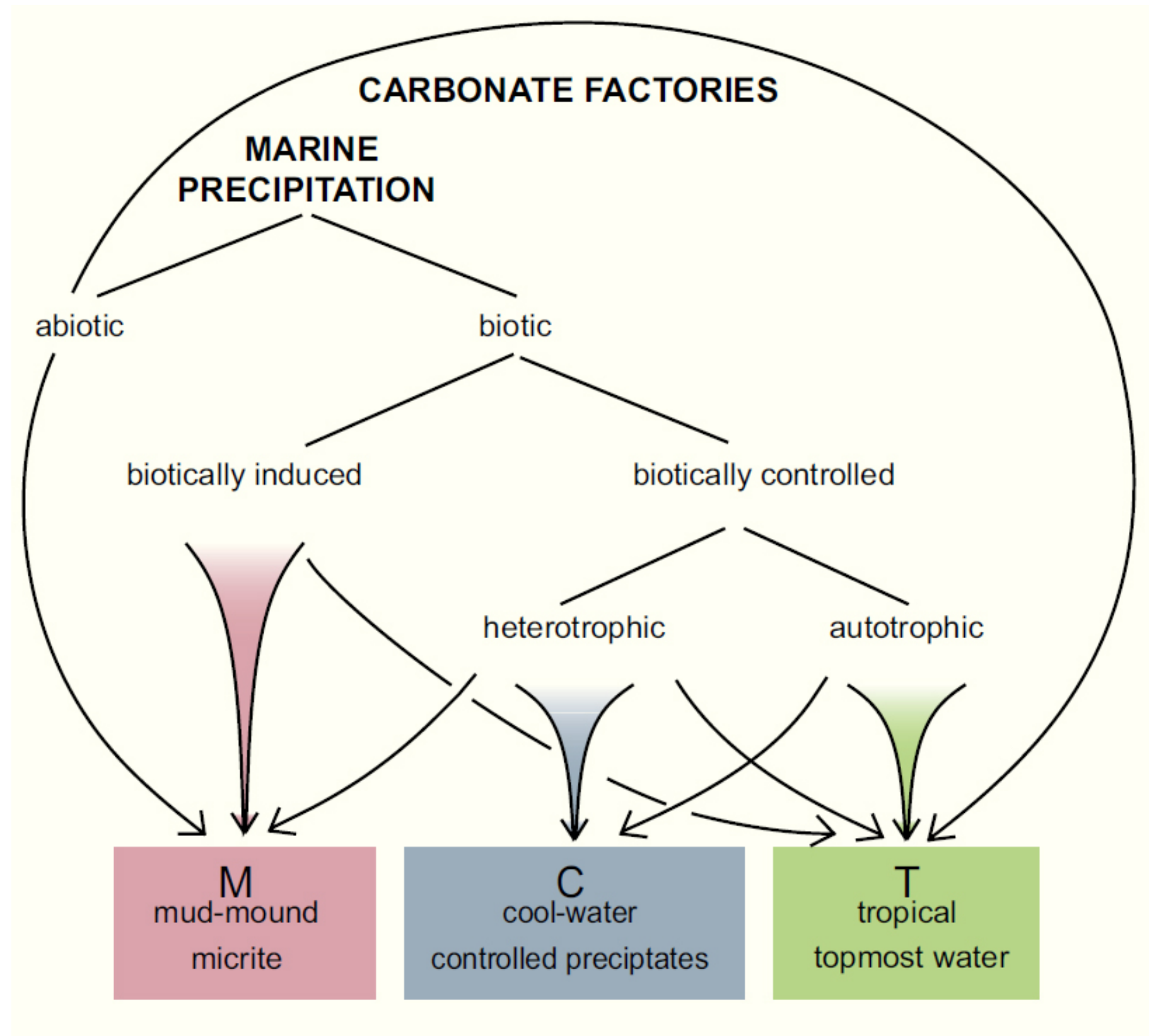


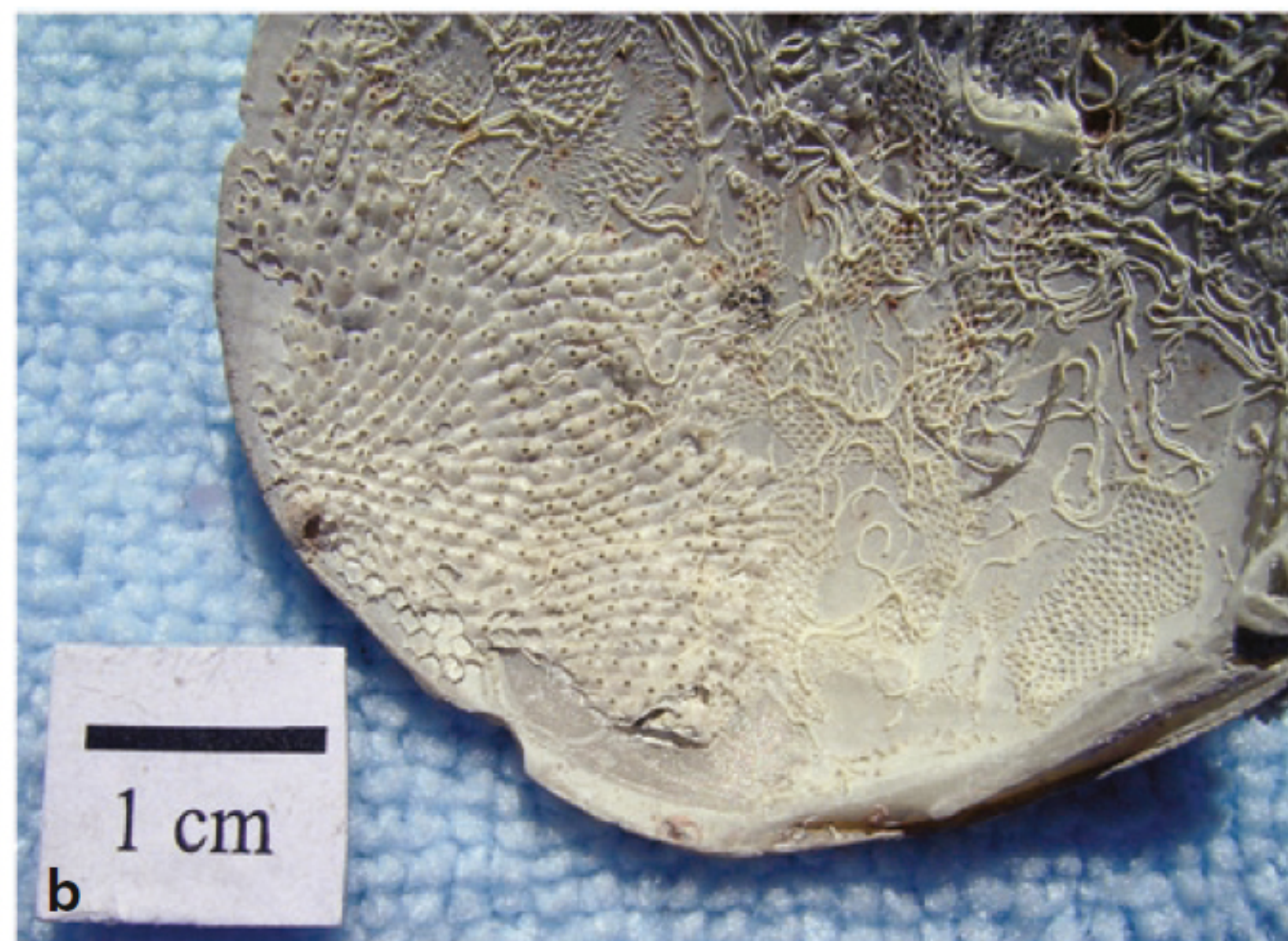
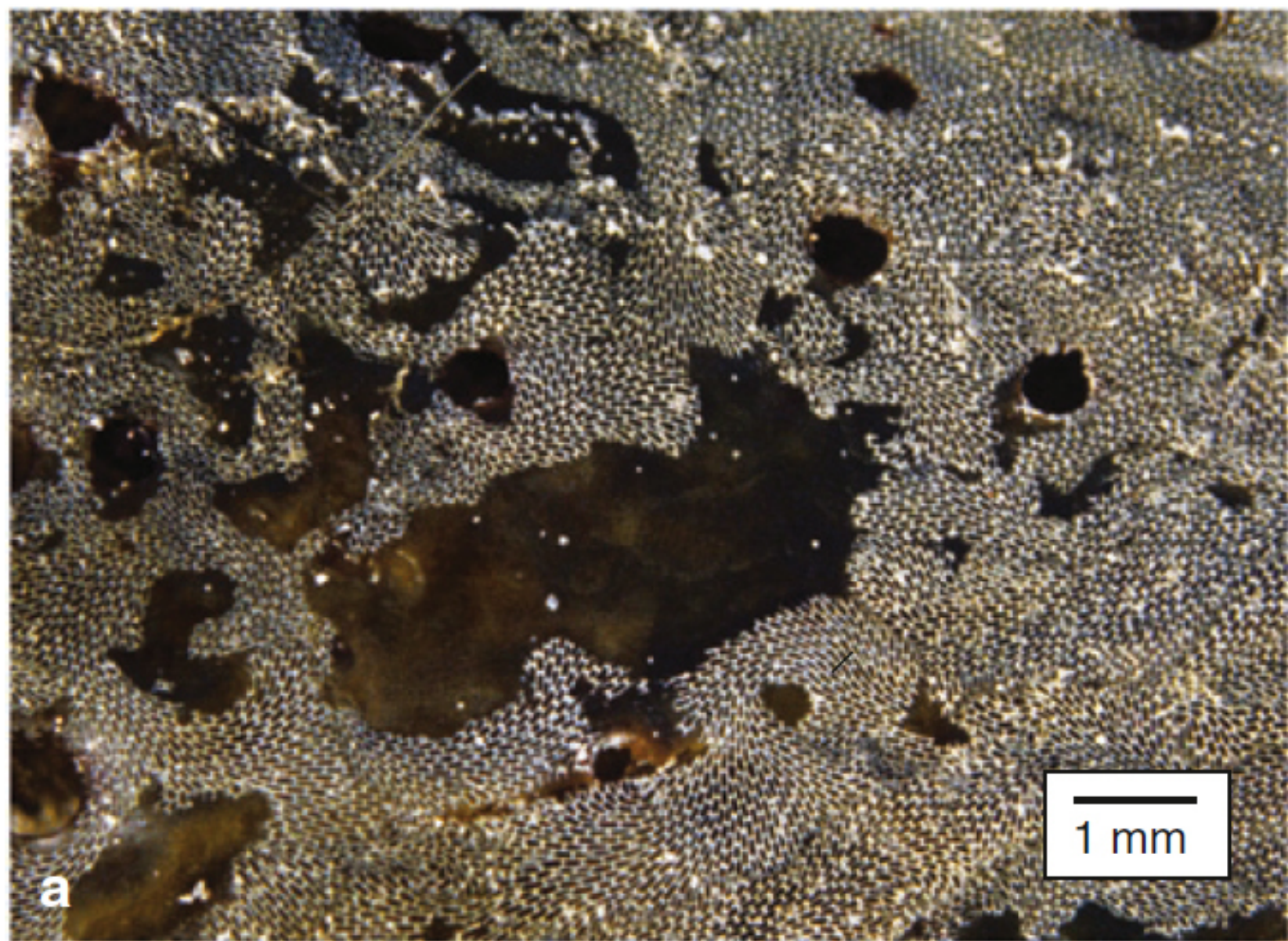




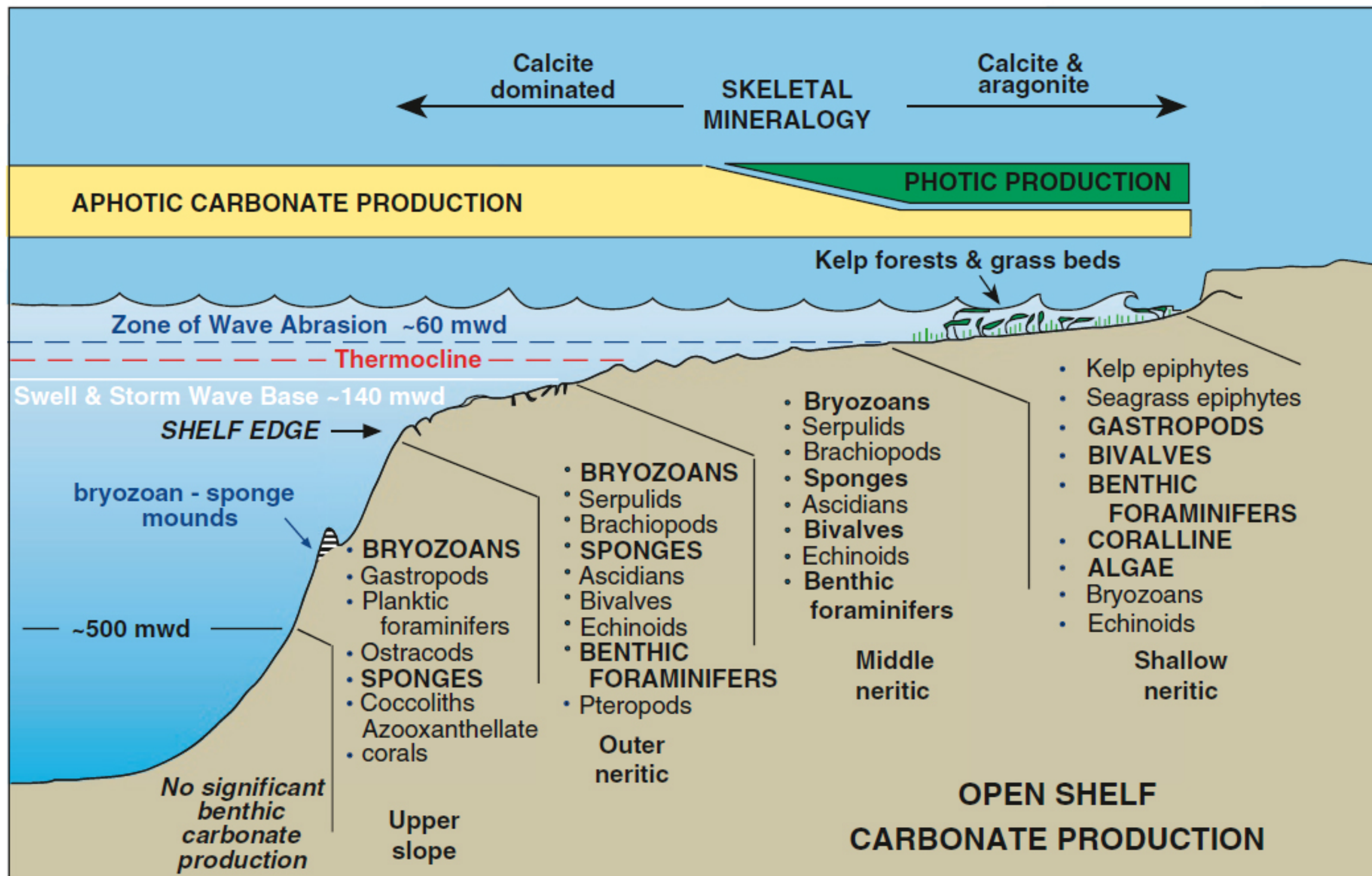
World Reef Map <https://maps.lof.org/lof>



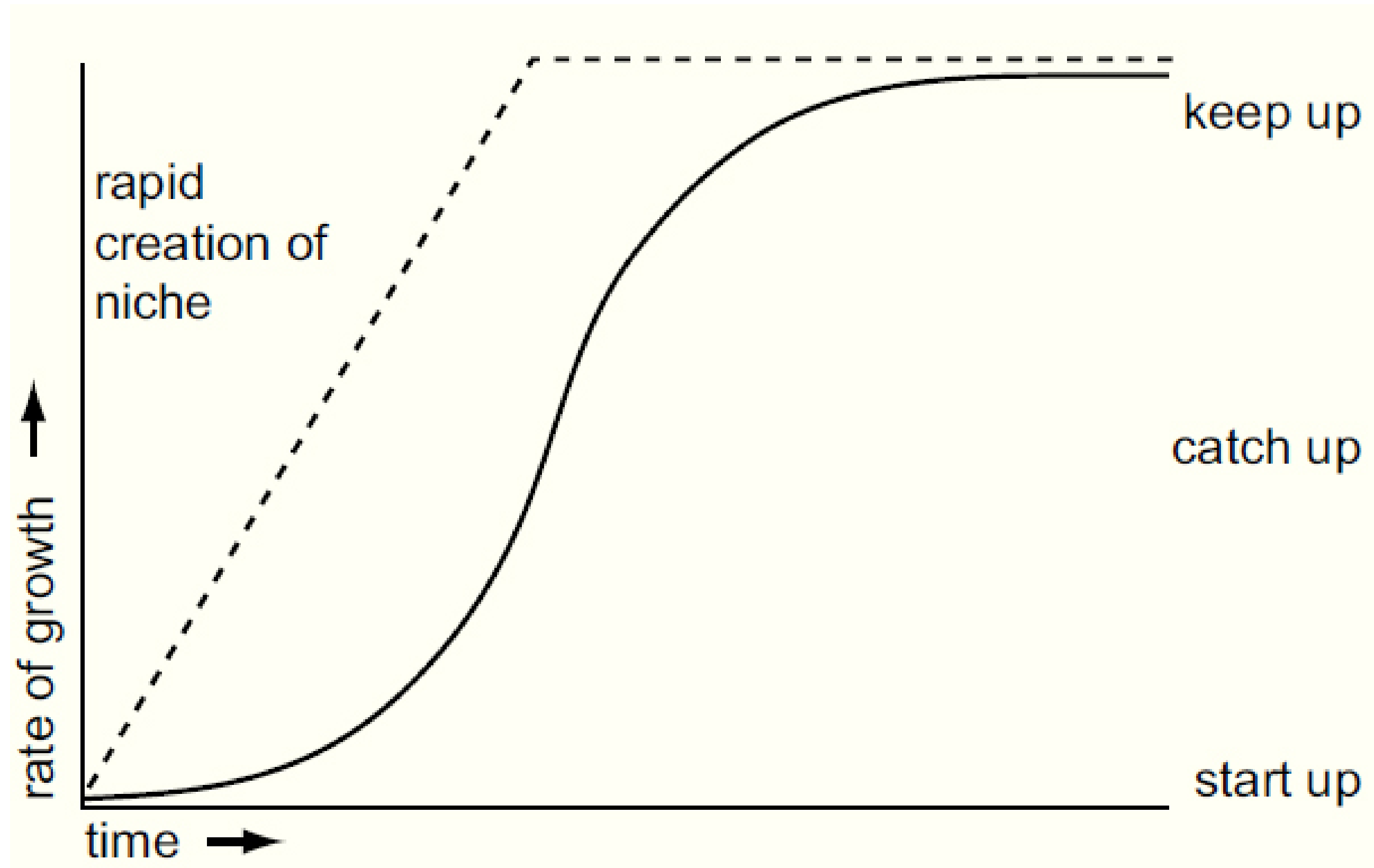


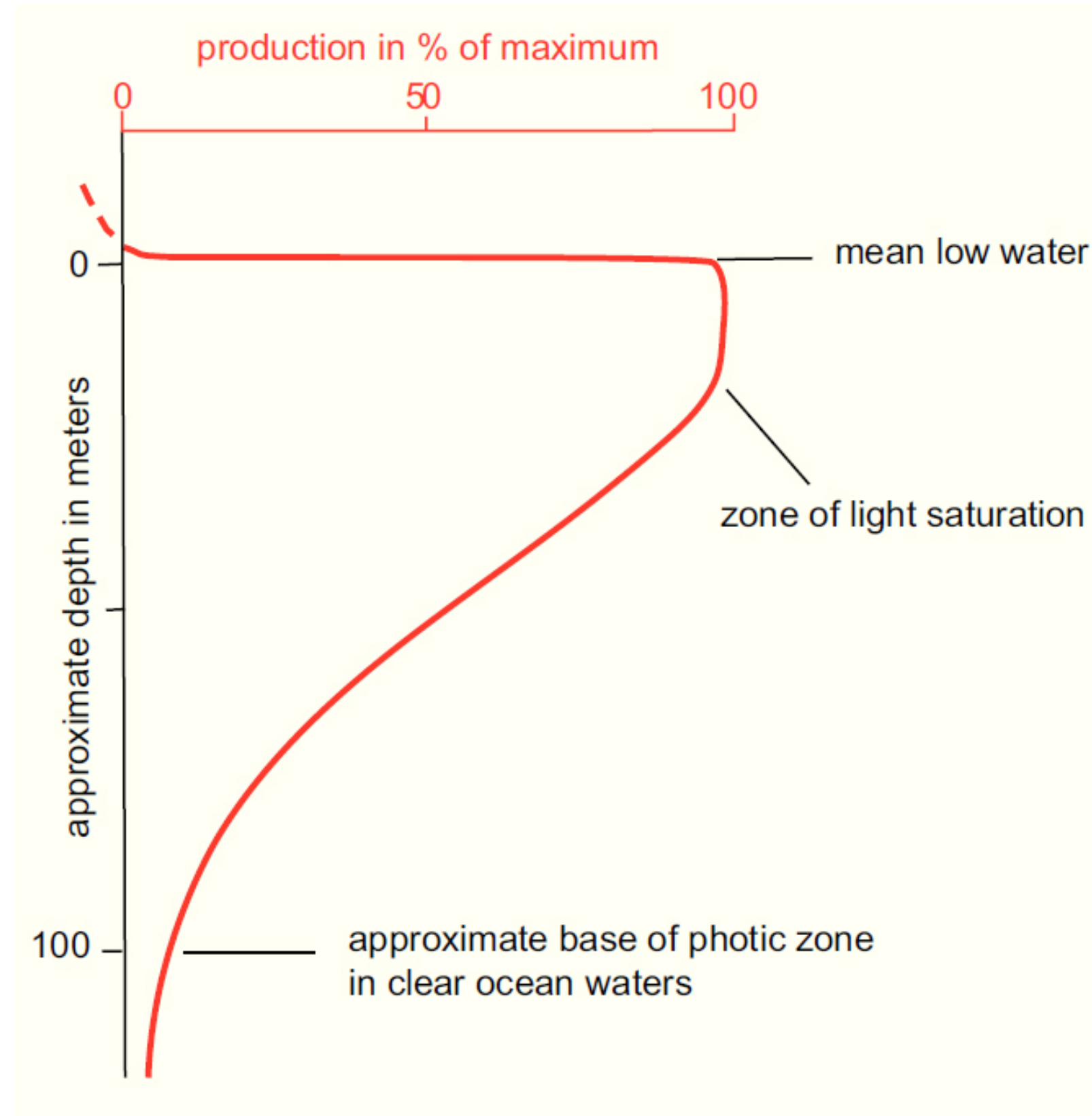




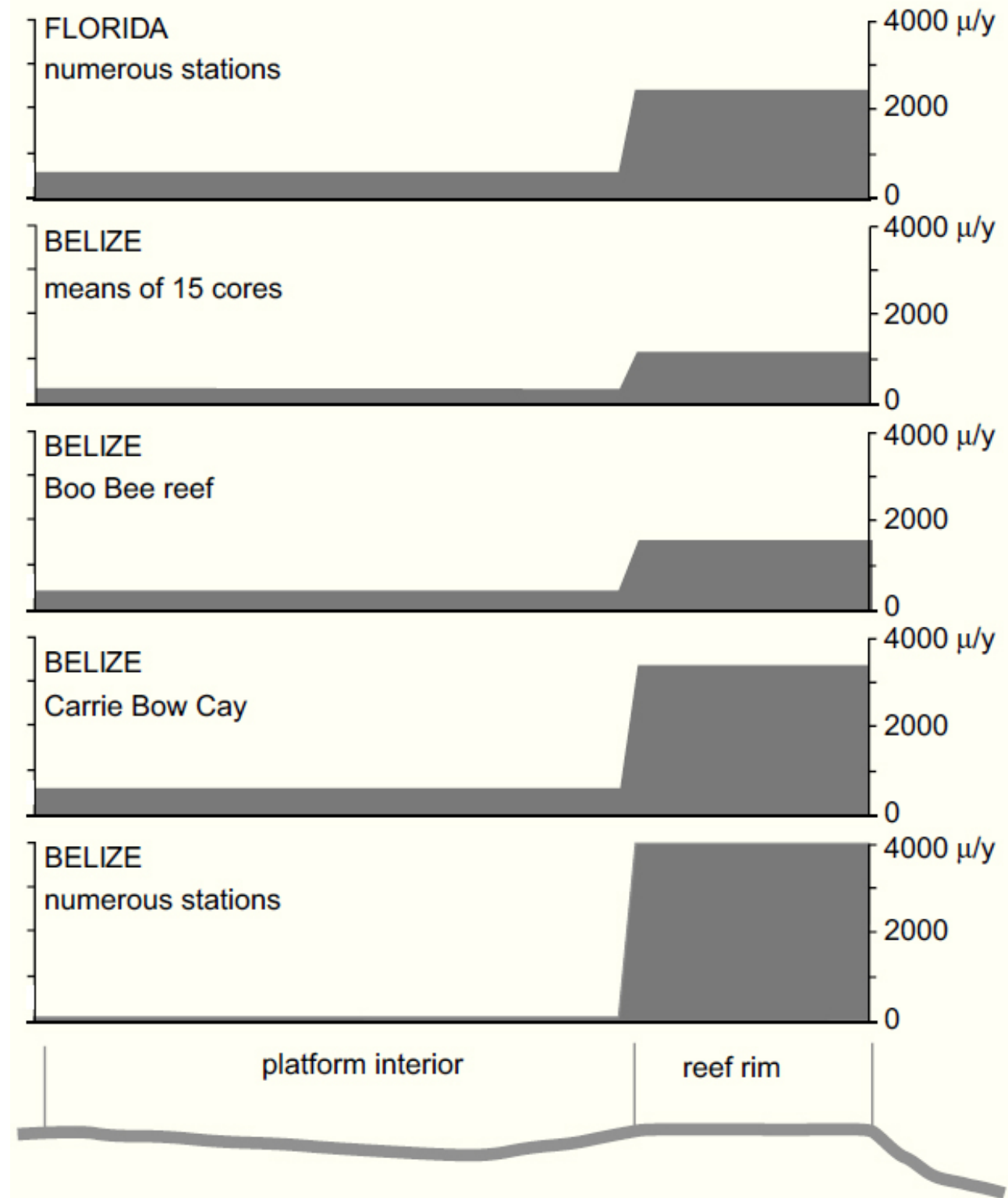






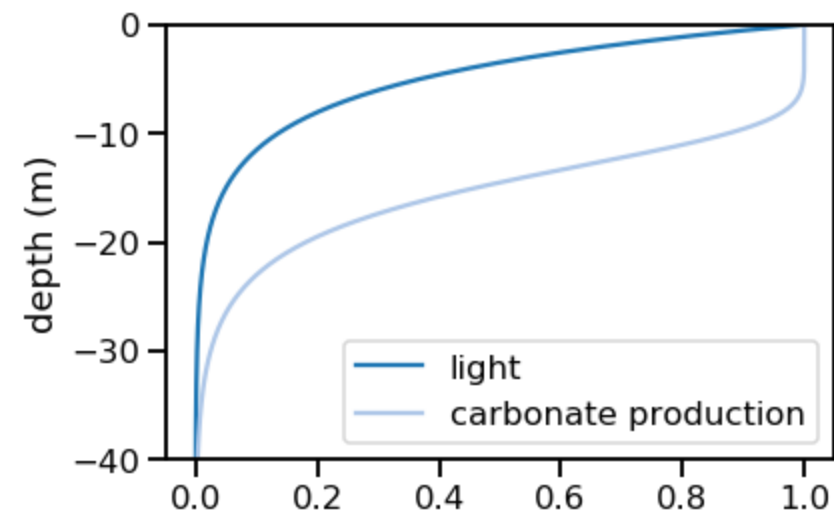


CARBONATE PRODUCTION



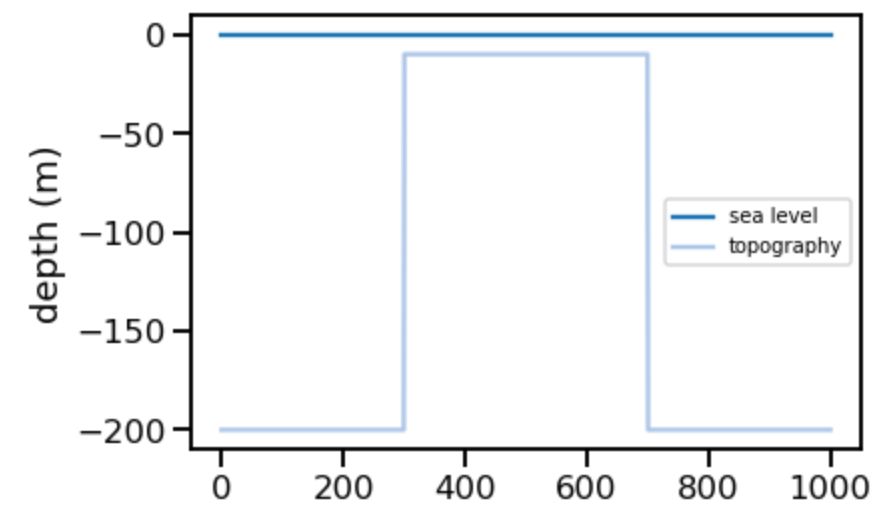
```
In [245]: import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
dz = np.linspace(0,100,1000)
extinction_coeff = 0.2
light_z = np.exp(-extinction_coeff*dz)
plt.plot(light_z,-dz,label='light')
light_base=.1
production_capacity = 1
production_z = production_capacity * np.tanh(light_z/light_base)
plt.plot(production_z,-dz,label='carbonate production')
plt.gca().set_ylim([-40,0]); plt.legend(loc='best'); plt.gca().set_ylabel('depth (m)')
```

Out[245]: Text(0, 0.5, 'depth (m)')



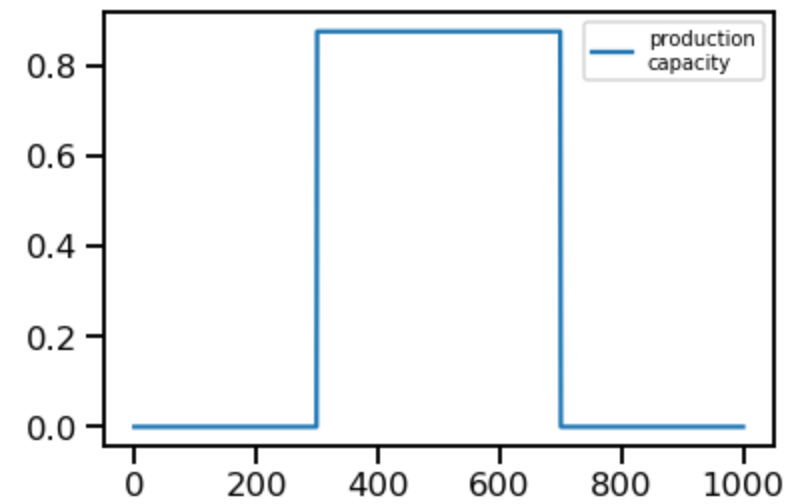
```
In [247]: plt.plot([0,1000],[0,0],label='sea level')
dx = np.linspace(0,1000,1000)
topography = np.zeros(1000)-200
topography[dx>300]=-10
topography[dx>700]=-200
plt.plot(dx,topography,label='topography')
plt.gca().set_ylabel('depth (m)');plt.legend(loc='best',fontsize=10)
```

Out[247]: <matplotlib.legend.Legend at 0x7f28c2d73510>



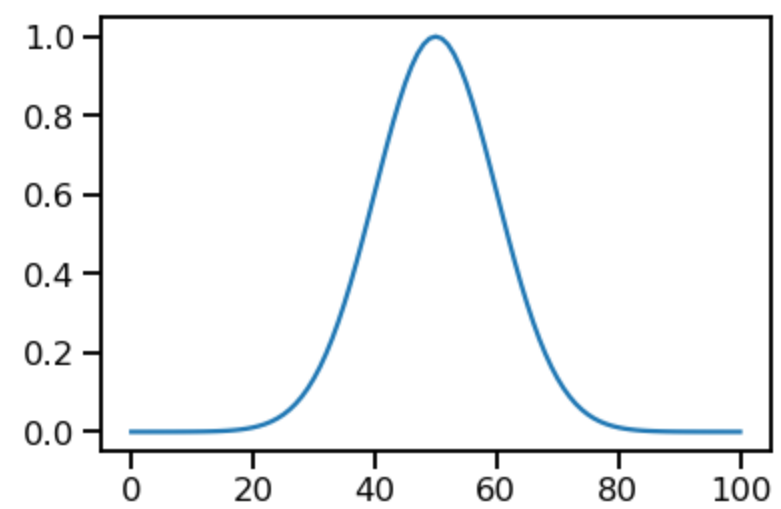
```
In [248]: plt.plot(dx,prod_func(-topography),label='production\ncapacity')  
plt.legend(loc='best',fontsize=10)
```

Out[248]: <matplotlib.legend.Legend at 0x7f28c19abc10>



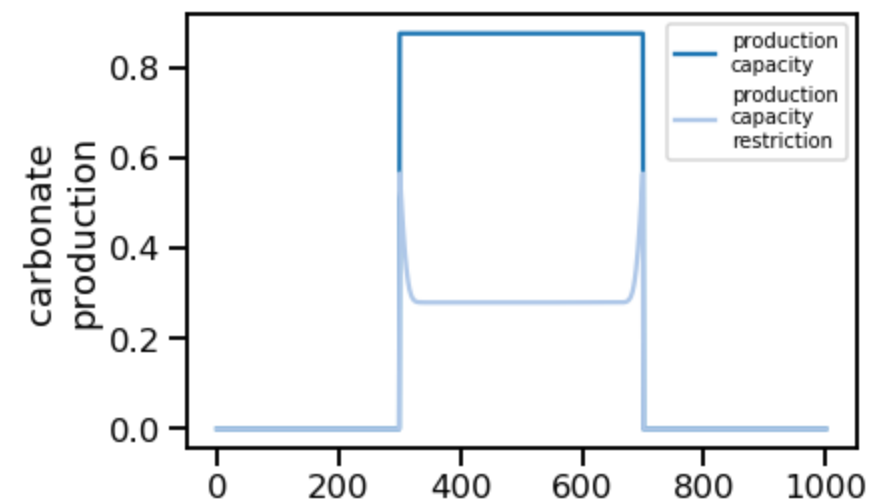
```
In [249]: from scipy.signal import gaussian  
window_size = 50  
standard_deviation = 10  
plt.plot(gaussian(1+2*window_size,standard_deviation))
```

Out[249]: [<matplotlib.lines.Line2D at 0x7f28c2db5510>]



```
In [252]: restriction = np.convolve(topography, gaussian(1+2*window_size, standard_deviation), mode='valid')
restriction -= np.min(restriction)
restriction /= 7000 #multiply to scale shelf production
productivity = prod_func(-topography)
plt.plot(dx, productivity, label='production\ncapacity')
productivity[window_size:-window_size] = productivity[window_size:-window_size] * (1 - restriction)
plt.plot(dx, productivity, label='production\ncapacity\nrestriction')
plt.legend(loc='best', fontsize=10); plt.gca().set_ylabel('carbonate\nproduction')
```

Out[252]: Text(0, 0.5, 'carbonate\nproduction')



In [253]:

```
r = 1  
capacity=2  
N=.5  
growth_rate = r*N*(1-N/capacity)
```



```
In [270]: dt = 1 # timestep
base_level_rise = 100 # long term subsidence in meters
dx = 10 # x grid spacing
total_time = 3e6 # duration of simulation in years
initial_baselevel = 1 # in meters
sed_Q = 0.0 # sedimentation flux

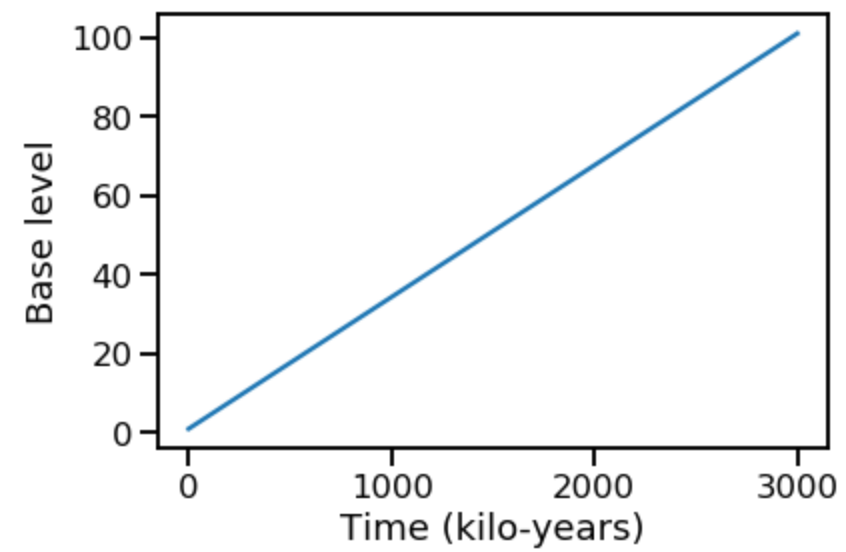
#create an instance of Diffuse1D (defined above)
model = Diffuse1D(
    length=10000,
    spacing=dx,
    timestep=dt,
    left=0,
    right=0,
    K=2e-2,
    sed_Q=sed_Q,
    no_flux_boundary=True,
)

xt = np.linspace(0, total_time, 10000) # creating uniform timegrid
RSL1 = -1.5 * np.sin(xt * 2 * np.pi * (1 / 20000)) # cyclic sea level component 1
RSL2 = 1 * sawtooth_wave(5, xt * 2 * np.pi * (1 / 100000)) # cyclic sea level component 2
RSL = (base_level_rise / (total_time) * xt + initial_baselevel) # cyclic sea level + subsidence
# creates a function in the Diffuse1D model mapping your sea level boundary condition to time
model.set_baselevel(xt, RSL)
```



```
In [271]: #to plot your model sea level boundary condition  
plt.plot(xt/1000,model.base_level_fun(xt))  
plt.gca().set_xlabel('Time (kilo-years)')  
plt.gca().set_ylabel('Base level')
```

Out[271]: Text(0, 0.5, 'Base level')



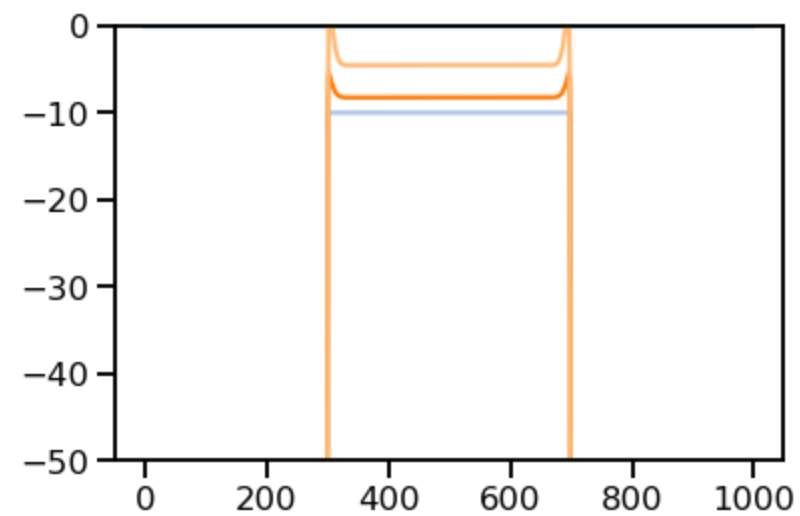
```
In [272]: plt.plot(model.u)
model.u=topography
plt.plot(model.u)
for i in tqdm(range(1000)):
    model.run_step()
plt.plot(model.u)

for i in tqdm(range(2000)):
    model.run_step()
plt.plot(model.u)
plt.gca().set_ylim([-50,0])
```

```
HBox(children=(FloatProgress(value=0.0, max=1000.0), HTML(value='')))
```

```
HBox(children=(FloatProgress(value=0.0, max=2000.0), HTML(value='')))
```

```
Out[272]: (-50, 0)
```



```
In [302]: dt = 10 # timestep
base_level_rise = 60 # long term subsidence in meters
dx = 10 # x grid spacing
total_time = 1e6 # duration of simulation in years
initial_baselevel = 1 # in meters
sed_Q = 0.0 # sedimentation flux

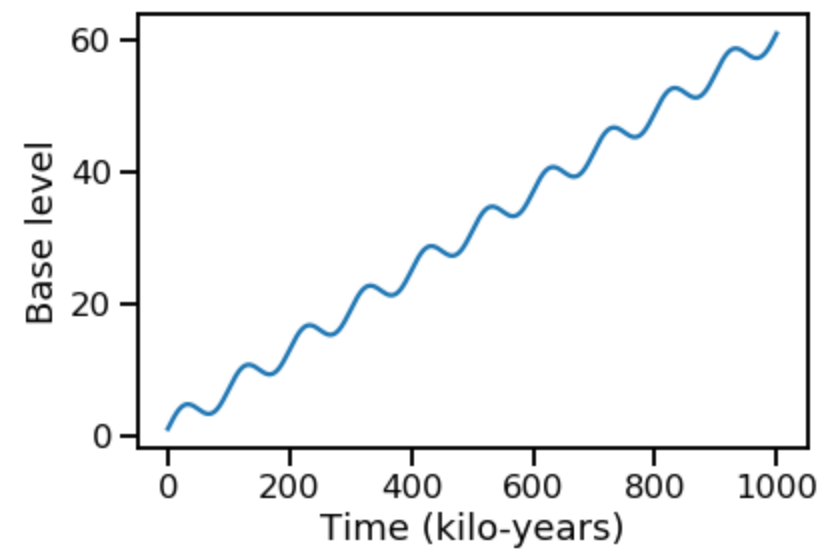
#create an instance of Diffuse1D (defined above)
model = Diffuse1D(
    length=10000,
    spacing=dx,
    timestep=dt,
    left=0,
    right=0,
    K=2e-2,
    sed_Q=sed_Q,
    no_flux_boundary=True,
)

xt = np.linspace(0, total_time, 10000) # creating uniform timegrid
RSL1 = 2 * np.sin(xt * 2 * np.pi * (1 / 1e5)) # cyclic sea level component 1
RSL = (base_level_rise / (total_time) * xt + initial_baselevel + RSL1) # cyclic sea level + subsidence
# creates a function in the Diffuse1D model mapping your sea level boundary condition to time
model.set_baselevel(xt, RSL)
model.u=topography
model.carb_rate = .0015
```



```
In [303]: #to plot your model sea level boundary condition  
plt.plot(xt/1000,model.base_level_fun(xt))  
plt.gca().set_xlabel('Time (kilo-years)')  
plt.gca().set_ylabel('Base level')
```

Out[303]: Text(0, 0.5, 'Base level')

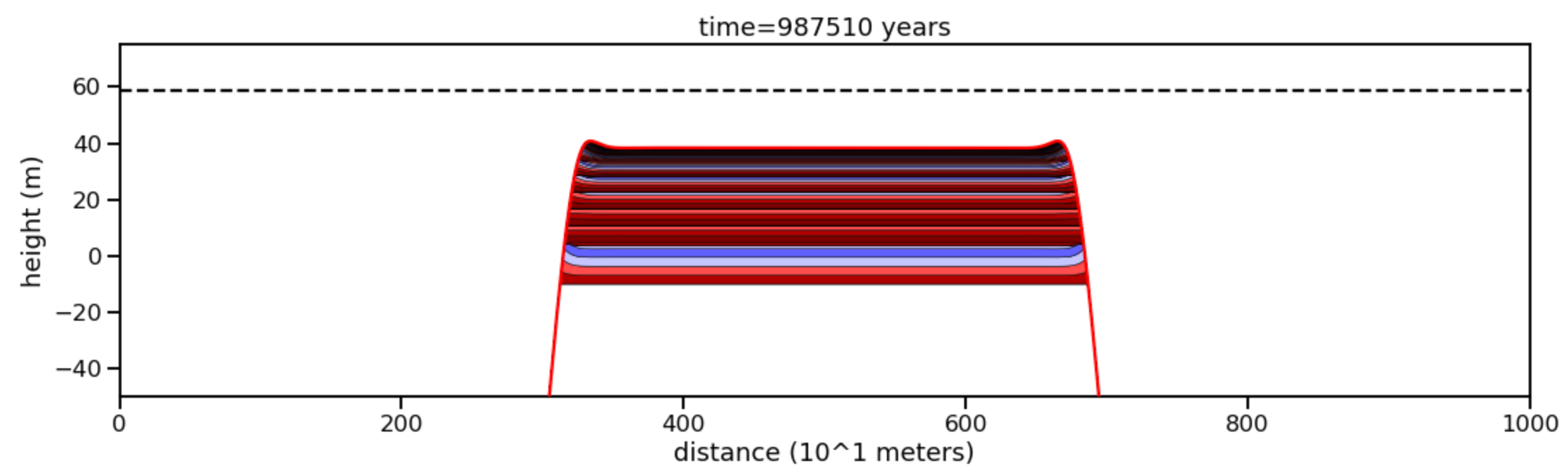


```
In [304]: # initial lists to store model outputs throughout the simulation
beds = [] #model topography
age = [] #model time
rsl = [] #relative (local) sea level
sed_on = True
progress_bar = tqdm(range(int(total_time / dt / 1))) #run the model for the full duration, the tqdm wrapper provides a progress bar
for i in progress_bar:
    model.run_step() #run 1 timestep dt
    if i%50==0: #save every 50 steps to our lists
        beds.append(model.u)
        age.append(model.time)
        rsl.append(model.base_level)
```

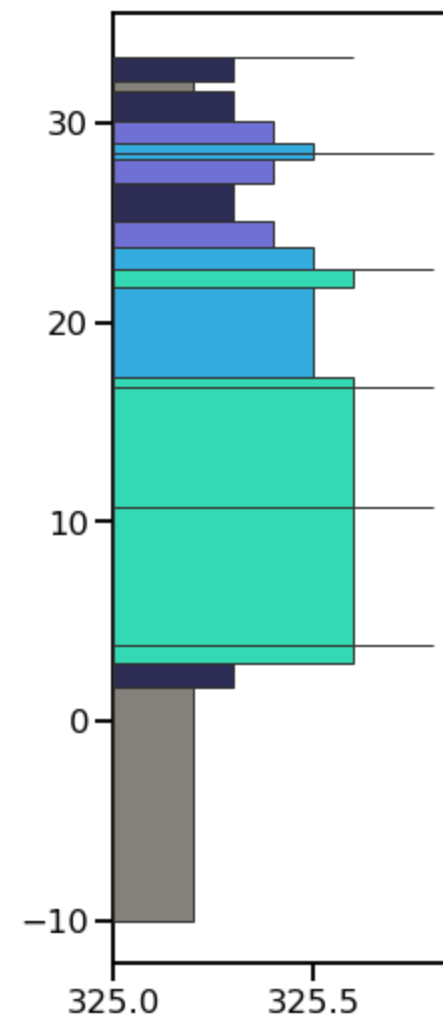
```
HBox(children=(FloatProgress(value=0.0, max=100000.0), HTML(value='')))
```



```
In [321]: skip=25  
animate_beds(beds=beds[::skip],otime=age[::skip],rsl=rsl[::skip],aspect=2, ymin=-50, ymax=75, color=False)
```



```
In [319]: column_number = 325
bed_facies, bed_bottom, bed_thickness, bed_colors = get_strat_column(beds, age, rsl, column_number, skip=10)
plot_column(bed_facies, bed_bottom, bed_thickness, bed_colors, left=column_number)
plt.gca().set_aspect(.05)
```



In [320]:

```
column_number = 500  
bed_facies, bed_bottom, bed_thickness, bed_colors = get_strat_column(beds, age, rsl, column_number, skip=10)  
plot_column(bed_facies, bed_bottom, bed_thickness, bed_colors, left=column_number)  
plt.gca().set_aspect(.05)
```

