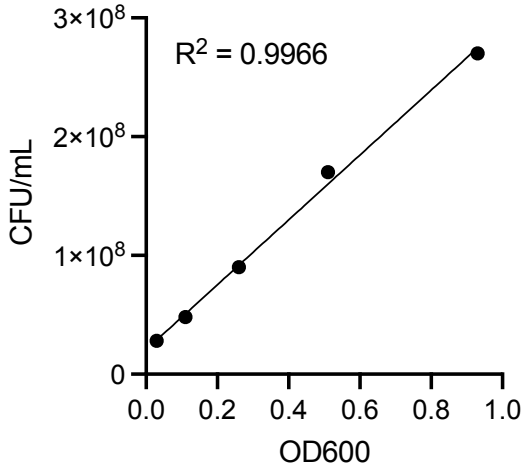
nature portfolio

https://doi.org/10.1038/s41596-022-00723-5

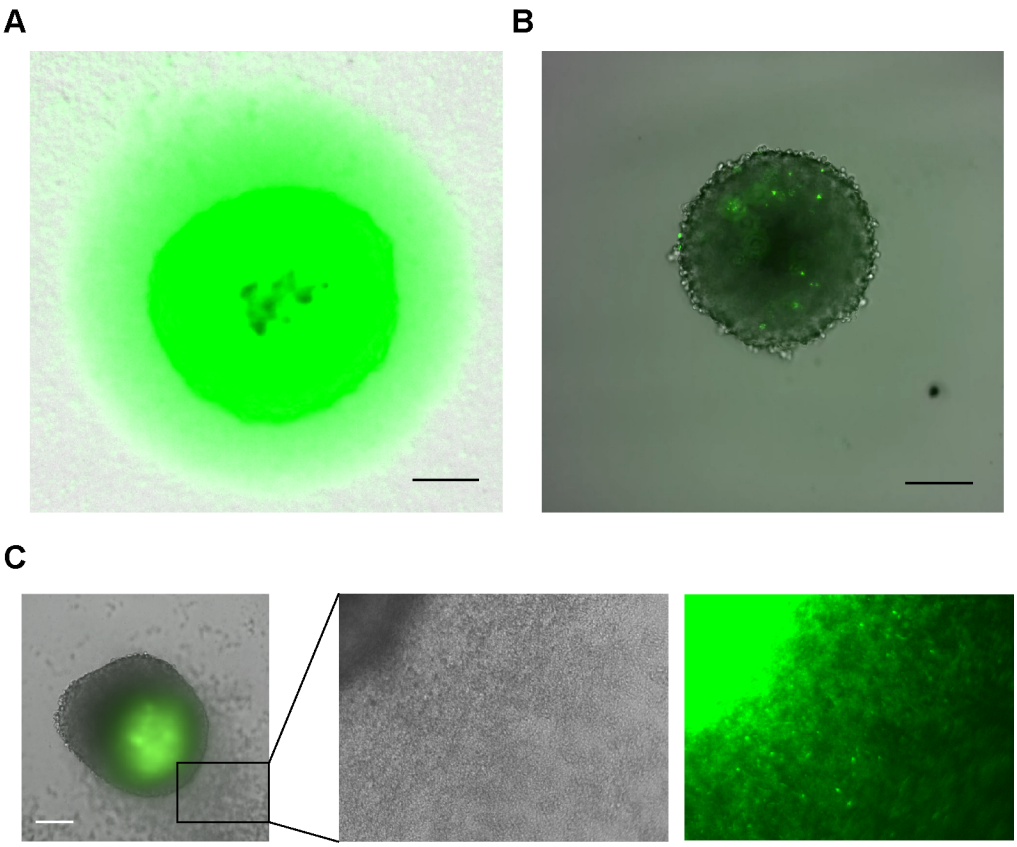
Supplementary information

A rapid screening platform to coculture bacteria within tumor spheroids

In the format provided by the authors and unedited



Supplementary Figure 1. OD600 to CFU/mL conversion. Analysis of correlation between OD600 values and CFU/mL of *S. typhimurium*. Line represents linear regression ($R^2 = 0.9966$).



Supplementary Figure 2. Bacteria spheroid coculture (BSCC) optimization. (A) Representative image of the coculture that failed to contain bacteria within CT26 tumor spheroids due to low gentamicin concentration in the media, leading to overgrowth of GFP-expressing *S. typhimurium* in the media. Scale bar, 100 μm. **(B)** Representative image of unsuccessful bacteria colonization of the tumor spheroid due to excessive amount of gentamicin in the media, leading to eradication of the bacteria within the tumor spheroid. Scale bar, 100 μm. **(C)** Representative image of bacteria escaping from within the tumor spheroid after colonization due to lack of gentamicin in the media during the course of coculture maintenance. Scale bar, 100 μm. Zoomed images (2.7x) show the area of GFP-expressing *S. typhimurium* leakage in brightfield and fluorescence channels.

In [1]:

```
import glob
import os

from scipy.misc import imresize
import scipy.ndimage as ndi
from imageio import imread
from scipy.ndimage.filters import gaussian_filter
from scipy.ndimage.morphology import grey_closing, grey_opening, distance_transf
from scipy.signal import medfilt
from skimage.measure import label, regionprops
from skimage.filters import threshold_otsu, threshold_minimum, threshold_local,
from skimage.segmentation import clear_border

from imageTools import absGradient, scaleDown

import pickle
import  time

import cv2 as cv

import numpy as np

import matplotlib as mpl
mpl.rcParams['pdf.fonttype'] = 42
import matplotlib.pyplot as plt
```

In [2]:

```
ALLDIR = glob.glob('../*')


INDIR = '../A1/'


bClearBorder = True
RMAX_SCAN = 600
RN_SCAN = 300


 bAltFormat = True

 OUTDIR =
 INDIR.replace('../','')
 OUTDIR = OUTDIR.replace('
 ','_')

 if not
     os.path.isdir(OUTDIR):
     os.system('mkdir ' +
     OUTDIR)
```

In [3]:
```
if (bAltFormat):

    FILES1 = glob.glob(INDIR + "*c1.*")
    FILES1 = sorted(FILES1)

    FILES2 = glob.glob(INDIR + "*c2.*")
    FILES2 = sorted(FILES2)

else:

    FILES1 = glob.glob(INDIR + "data/*c001.*")
    FILES1 = sorted(FILES1)

    FILES2 = glob.glob(INDIR + "data/*c002.*")
    FILES2 = sorted(FILES2)

print(FILES1)
print(FILES2)
```

In [4]:
```
print(len(FILES1))
print(len(FILES2))
```

In [5]:
```
def makeMask(img, bOtsu=False):

    if (bOtsu):
        imgmask  =  (img>TWEAK*threshold_otsu(img))*1
    else:
        imgmask  =  (img>TWEAK*threshold_yen(img))*1

    if (bClearBorder):
        imgmask  =  1-clear_border(1-imgmask)

    return imgmask
```

In [6]:

```
PLOTDIR = OUTDIR + 'debugPlots/'
if not os.path.isdir(PLOTDIR):
    os.system('mkdir ' + PLOTDIR)


def  visMask(ijk):

    print(ijk)

    img1  =  imread(FILES1[ijk])
    img2  =  imread(FILES2[ijk])

    img1mask = makeMask(img1)



    dirmask = OUTDIR + 'mask/'
    if (not os.path.isdir(dirmask)):
        os.system('mkdir ' + dirmask)
    fnamemask1_out  =  dirmask  +  (FILES1[ijk].split('/')[-1])
    cv.imwrite(fnamemask1_out,  img1mask.astype(np.uint16))



    img1mask_display  =  np.ma.masked_where(img1mask,img1)img1maskedt
    = distance_transform_edt(1-img1mask)

    img1mask_otsu = makeMask(img1, bOtsu=True) img1mask_otsu_display
    =  np.ma.masked_where(img1mask_otsu,img1)


    plt.close('all')
    plt.figure(figsize=(12,10))

    plt.subplot('221')
    plt.imshow(img1,  'gray')
    plt.title('tl')

    plt.subplot('222')
    plt.imshow(img1,  'gray')
    plt.imshow(img1mask_display,  'hot')
    plt.title('threshold_yen')

    plt.subplot('223')
    plt.imshow(img1maskedt, 'hot')
    plt.title('distance  transform')
```

```
    plt.subplot('224')
    plt.imshow(img1, 'gray')
    plt.imshow(img1mask_otsu_display, 'hot')
    plt.title('threshold_otsu')

    plt.savefig(PLOTDIR + ('mask_%04d.png' % ijk))

    return img1mask
```

In [7]:
```
JUNK = visMask(9)
```

In [8]:
```
pklfile = OUTDIR + 'RES.pkl'
print(pklfile)

if (not os.path.isfile(pklfile)):
    mask = []
    for ijk, fname1 in enumerate(FILES1):
        mask.append(visMask(ijk))
    mask = 1-np.array(mask)
```

```
      A1/RES.pkl
```

In [9]:
```
def maxAreaLabel(mask1_label):
    maxArea = 0
    maxLabel = -1
    maxProps = None
    for region in regionprops(mask1_label):
        if (maxArea<region.area):
            maxArea = region.area
            maxLabel = region.label
            maxProps = region
    return (mask1_label==maxLabel)*1, maxProps
```

In [10]:
```
if (False):

    ijk = 0
    img1 = imread(FILES1[ijk])
    img2 = imread(FILES2[ijk])

    mask1 = np.squeeze(mask[ijk,:,:])

    mask1_label = label(mask1)
    mask1_label, maxProps = maxAreaLabel(mask1_label)

    img1_mask = np.ma.masked_where(mask1_label==0,img1)
    img2_mask = np.ma.masked_where(mask1_label==0,img2)

    print(maxProps.centroid)

    plt.imshow(img1_mask)
```

In [11]:
```
def makeRCoords(mask1_label, centroid):
    SZ = mask1_label.shape
    YY = np.arange(SZ[0])
    XX = np.arange(SZ[1])
    XX = np.outer(np.ones(SZ[0]),XX)
    YY = np.outer(YY,np.ones(SZ[1]))

    print(YY)
    print(XX)
    print(np.sqrt((XX-centroid[1])**2+(YY-centroid[0])**2))

    return  np.sqrt((XX-centroid[1])**2+(YY-centroid[0])**2)
```

In [12]:
```
def returnImageHist(mask1_label, maxProps, img):

    Rimg1 = makeRCoords(mask1_label, maxProps.centroid)

    RMAX = RMAX_SCAN
    RN = RN_SCAN
    dR = RMAX/ (RN-1)



    spanR1 = np.linspace(0,RMAX,RN) spanR2
    = np.linspace(dR,RMAX+dR,RN)
    measurementsR = []
    measurementsY = []
    for xyz, val in enumerate(spanR1):
        r1 = spanR1[xyz]


        r2 = spanR2[xyz]

        meas1 = regionprops(((Rimg1>=r1)*(Rimg1<r2))*1, (Rimg1>=r1)*(Rimg1<r2)*i
        for region in meas1:
            measurementsR.append(r1)
            measurementsY.append(region.mean_intensity)

    return (measurementsR,measurementsY)
```

In [13]:
```
def extract1(parms1):

    ijk,fname1,fname2 = parms1
    print(fname1,fname2)

    img1 = imread(fname1)
    img2 = imread(fname2)

    img2 = img2 - np.percentile(img2,1)

    mask1 = np.squeeze(mask[ijk,:,:])

    mask1_label = label(mask1)
    mask1_label, maxProps = maxAreaLabel(mask1_label)

    print(ijk, maxProps.centroid)

    xyimg = makeRCoords(mask1_label, maxProps.centroid)

    stats1 = returnImageHist(mask1_label, maxProps, img1)
    stats2 = returnImageHist(mask1_label, maxProps, img2*mask1_label)
    statsmask = returnImageHist(mask1_label, maxProps, 1-mask1_label)

    return((stats1,stats2,statsmask))
```

In [14]:
```
if (os.path.isfile(pklfile)):
    print('loading results!')
    RES = pickle.load(open(pklfile,'rb'))
else:
    print('computing results!')
    PARMS = []
    for ijk, fname1 in enumerate(FILES1):
        fname1 = FILES1[ijk]
        fname2 = FILES2[ijk]
        PARMS.append((ijk,fname1,fname2))

    from multiprocessing import Pool
    pool = Pool()
    RES = pool.map(extract1,PARMS)
    pool.close()

    with open(pklfile,'wb') as fd:
        pickle.dump(RES,fd,protocol=pickle.HIGHEST_PROTOCOL)
```

In [15]:
```
YPLOT1 = []
YPLOT2 = []
YPLOTMASK = []
for ijk, val in enumerate(RES):
    x1,x2,xmask = val
    XVALS = x1[0]
    YPLOT1.append(x1[1])
    YPLOT2.append(x2[1])
    YPLOTMASK.append(xmask[1])
YPLOT1 = np.array(YPLOT1)
YPLOT2 = np.array(YPLOT2)
YPLOTMASK = np.array(YPLOTMASK)
```

In [16]:
```
print(YPLOT1.shape)
SZ = YPLOT1.shape
for ijk in range(SZ[0]):
    YPLOT1[ijk,:] = YPLOT1[ijk,:] / np.max(YPLOT1[ijk,:])

PLX = np.array(XVALS)*1
PLY  =  np.linspace(0,len(FILES1),len(FILES1))
print(PLX.shape, PLY.shape, YPLOT1.shape)
```

In [17]:
```
MASKCUTOFF = 0.95

MASKMAX = PLX[np.max(np.where(YPLOTMASK<=MASKCUTOFF)[1])]
print(np.max(np.where(YPLOTMASK<=MASKCUTOFF)[1]), MASKMAX)
plt.imshow(1*(YPLOTMASK.T<=MASKCUTOFF), aspect='auto')
plt.colorbar()
```

In [18]:
```
cdict = {'red':    ((0.0,  0.0, 0.0),
                    (0.5,  0.0, 0.0),
                    (1.0,  1.0, 1.0)),

         'green': ((0.0,  0.0, 0.0),
                   (0.5,  0.7, 0.7),
                   (1.0,  1.0, 1.0)),

         'blue':  ((0.0,  0.0, 0.0),
                   (0.5,  0.0, 0.0),
                   (1.0,  1.0, 1.0))}


cmap  =  mpl.colors.LinearSegmentedColormap('GreenDark',  cdict)
```

In [19]:
```
DT = (20)

T0 = 0
```

In [20]:
```
plt.figure(figsize=(6,2.5))

plt.imshow(YPLOT1.T, extent=[T0,T0+DT*len(FILES1),np.min(XVALS),np.max(XVALS)],
           aspect='auto', \
           origin='lower',  cmap=cmap)
plt.colorbar()

plt.contour(T0+DT*PLY,  PLX,  YPLOTMASK.T,  levels=[0.95,],  colors=['k',])

SLUG = OUTDIR.split('/')[-2]

plt.title(SLUG + ': ' + 'tl') plt.xlabel('time
(hours)') plt.ylabel('distance  from  centroid
(pixels)')

plt.ylim([0,1.1*MASKMAX])

plt.savefig(OUTDIR + SLUG + '_channel1.png')
plt.savefig(OUTDIR + SLUG + '_channel1.pdf')
```

In [21]:
```
YPLOT2_zoom = ndi.zoom(YPLOT2,10,order=0)
plt.imshow(YPLOT2_zoom.T,aspect='auto',origin='bottom')
```

In [22]:
```
um_per_pixel = 2.1812977099

plt.figure(figsize=(10,2.5))

plt.imshow(YPLOT2_zoom.T,
           extent=[T0,T0+DT*len(FILES1),um_per_pixel*np.min(XVALS),um_per_pixel*
           aspect='auto', \
           clim=[np.min(YPLOT2),0.9*np.max(YPLOT2)], \
           origin='lower',
           cmap=cmap,
           interpolation='none',  resample=True)
plt.colorbar()

plt.contourf(T0+DT*PLY,  um_per_pixel*PLX,  YPLOTMASK.T,  levels=[0.95,2],  colors=[
plt.contour(T0+DT*PLY,  um_per_pixel*PLX,  YPLOTMASK.T,  levels=[0.95],  colors=['gr

plt.title(SLUG + ': ' + 'gfp')
plt.xlabel('time (hr)')
plt.ylabel('distance from centroid (um)')

plt.ylim([0,um_per_pixel*1.0*MASKMAX])

plt.xlim([110,120])

plt.tight_layout()
```

In [25]:

```
plt.figure(figsize=(10,2.5))

plt.imshow(YPLOT2_zoom.T,
           extent=[T0,T0+DT*len(FILES1),np.min(XVALS),np.max(XVALS)], \
           aspect='auto', \ \ origin='lower',
           cmap=cmap,
           interpolation='none',  resample=True)
plt.colorbar()

plt.contourf(T0+DT*PLY,  PLX,  YPLOTMASK.T,  levels=[0.95,2],  colors=['w',])
plt.contour(T0+DT*PLY,  PLX,  YPLOTMASK.T,  levels=[0.95],  colors=['gray',])

plt.title(SLUG + ': ' + 'gfp')
plt.xlabel('time (hr)')
plt.ylabel('distance  from  centroid  (pixels)')

plt.ylim([0,1.0*MASKMAX])


plt.tight_layout()

plt.savefig(OUTDIR + SLUG + '_channel2.png')
plt.savefig(OUTDIR + SLUG + '_channel2.pdf')


np.savetxt(OUTDIR + SLUG + 'GFPdata.dat', YPLOT2_zoom.T)
np.savetxt(OUTDIR + SLUG + 'GFPmask_data.dat', YPLOT1.T)
```