[231]:
```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[232]:
```
%matplotlib inline
```

Ejemplo de Patching

A continuación se muestran los resultados de la división en *patches* de una única imagen SVS.

Librerías

[233]:
```
from IPython.display import display

import yaml
import os
import random

import numpy as np
import pandas as pd

from PIL import ImageOps
import openslide
from matplotlib import pyplot as plt

from wsi.slide import thumbnail
from wsi.patch import get_slide_patches_params
from wsi.patch import get_white_pixel_percetange
```

Configuración

[234]:
```
with open('conf/user_conf.yaml', 'r') as f:
    conf = yaml.load(f)
```

[235]:
```
slides_path = os.path.join(conf['data_path'], 'slides', 'svs')
```

Selección aleatoria de una imagen

Se lee aleatoriamente una imagen del directorio de imágenes SVS. Para leer y procesar imágenes SVS se utiliza la librería *openslide*.

[236]:
```
files = os.listdir(slides_path)
file= random.choice(files)
print(file)

os_img = openslide.open_slide(os.path.join(slides_path, file))
```

```
TCGA-HZ-7919-01A-01-BS1.svs
```

Propiedades OpenSlide

Una vez leída la imágen se pueden obtener una miniatura del tamaño que se quiera.

[204]:
```
os_img.get_thumbnail((1000,500))
```

[204]:

Una imagen SVS tiene asociadas también la imagen de la etiqueta de la muestra y una imágen de la lámina al completo (macro).

[135]:
```
os_img.associated_images
```

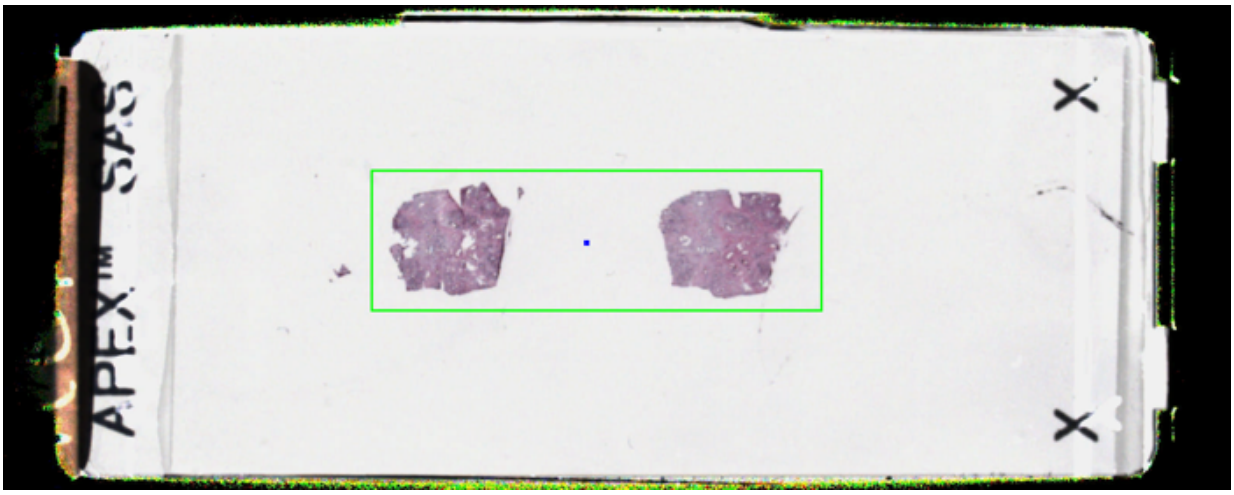
[135]:
```
<_AssociatedImageMap {'label': <PIL.Image.Image image mode=RGBA size=663x652 at
0x1C3F47B860>, 'macro': <PIL.Image.Image image mode=RGBA size=1600x631 at 0x1C42E72240>,
'thumbnail': <PIL.Image.Image image mode=RGBA size=1024x310 at 0x1C42ECF748>}>
```

[136]:
```
label = os_img.associated_images['label']
label.thumbnail((300,300))
label
```

[136]:

[137]:
```
macro = os_img.associated_images['macro']
macro.thumbnail((700,700))
macro
```

[137]:



También podemos obtener el ratio de de submuestreo de en cada uno de los niveles, esto es el atributo *level_downsamples*. Normalmente veremos 3 o 4 niveles y en el primero de ellos el ratio de submuestreo será 1, es decir, la imagen a máxima resolución. Entre cada nivel normalmente el tamaño de la imágen se reduce cuatro veces (1, 4, 16...)

[139]:
```
os_img.level_downsamples
```

[139]:
```
(1.0, 4.000062753629251, 16.001255230125523, 32.005189152996316)
```

Junto con los ratios de submuestreo es posible también ver el tamaño total en píxeles de la imágen en cada nivel.

[53]:
```
os_img.level_dimensions
```

[53]:
```
((114240, 36984), (28560, 9246), (7140, 2311), (3570, 1155))
```

[205]:
```
114240 / 5
```

[205]:
```
22848.0
```

Entre las propiedades de fichero encontramos el aumento del microscopio utilizado para obtener la imágene. Si quisieramos calcular el aumento correspondiente a cada uno de los niveles bastaría con dividirlo por el ratio de submuestreo.

[145]:
```
os_img.properties['aperio.AppMag']
```

[145]:
```
'40'
```

[146]:
```
levels_magnification = [int(os_img.properties['aperio.AppMag'])/int(x) for x in os_img.
↪level_downsamples]
```

```
levels_magnification
```

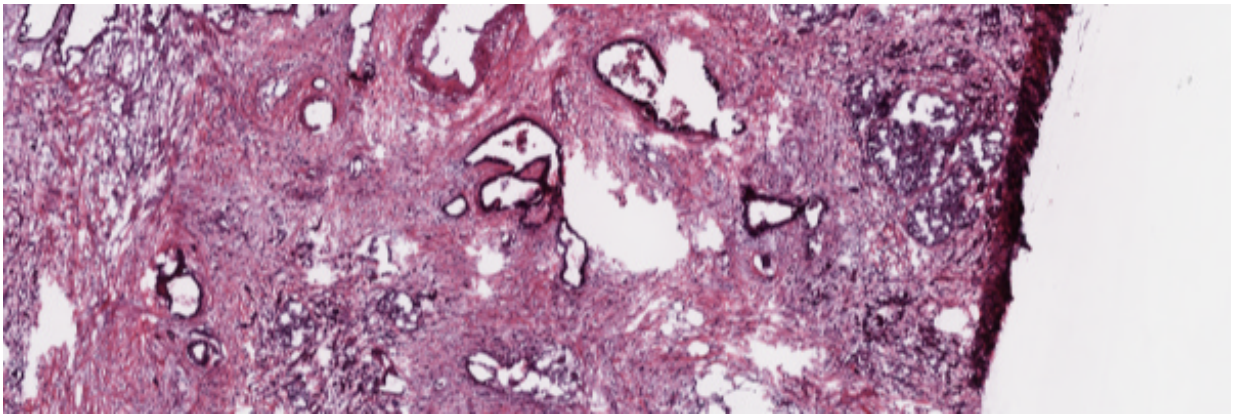
[146]:
```
[40.0, 10.0, 2.5, 1.25]
```

Un método muy útil para la división en patches de una imágen sin tener que cargar en memoria la imagen de un nivel al completo es el *read_region*. Los parámetros son los siguientes: * location: posición del pixel de la esquina superior izquierda en el nivel 0 (máxima resolución). * level: nivel del que obtener la imágen. * size: dimensiones de la imagen a obtener

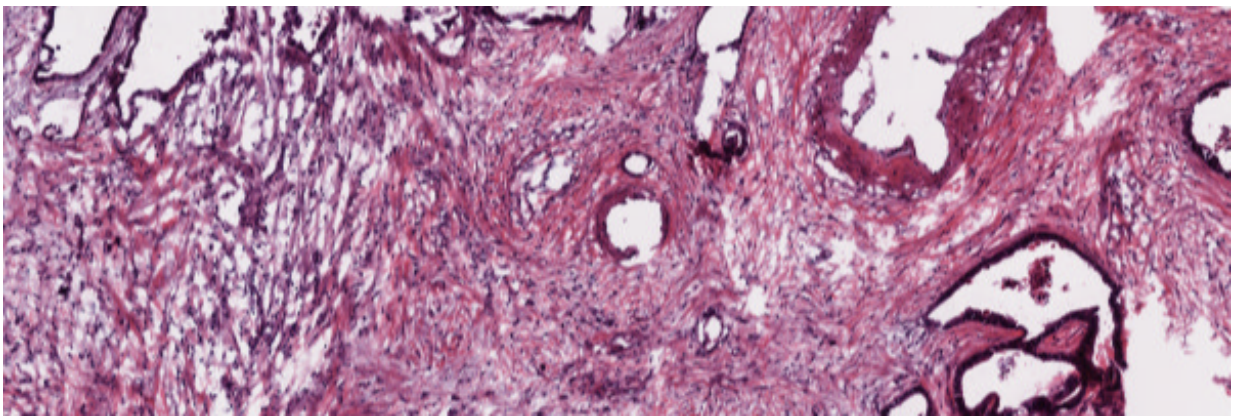
Como se puede ver al elegir un nivel de mayor aumento se está haciendo un zoom sobre la imagen anterior en la esquina superior izquierda.

[216]:
```
for level,mag in list(enumerate(levels_magnification))[::-1]:
    patch = os_img.read_region(location=(12000,16000), level=level, size=(600,200))
    print('Nivel', level, ' Aumento', 'x{}'.format(mag))
    display(patch)
```

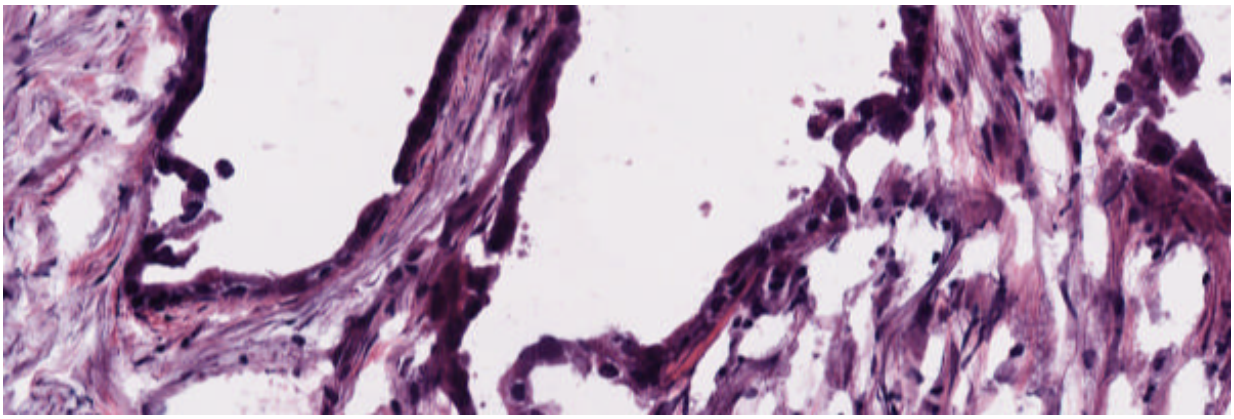
```
Nivel 3  Aumento x1.25
```



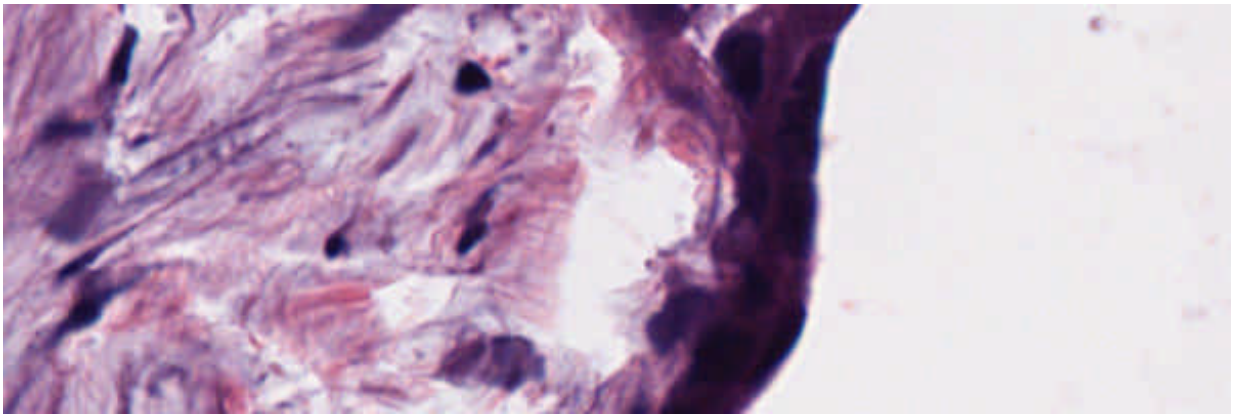
```
Nivel 2  Aumento x2.5
```



```
Nivel 1  Aumento x10.0
```



```
Nivel 0  Aumento x40.0
```



División en patches

La función creada para dividir una imagen en *patches* necesita dos parámetros, el tamaño del *patch* en píxeles y el aumento que se desea. No es necesario limitarse a los niveles de aumento disponibles. En caso de que el aumento no esté entre los niveles se seleccionará el nivel inmediatamente superior y posteriormente se redimensionará la imagen según sea necesesario.

[217]:
```
MAGNIFICATION = 1.25
PATCH_SIZE = 128

patches_params = get_slide_patches_params(os_img, patch_size=PATCH_SIZE,␣
 ↪magnification=MAGNIFICATION)
```

Esta función devuelve una lista con la info necesaria de cada uno de los *patches* para hacer las llamadas a la función *read_region* vista anetiormente. Cuando los valores *level_patch_size* y *patch_size* difieren es porque se ha tenido que leer de un nivel de aumento distinto al seleccionado.

[219]:
```
patches_params[:3]
```

[219]:
```
[{'index': (0, 0),
  'location': [0, 0],
  'level': 3,
```

```
  'level_patch_size': (128, 128),
  'patch_size': (128, 128),
  'resize': 1.0},
 {'index': (0, 1),
  'location': [4096, 32],
  'level': 3,
  'level_patch_size': (128, 128),
  'patch_size': (128, 128),
  'resize': 1.0},
 {'index': (0, 2),
  'location': [8193, 32],
  'level': 3,
  'level_patch_size': (128, 128),
  'patch_size': (128, 128),
  'resize': 1.0}]
```

[220]:
```
len(patches_params)
```

[220]:
```
161
```

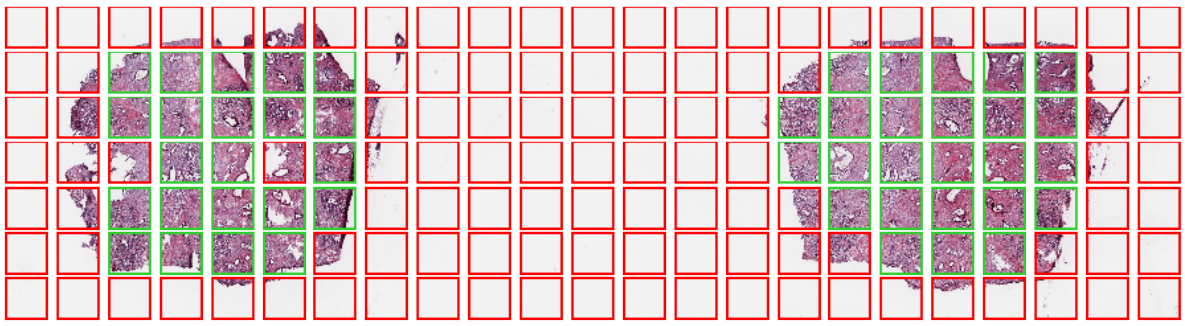
En la miniatura se podía ver que hay un porcentaje muy alto de la imagen que no pertenece al tejido, es decir, no es informativa. Para evitar entrenar las redes neuronales con *patches* de estas zonas se realiza un filtro. Aquellas imágenes con un porcentaje de píxeles blancos (o casi blancos) superior al porcentaje BLANK_PIXEL_THRESHOLD son descartadas.

El siguiente código lee cada uno de los *patches* definidos anteriormente y los dibuja añadianendo un marco de color en rojo o verde indicando si el *patch* es considerado válido. Cada una de las imágenes generadas que sean válidas podrán ya ser guardadas en formato png.

[229]:
```
BLANK_PIXEL_THRESHOLD = 30
```

[230]:
```
nrows,ncols = patches_params[-1]['index']
aspect_ratio = nrows / ncols

fig = plt.figure(figsize=(16, 16 * aspect_ratio))
valid = 0

for i,patch_params in enumerate(patches_params):

    patch_img = os_img.read_region(patch_params['location'], patch_params['level'],␣
 ↪patch_params['level_patch_size'])
    patch_arr = np.asarray(patch_img.convert('RGB'))

    blank_perc = get_white_pixel_percetange(patch_arr)

    frame_color = 'limegreen' if blank_perc <= BLANK_PIXEL_THRESHOLD else 'red'

    if blank_perc <= BLANK_PIXEL_THRESHOLD:
        valid += 1

    patch_img_framed = ImageOps.expand(patch_img, border=8, fill=frame_color)
    patch_arr_framed = np.asarray(patch_img_framed.convert('RGB'))
```

```
    plt.subplot(nrows + 1, ncols + 1, i+1)

    plt.imshow(patch_arr_framed)
    plt.axis('off')

plt.subplots_adjust(wspace=0.05, hspace=0.05)
plt.show()

print('Total Patches:', len(patches_params))
print('Valid Patches: {} ({:.2f}%)'.format(valid, 100 * valid / len(patches_params)))
```



```
Total Patches: 161
Valid Patches: 47 (29.19%)
```