I. Introduction

Meiotic drive represents a major deviation from Mendel's rules by resulting in the unequal segregation of alleles to progeny (Lyttle 1991). The maize Abnormal chromosome 10 (Ab10) system is one of the most classic example of meiotic drive, which target knobs, the condensed heterochromatic regions, to preferentially transmit knobs with linked drive loci (Yu 1997). As the knob regions are mainly consisted of repetitive sequence, the characterization of knobs remains as a conundrum for next generation sequencing methods. On the other hand, cytological studies, especially fluorescence *in situ* hybridization (FISH), are playing the most important role in knob identification and quantification. According to the cytological analysis, Ab10 has extended long arm, approximately 1.3 times the size of normal chromosome 10L. Knobs, including the 180-bp knob repeats and the 350-bp TR-1 repeats, are present in the long arm of Ab10 while absent from normal chromosome 10 (Rhoades 1985, Dawe 1996, Kanizay 2013). To study the distribution of Ab10 subtypes in maize population, we did visual assays of the chromosomal knobs in root tip cells across maize landraces and teosinte using fluorescence *in situ* hybridization (FISH). With the FISH images of samples across maize subspecies in hand, we are aiming to use image processing to quantify the knob size of different chromosomes across species, predict the size of knobs in base pair units, and identify the presence of Ab10 based on the karyotype difference and knob pattern features.

II. Methods

Scikit-image package (Van der Walt 2014, Gouillart 2017) was used for FISH image processing. Skimage.io was utilized for reading and writing original FISH images in numpy matrix format. RGB channels were separated and converted to grayscale by skimage.color function. Skimage.filters.threshold_adaptive function was then exploited for thresholding and binary image generation. The individual chromosomes in their bonding boxes were isolated through the utilization of label and regionprops functions from skimage.measure. Following the isolation, the area and centroid of chromosomes, as well as the area and total intensity of knobs on each chromosome were measured by regionprops sub-functions, which were then plotted as bar plot with matplotlib.pyplot. Rotation of the chromosomes were accomplished by integrating skimage.transform.rotate function with regionprops.orientation function. The RGB channels of rotated regions were then separated and labelled, followed by the centroid calculation for each chromosome and knobs on them. Through the comparison of centroid coordinates of 180-bp knob and chromosomes, the green knobs were further rotated to place at the bottom of each bounding box. The centroid coordinates of 180-bp knob and TR-1 were compared and the chromosomes whose y-axis value of green regions higher than that of red regions were originally classified as Ab10 candidates and labelled with red rectangles.

III. Results

As shown in the original FISH image (Figure 1), the knobs could not be visualized in the case that their intensity were not high enough or area size was too small. After the thresholding process (Figure 1), the image was more viewable with knob regions exhibited more clearly. Individual chromosomes in their bounding boxes were isolated (Figure 2) and rotated to make them perpendicular to the x-axis, with 180-bp knobs placed at the bottom of each region (Figure 3). Based on the cytological patterns of the relative locations for knobs on Ab10 (Kanizay

2013), the TR-1 repeat region should be located above the 180-bp repeat knob after the chromosome rotation. Through the implementation of this feature, three chromosomes were identified as Ab10 candidates (Figure 4).

IV. Discussion

As we all know, maize is a diploid plant and the probability that three Ab10 are present in one single cell, where 20 chromosomes were found, is relatively low. Actually, only the chromosomes located in the bottom row (Figure 4) are real abnormal chromosome 10, even though the one present in the first row also agreed with the model we set, which specified the chromosomes whose y-axis centroid coordinate of 180-bp knob higher than that of TR-1 as potential Ab10. It indicated that the model we used for Ab10 identification among all 20 chromosomes was not ideal. To improve the accuracy, we should utilize the cytological features of chromosomes and knobs, such as the chromosomal length, the long/short arm ratio to differentiate each chromosome. In addition, we could also make use of the phenomenon that TR-1 repeat region and 180-bp knob region almost never overlap to identify Ab10 among the chromosomes with knobs.

Intriguingly, based on the quantification results of total knob area and intensity in each chromosomal region (Figure 5), we can make rough predictions for the homologous chromosome pairs in cell nuclei, even without the detailed information of chromosome length or arm ratio. For example, the knob intensity of region 8 and region 16 are ranked highest, which are respectively corresponding to the first chromosome in the second row and the second chromosome in the first row in Figure 3. Consistent with the cytological studies, these two chromosomes are both chromosome 3 in maize haploid genome.

V. Conclusion and Future Work

As stated in the results, we have successfully performed image thresholding to generate binary image data and improve image visualization quality. In addition, we quantified the knob size and intensity located on each chromosome, and presented the data in bar plot. However, we didn't get the desired result for Ab10 identification, since chromosome 3 was mistaken as an additional Ab10. Therefore, we need to improve our model for Ab10 characterization in the future through the incorporation of the cytological features for each chromosome in haploid genome, such as chromosome length and arm ratio. In addition, we will integrate our analysis results for knob intensity as well as area size present in maize genome with k-mer analysis for repetitive regions (Liu 2017) using next generation sequencing data, to get a more precise estimation of repeats size in base pair unit.

VI. Reference

Dawe, R. K., & Cande, W. Z. (1996). "Induction of centromeric activity in maize by suppressor of meiotic drive 1." Proc. Natl. Acad. Sci. USA **93**: 8512–8517.

Gouillart, E., Nunez-Iglesias, J., & van der Walt, S. (2017). "Analyzing microtomography data with Python and the scikit-image library." Advanced Structural and Chemical Imaging **2**(1): 18

Kanizay, L. B., Pyhäjärvi, T., Lowry, E. G., Hufford, M. B., Peterson, D. G., Ross-Ibarra, J., & Dawe, R. K. (2013). "Diversity and abundance of the Abnormal chromosome 10 meiotic drive complex in Zea mays." Heredity **110**(6): 570-577

Liu, S., Zheng, J., Migeon, P., Ren, J., Hu, Y., He, C., ... & Wang, G. (2017). "Unbiased K-mer Analysis Reveals Changes in Copy Number of Highly Repetitive Sequences During Maize Domestication and Improvement. ." Scientific Reports, **7**.

Lyttle, T. W. (1991). "SEGREGATION DISTORTERS." Annu. Rev. Genet **25**: 51 I-57

Rhoades, M. M., & Dempsey, E. (1985). "Structural heterogeneity of chromosome 10 in races of maize and teosinte." UCLA symposia on molecular and cellular biology.

Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., ... & Yu, T. (2014). "scikit-image: image processing in Python." PeerJ **2**: e453.

Yu, H. G., Hiatt, E. N., Chan, A., Sweeney, M., & Dawe, R. K. (1997). "Neocentromere-mediated chromosome movement in maize." J Cell Biol **139**(4): 831-840.

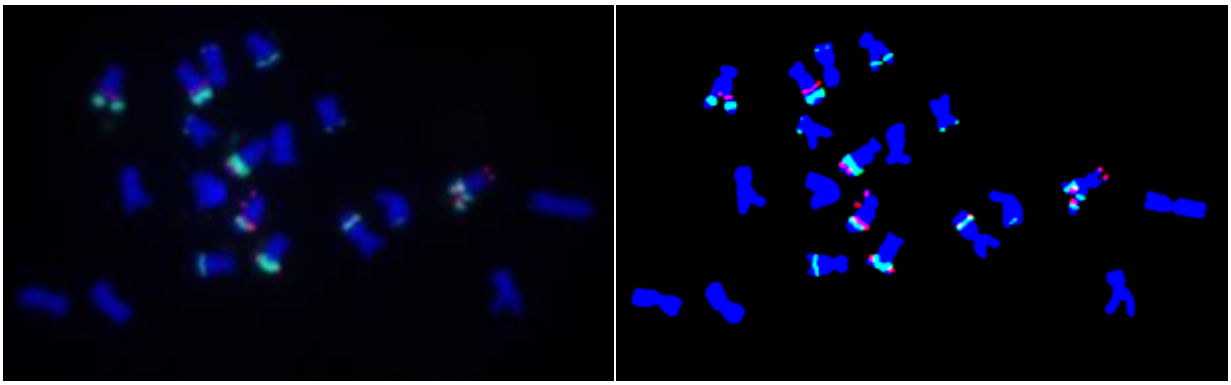

Figure 1. Original FISH image and thresholded image. Chromsomes were stained by DAPI and colored blue. Green regions are 180bp-knob repeats and TR-1 repeats were shown as red dots.

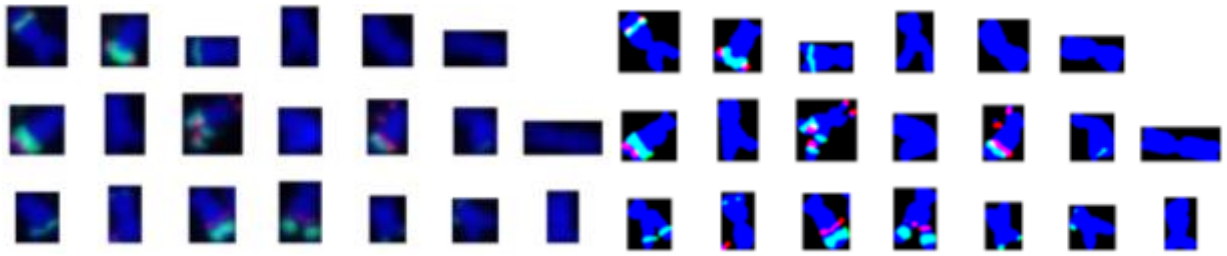

Figure 2. Individual chromosomes in their individual bounding box. The image on the left are bounding boxes isolated from the original FISH image and the one on the right are bounding boxes isolated from the FISH image after thresholding. The red, green and blue channels are representing 350-bp TR-1 repeats, 180-bp knob repeats and chromosomes respectively.

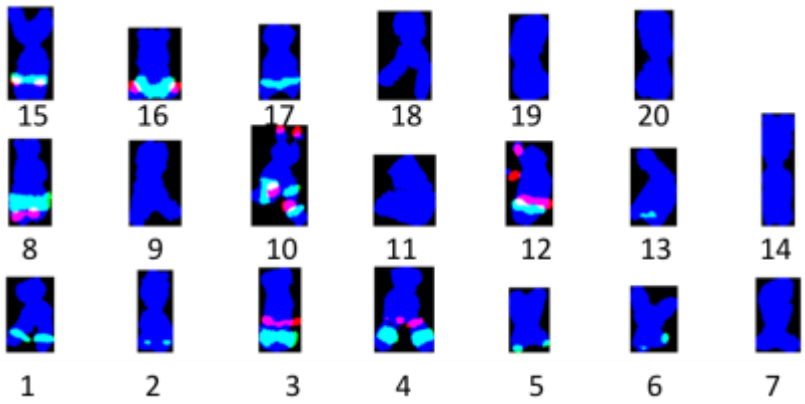

Figure 3. Chromosomal regions after rotation. Chromosomes were rotated to be perpendicular to the x-axis and the 180-bp knobs were placed at the bottom of each bounding box.

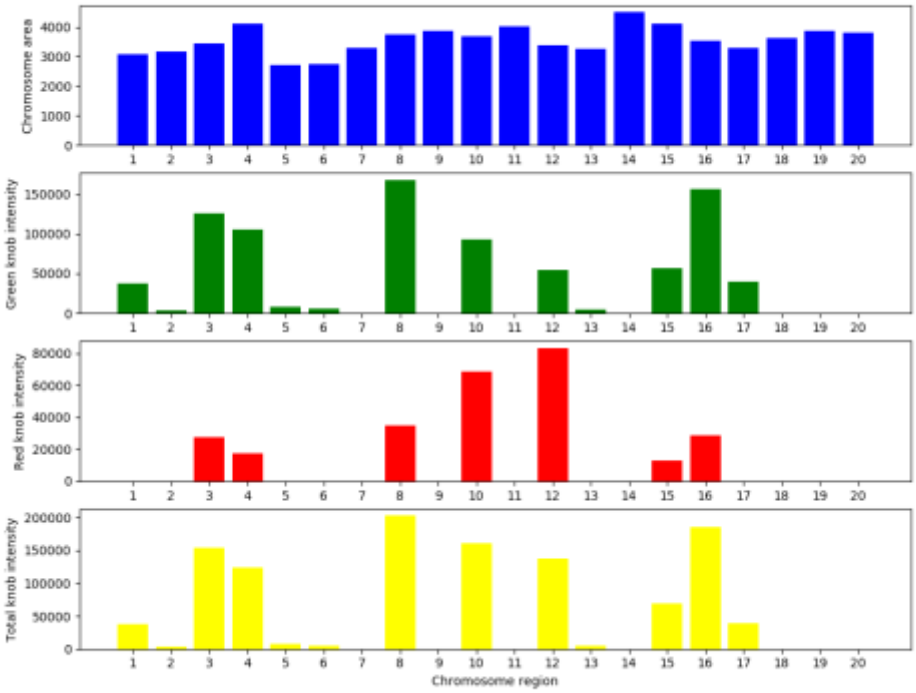

Figure 4. The bar plots of each chromosomal area (colored blue), the green knob (180-bp knob, colored green) total intensity and the red knob (TR-1, colored red) total intensity as well as the sum of total intensity (colored yellow) for both knob types on each chromosome. The labels of 20 chromosomes are corresponding to the ones labelled in Figure 3.

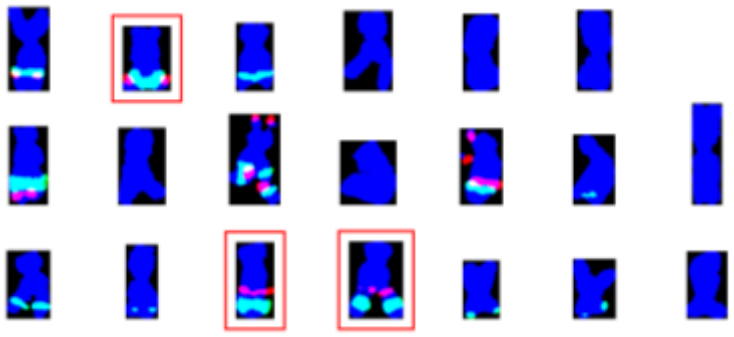

Figure 5. Ab10 chromosome candidates identified by centroid coordinate comparison. The chromosomes with red rectangle labels are the potential Ab10 present in genome. 180-bp knob shown in green and TR-1 shown in red.

```
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
from skimage import io
from skimage import filters
from skimage.measure import label, regionprops
from skimage.filters import threshold_adaptive
from skimage.transform import rotate
from skimage import color
import math

# Load the image by skimage io
ab10_2 = io.imread('/Users/JianingLiu/Downloads/ab10-typeI.png')
ab10_2_gray = color.rgb2gray(ab10_2)
# Separate out the channels
r = ab10_2[:, :, 0]
g = ab10_2[:, :, 1]
b = ab10_2[:, :, 2]
# Read in an RGB image and convert it to grayscale
r_gray = color.rgb2gray(r)
g_gray = color.rgb2gray(g)
b_gray = color.rgb2gray(b)
print(r_gray.shape)
# Finding edges with Sobel filters
r_edges = filters.sobel(r_gray)
g_edges = filters.sobel(g_gray)
b_edges = filters.sobel(b_gray)
# Apply threshold to each channel and change boolean image from threshold_adaptive to 0/1 binary image
g_bw = threshold_adaptive(g_gray, 47, offset=-20) * 1
r_bw = threshold_adaptive(r_gray, 65, offset=-25) * 1
b_bw = threshold_adaptive(b_gray, 83, offset=-10) * 1
combined_bw = np.zeros((g.shape[0], g.shape[1]), dtype=int)
# Merge channels
ab10_2_merge = np.ones((r.shape[0], r.shape[1], 3), 'uint8')
ab10_2_merge[:, :, 0] = r_bw * 255
ab10_2_merge[:, :, 1] = g_bw * 255
ab10_2_merge[:, :, 2] = b_bw * 255
ab10_2_merge_fig = Image.fromarray(ab10_2_merge)
ab10_2_merge_fig.save("/Users/JianingLiu/Downloads/ab10_1_merge.png")
# Plot individual chromosomes with red/green fluorescence in bbox
combined_bw = b_bw + g_bw + r_bw
combined_bw[combined_bw > 1] = 1
combined_label_image = label(combined_bw)
combined_regions = regionprops(combined_label_image)
regions_sliced = []
regions_sliced_widths = []
regions_centroid = []

ab10 = np.zeros(shape=len(combined_regions), dtype=int) * False
#chr_color = np.matrix(
   #[[0, 102, 204], [76, 0, 153], [66, 69, 129], [66, 69, 129], [135, 69, 68], [135, 69, 68], [133, 103, 70], [57, 130, 130],
[102, 255, 255],
    #[255, 128, 0], [255, 0, 255], [255, 153, 204], [255, 204, 204], [153, 255, 204], [51, 0, 102], [0, 102, 51],
    #[102, 102, 0], [143, 188, 143], [176, 196, 222], [128, 0, 0]])
for l, region in enumerate(combined_regions):
   if region.area > 100:
      regions_centroid.append(region.centroid)
```

```
    # Rotate the chromosomes in bbox by using orientation degree
    rtimg = rotate(ab10_2_merge[region.bbox[0]:region.bbox[2], region.bbox[1]:region.bbox[3]],
                -math.degrees(region.orientation) + 90, resize=True)
    # Separate out the channels of roated image, output the bbox which doesn't fit the chromosomes
    rtimg_r = rtimg[:, :, 0]
    rtimg_r.astype(int)
    rtimg_g = rtimg[:, :, 1]
    rtimg_g.astype(int)
    rtimg_b = rtimg[:, :, 2]
    rtimg_b.astype(int)
    # Label the binary image of rotated region, and slice the labelled region with bbox
    rtbw = np.zeros((rtimg_b.shape[0], rtimg_b.shape[1]), dtype=int)
    rtbw = rtimg_r + rtimg_g + rtimg_b
    rtbw[rtbw > 1] = 1
    rtbw_label = label(rtbw)
    rtbw_regions = regionprops(rtbw_label)
    max_area = 0.
    max_index = 0
    for i, rt_region in enumerate(rtbw_regions):
        if rt_region.area > max_area:
            max_index = i
            max_area = rt_region.area
    max_region = rtbw_regions[max_index]
    rtimg_chr = rtimg[max_region.bbox[0]:max_region.bbox[2], max_region.bbox[1]:max_region.bbox[3]]
    # Separate the channels of the rotated image (bbox)
    chr_r = rtimg_chr[:, :, 0]
    chr_r.astype(int)
    chr_g = rtimg_chr[:, :, 1]
    chr_g.astype(int)
    chr_b = rtimg_chr[:, :, 2]
    chr_b.astype(int)
    # Label chromosome (blue region), calculate the centroid of the largest blue area
    labels = label(chr_b)
    chr_regions = regionprops(labels)
    max_area = 0.
    max_index = 0
    for i, chr_region in enumerate(chr_regions):
        if chr_region.area > max_area:
            max_index = i
            max_area = chr_region.area
    chr_b_region = chr_regions[i]
    chr_b_cen_y = chr_b_region.centroid[0]
    # Label the knobs (green region), and calculate the centriod of the green areas in each chromosome by adding
up the centroid multiplying the area ratio/ total area of each green region
    labels = label(chr_g)
    chr_regions = regionprops(labels)
    max_area = 0
    area_weight = []
    for chr_region in chr_regions:
        max_area += chr_region.area
        area_weight.append(float(chr_region.area))
    area_weight = np.array(area_weight) / max_area
    chr_g_cen_y = 0.
    # Compare the centroid of green region vs blue region, make the green area located in the bottom of each
image
    for i, chr_region in enumerate(chr_regions):
        chr_g_cen_y += chr_region.centroid[0] * area_weight[i]
```

```
        if chr_g_cen_y < chr_b_cen_y:
            rtimg_chr = rotate(rtimg_chr, 180)
            chr_g_cen_y = max_region.bbox[2] - max_region.bbox[0] - chr_g_cen_y
            chr_b_cen_y = max_region.bbox[2] - max_region.bbox[0] - chr_b_cen_y
        # Separate the channels after second rotation
        chr_r = rtimg_chr[:, :, 0]
        chr_r.astype(int)
        chr_g = rtimg_chr[:, :, 1]
        chr_g.astype(int)
        chr_b = rtimg_chr[:, :, 2]
        chr_b.astype(int)
        # # Change colors of the rotated image (green area at bottom)
        # chrimg = np.zeros((chr_b.shape[0], chr_b.shape[1], 3), 'uint8')
        # # blue area minus the overlapped regions between r,g with b
        # chr_b1 = chr_b - chr_g - chr_r
        # chr_b1[chr_b1 < 0] = 0
        # ci = l % chr_color.shape[0]
        # chrimg[:, :, 0] = chr_b1 * chr_color[ci, 0] + chr_r * 255
        # chrimg[:, :, 1] = chr_b1 * chr_color[ci, 1] + chr_g * 255
        # chrimg[:, :, 2] = chr_b1 * chr_color[ci, 2]
        regions_sliced.append(rtimg_chr)
        regions_sliced_widths.append(max_region.bbox[3] - max_region.bbox[1])
        # Label the TR1 (red region), and calculate the centriod of the red regions in each chromosome by adding up
the centroid multiplying the area ratio/ total area of each region region
        labels = label(chr_r)
        chr_regions = regionprops(labels)
        max_area = 0
        area_weight = []
        noTR1onTop = True
        for chr_region in chr_regions:
            if chr_region.centroid[0] > chr_b_cen_y:
                max_area += chr_region.area
                area_weight.append(float(chr_region.area))
            else:
                area_weight.append(0.)
                if chr_region.area > 5:
                    noTR1onTop = False
        area_weight = np.array(area_weight) / max_area
        chr_r_cen_y = 0.
        for i, chr_region in enumerate(chr_regions):
            chr_r_cen_y += chr_region.centroid[0] * area_weight[i]
        if chr_g_cen_y > chr_r_cen_y and chr_r_cen_y != 0:
            ab10[len(regions_sliced) - 1] = True and noTR1onTop
# Plot image
mMargin = 15
borderW = 3
for i, img in enumerate(regions_sliced):
    x = i % 7
    y = int(i / 7)
    if ab10[i]:
        rect = np.ones((img.shape[0] + mMargin * 2, img.shape[1] + mMargin * 2), 'uint8')
        for i in range(borderW):
            rect[i, :] = 0
            rect[:, i] = 0
            rect[-i - 1, :] = 0
            rect[:, -i - 1] = 0
        rectimg = np.ones((rect.shape[0], rect.shape[1], 3), 'uint8') * 255
```

```
        rectimg[:, :, 1] = rect * 255
        rectimg[:, :, 2] = rect * 255
        plt.figimage(rectimg, 150 - mMargin + 150 * x - regions_sliced_widths[i] / 2, 75 - mMargin + 150 * y)
    plt.figimage(img, 150 + 150 * x - regions_sliced_widths[i] / 2, 75 + 150 * y)
plt.show()
# Label each channel
b_label_image = label(b_bw)
g_label_image = label(g_bw)
r_label_image = label(r_bw)
# b_image_label_overlay = label2rgb(b_label_image, image=b_bw)
# fig, ax = plt.subplots(figsize=(10, 6))
# ax.imshow(b_image_label_overlay, alpha=0.8)
# Calculate the mean intensity,total intensity and area (number of pixels) of each region in blue/green/red channel
b_regions = regionprops(b_label_image, b_gray)
b_area = []
b_mean_intensity = []
b_centroid = []
b_bbox = []
b_regions_sliced = []
b_regions_sliced_widths = []
for region in b_regions:
    if region.area > 100:
        b_area.append(region.area)
        b_mean_intensity.append(region.mean_intensity)
        b_centroid.append(region.centroid)
        b_bbox.append(region.bbox)
        b_regions_sliced.append(ab10_2_merge[region.bbox[0]:region.bbox[2], region.bbox[1]:region.bbox[3]])
        b_regions_sliced_widths.append(region.bbox[3] - region.bbox[1])
g_regions = regionprops(g_label_image, g_gray)
g_area = []
g_mean_intensity = []
g_centroid = []
for region in g_regions:
    if region.area > 10:
        g_area.append(region.area)
        g_mean_intensity.append(region.mean_intensity)
        g_centroid.append(region.centroid)

r_regions = regionprops(r_label_image, r_gray)
r_area = []
r_mean_intensity = []
r_centroid = []
for region in r_regions:
    if region.area > 10:
        r_area.append(region.area)
        r_mean_intensity.append(region.mean_intensity)
        r_centroid.append(region.centroid)
r_total_area = np.zeros(len(b_area))
r_total_intensity = np.zeros(len(b_area))
r_belongs_to = np.zeros(len(r_centroid), dtype=np.int)
i = 0
for rcen in r_centroid:
    j = 0
    for box in b_bbox:
        if rcen[0] > box[0] and rcen[1] > box[1] and rcen[0] < box[2] and rcen[1] < box[3]:
            r_belongs_to[i] = j
        j += 1
```

```
    i += 1
for i in range(len(r_belongs_to)):
    r_total_area[r_belongs_to[i]] += r_area[i]
    r_total_intensity[r_belongs_to[i]] += r_mean_intensity[i] * r_area[i]

# Calculate the total area & intensity of green, red channel in each blue region (chromosome)
g_total_area = np.zeros(len(b_area))
g_total_intensity = np.zeros(len(b_area))
g_belongs_to = np.zeros(len(g_centroid), dtype=np.int)
i = 0
for gcen in g_centroid:
    j = 0
    for box in b_bbox:
        if gcen[0] > box[0] and gcen[1] > box[1] and gcen[0] < box[2] and gcen[1] < box[3]:
            g_belongs_to[i] = j
        j += 1
    i += 1
for i in range(len(g_belongs_to)):
    g_total_area[g_belongs_to[i]] += g_area[i]
    g_total_intensity[g_belongs_to[i]] += g_mean_intensity[i] * g_area[i]
knob_total_intensity = r_total_intensity + g_total_intensity
# Plot bar plot
fig = plt.figure(figsize=(12, 4))
ax = fig.add_subplot(411)
xTickMarks = [str(i) for i in range(1, 21)]
chr = np.arange(1, len(g_total_intensity) + 1)
b_area_ratio = b_area
ax.bar(chr, b_area_ratio, color='b', tick_label=xTickMarks)
ax.set_ylabel('Chromosome area')
ax = fig.add_subplot(412)
ax.bar(chr, g_total_intensity, color='g', tick_label=xTickMarks)
ax.set_ylabel('Knob180 intensity')
ax = fig.add_subplot(413)
ax.bar(chr, r_total_intensity, color='r', tick_label=xTickMarks)
ax.set_ylabel('TR1 intensity')
ax = fig.add_subplot(414)
ax.bar(chr, knob_total_intensity, color='yellow', tick_label=xTickMarks)
ax.set_ylabel('Total knob intensity')
plt.xlabel('Chromosome region')
plt.show()
```