Redlining's Legacy in Urban Heat

Jason Kao (jk4248)
Professor John Kender
T.A. Ruochen (Nick) Liu
COMS E6735: Visual Databases
April 26, 2021

I. Introduction

Increasingly warm environments in cities pose a danger to human health. Urban heat amplifies the concentration of air pollutants [1] and increases mortality during heat waves [2]–[4]. Statistical analyses suggest dense urban areas without canopy coverage are most susceptible to greater heat exposure [5], but few studies explore the reasons why many historically marginalized areas reside in those areas [6].

Recent research by Hoffman, Shandas, and Pendleton [7] points to one explanation for the heat: historical federal housing segregation. After the Great Depression, New Deal legislation created the Home Owners' Loan Corporation (HOLC) to be a standardized appraisal tool for home buyers and lenders. Between 1935 and 1940, HOLC drew mortgage security maps in over 200 major cities grading neighborhoods from A to D. Areas that received the grade "A" were considered low-risk for lenders and colored green. Areas with the grade "D" were considered "harzardous" and colored red.

Government surveyors assigned grades according to a number of factors, including access to transportation, quality of housing, and interviews with local officials. However, a primary factor in these decisions was race. Much racist and xenophobic language can be found in area reports written by HOLC staffers. Words such as "infiltration", "inharmonious", and "lower-grade populations" are pervasive[1]. White neighborhoods, on the other hand, were described as containing "respectable people" [8]. But while Shelley v. Kraemer (1948) [9] made it illegal for states to enforce racially restrictive housing covenants, and the Fair Housing Act of 1968 [10] prohibited private entities from discriminating the sale of houses based on race, the economic and spatial segregation created by HOLC's maps persist. Borrowers within red areas have been more likely to be denied credit and mortgages, which have contributed to a host of adverse social outcomes such as poverty and high unemployment. Today, a majority of these areas are still largely inhabited by low-income and middle-income people of color, while grade "A" areas remain predominantly upper-class

[1]"Of the Bedford–Stuyvesant in Brooklyn, for instant, agents explained that 'Colored infiltration a definitely adverse influence on neighborhood desirability although Negroes will buy properties at fair prices and usually rent rooms.' In the Tompkinsville neighborhood in Staten Island, 'Italian infiltration depress residential desirability in this area.' In a south Philadelphia neighborhood 'Infiltration of Jewish into area have depressed values.' The assessors of a Minneapolis neighborhood attributed the decline of a 'once a very substantial and desirable area' to the 'gradual infiltration of negroes and Asiatics."' [8]

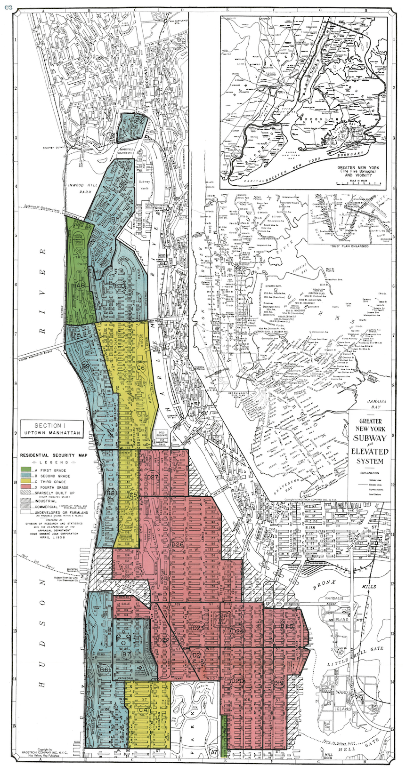


Fig. 1. A scan of a HOLC map drawn for upper Manhattan.

white [11]. Redlined neighborhoods have seen less investment in public health, such as tree canopy coverage [12] and public transportation, and more air pollution [1]. Moreover, greenspace [13] and highways [7] are associated with greater urban heat.

I seek to investigate the relationship between redlining and urban heat concentration by researching the following questions:

1) To what extent can redlining, and the resulting ecological segregation, explain heat exposure in urban area?
2) How has the pattern changed over time?

I will derive land surface temperatures from public satellite imagery collected by NASA's Landsat program. I will then conduct a comparative, time-series analysis of the temperatures in different HOLC grades and compare them to different ecological factors. Lastly, I will build an interactive website that visualizes and enables city-level exploration into this data. I explain these methods in-depth in the following sections.

II. Materials and methods

A. HOLC's mortgage security maps

The boundaries on the original HOLC maps were drawn over city maps with marker. Prior to 2016, these maps were only available in this raster form. Since then, the University of Richmond's Mapping Inequality project has digitized all of these maps into geolocated polygon boundaries. Each polygon is assigned several data fields, including neighborhood grade and qualitative comments by the government surveyors. These polygons are available in all 202 cities as a shapefile [8]. Hereafter I will refer to HOLC polygons and HOLC neighborhoods interchangeably.

B. Satellite imagery

The NASA/USGS Landsat program, begun in 1972, consists of eight imaging satellites. The latest satellite to be launched, Landsat 8, can capture an image of the entire Earth every sixteen days. The satellites capture eleven bands, ranging from a deep blue to near-infrared and thermal-infrared. Visible spectrum bands are captured at a spatial resolution of thirty meters. The thermal bands, which have the longest wavelengths, are captured at a spatial resolution of one hundred meters.

Landsat seems to be the best option for this project compared to newer, more popular imagery missions. NASA's MODIS (Moderate Resolution Imaging Spectroradiometer) instrument, aboard the Terra and Aqua satellites [14], can capture thermal bands at a much higher temporal resolution than Landsat (one global image per day) but at a prohibitively low spatial resolution (1000m). The Sentinel missions of the European Union also capture thermal bands at a 1000m spatial resolution. NASA's most recent remote sensing system, the Geostationary Operational Environmental Satellite (GOES), images the Earth at an astonishing frequency of fifteen minutes, but at the cost of a 4km spatial resolution [15].

Across the eight Landsat satellites are a set of imaging instruments that have captured thermal bands at varying spatial resolutions. The first Landsat satellite to capture the thermal band reliably was Landsat 4, launched in 1982 and decommissioned in 1993 [16]. Landsat 4, as well as Landsat 5 (1984–2013) carried the Thematic Mapper instrument (TM), which captured the thermal infrared channel at a spatial resolution of 120m [17]. Landsat 6 fell into the Indian Ocean [18]. Landsat 7 (1999–present) carries the Enhanced Thematic Mapper Plus instrument (ETM+), which better detected the thermal band at a spatial resolution of 60m [19]. Landsat 8 (2013–present) carries the Thermal Infrared Sensor (TIRS), which collects the data of two thermal infrared bands at a resolution of 100m.

I used the Google Earth Engine platform (GEE) [20] to access and process the desired satellite imagery. The imagery on Google's servers has already been preprocessed to compute cloud masks, cloud coverage percentage, and other derivations that are useful for temperature calculation. Moreover, the platform offers high parallelizability and automatic optimization of computation throughout its servers.

I analyze satellite imagery over time for the years 2020, 2000, and 1990. I did not analyze the year 2010 due to the Scan Line Corrector Failure [19]. Table 1 lists all the datasets from GEE I will use.

TABLE I
Google Earth Engine Datasets

Dataset name	**Temporal availability**
USGS Landsat 8 Collection 1 Tier 1 Raw Scenes	2013–present
USGS Landsat 7 Collection 1 Tier 1 Raw Scenes	1999–present
USGS Landsat 5 TM Collection 1 Tier 1 Raw Scenes	1984–2012

I processed the imagery according to the flowchart in Fig. 2. First, I filtered each collection of satellite imagery to those that covered a HOLC polygon. I then filtered each collection to the summertime (June 1 to August 31) of the target year, as well as that of the year before and that of the year after. To account for cloudy images[2], I initially filtered the collections using GEE's precomputed cloud coverage percentages. However, I found that this method was overbroad, as Landsat images are much larger than cities. Landsat 7 images, for instance, are approximately 170 km north-south by 183 km east-west — an area of 31,110 km^2. Cities in the HOLC shapefile have a median area of 34 km^2. Thus, an image with a cloud percentage score of even 0.1% could still block out an entire city. For a more narrowly tailored approach, I sought to compute cloud coverage scores that only considered image areas above cities.

Each Tier 1 Landsat image comes with a 16-bit Quality Assessment Band [21]. The fourth bit is cloud (0 means no, 1 means yes). I reduced each image within all intersecting city boundaries and found the percentage of cloudy pixels. I excluded an image from the analysis if it the percentage exceeded 10.

C. Temperature extraction

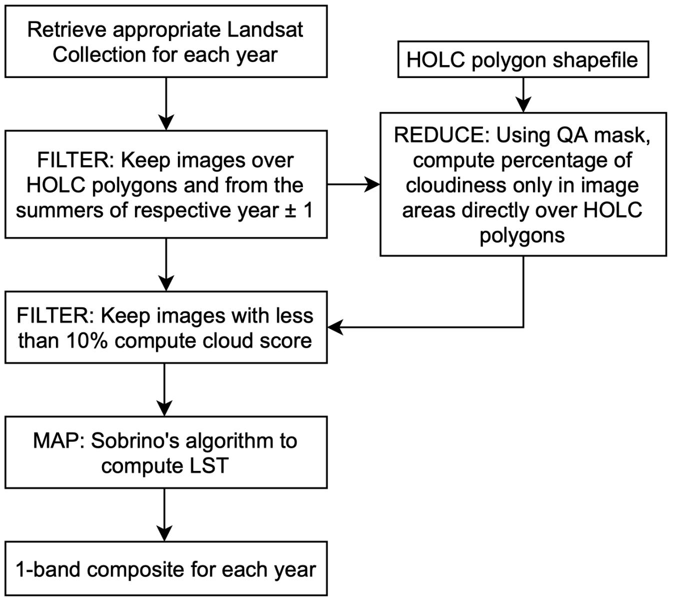
Using models provided by Sobrino et al. [22], and the open-source library Google Earth Engine Toolbox [23], I transformed each image into a float band representing land surface temperature (LST) in Fahrenheit. The general methodology is as follows.

I first compute a Top of Atmosphere (L_λ) radiance band:

$$L_\lambda = \text{RADIANCE\_MULT} * BT + \text{RADIANCE\_ADD}$$

where the additive and multiplicative RADIANCE constants are provided by NASA to help convert pixel values to radiance, and BT is the thermal infrared band (band 10 for Landsat 8,

[2]Clouds artificially decrease land surface temperature measurements.



Retrieve appropriate Landsat Collection for each year
HOLC polygon shapefile
FILTER: Keep images over HOLC polygons and from the summers of respective year ± 1
REDUCE: Using QA mask, compute percentage of cloudiness only in image areas directly over HOLC polygons
FILTER: Keep images with less than 10% compute cloud score
MAP: Sobrino's algorithm to compute LST
1-band composite for each year

Fig. 2. Processing steps to retrieve temperature composites from raw images.

but band 6L for Landsat 7). Next, I calculate the Top of Atmosphere Brightness Temperature (T_B) under the assumption of uniform surface emissivity:

$$T_B = \frac{\mathrm{K1}}{\ln\left(\mathrm{K2}/L_\lambda + 1\right)}$$

where K1 and K2 are calibration constants provided by NASA to help convert radiance to temperature. Lastly, to correct for emissivity, I first calculate the normalized difference vegetation index (NDVI, the normalized difference between the near infrared and red band). Next, we calculate the vegetation proportion using two constants for NDVI values of vegetation and soil ($\mathrm{NDVI}_v = 0.5$ and $\mathrm{NDVI}_s = 0.2$ respectively):

$$P_v = \left(\frac{\mathrm{NDVI} - \mathrm{NDVI}_s}{\mathrm{NDVI}_v - \mathrm{NDVI}_s}\right)^2$$

Emissivity is derived by the following equation:

$$\epsilon = 0.004 \cdot P_v + 0.987$$

Finally, LST in Kelvin is calculated by correcting T_B with emissivity:

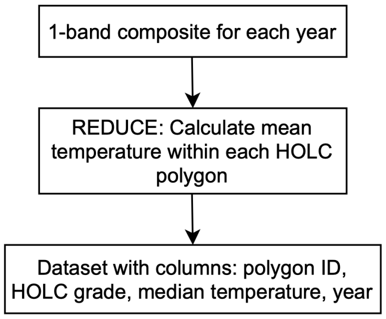
$$\mathrm{LST} = \frac{T_B}{1 + (\lambda T_B/\rho) \cdot \ln \epsilon}$$

where $\lambda = 11.5\mu m$ and $\rho = 1.438 * 10^{-2} m \cdot K$.

I then composited the images in each 3-year grouping using a median value function. Finally, I masked the composite with the JRC Global Surface Water dataset [24] (which is available on GEE) to hide many irrelevantly low temperature values

The temperature computation code — which must be run in the Earth Engine Code Editor — is in Appendix A. I used the open-source Google Earth Engine Toolbox (GEET) to perform the transformations necessary to calculate temperature [**?**].

I use the final temperature composite for data visualization (see section IV), however I further reduce the composite in preparation for statistical analysis. This processing is summarized in Fig. 3. I first calculated the median temperature within each HOLC polygon. I then exported the object from the GEE platform and cleaned and formatted the data with ndjson-cli [25].



1-band composite for each year
REDUCE: Calculate mean temperature within each HOLC polygon
Dataset with columns: polygon ID, HOLC grade, median temperature, year

Fig. 3. Processing steps for the statistical analysis of temperature.

D. Impervious surfaces

Impervious surfaces are important to analyze in urban temperature research because they absorb and store more heat, enhancing the urban heat island effect [26]. Certain kinds of impervious surfaces, such as buildings and major highways, are important to look at for this paper in particular. Highways, during the initial wave of interstate infrastructure construction in the 1950s, had been placed by some state and municipal governments directly through formerly redlined low-income or Black communities [27]. Redlined areas, which largely are inexpensive land, also saw the construction of large building complexes for housing or industry [28]. These two patterns motivate a careful analysis of not only how impervious surfaces affect urban heat, but also the individual and compounding effects of different kinds of impervious surfaces.

The USGS National Land Cover Database (NLCD) provides a nationwide dataset on land cover and impervious surfaces at a 30-meter resolution. In its newest iteration, NLCD 2016, the data not only detail what percentage of each pixel contains impervious surfaces (the impervious percentage layer, or IPL), but also what kinds of surfaces they are (the impervious surface descriptor layer, or ISDL). The ISDL identifies different kinds of roads, energy production sites, and non-road surfaces (e.g. buildings). [29]

The entire database is available on the GEE platform. The ISDL is coded as an integer band with thirteen possible values. A band value of 0, for instance, represents “not impervious”; 1 represents a “primary road” (e.g. interstates and other major roads); and 11 represents a “urban energy production site”.

I use this raster database for data visualization (see section IV), however I further reduce the dataset in preparation for statistical analysis. I used a GEE reducer to compute the frequency of each band value in ISDL in every HOLC polygon. I exported the object as GeoJSON from the platform to my local computer. I cleaned and formatted it into a table with ndjson-cli [25] and read it into RStudio [30].

E. Tree cover

The NLCD also has a tree cover layer, which is a 30-meter raster image where pixel values are the percentages of tree cover in each pixel. I use this raster database for data visualization (see section IV), however I further reduce the dataset in preparation for statistical analysis. I used a GEE reducer to compute the frequency of each percent value in every HOLC polygon. I exported the object as GeoJSON from the platform to my local computer. I re-formatted it into a table with ndjson-cli and used Python to compute the average percent value within each polygon.

III. Results

I grouped the temperature data by the HOLC grades of the neighborhoods and conducted a Tukey's honestly significant difference (HSD) post-hoc test at a 95% confidence interval ($p = 0.05$). Tukey's test allows us to understand the extent and significance of the temperature disparities between different HOLC grades. I accounted for the unequal sample sizes across HOLC grades (1,040 polygons rated A; 2,331 rated B; 3,380 rated C; 2,116 rated D) by using a slight variation of the Tukey test called the Tukey-Kramer Method. The results of Tukey's test on temperature data (section II-C) from 1990 are as follows.

```
      diff         lwr      upr     p adj
B-A 1.1824 0.306236239 2.058594 0.0029649
C-A 1.8193 0.986386181 2.652215 0.0000001
D-A 3.1980 2.312439781 4.083634 0.0000000
C-B 0.6368 0.004940516 1.268830 0.0473943
D-B 2.0156 1.315700166 2.715543 0.0000000
D-C 1.3787 0.733796966 2.023676 0.0000002
```

An example interpretation of this table would be that neighborhoods graded “D” are between 2.3 and 3.2 degrees hotter than neighborhoods graded “A” at a 95% confidence interval (see the third row) .

The test results for 2000 are as follows.

```
      diff       lwr      upr     p adj
B-A 1.2559 0.6454921 1.866379 0.0000008
C-A 2.4665 1.8851086 3.048036 0.0000000
D-A 3.0236 2.4025991 3.644723 0.0000000
C-B 1.2106 0.7706446 1.650629 0.0000000
D-B 1.7677 1.2765932 2.258857 0.0000000
D-C 0.5570 0.1024794 1.011698 0.0089250
```

The test results for 2020 are as follows.

```
      diff        lwr       upr     p adj
B-A 1.6225  0.8032040 2.4419315 0.0000022
C-A 2.6611  1.8804397 3.4418506 0.0000000
D-A 2.8008  1.9671543 3.6344884 0.0000000
C-B 1.0385  0.4481411 1.6290138 0.0000371
D-B 1.1782  0.5193798 1.8371275 0.0000260
D-C 0.1396 -0.4704539 0.7498064 0.9356703
```

We see that in each pairwise comparison, for all years and most comparisons, there is a statistically significant positive increase in temperature when stepping down in HOLC grade. These differences have decreased over time.

I computed Pearson correlations between temperature and the proportion of impervious surfacing for roads and nonroads (I ignored energy production sites as they rarely appeared in HOLC cities.) Under a 95% confidence interval, temperature correlated with any impervious surfacing at $R = (0.2515619, 0.2741793)$. Temperature correlated with impervious surfaces that were roads at $R = (0.1919271, 0.2152166)$. Temperature correlated with impervious surfaces that weren't roads at $R = (0.2406727, 0.2634255)$. Thus I conclude that nonroad impervious surfaces, like buildings, are a major determinant of temperature.

I computed the Pearson correlation between temperature and the proportion of tree cover in each HOLC polygon as well. The result is $R = (-0.3262793, -0.3043991)$. Thus trees are well-correlated with temperature.

IV. Visualizations

I built an interactive website for users to view statistical results and visualizations of the data: https://jsonkao.github.io/redlining-heat/.

In the first part of the website, a user can choose a city and year from a dropdown menu and see a map of that city with summer temperatures and impervious surfacing. The map also displays the HOLC grades of each neighborhood. The process that generates this map for each city requires five inputs, as shown in Fig. 4.

Before generating all the inputs, I required the bounding box of the city for which we want to generate a map. I used mapshaper [31] to read the HOLC shapefile from section II-A and transform it into a mapping from city to bounding box.

The first input for the map is the temperature information for each city. I used the Earth Engine's JavaScript API [32] to crop the temperature composites from section II-C to fit each city's bounding box and download the resulting image. (I used the API instead of the platform to be able to do the operation from the command line, within a Makefile.) I then colorized the data, which I describe in section IV-A.

The second input required is the impervious surfaces data. I used the Earth Engine JavaScript API to crop the impervious surfaces descriptor layer to fit the city's bounding box. The NLCD data is already pre-projected into Conus Albers. After downloading the image, I projected it back into WGS84 (latitude-longitude) and ignored any data that consequently overflowed the city's bounding box. I then colorized the data, which I describe in section IV-A. For the web visualization, I allow users to select which category of impervious surfaces they would like to see. I ignored any kind of impervious surfacing that was an energy site or not urban. Thus I grouped 8 values of impervious surfacing into two categories: urban roads (bits with values 1 through 6) and urban non-roads (values 9 and 10). I generated 3 image assets: one for each category, and one to show impervious surfacing as a whole. I found that the third image was necessary because simply

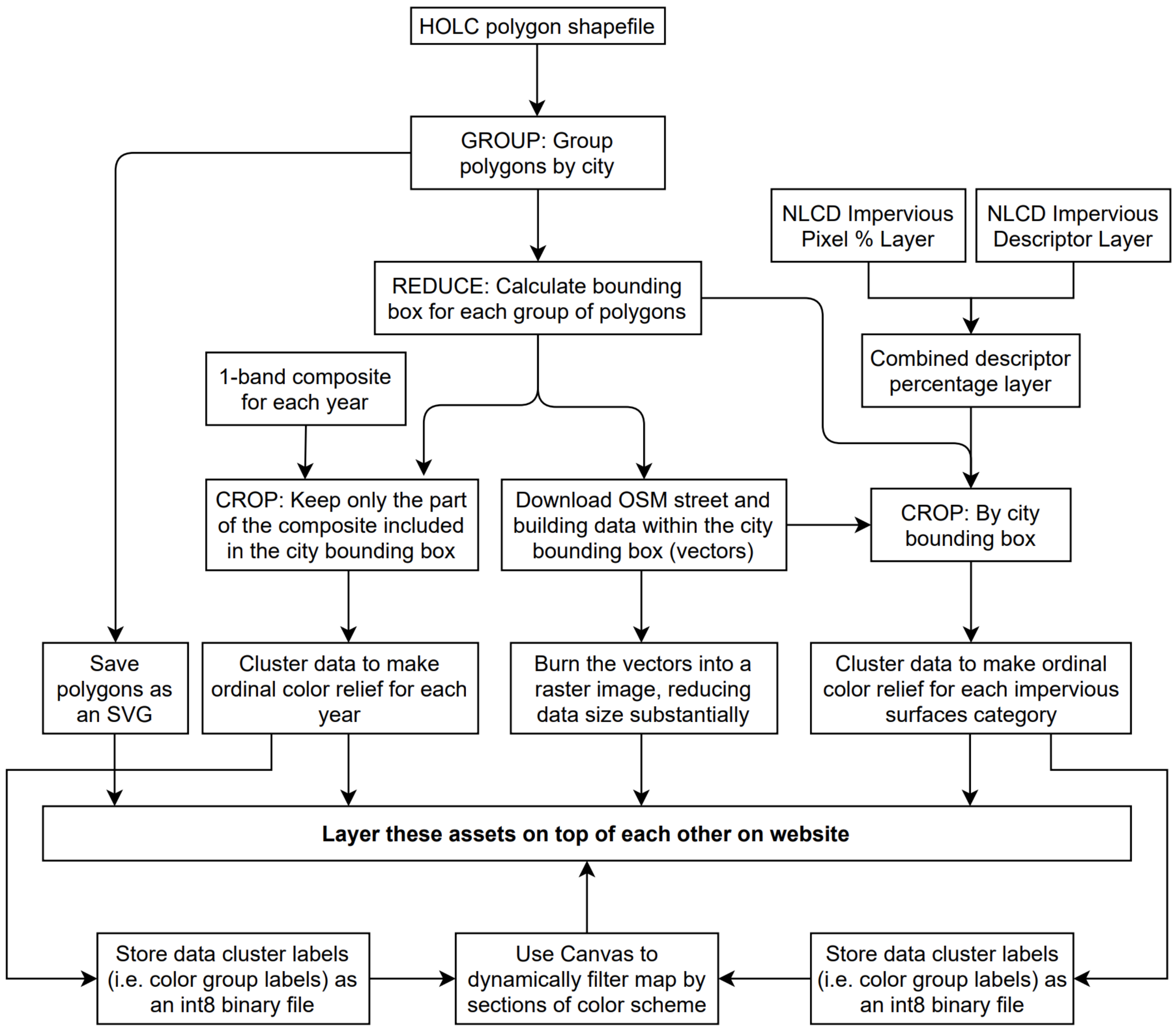

HOLC polygon shapefile
GROUP: Group polygons by city
NLCD Impervious Pixel % Layer
NLCD Impervious Descriptor Layer
REDUCE: Calculate bounding box for each group of polygons
Combined descriptor percentage layer
1-band composite for each year
CROP: Keep only the part of the composite included in the city bounding box
Download OSM street and building data within the city bounding box (vectors)
CROP: By city bounding box
Save polygons as an SVG
Cluster data to make ordinal color relief for each year
Burn the vectors into a raster image, reducing data size substantially
Cluster data to make ordinal color relief for each impervious surfaces category
Layer these assets on top of each other on website
Store data cluster labels (i.e. color group labels) as an int8 binary file
Use Canvas to dynamically filter map by sections of color scheme
Store data cluster labels (i.e. color group labels) as an int8 binary file

Fig. 4. Process for generating the full map on the website.

overlaying the two category-specific images on the web led to miniscule but noticeable overlap at pixel boundaries.

The third input is a base map[3]. I first downloaded street and building data (line and polygon data, respectively) from the Overpass API, a public API for OpenStreetMap data [33], within each city's bounding box. I used Python to convert the OpenStreeMap data, which is in XML format, into GeoJSON. I then used GDAL to burn this vector data into a PNG.

The fourth input to the map is the HOLC boundary data. I used mapshaper to read the HOLC shapefile, filter the polygons within the desired city, and output an SVG.

The fifth input to the map — cluster labels — I discuss in the section IV-A.

In summary, for each city I have a base map, a boundary map, a temperature map for each year, and an impervious surfaces map. I layer them on top of each other using HTML and CSS to create the final map. I used JavaScript for interactivity: once a user chooses the desired city and year, the layers of the map are set to that particular city.

The maps load nearly instantaneously, as all raster layers have been pregenerated to PNGs that are 3000 pixels in width (the maximum size of a basemap asset is 1.3M). I used Spectate[4] [34] to develop the website and Snowpack [35] to

[3]A base map is a background layer to the main map that contains contextual information like streets, buildings, and neighborhood labels.

[4]Spectate is a tool that provides a workflow for building and publishing freeform stories. It allows reporters to code locally while writing stories in Google Docs. Spectate was created in the Graphics department of the Columbia Daily Spectator.

build the website.

Generating the raster layers takes an average of 130 seconds for each of the 202 cities. The entire process is facilitated by GNU `make` on a computer with 8 GB of RAM and 4 CPU cores.

I've also run the Tukey test and created a visualization of that result with ggplot2 [36] for each city. That graphic appears after the map. The y-axis represents the pair being compared. The x-axis represents the difference in temperature those two HOLC grades had. The horizontal bands represents the confidence interval.

A. Color blending on the web

Initially, I used a continuous color scale from blue to white to red to show temperature. However, I later decided to bin the data into a discrete set of color groupings to reduce the amount and complexity of color information on the map. Thus arose the question of how to group temperature values into the bins as to maximize the map's visual accuracy and effectiveness. The problem is well-studied in cartography. (In computer science, it can be called clustering.) I first used a quantile method. However, in a dataset with values `[1, 10, 11, 12]` for instance, quantile grouping with 2 bins would group 1 with 10. This is a suboptimal "hard break" because 10 is much more intuitively grouped with 11 and 12 than 1. A commonly used algorithm in cartography for finding "natural breaks" in the data is the Jenks optimization method [37]. Jenks finds groupings of data that minimize intra-group variance and maximize inter-group variance. In 2011, Haizhou Wang and Mingzhou Song published a new algorithm, Ckmeans, in R for 1-dimensional k-means clustering (a polynomial-time special case of k-means) [38] that was shown to be more optimal than [39] (the default k-means implementation in R) and seen to be a slight improvement over Jenks[5]. Thus I decided to bin temperature data with the Python package kmeans1d — a C++ implementation of a 2018 Ckmeans improvement [41]. In Fig. 5, which compares quantile grouping (top) and Ckmeans grouping (bottom), one can observe how Ckmeans offers a more uniform distribution of data across each color while knowing that no "hard breaks" are occurring in the scale.

Furthermore, I wanted to display the impervious surfaces raster data together with temperature so users could explore the patterns between temperature, HOLC grade, and physical infrastructure. Here, I took inspiration from bivariate choropleth maps, which are choropleth maps that show two variables at once and can help bring out relationships between the variables. Fig. 6 is an example of how two univariate choropleth maps are combined into one bivariate choropleth.

The visual complexity of bivariate maps further motivates a use of a discrete rather than continuous color space. Yet two major questions were to be resolved before the redlining maps could adopt the bivariate choropleths' concept. For one, what would the colors be? (For the rest of the paper, I will interpret color with respect to the HSV color space. The HSV space is cylindrical: hue is the angular dimension, value is the vertical axis from black to white, and saturation is the radial axis from white to a hue.) In [42] Stephens recommends using two single-hue sequential color schemes. However, I wanted to express the temperature variable using diverging colors, which requires at least two hues. Stephens' custom bivariate color schemes all involve one hue for each scale, and the two hues are taken from an approximate triadic color scheme[6] I understood these pairings to be "bivariate-compatible". In the HSV space, bivariate-compatible colors are 120° apart in hue. Therefore for any hue $h°$ there exists two distinct bivariate-compatible hues $h° \pm 120°$.

I built a tool in HTML and CSS (see Fig. 7) to determine which absolute value hue offsets would beget the most appeal-

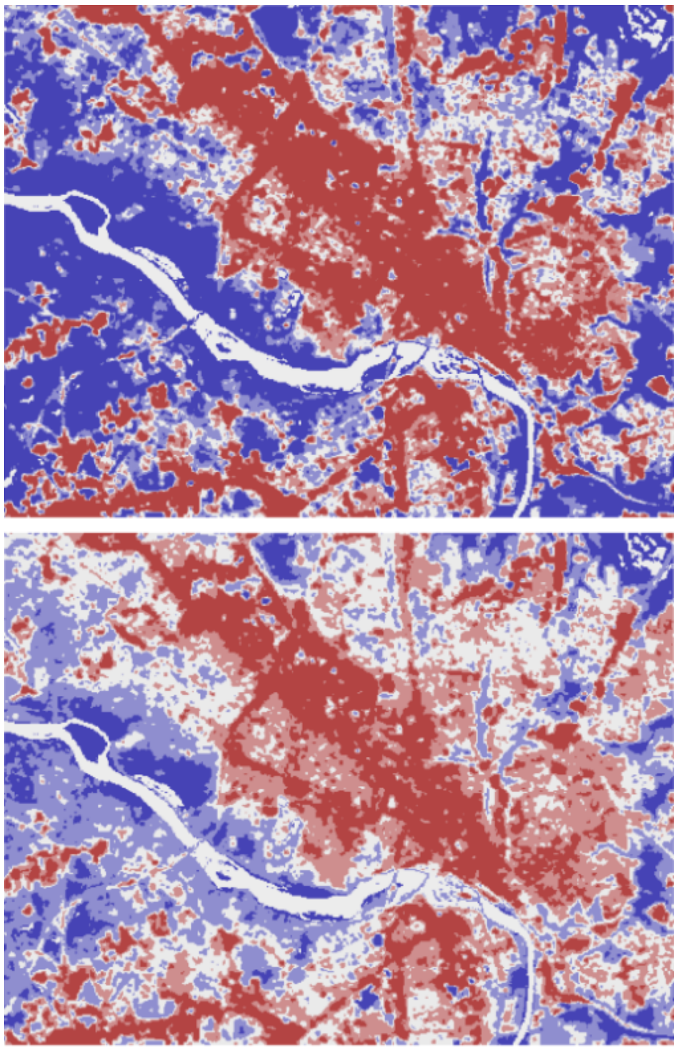


Fig. 5. Top: The temperature composite for Richmond, 2000, mapped to 5 colors using a quantile scale. Bottom: The same but using Ckmeans. (These images were from an earlier stage of development, and the choice of colors and bin counts do not reflect the final product.)

[5]In 2015, Tom MacWright, a well-known contributor to Mapbox and D3, endorsed Ckmeans and replaced Jenks with Ckmeans in his Simple Statistics JavaScript library [40]

[6]Triadic colors are colors that can be connected by an equilateral triangle on the color wheel (e.g. red and blue, cyan and magenta).

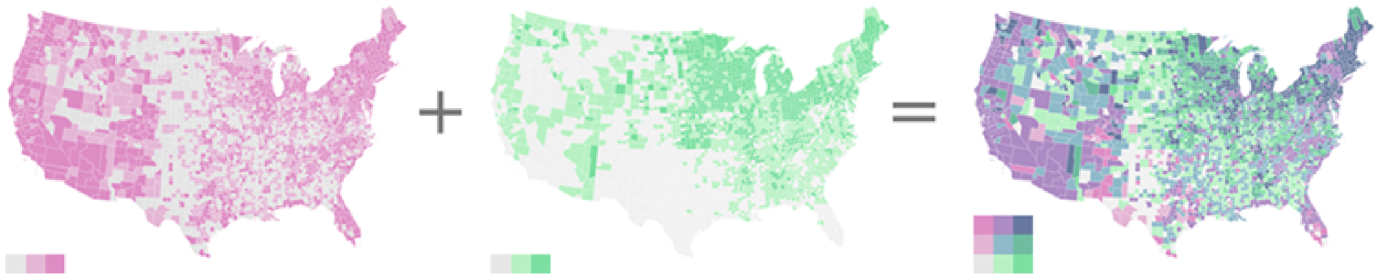


Fig. 6. “And then there were nine: Combining two 3-class univariate maps produces one 9-class bivariate map.” Source: [42]

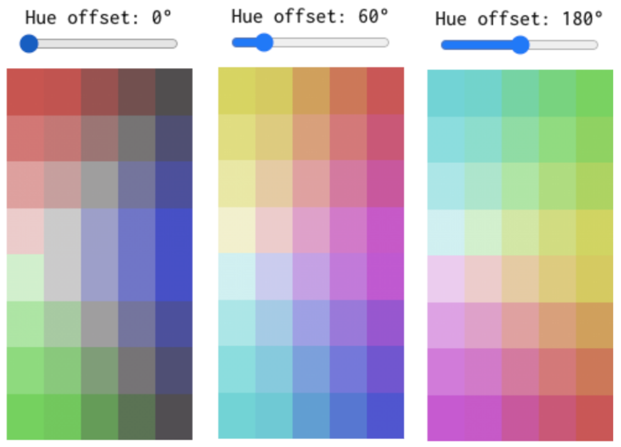


Fig. 7. A range slider tool through which one can view temperature colors on the y-axis overlaid with impervious surfaces colors on the x-axis.

ing and understandable color scheme.

I noticed that hue offsets that were multiples of 60, but not 120, often yielded the prettiest and most distinct colors. Thus I chose a hue of 60° for the high end of the temperature scale, $60+120°$ for the low end of the temperature scale, and $60+240°$ for the impervious surfaces scale.

Stephens’ color schemes increase in saturation and decrease a little in value [42]. Thus for each mapping to b color groups (if we understand the diverging temperature scale to be a combination of two b-group sequential scales) I generated a color scheme $c_1, \ldots, c_b$ (where each c_i is a tuple of hue, saturation, and value) with the following formula:

$$c_i = \left(H, S_i + \frac{i}{b}(S_f - S_i), V_i + \frac{i}{b}(V_i - V_f)\right)$$

where H is hue S is saturation, and V is value. The subscripts represent their initial and final values, which can be found in the code in `./data/scripts/create-color-config.py` (see section VI).

I used GDAL in Python [43] with the aforementioned kmeans1d library to cluster the values of each raster image. I then used GDAL’s gdaldem program to map those clusters to either the single-hue sequential scale for impervious surfaces or the double-hue diverging scale for temperatures.

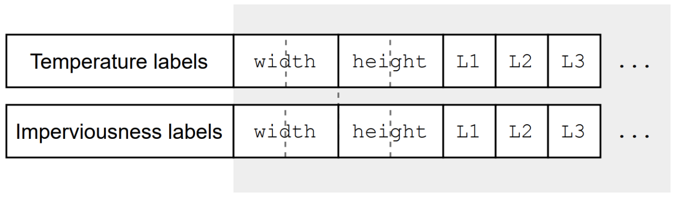


Fig. 8. Byte structure of binary label files. One block represents one byte.

I decided to use six ordinal bins for temperature and four bins for the percentage of impervious surfaces. On the web, I use CSS blend the layers with the darken blend, as recommended by [42].

The website offers a legend very similar to Fig. 7 to help a user interpret the meaning of the different colors. Given the color complexity of the legend and map, I implemented an image filtering tool as suggested by [44]. Users can also click individual squares of the legend to only view data of that color (i.e. of that temperature and impervious surfaces bin). What enables this is the fifth input to the map: cluster labels. When I generate the clusters for the temperature and impervious surfaces color reliefs, I store the cluster labels of each pixel in a binary file, whose format is shown in Fig. 8. In the file, the first four bytes are two Uint16’s, `width` and `height`. Afterwards are `width * height` Uint8’s representing cluster labels.

When a user clicks on one (or multiple) color squares in the legend, I create an array of length `width * height * 4` — R, G, B, A for each label — where a pixel becomes black if its labelling is selected and white otherwise. I paint this data onto a semi-opaque HTML Canvas. With a lighten blending mode, users can see through the black areas and can barely see through the white areas.

The coloring of the basemap (roads and buildings) is purely aesthetic. I chose `hsl(0, 0, 93)` for the basemap.

V. Discussion and Critique

In this research I sought to investigate the relationship between redlining and urban heat concentration. I tried to determine to what extent redlining and other forms of ecological segregation like physical infrastructure could explain heat exposure in urban areas, and whether that trend changed over time. My findings show that, even today, there is a statistically significant positive increase in temperature when stepping down in HOLC grade. These differences have decreased over time from 1990 and 2000. I also found impervious surfaces, in particular non-road surfaces, are a major determinant of heating and that tree canopy is a major determinant of cooling.

I am proud of some of the technological accomplishments this research accomplished. The project presented novel forms of data visualizations and interaction that allow users to view any HOLC city in-depth. In particular, I think the pixel filtering is a good step for people who want to explore the data for their own localities rather than see statistical aggregations. Moreover, past research in this area does not filter images by a custom cloud cover percentage.

A major issue this research overlooks is the current demographics of HOLC polygons. (One can derive that from census tracts using dasymetric reaggregation [45].) Some formerly redlined areas have gone through "urban renewal" and slum-clearing, i.e. displacement and gentrification, which likely confounds some of the results of this paper, especially correlations with tree canopy.

Secondly, this research only conducts statistical analyses at the nation and city levels. This likely hides, exaggerates, or distorts many patterns in the data. [7] disaggregates their analysis by region (e.g. midwest, northeast), which already teases out some interesting trends. Future research should attempt a more meaningful and precise disaggregation, such as by the different kinds of policies by which individual cities have contributed to heat inequality. In their discussion [7] offer good starting points with regards to this. The disaggregation discussed here applies to visualization as well.

Another issue of this project is its lack of colorblindness or accessibility research. Initially, I had included the hue offset slider tool at the top of the website to allow users to dynamically choose the $h°$ offset of the maps. This was possible because of CSS' `hue-rotate` filter function, which purports to rotate the hue of any image. However, the saturation and lightness of the images were being unreasonably distorted. I learned that CSS does not do true hue rotation (i.e. convert RGB to HSV, rotate, then convert HSV back to RGB), but rather uses a matrix transformation to approximate it. As the RGB to HSV transformation is nonlinear, this approximation was wholly inadequate for my use case. Future iterations of the maps could implement a true hue rotation with WebGL as to continue to take advantage of GPU computation.

VI. Code

All the code, data, and statistical analyses for this project is at https://github.com/jsonkao/redlining-heat. The README explains the directory structure.

References

[1] C. Sarrat, A. Lemonsu, V. Masson, and D. Guedalia, "Impact of urban heat island on regional atmospheric pollution," *Atmospheric Environment*, vol. 40, no. 10, pp. 1743–1758, Mar. 2006.

[2] J. Tan, Y. Zheng, X. Tang, C. Guo, L. Li, G. Song, X. Zhen, D. Yuan, A. J. Kalkstein, F. Li, and H. Chen, "The urban heat island and its impact on heat waves and human health in Shanghai," *International Journal of Biometeorology*, vol. 54, no. 1, pp. 75–84, Jan. 2010.

[3] Y. Zhou and J. M. Shepherd, "Atlanta's urban heat island under extreme heat conditions and potential mitigation strategies," *Natural Hazards*, vol. 52, no. 3, pp. 639–668, Mar. 2010.

[4] J. A. Patz, D. Campbell-Lendrum, T. Holloway, and J. A. Foley, "Impact of regional climate change on human health," *Nature*, vol. 438, no. 7066, pp. 310–317, Nov. 2005.

[5] N. Debbage and J. M. Shepherd, "The urban heat island effect and city contiguity," *Computers, Environment and Urban Systems*, vol. 54, pp. 181–194, Nov. 2015.

[6] J. Voelkel, D. Hellman, R. Sakuma, and V. Shandas, "Assessing Vulnerability to Urban Heat: A Study of Disproportionate Heat Exposure and Access to Refuge by Socio-Demographic Status in Portland, Oregon," *International Journal of Environmental Research and Public Health*, vol. 15, no. 4, p. 640, Apr. 2018.

[7] J. S. Hoffman, V. Shandas, and N. Pendleton, "The Effects of Historical Housing Policies on Resident Exposure to Intra-Urban Heat: A Study of 108 US Urban Areas," *Climate*, vol. 8, no. 1, p. 12, Jan. 2020.

[8] R. Nelson, L. Winling, R. Marciano, and N. Connolly, "Mapping Inequality," https://dsl.richmond.edu/panorama/redlining/.

[9] "Shelley v. Kraemer," 1948.

[10] "Fair Housing Act of 1968," 42 U.S.C.§ 3604.

[11] B. Mitchell and J. Franco, "HOLC "Redlining" Maps: The Persistent Structure Of Segregation And Economic Inequality," National Community Reinvestment Coalition, Tech. Rep., Mar. 2018.

[12] A. Nardone, K. E. Rudolph, R. Morello-Frosch, and J. A. Casey, "Redlines and Greenspace: The Relationship between Historical Redlining and 2010 Greenspace across the United States," *Environmental Health Perspectives*, vol. 129, no. 1, Jan. 2021.

[13] Jesdale Bill M., Morello-Frosch Rachel, and Cushing Lara, "The Racial/Ethnic Distribution of Heat Risk–Related Land Cover in Relation to Residential Segregation," *Environmental Health Perspectives*, vol. 121, no. 7, pp. 811–817, Jul. 2013.

[14] N. G. S. F. Center, "Moderate Resolution Imaging Spectroradiometer (MODIS) - LAADS DAAC," https://ladsweb.modaps.eosdis.nasa.gov/missions-and-measurements/modis/.

[15] National Oceanic and Atmospheric Administration, "Advanced Baseline Imager (ABI) — GOES_R Series," https://www.goes-r.gov/spacesegment/abi.html, Aug. 2019.

[16] N. L. Science, "Landsat 4," https://landsat.gsfc.nasa.gov/landsat-4-5/landsat-4.

[17] ——, "The Thematic Mapper - Landsat Science," https://landsat.gsfc.nasa.gov/landsat-4-5/tm.

[18] A. Press, "EROS center to help design new satellite," *Sioux City Journal*, Jan. 2006.

[19] N. L. Science, "Landsat 7 ETM+ — Landsat Science," https://landsat.gsfc.nasa.gov/landsat-7/landsat-7-etm.

[20] N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore, "Google Earth Engine: Planetary-scale geospatial analysis for everyone," *Remote Sensing of Environment*, vol. 202, pp. 18–27, Dec. 2017.

[21] U. G. Survey, "Landsat Collection 1 Level-1 Quality Assessment Band," https://www.usgs.gov/core-science-systems/nli/landsat/landsat-collection-1-level-1-quality-assessment-band?qt-science_support_page_related_con=0#qt-science_support_page_related_con.

[22] J. A. Sobrino, J. C. Jimenez-Munoz, G. Soria, M. Romaguera, L. Guanter, J. Moreno, A. Plaza, and P. Martinez, "Land Surface Emissivity Retrieval From Different VNIR and TIR Sensors," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 2, pp. 316–327, Feb. 2008.

[23] E. Lacerda, "Google Earth Engine Toolbox," Apr. 2021.

[24] J.-F. Pekel, A. Cottam, N. Gorelick, and A. S. Belward, "High-resolution mapping of global surface water and its long-term changes," *Nature*, vol. 540, no. 7633, pp. 418–422, Dec. 2016.

[25] M. Bostock, "Mbostock/ndjson-cli," Mar. 2021.

[26] X. Li, Y. Zhou, G. R. Asrar, M. Imhoff, and X. Li, "The surface urban heat island response to urban expansion: A panel analysis for the conterminous United States," *Science of The Total Environment*, vol. 605-606, pp. 426–435, Dec. 2017.

[27] D. N. Archer, "'White Men's Roads Through Black Men's Homes': Advancing Racial Equity Through Highway Reconstruction," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3539889, Feb. 2020.

[28] M. Zuk, A. H. Bierbaum, K. Chapple, K. Gorska, and A. Loukaitou-Sideris, "Gentrification, Displacement, and the Role of Public Investment," *Journal of Planning Literature*, vol. 33, no. 1, pp. 31–44, Feb. 2018.

[29] L. Yang, S. Jin, P. Danielson, C. Homer, L. Gass, S. M. Bender, A. Case, C. Costello, J. Dewitz, J. Fry, M. Funk, B. Granneman, G. C. Liknes, M. Rigge, and G. Xian, "A new generation of the United States National Land Cover Database: Requirements, research priorities, design, and implementation strategies," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 146, pp. 108–123, Dec. 2018.

[30] R. Team, "RStudio: Integrated Development Environment for R," RStudio, PBC., Boston, MA, 2020.

[31] M. Bloch, "Mapshaper," Mar. 2021.

[32] "NPM Installation - Google Earth Engine," https://developers.google.com/earth-engine/guides/npm_install.

[33] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org," 2017.

[34] J. Kao, "Spectate: Providing a workflow for building and publishing freeform stories," Columbia Daily Spectator Graphics Desk, Apr. 2021.

[35] F. K. Schott, "Snowpack," 2021.

[36] H. Wickham, *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[37] G. F. Jenks and U. o. K. D. of Geography, *Optimal Data Classification For Choropleth Maps*, ser. University of Kansas Department of Geography Occasional Paper. University of Kansas, 1977, vol. 2.

[38] H. Wang and M. Song, "Ckmeans.1d.dp: Optimal k-means Clustering in One Dimension by Dynamic Programming," *The R Journal*, vol. 3, no. 2, p. 29, 2011.

[39] J. A. Hartigan and M. A. Wong, "A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[40] T. MacWright, "Reimplement Jenks · Issue #117 · simple-statistics/simple-statistics," https://github.com/simple-statistics/simple-statistics/issues/117, Jul. 2015.

[41] A. Grønlund, K. G. Larsen, A. Mathiasen, J. S. Nielsen, S. Schneider, and M. Song, "Fast Exact k-Means, k-Medians and Bregman Divergence Clustering in 1D," *arXiv:1701.07204 [cs]*, Apr. 2018.

[42] J. Stephens, "Bivariate Choropleth Maps: A How-to Guide," https://www.joshuastevens.net/cartography/make-a-bivariate-choropleth-map/, Feb. 2015.

[43] GDAL/OGR contributors, "GDAL/OGR Geospatial Data Abstraction software Library," Open Source Geospatial Foundation, 2021.

[44] A. Kirk, "Bivariate choropleth maps," Jun. 2017.

[45] D. Lu, "Dasymetric reaggregation using Mapshaper," https://medium.com/@DeniseDSLu/dasymetric-reaggregation-using-mapshaper-218e87babaa3, Mar. 2020.

VII. Appendix

The following code depends on a variable, `holc_vectors`, that is only accessible within the Earth Engine Code Editor. The script and variable are available on the platform here. Accessing the platform requires signing up as an Earth Engine user.

```
var geet = require('users/jasonkao/landsat:geet');

var l8 = ee.ImageCollection("LANDSAT/LC08/C01/T1");
var l7 = ee.ImageCollection("LANDSAT/LE07/C01/T1");
var l5 = ee.ImageCollection("LANDSAT/LT05/C01/T1");
var nlcd = ee.ImageCollection("USGS/NLCD_RELEASES/2016_REL");

/* Temperature computations */

function getQABits(image, start, end, newName) {
  // Compute the bits we need to extract.
  var pattern = 0;
  for (var i = start; i <= end; i++) {
    pattern += Math.pow(2, i);
  }
  // Return a single band image of the extracted QA bits, giving the band
  // a new name.
  return image.select([0], [newName]).bitwiseAnd(pattern).rightShift(start);
}

function calculateMeans(collection, lst_calc, year) {
  // Filter images by time and cloud cover
  var images = collection
    .filterBounds(holc_vectors.geometry())
    .filterDate(year + '-06-01', year + '-08-31')
    .map(function (image) {
      var bqa = getQABits(image.select('BQA'), 4, 4, 'cloud');
      var reducers = ee.Reducer.sum().combine({
        reducer2: ee.Reducer.count(),
        sharedInputs: true
      });
      var stats = bqa
      .reduceRegion({
        reducer: reducers,
        geometry: holc_vectors.filterBounds(
          ee.Geometry.Polygon(ee.Geometry(image.get('system:footprint')).coordinates
    ())
        ),
        scale: 30,
      })
      return image.set({
        proportionCloud: ee.Number(stats.get('cloud_sum')).divide(ee.Number(stats.get
    ('cloud_count')))
      });
    })
    .filter(ee.Filter.lte('proportionCloud', 0.4));
  // Map LST computation across images. Take the mean for overlaps
  var composite_computed = images.map(function (image) {
    return lst_calc(image).select('LST').multiply(9/5).add(32);
  }).mean();

  var mask = ee
```

```
    .Image('JRC/GSW1_2/GlobalSurfaceWater')
    .select('max_extent')
    .eq(0);

  // Calculate and export a reduceRegions FeatureCollection
  var meanDictionary = composite_computed.mask(mask).reduceRegions({
    reducer: ee.Reducer.mean(),
    collection: holc_vectors,
    scale: 30
  });

  Export.table.toDrive({collection: meanDictionary, description: 'tempTask-' + year,
   fileFormat: 'GeoJSON'});
}

calculateMeans(l7, geet.lst_calc_ls7, 2000);
calculateMeans(l8, geet.lst_calc_ls8, 2020);
calculateMeans(l5, geet.lst_calc_ls5, 1990);

/* NLCD computations */

function getImpervious(year) {
  // Filter dataset to impervious descriptor of 2016 product.
  var nlcdYear = nlcd
    .filter(ee.Filter.eq('system:index', year))
    .first();
  var descriptor = nlcdYear.select('impervious_descriptor');
  var pct = nlcdYear.select('impervious').divide(100);

  var frequencies = descriptor.addBands(pct).reduceRegions({
    reducer: ee.Reducer.frequencyHistogram().splitWeights(),
    collection: holc_vectors
  });

  Export.table.toDrive({
    collection: frequencies,
    description: 'imperviousTask-' + year,
    fileFormat: 'GeoJSON'
  });
}

getImpervious('2016');

function getCanopy(year) {
  // Filter dataset to impervious descriptor of 2016 product.
  var nlcdYear = nlcd
    .filter(ee.Filter.eq('system:index', year))
    .first();
  var tree_cover = nlcdYear.select('percent_tree_cover');

  var frequencies = tree_cover.reduceRegions({
    reducer: ee.Reducer.frequencyHistogram(),
    collection: holc_vectors
  });

  Export.table.toDrive({
    collection: frequencies,
    description: 'treeTask-' + year,
    fileFormat: 'GeoJSON'
  });
}

getCanopy('2016');
```

Listing 1. GEE-flavored JavaScript Listing