Final Project: Storms and Global Warming

Aaron, Coco, Kevin

Bias

As biases are unavoidable, it is important to be aware of their influence on us and the work that we do, especially when it comes to predictive algorithms. The most common types of bias are interaction bias, latent (or prejudice) bias, and selection bias. In this article by Annie Brown, she discusses bias with Corey White and the impact that it has on marginalized communities. Since the majority of the individuals creating the algorithms are white men, they are frequently misrepresenting populations by using only white men as the standard in training data. This impacts facial recognition software that is being more frequently used by police forces to find "criminals" but is incorrectly identifying Black men and women as culprits when they do not match the suspects description. The use of historical data to train algorithms also poses issues as it does not reflect the current societal changes. By using this data, programs are learning that more men are doctors because historically this was true. Using outdated data is dangerous because if a program is used to look for the best candidates for a position it will be skewed in favor of men as historically they were the individuals who filled the positions. Data needs to be looked at by a diverse group of individuals to ensure there is appropriate representation in the way the algorithm is working.

Annie Brown: Biased Algorithms Learn from Biased Data

Problem Background

We are examining trends in temperature changes and how it relates to increasing storm strength and frequency. It is important to look at these changes as they impact everyone globally. As storm intensity and frequency increases due to rising temperatures, we are going to need to change the structural integrity of buildings, flight patterns, as well as expand the areas considered flood zones. Having an awareness of these changes will help everyone better prepare for stronger storms, and the use of data may help to convince those who are skeptical of global warming's impact.

Our goal is to use machine learning to train the algorithm to predict the potential number of storms per year based on both of the temperature changes on land and in the ocean. We hope to be able to do this by training the algorithm to correctly classify storms based on land and ocean temperatures as well as using temperature data and wind speed to predict future storms.

Data Scrubbing

#include <matlab.h>

The data-scrubbing script starts with a load function to keep all structure and table names consistent. It should, most importantly, return three structures named 'atlantic', 'pacific', and 'globaltemp'. The two ocean storm structures contain information on wind speed, pressure, wind direction, datetime, and naming conventions. Each function called will have a description in the local functions section of this livescript.

We start by removing negative values for real measurements and replacing them with NaN. Values scrubbed include negative wind speeds (note: speed not velocity) and negative pressures that seem to be placeholder

values. We also removed the names of the storm since we were already aware of the issue that storms with feminine names are taken less seriously. Referencing names of storms can also contribute to bias, as there is an expectation of severity when it comes to household names like Hurricane Katrina or Irma. The societal impacts of these storms can influence our judgment of their severity. Using storm IDs instead can keep storms catalogued bias-free. Additionally, we removed the field "Events" which displayed whether or not the storm made landfall, as storms that make landfall are inherently seen as stronger due to the destruction they can cause in populated areas.

Datetimes and storm classifications are split and one-hot encoded for easier data manipulation. Redundant features and rows with only NaN values are removed. Storm data is shortened significantly to only include one row per storm. Global temperature averages by month and year are appended to each storm structure. Names of storms are removed to avoid bias.

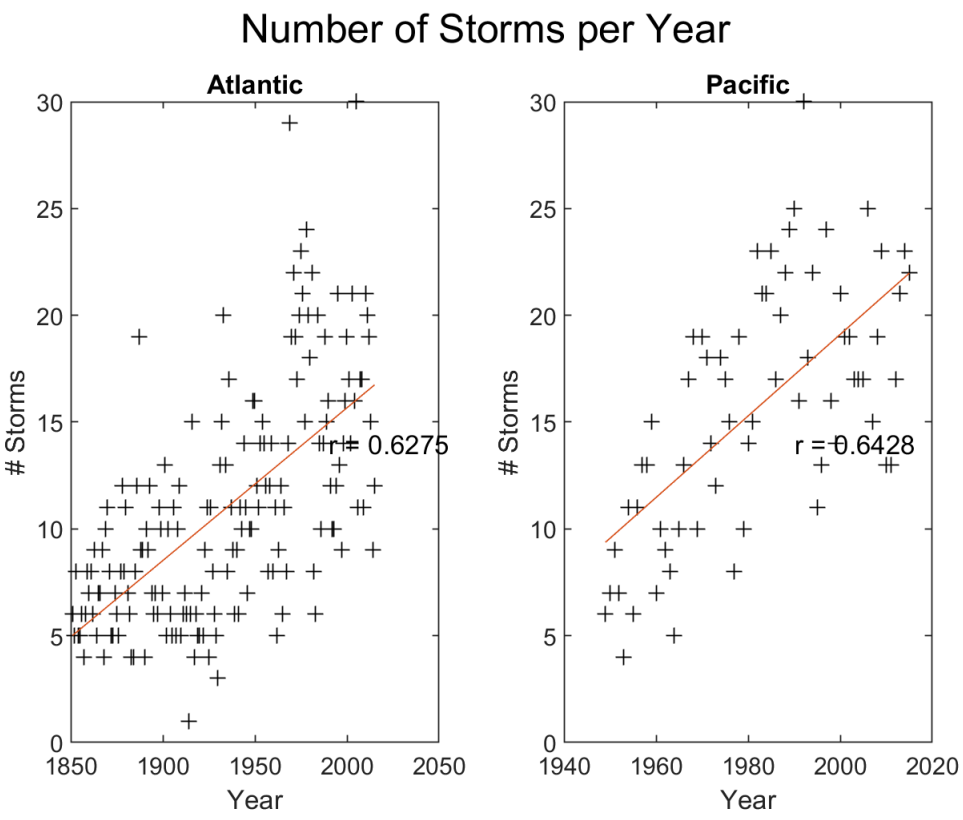
```
clear
clc

load finalprojdata.mat;

pacific = neinnine(pacific);
atlantic = neinnine(atlantic);
pacific = splittime(pacific);
atlantic = splittime(atlantic);
pacific = oceanscrub(pacific);
atlantic = oceanscrub(atlantic);
pacific = onespicyencoder(pacific,'Status');
atlantic = onespicyencoder(atlantic,'Status');
pacific = scrubbigwinds(pacific);
atlantic = scrubbigwinds(atlantic);
pacific = hurricane_cats(pacific);
atlantic = hurricane_cats(atlantic);
pacific = singlestorms(pacific);
atlantic = singlestorms(atlantic);

globaltemp = nanworldwide(globaltemp);
globaltemp = splitdatetime(globaltemp);

pacific = addtemps(pacific,globaltemp);
atlantic = addtemps(atlantic, globaltemp);
```

Exploratory Data Analysis

Below are fitted plots showing the number of storms per year in the Atlantic and Pacific ocean going as far back as their respective records allow. As with many physical quantities, the plots have a large and relatively random distribution. The plots do, however, show a clear upward trend. This data may be a bit skewed by our increased ability to track storms in recent years, as well as our decision in the 1950s to start tracking tropical depressions. However, we can not discount the trend.

```
clf
subplot(1,2,1)
frequencyperyear(atlantic, 'Atlantic');
subplot(1,2,2)
```

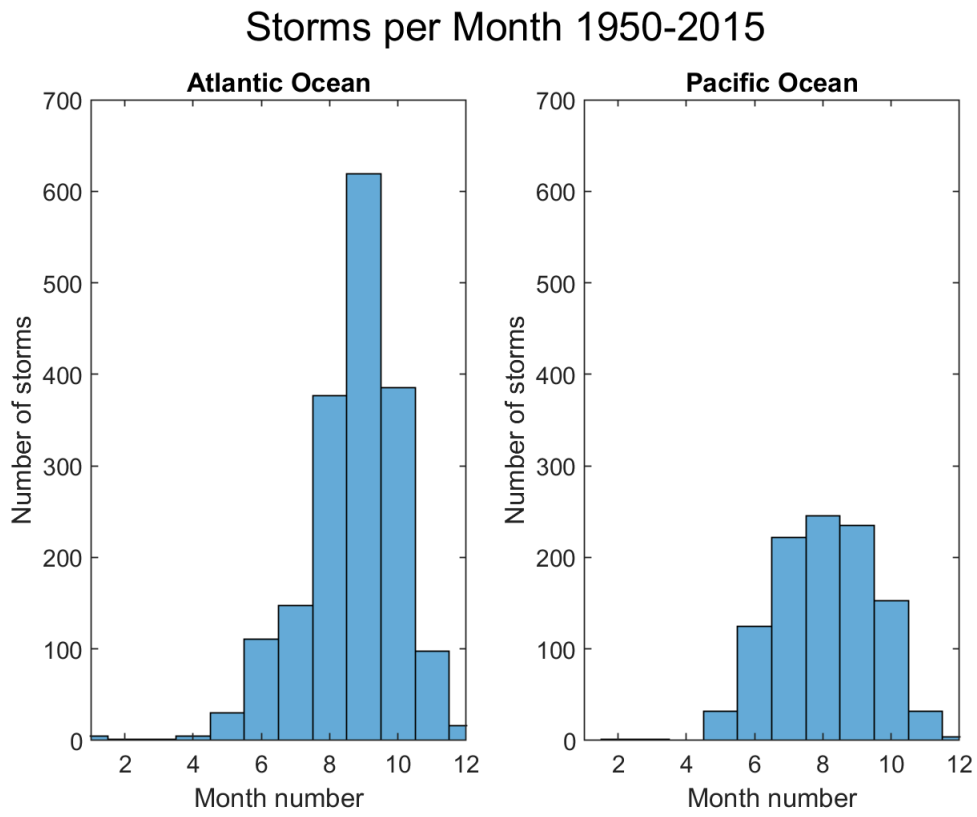
```
frequencyperyear(pacific, 'Pacific');
sgtitle('Number of Storms per Year');
```




```
clf
subplot(1,2,1);
totalhist(atlantic);
title('Atlantic Ocean');

subplot(1,2,2);
totalhist(pacific);
title('Pacific Ocean');
```

Below are histograms displaying the number of storms per month (listed numerically) logged in the period from 1950 to the end of the dataset. This period was chosen, as this is about when records start populating for the Pacific Ocean data structure. The y-axis has been normalized between the histograms to highlight the difference in overall frequency of storms. As you can see, the Atlantic Ocean has significantly more storms per any given time. Moreover, storms tend to occur during the Northern Hemisphere's summer months. This coincides with "Hurricane Season".

```
sgtitle('Storms per Month 1950-2015');
```

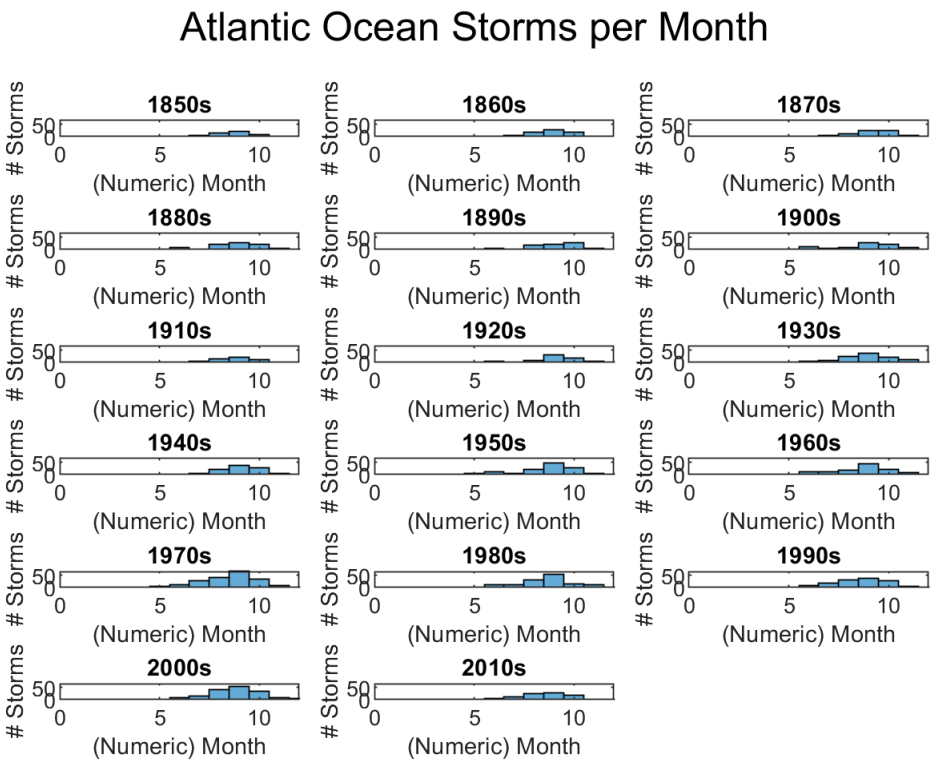



```
clf
tenhist(atlantic, 70, 6, 3);
```

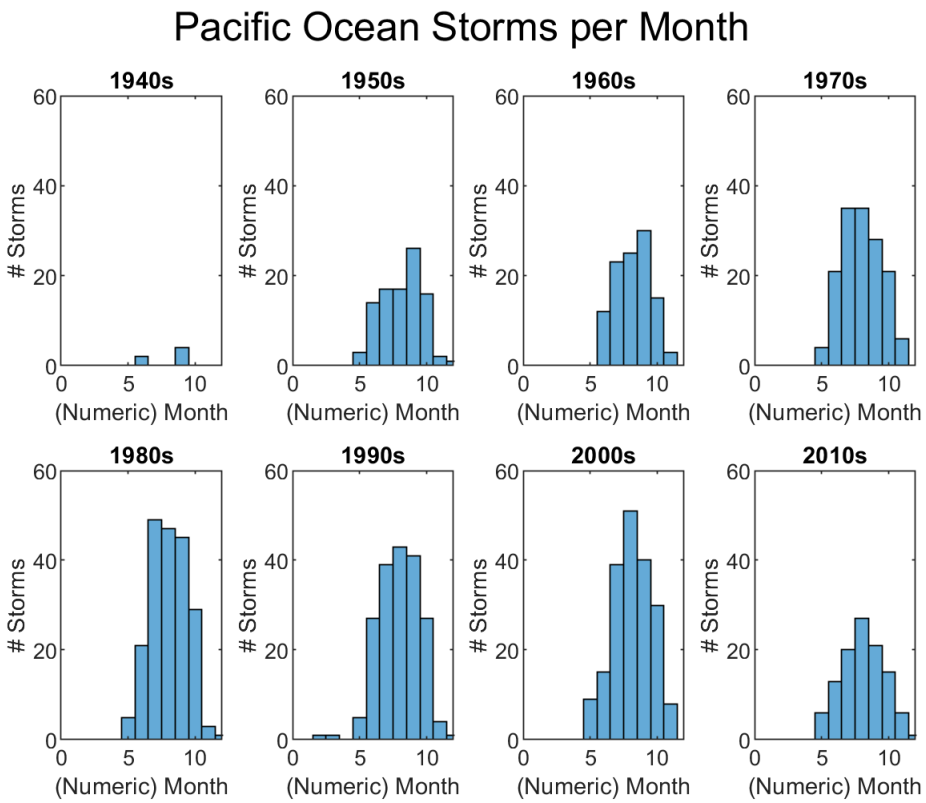
Below are more specific representations of the above data. Data are binned by month, and separated by decade as early as the respective data structures allow. It is once again clear that the number of storms logged overall is increasing over time.

It is important to note that since the category of Tropical Depression was added, it increases the number of overall storms recorded causing a noticable change between the 1960's and 1970's in the Atlantic Storm data.

```
sgtitle('Atlantic Ocean Storms per Month');
```

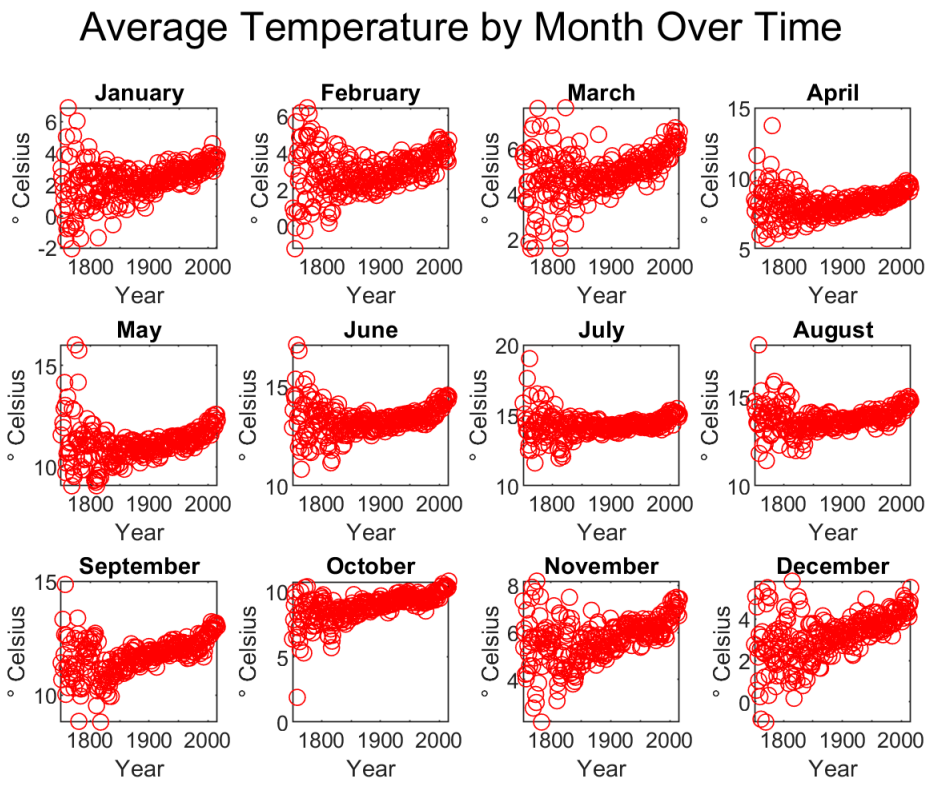



```
clf
tenhist(pacific, 60, 2, 4);
sgtitle('Pacific Ocean Storms per Month');
```



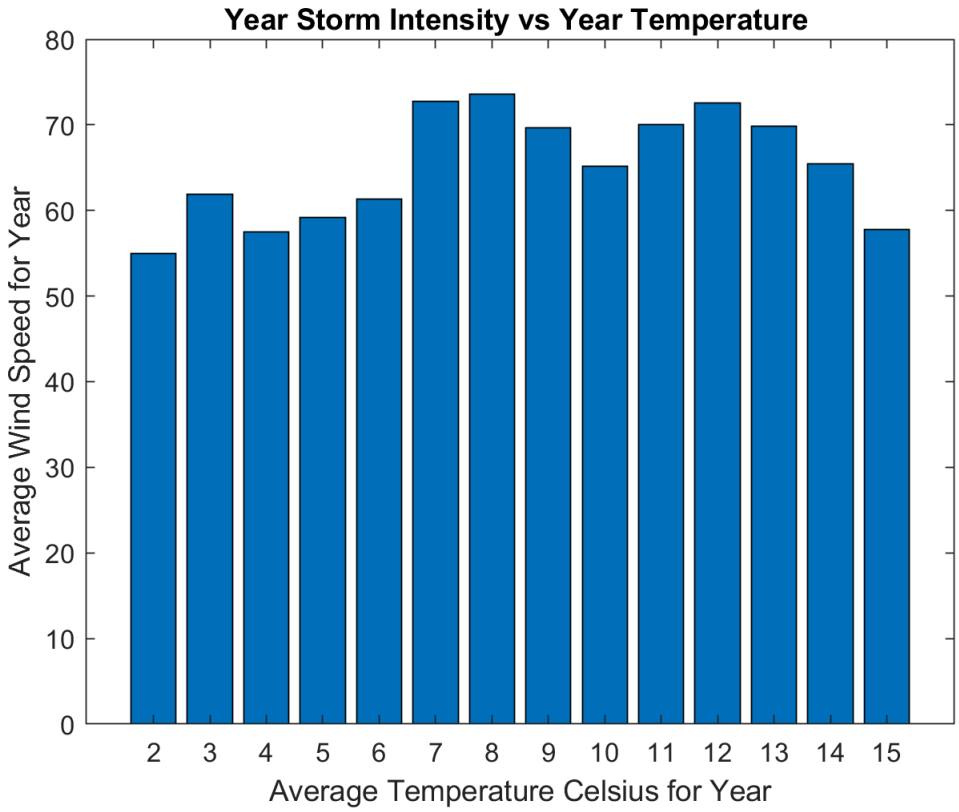
Below are the changes in monthly temperature over time. We wanted to see if there was a significant change in general temperatures and it is easy to see that there are for every single month. The x and y axis are not normalized to account for the differences between usual monthly temperatures.

```
clf
plotbymonth(globaltemp);
```



The graph below explores whether an increase temperature does relate to a higher wind speed. It does seem that the higher the yearly average temperature is, the faster the wind speeds are (i.e. there is a slight trend).

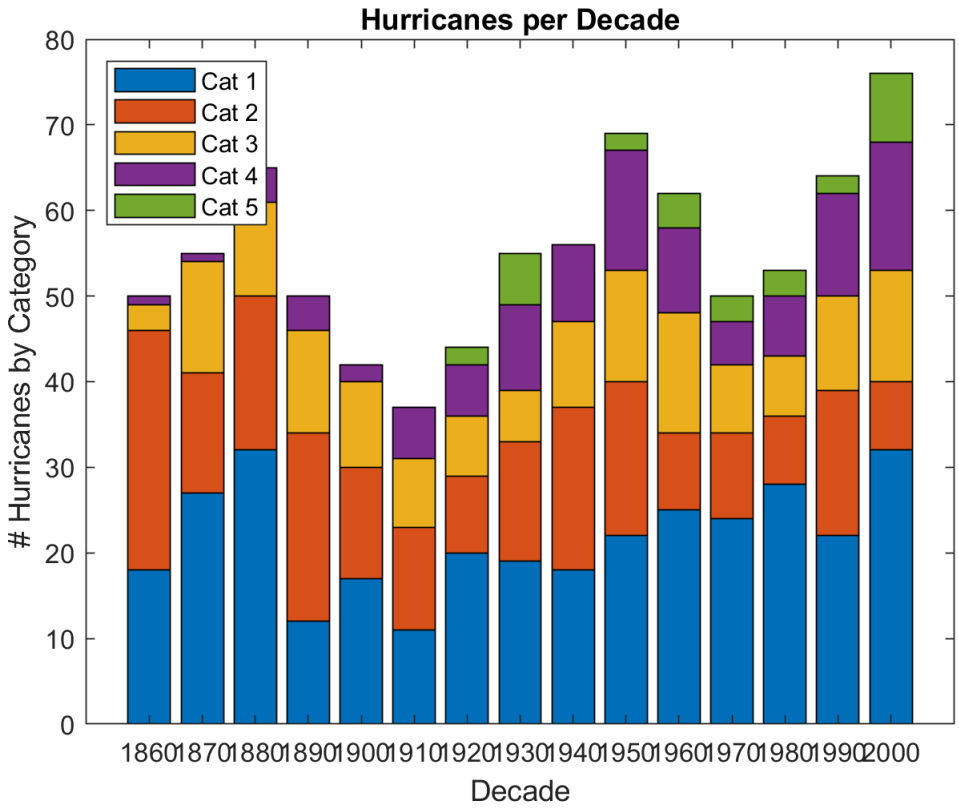
```
clf
intensevtemp(atlantic);
```



Data Analysis

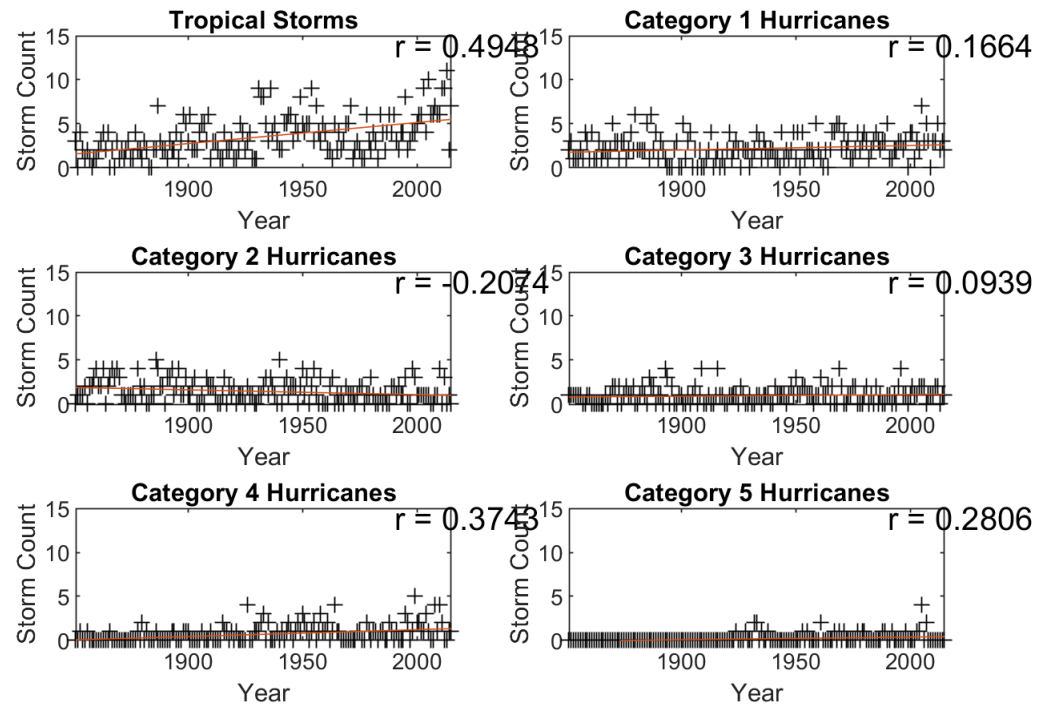
Below, we compared the number of hurricanes in each decade across all 5 categories. It is clear to see that there are an increasing number of category 5 hurricanes occurring every decade as global temperatures continue to increase. It must also be noted that data for the 2010's only goes up to 2015 so there are only half of the represented years in that decade compared to every other. This bin has been omitted.

```
clf
intensityovertime(atlantic);
```



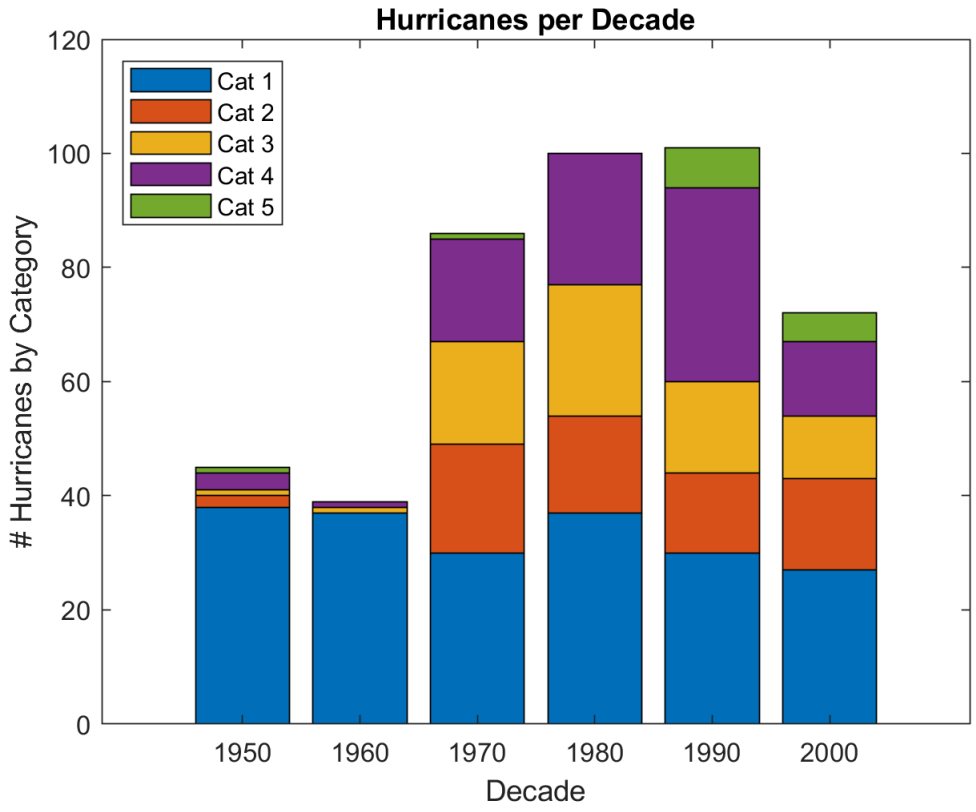
We broke the data down further and compared the occurrences of the storms below over time. We can see that there is a steady increase in the number of storms over time, especially Tropical Storms, Category 4, and Category 5. While it is important to note that as technologies has gotten better it is easier to capture more storm information, the number of stronger storms is increasing.

```
sgtitle('Frequency of Atlantic Storms and Intensities over Time')
```

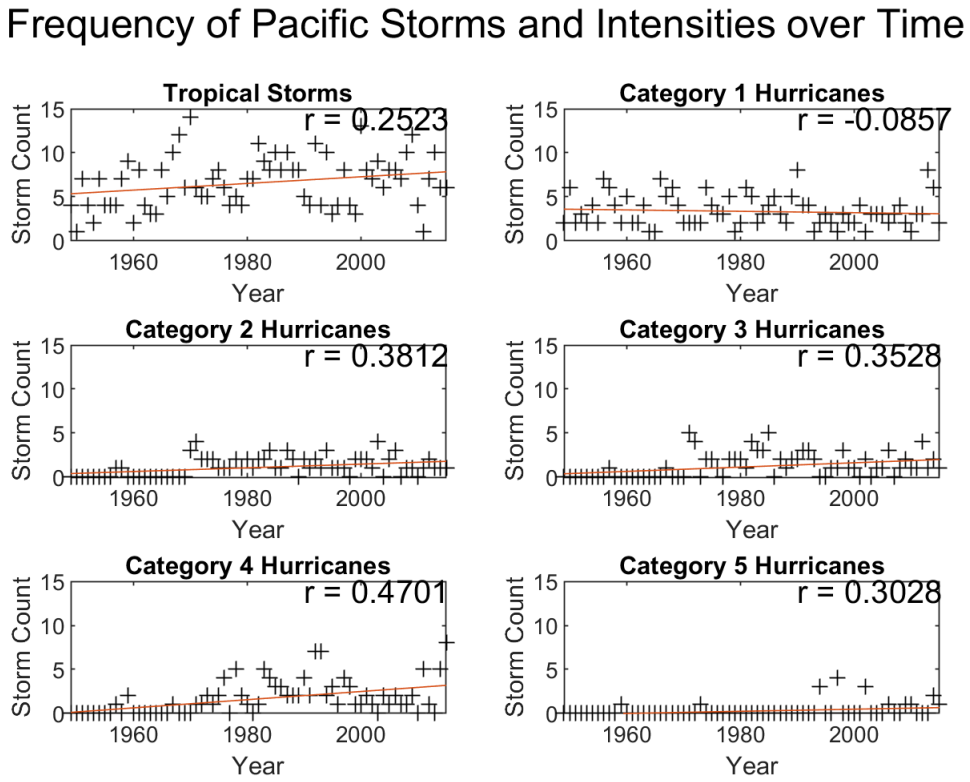


Below are the Pacific Ocean graphs for intensity and frequency.

```
intensityovertime(pacific);
```




```
sgtitle('Frequency of Pacific Storms and Intensities over Time')
```



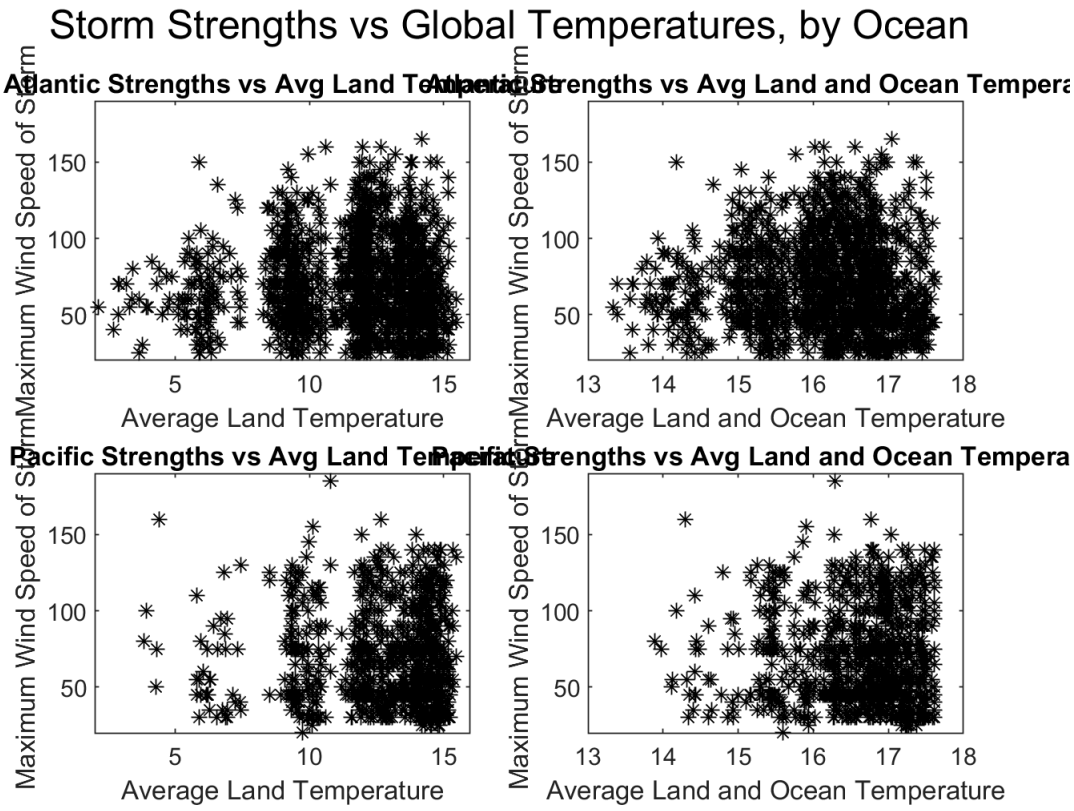
```
clear intensityovertime;
clf
```

We next wanted to understand how storm intensity (as indicated by maximum winds attained by the storm) and storm frequency behave as global temperatures change. To do so, we plotted the maximum wind speeds of each storm against global temperatures at the time of each storm. We created a total of four subplots for this analysis:

- Atlantic Storm Maximum Winds vs. Average Land Temperatures
- Atlantic Storm Maximum Winds vs. Average Land and Ocean Temperatures
- Pacific Storm Maximum Winds vs. Average Land Temperatures
- Pacific Storm Maximum Winds vs. Average Land and Ocean Temperatures

Looking at the scatterplots, one can see the occurence of storms with very high wind speeds increasing with global temperatures, implying that higher temperatures may provide conditions for stronger storms. Also, the body of all of the scatterplots becomes increasingly dense as the temperature increases from left to right in the plot, showing that the overall frequency of storms also tends to increase with increases in global temperatures.

```
subplot(2,2,1)

plot([atlantic.LandAverageTemperature],[atlantic.MaximumWind],'k*')
title('Atlantic Strengths vs Avg Land Temperature')
xlabel('Average Land Temperature')
ylabel('Maximum Wind Speed of Storm')
axis([2 16 20 190])

subplot(2,2,2)

plot([atlantic.LandAndOceanAverageTemperature],[atlantic.MaximumWind],'k*')
title('Atlantic Strengths vs Avg Land and Ocean Temperature')
xlabel('Average Land and Ocean Temperature')
ylabel('Maximum Wind Speed of Storm')
axis([13 18 20 190])

subplot(2,2,3)

plot([pacific.LandAverageTemperature],[pacific.MaximumWind],'k*')
title('Pacific Strengths vs Avg Land Temperature')
xlabel('Average Land Temperature')
ylabel('Maximum Wind Speed of Storm')
axis([2 16 20 190])

subplot(2,2,4)

plot([pacific.LandAndOceanAverageTemperature],[pacific.MaximumWind],'k*')
title('Pacific Strengths vs Avg Land and Ocean Temperature')
xlabel('Average Land and Ocean Temperature')
ylabel('Maximum Wind Speed of Storm')
axis([13 18 20 190])

sgtitle('Storm Strengths vs Global Temperatures, by Ocean')
```



Machine Learning

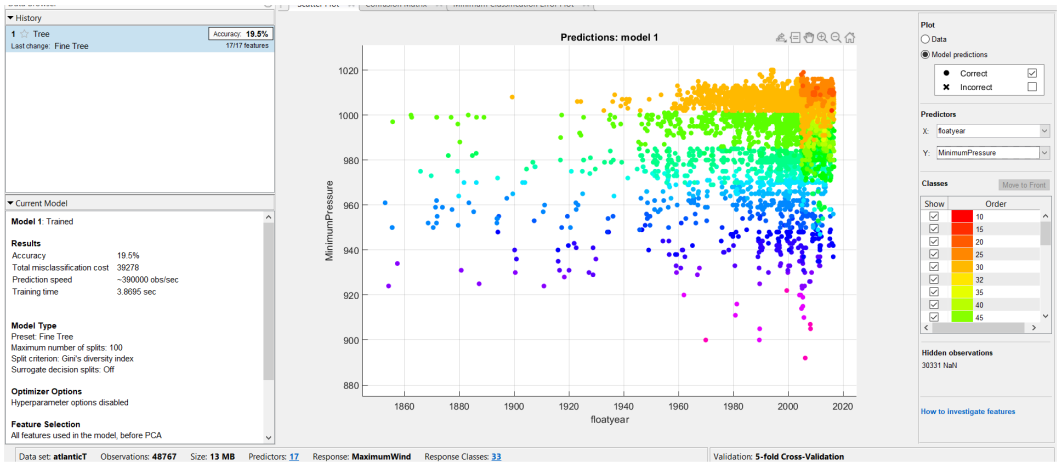
Classification Learner:

We looked into whether we could train the algorithm to accurately return maximum wind speed based off of minimum pressure in storms over time. To allow the algorithm to have a higher rate of accuracy, we adjusted our data scrubbing script to leave in the repeated measures of each storm, removing the function singlestorms. This gave the algorithm a wider variety of data points to analyze, potentially allowing for higher rates of accuracy.

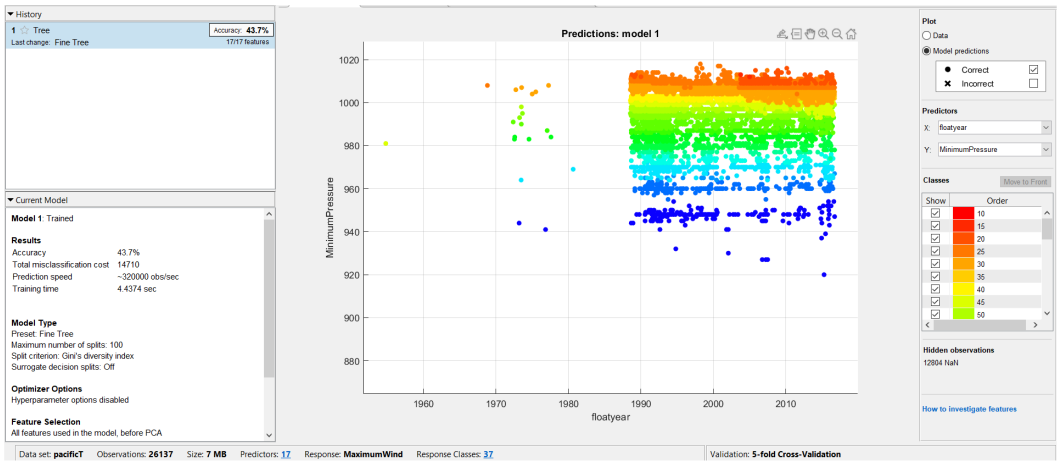
Once the data was changed from a structure to a table, we unselected the logically classified features, leaving only the numerical data for the computer to analyze and ran a fine tree training to evaluate the data.

Unfortunately, we only returned a 19.4% accuracy for Atlantic and 43.4% accuacy for Pacific as seen below:

```
clf
Acl = imread('Atlantic_TrainChange.png');
imshow(Acl, []);
```




```
clf
Pcl = imread('Pacific_TrainChange.png');
imshow(Pcl, []);
```



This is however an improvement to the initial tests with machine learning when we were only returning less than 10% accuracy for both sets of data.

Regression Analysis

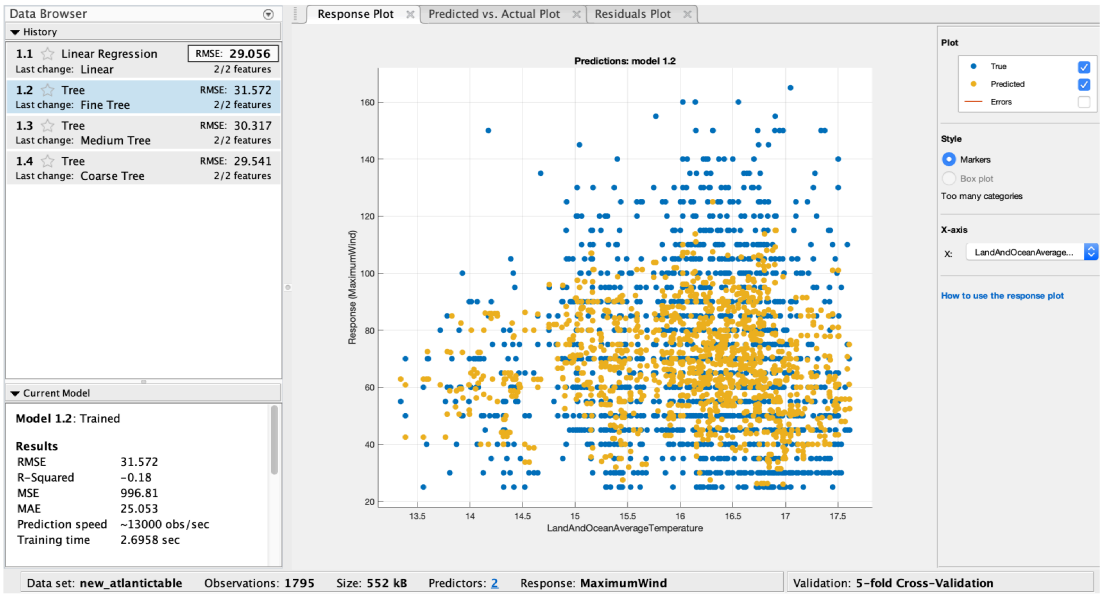
Having observed that both the frequency and intensity of storms tend to increase with increasing temperature, we decided to run our data through the Regression Learner in order to see how well we could actually predict a storm's expected strength given values for global temperature. We created separate prediction models for storms in the Atlantic and storms in the Pacific.

To generate these prediction models, we first converted the data structs we have been using for Atlantic and Pacific storms into tables using the struct2table function. We then started a session within the Regression Learner and imported the relevant table, selected "Maximum Winds" as the response variable, selected our predictor variables, opted for 5-fold Cross Validation, and started the session.

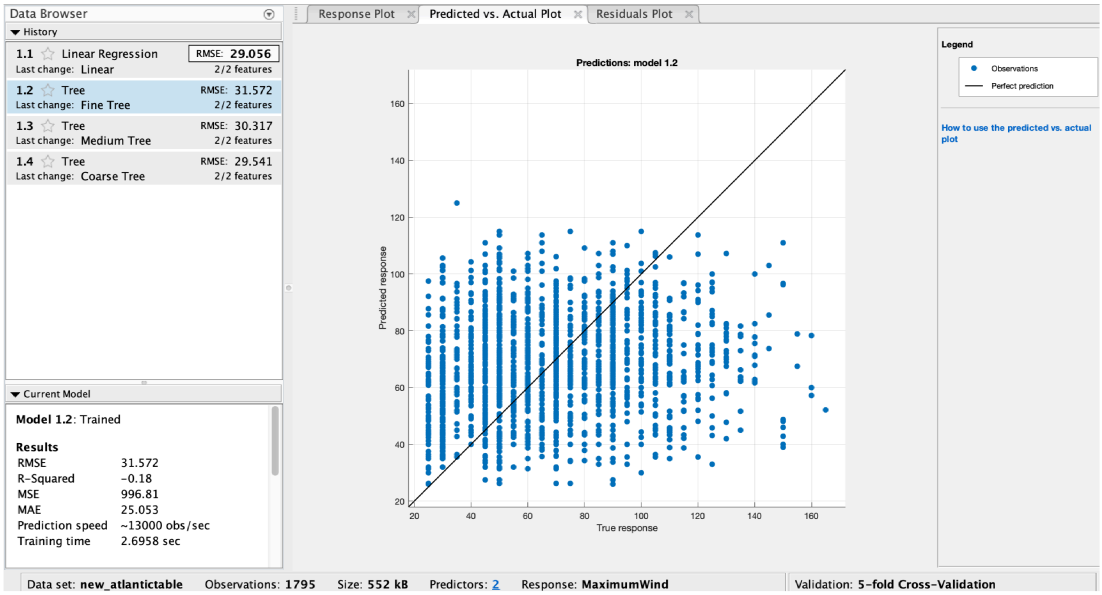
To decide which predictor variables we would ultimately use to create our model, we performed several iterations of regression learning using different combinations of predictor variables to see which combination would yield the prediction model with the highest accuracy. We first started with the predictor variables month, year, LandAverageTemperature, LandAverageTemperatureUncertainty, LandAndOceanAverageTemperature, and LandAndOceanAverageTemperatureUncertainty. However, through various trials we found that simply using the predictor variables of LandAverageTemperature and LandAndOceanAverageTemperature yielded the prediction model with the highest prediction accuracy.

We then ran the data through all of the Quick-To-Fit Models, for both the Atlantic and Pacific. Of the four models produced, the Fine Tree prediction model wound up being the most accurate, with an RMSE of 31.572 for Atlantic and 34.096 for Pacific. However, the overall accuracy was not very good, which can be seen in the Response, Prediction vs Actual, and Residual plots for each group.

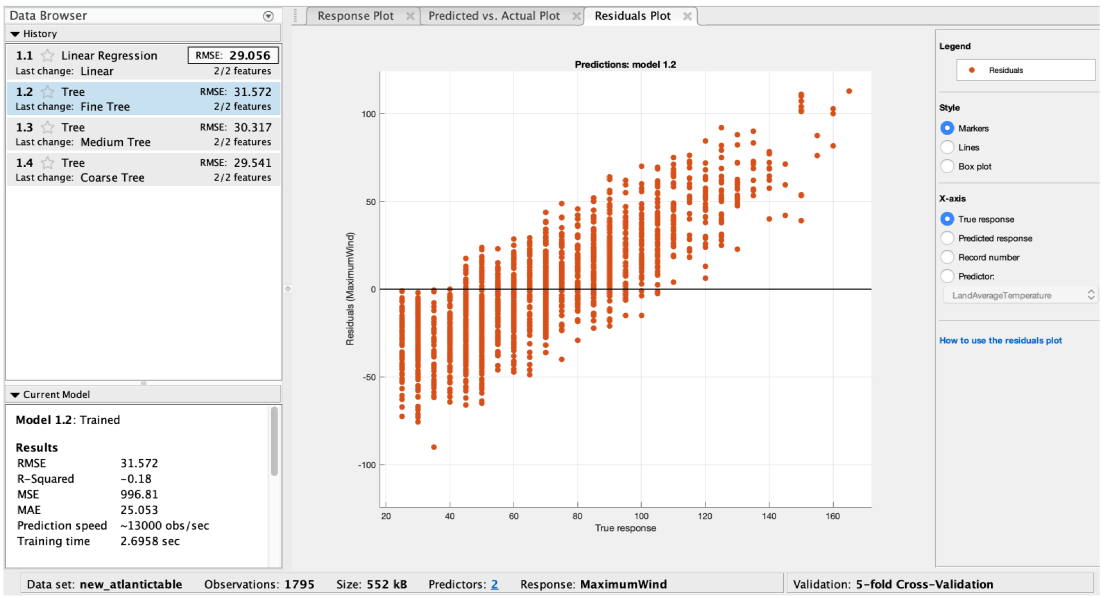
```
clf
A = imread('Atlantic_Response_Plot.png');
imshow(A,[]);
```



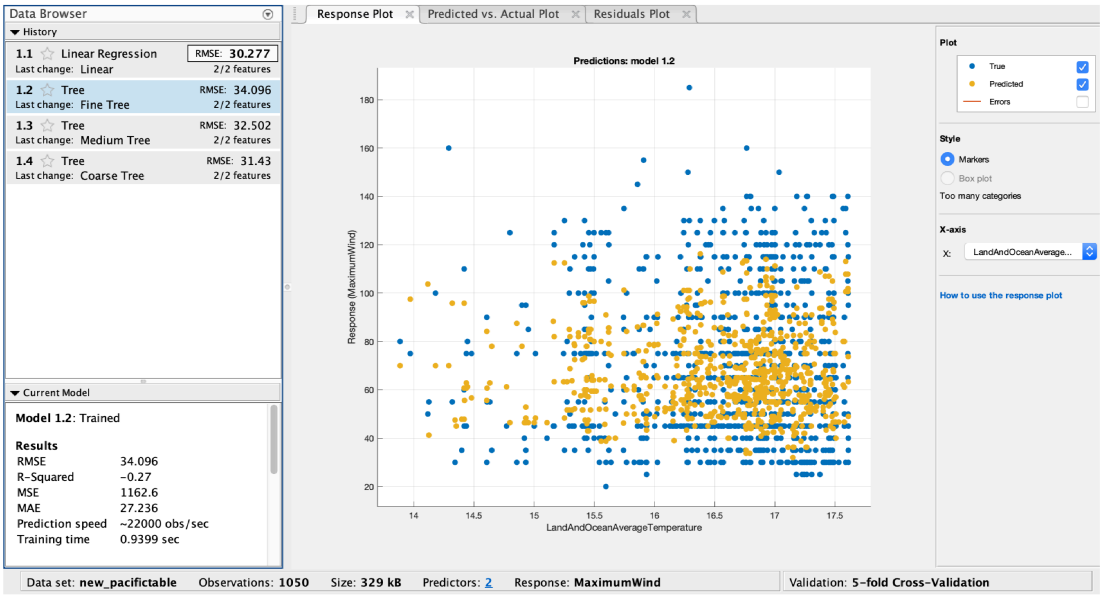

```
clf
B = imread('Atlantic_Prediction_Plot.png');
imshow(B,[]);
```



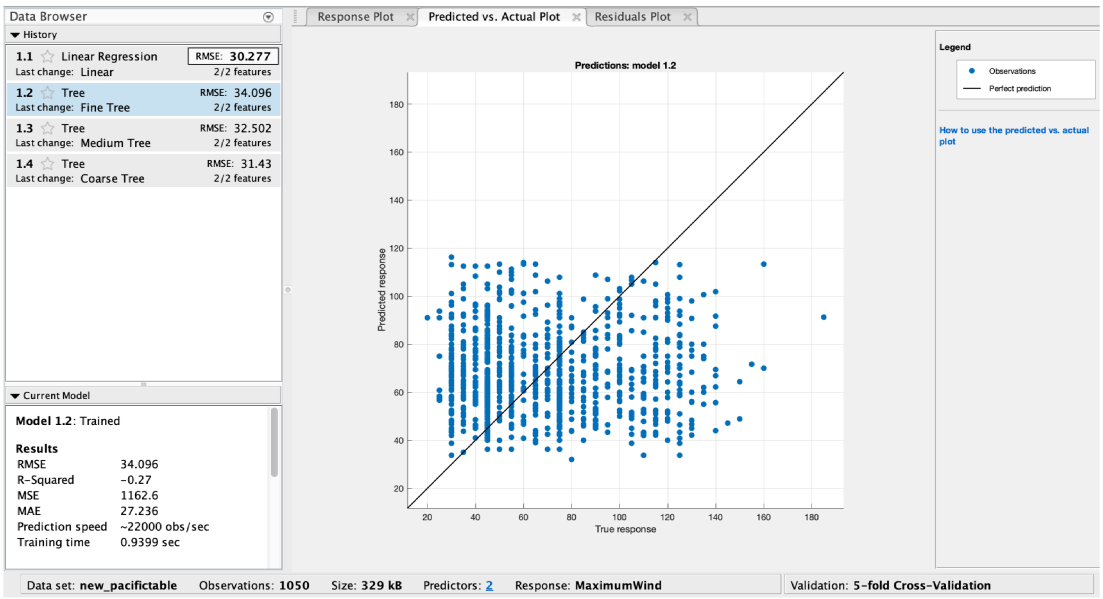

```
clf
C = imread('Atlantic_Residuals_Plot.png');
imshow(C,[]);
```



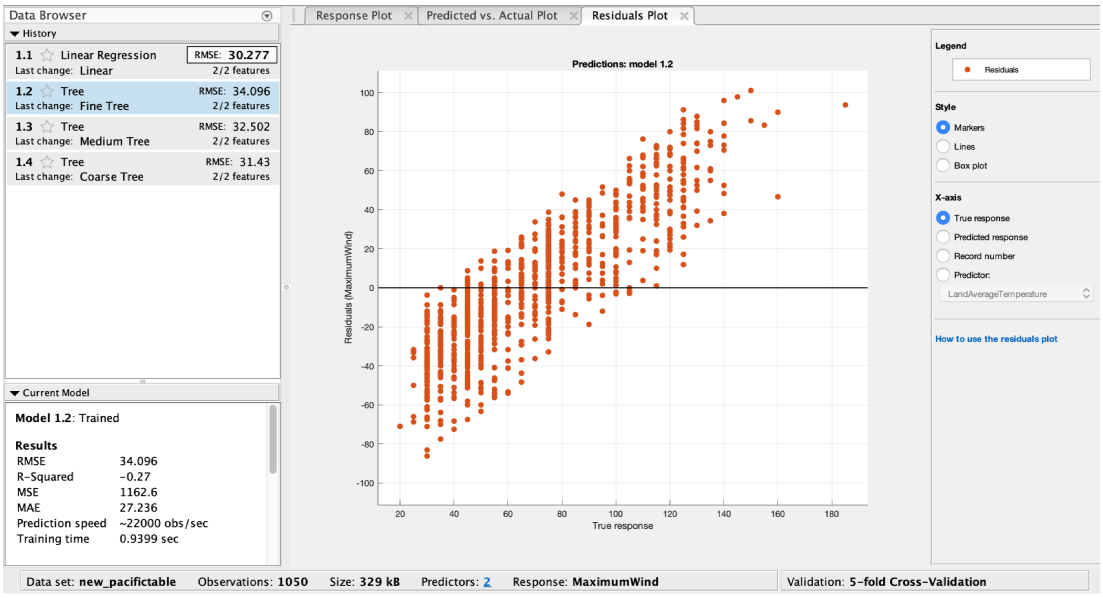

```
clf
D = imread('Pacific_Response_Plot.png');
imshow(D,[]);
```




```
clf
E = imread('Pacific_Prediction_Plot.png');
imshow(E,[]);
```




```
clf
F = imread('Pacific_Residuals_Plot.png');
imshow(F,[]);
```



Data Scrubbing Functions

Function nanworldwide removes the NaN values from the temperature data. Often, all the values in the row were NaN.

```
function outstruct = nanworldwide(structure)
index = 1;
for i = length(structure):-1:1
    if ~isnan(structure(i).LandAverageTemperature)
        outstruct(index) = structure(i);
        index = index + 1;
    end
end
end
```

Function splitdatetime operates on the datetime values present in globaltemp. It splits the values into features of class 'double' that store year and month, respectively.

```
function outstruct = splitdatetime(structure)
for i = 1:length(structure)
    structure(i).year = year(structure(i).dt);
    structure(i).month = month(structure(i).dt);
end
outstruct = structure;
end
```

Function neinnine removes the -99 and -999 values from storm structure wind speed and pressure, respectively. They are replaced with NaN.

```
function outstruct = neinnine(structure)
field_names = fieldnames(structure);
```

```
for i = 1:length(field_names)
    for j = 1:length(structure)
        if structure(j).(field_names{i}) < 0
            structure(j).(field_names{i}) = NaN;
        end
    end
end
outstruct = structure;
end
```

Function splittime operates on the strange datetime-like number string present in the storm structures. The function converts the time code to a character vector and splits it into its year, month and day. These are added as features to the structures, along with a feature 'floatyear' which is a decimal value made to easily plot continuous, unique time.

```
function outstruct = splittime(structure)
for i = 1:length(structure)
    str = int2str(structure(i).Date);
    year = str2double(str(1:4));
    month = str2double(str(5:6));
    day = str2double(str(7:8));
    time = structure(i).Time;
    structure(i).floatyear = (year + month/12 + day/(365) + time/2400); %splits into decimal ye
    structure(i).year = year;
    structure(i).month = month;
    structure(i).day = day;
end
outstruct = structure;
end
```

Function oceanscrub operates on storm data structures. It removes fields that have already been split into more usable values, and fields that are unused.

```
function outstruct = oceanscrub(instruct)
%Removes fields we won't use, or that have been split. Format for call is
%outstruct = oceanscrub(instruct)
fields = {'Name', 'Event', 'Date', 'Time', 'Latitude', 'Longitude'};
outstruct = rmfield(instruct, fields);
end
```

Function onespicyencoder operates on the categorical features in our storm structures. It one-hot encodes them into features for each unique value in the categorical feature. In this case, we use it to split the storms by category into Tropical Storms, Hurricanes, Tropical Depressions, et al. It removes the feature it used to one-hot encode.

```
function new_struct = onespicyencoder(struct,fieldname)
%Grab the unique categories of the struct field to be hot encoded
categories_ca = unique({struct.(fieldname)});

for i = 1:length(categories_ca)
    identifier = ['is' categories_ca{i}];
```

```
    for j = 1:length(struct)
       if struct(j).(fieldname) == categories_ca{i}
           struct(j).(identifier) = true;
       else
           struct(j).(identifier) = false;
       end
    end
end

struct = rmfield(struct,'Status');
new_struct = struct;

end
```

Function scrubbigwinds removes all rows from storm data that have a NaN value for maximum wind. Often, these rows had ALL NaN values, as maximum wind was the most rudimentary measurement.

```
function subset = scrubbigwinds(structure)
%gets rid of all rows that have NaNs for max wind

subset_index = 1;

for i = 1:length(structure)
    if ~isnan(structure(i).MaximumWind)
        subset(subset_index) = structure(i);
        subset_index = subset_index + 1;
    end
end
end
```

Function hurricane_cats one-hot encodes storms classified as hurricanes into Category 1-5 features. Wind speed is in knots, and thresholds have been placed to correctly classify storms.

```
function new_struct = hurricane_cats(structure)

cat_fields = {'isCat1HU', 'isCat2HU', 'isCat3HU', 'isCat4HU', 'isCat5HU'};

for j = 1:length(structure)
    if structure(j).MaximumWind >= 64 && structure(j).MaximumWind <= 82
        structure(j).(cat_fields{1}) = true;
        structure(j).(cat_fields{2}) = false;
        structure(j).(cat_fields{3}) = false;
        structure(j).(cat_fields{4}) = false;
        structure(j).(cat_fields{5}) = false;

    elseif structure(j).MaximumWind >= 83 && structure(j).MaximumWind <= 95
        structure(j).(cat_fields{1}) = false;
        structure(j).(cat_fields{2}) = true;
        structure(j).(cat_fields{3}) = false;
        structure(j).(cat_fields{4}) = false;
        structure(j).(cat_fields{5}) = false;

    elseif structure(j).MaximumWind >= 96 && structure(j).MaximumWind <= 112
```

```
        structure(j).(cat_fields{1}) = false;
        structure(j).(cat_fields{2}) = false;
        structure(j).(cat_fields{3}) = true;
        structure(j).(cat_fields{4}) = false;
        structure(j).(cat_fields{5}) = false;

    elseif structure(j).MaximumWind >= 113 && structure(j).MaximumWind <= 136
        structure(j).(cat_fields{1}) = false;
        structure(j).(cat_fields{2}) = false;
        structure(j).(cat_fields{3}) = false;
        structure(j).(cat_fields{4}) = true;
        structure(j).(cat_fields{5}) = false;

    elseif structure(j).MaximumWind >= 137
        structure(j).(cat_fields{1}) = false;
        structure(j).(cat_fields{2}) = false;
        structure(j).(cat_fields{3}) = false;
        structure(j).(cat_fields{4}) = false;
        structure(j).(cat_fields{5}) = true;
    else
        structure(j).(cat_fields{1}) = false;
        structure(j).(cat_fields{2}) = false;
        structure(j).(cat_fields{3}) = false;
        structure(j).(cat_fields{4}) = false;
        structure(j).(cat_fields{5}) = false;
    end
end

new_struct = structure;

end
```

Function singlestorms refines the data to about 1/20th of the original dataset. This is a significant amount of data to remove, but it is much more useful. Storms were being tracked every six hours over the span of their lifetime. Tracking a storm before it reaches its ultimate strength (e.g. watching a storm build up to a category 5 hurricane) would flood the data with many "storms" of low intensity, when they were in actuality very intense storms. We refined this data down to one row per storm--in particular, the row with the highest wind speed. This is most indicative of how intense the storm was.

```
function newstormstruct = singlestorms(structure)
%Takes storm data struct and returns a new struct that only has one record
%per storm ID, and this record represents the storm's strongest point

%Grab unique storm IDs and put into cell array
storm_ids = unique({structure.ID},'stable');
total_index = length(structure);

for i = length(storm_ids):-1:1
    bin_index = 1;

    %create a bin for the storm, and a vector that stores the max winds for the relevant storm

    while isequal(structure(total_index).ID, storm_ids{i}) && total_index > 1
```

```
        storm_bin(bin_index) = structure(total_index);
        bin_index = bin_index + 1;
        total_index = total_index - 1;
    end

    %find the index for the max wind values

    strongest_index = find([storm_bin.MaximumWind] == max([storm_bin.MaximumWind]));

    %use the index from the max wind value to choose the struct row
    %with the highest wind, since the storm bin and the max wind vectors will be the same lengt

    newstormstruct(i) = storm_bin(strongest_index(1));
    clear storm_bin

end
end
```

Function addtemps appends the average temperature data from 'globaltemp' to the storm structures 'atlantic' and 'pacific'. They are matched by month and year to the corresponding storm ID. This allows us to track the intensity and frequency of storms versus temperature.

```
function new_structure = addtemps(storm_structure,temps_structure)

rmfield(temps_structure, 'LandMaxTemperature');
rmfield(temps_structure, 'LandMaxTemperatureUncertainty');
rmfield(temps_structure, 'LandMinTemperature');
rmfield(temps_structure, 'LandMinTemperatureUncertainty');

for i = 1:length(temps_structure)
    for j = 1:length(storm_structure)
        if isequal(storm_structure(j).year,temps_structure(i).year) && isequal(storm_structure(
            storm_structure(j).LandAverageTemperature = temps_structure(i).LandAverageTemperatu
            storm_structure(j).LandAverageTemperatureUncertainty = temps_structure(i).LandAvera
            storm_structure(j).LandAndOceanAverageTemperature = temps_structure(i).LandAndOcean
            storm_structure(j).LandAndOceanAverageTemperatureUncertainty = temps_structure(i).L
            storm_structure(j).roundlandtemp = round(temps_structure(i).LandAverageTemperature)
            storm_structure(j).roundworldtemp = round(temps_structure(i).LandAndOceanAverageTem
        end
    end
end

new_structure = storm_structure;

end
```

Exploratory Data Analysis Functions

Function frequencyperyear takes a structure input and a character vector plot title input. The function runs through the storm structure year by year and counts how many storms occurred. This data is plotted as a scatter plot and fitted with a polyfit line of degree 1. The r value is displayed on the plot.

```
function frequencyperyear(structure, inpttitle)
```

```
x = [];
y = [];
for i = min([structure.year]):max([structure.year])
    counter = 0;
    for j = 1:length(structure)
        if structure(j).year == i
            counter = counter + 1;
        end
    end
    x = [x i];
    y = [y counter];
end

coefs = polyfit(x,y,1);
fit_line = polyval(coefs,x);
r = corrcoef(x,y);
r_text = sprintf("r = %.4f",r(1,2));
plot(x,y,'k+',x,fit_line)
title(inpttitle);
xlabel('Year');
ylabel('# Storms');
text((max([structure.year])-25),14,r_text,'FontSize',10)
end
```

Function intensevtemp bins average wind speed based on that year's average temperature in degrees celsius. The temperature codes called on are included in the scrubbing functions. The function loops through the range of temperatures and then through the structure, returning an average wind value for each temperature bin (sum of max winds divided by number of data utilized).

```
function intensevtemp(structure);
%Round the average land temperatures in a struct to the nearest integer,
%bin wind speeds into those groups.
x = unique([structure.roundlandtemp]);
y = zeros(1,length(x));
z = zeros(1,length(x));
avg = zeros(1,length(x));
for j = 1:length(x)
    for i = 1:length(structure)
        if structure(i).roundlandtemp == x(j)
            y(j) = y(j) + structure(i).MaximumWind;
            z(j) = z(j) + 1;
        end
    end
    avg(j) = y(j) / z(j);
end
bar(x,avg)
title('Year Storm Intensity vs Year Temperature ');
xlabel('Average Temperature Celsius for Year');
ylabel('Average Wind Speed for Year');

end
```

Function plotbymonth creates a plot (for each month of the year) of average land temperature over the range of time available to us in our data. No fit lines were plotted because the upward trend is easy to read for our purposes. The function separates relevant data by checking the month value in the structure.

```
function plotbymonth(structure)
titleca = {'January', 'February', 'March', 'April', 'May', 'June',...
    'July', 'August', 'September', 'October', 'November', 'December'};
for i = 1:12
    x = [];
    y = [];
    subplot(3,4,i);
    for j = 1:length(structure)
        if structure(j).month == i
            x = [x structure(j).dt];
            y = [y structure(j).LandAverageTemperature];
        end
    end
    plot(x,y,'ro');
    xlabel('Year');
    ylabel(sprintf("%c Celsius",char(176)));
    title(titleca{i});
end
sgtitle('Average Temperature by Month Over Time');
end
```

Function totalhist creates a histogram of all storms binned by month. The range of data goes back to 1950, as that is when the Pacific data starts to populate. The purpose of these plots is to compare Atlantic and Pacific data, so the range and ymax were fixed.

```
function totalhist(structure)
%Generates histogram of all storms by month for the last 69 years (nice)
histogram([structure.month]);
xlabel('Month number');
ylabel('Number of storms');
axis([1 12 0 700]);
end
```

Oh boy. The function tenhist creates histograms of all storms binned by month. This time, the data are separated by decade and subplotted according to a user-supplied subplot grid. The upper range is set by a user input, as well. The function creates unique datasets in the form of vectors, which allows the histogram function to operate on separate inputs. A switch-case is used to check the first 3 digits of the year code, separating data into decades. The function is a little messy, because it is one of the first ones we wrote! We could probably refine it down, but we think it's beautiful just the way it is.

```
function tenhist(structure, ymax, m, n)
% Generates histograms of storms per month in 10 year blocks. Bins are
% months. Specifies ymax for histograms, and subplot size (m x n)
eightfive = [];
eightsix = [];
eightseven = [];
eighteight = [];
```

```
eightnine = [];
ninek = [];
nineten = [];
ninetwo = [];
ninethree = [];
ninefour = [];
ninefive = [];
ninesix = [];
nineseven = [];
nineeight = [];
ninenine = [];
twok = [];
twokten = [];
for i = 1:length(structure)
    yr = int2str(structure(i).year);
    switch yr(1:3)
        case '185'
            eightfive = [eightfive structure(i).month];
        case '186'
            eightsix = [eightsix structure(i).month];
        case '187'
            eightseven = [eightseven structure(i).month];
        case '188'
            eighteight = [eighteight structure(i).month];
        case '189'
            eightnine = [eightnine structure(i).month];
        case '190'
            ninek = [ninek structure(i).month];
        case '191'
            nineten = [nineten structure(i).month];
        case '192'
            ninetwo = [ninetwo structure(i).month];
        case '193'
            ninethree = [ninethree structure(i).month];
        case '194'
            ninefour = [ninefour structure(i).month];
        case '195'
            ninefive = [ninefive structure(i).month];
        case '196'
            ninesix = [ninesix structure(i).month];
        case '197'
            nineseven = [nineseven structure(i).month];
        case '198'
            nineeight = [nineeight structure(i).month];
        case '199'
            ninenine = [ninenine structure(i).month];
        case '200'
            twok = [twok structure(i).month];
        case '201'
            twokten = [twokten structure(i).month];
        otherwise
            fprintf('Number outside of range\n');
    end
end
```

```
tempdata = {eightfive eightsix eightseven eighteight eightnine ninek nineten...
    ninetwo ninethree ninefour ninefive ninesix nineseven nineeight ninenine twok twokten};
temptitledata = ["1850s" "1860s" "1870s" "1880s" "1890s" "1900s" "1910s"...
    "1920s" "1930s" "1940s" "1950s", "1960s", "1970s", "1980s", "1990s", "2000s", "2010s"];
sum = 0;
```

Note: We're still in function tenhist! The function is now checking the size of the inputted data, as the amount of decades present is different between the two structures. The plotting and title data are reformed without any empty cells, so that the subplots fit together nicely.

```
for i = 1:length(tempdata)
    sum = sum + ~isempty(tempdata{i});
end
plotdata = cell(1,sum);
titledata = cell(1,sum);

idx = 1;
for i = 1:length(tempdata)
    if ~isempty(tempdata{i})
        plotdata{idx} = tempdata{i};
        titledata{idx} = temptitledata{i};
        idx = idx + 1;
    end
end

for i = 1:length(plotdata)
    subplot(m,n,i);
    histogram(plotdata{i});
    xlabel('(Numeric) Month');
    ylabel('# Storms');
    title(titledata(i));
    axis([0 12 0 ymax]);
end
end
```

Data Analysis Function

Function intensityovertime is a long one, and there will be text blocks throughout to explain it more thoroughly.

The function starts with a persistent figurecount which helps keep plots separate when being called. The main purpose of the first block is to count the number of Tropical Depressions, Tropical Storms, and Category 1-5 Hurricanes in the data. The counters are initialized and updated in a cell array that contains the fieldnames of the one-hot encoded features. Year-by-year, the counters are appended to a vector in order to log how many of each type of storm occurs. A vector of years is created for easy plotting.

```
function intensityovertime(structure)
persistent figurecount
if isempty(figurecount)
    figurecount = 1;
end
x = [];
```

```
TDs = [];
TSs = [];
cat1s = [];
cat2s = [];
cat3s = [];
cat4s = [];
cat5s = [];

for i = min([structure.year]):max([structure.year])

statuses_n_counters = {'isTD','isTS','isCat1HU','isCat2HU','isCat3HU','isCat4HU','isCat5HU';...
    0 0 0 0 0 0 0};

    for j = 1:length(structure)
        if structure(j).year == i
            for k = 1:length(statuses_n_counters)
                if structure(j).(statuses_n_counters{1,k}) == true
                    statuses_n_counters{2,k} = statuses_n_counters{2,k} + 1;
                end
            end
        end
    end

TDs = [TDs statuses_n_counters{2,1}];
TSs = [TSs statuses_n_counters{2,2}];
cat1s = [cat1s statuses_n_counters{2,3}];
cat2s = [cat2s statuses_n_counters{2,4}];
cat3s = [cat3s statuses_n_counters{2,5}];
cat4s = [cat4s statuses_n_counters{2,6}];
cat5s = [cat5s statuses_n_counters{2,7}];

x = [x i];

end
```

The data stored above is utilized here to create a stacked bar graph for each decade containing the number of each category hurricane. The vector 'decades' is created in a general form to ensure they are bins of decades which have complete data (i.e. skipping 1850s because data starts in 1851, skipping 2010s because data goes up to 2015), and to fit to the range of the two different structures.

The outer loop runs through the year range by decade, while the inner loop runs through the decades by increments of one year. This allows the function to bin each decade, while still running through each year.

```
hurricanes = [];
decades = (ceil(x(1)/10)*10):10:(floor(x(end)/10)*10 - 10); %bins by COMPLETE decades
idx = find(x==decades(1));
for i = decades(1):10:decades(end)
    a = 0;
    b = 0;
    c = 0;
    d = 0;
    e = 0;
    for j = 0:9
```

```
        if idx <= (find(x==decades(end))+10)
            a = a + cat1s(idx);
            b = b + cat2s(idx);
            c = c + cat3s(idx);
            d = d + cat4s(idx);
            e = e + cat5s(idx);
            idx = idx + 1;
        else
            a = [];
            b = [];
            c = [];
            d = [];
            e = [];
        end
    end
    hurricanes = [hurricanes; a b c d e];
end
figure(figurecount)
bar(decades,hurricanes,'stacked');
legend('Cat 1', 'Cat 2', 'Cat 3', 'Cat 4', 'Cat 5');
legend('Location', 'Northwest');
title('Hurricanes per Decade');
xlabel('Decade');
ylabel('# Hurricanes by Category');
figurecount = figurecount + 1;
```

This part of the function plots the number storms occurring within each intensity category over the range of years in each structure. In the for loop, the function uses a polyfit function of degree 1 to fit the trend, and displays the r value on the plots. A cell array is created which contains the vectors variables that capture the number of each category storm occuring each year. Titles are pulled from a string vector, and data is pulled from the cell-array.

```
cat_vars = {TSs cat1s cat2s cat3s cat4s cat5s};
titles = ["Tropical Storms","Category 1 Hurricanes", "Category 2 Hurricanes",...
    "Category 3 Hurricanes", "Category 4 Hurricanes", "Category 5 Hurricanes"];

figure(figurecount)
for i = 1:length(cat_vars)
coefs = polyfit(x,cat_vars{i},1);
fit_line = polyval(coefs,x);
r = corrcoef(x,cat_vars{i});
r_text = sprintf("r = %.4f",r(1,2));

subplot(3,2,i)
plot(x,cat_vars{i},'k+',x,fit_line)
xlabel('Year')
ylabel('Storm Count')
title(titles(i))
axis([min([structure.year]) max([structure.year]) 0 15])
text((max([structure.year])-25),14,r_text,'FontSize',12)
```

```
end
figurecount = figurecount + 1;
end
```