FINAL PROJECT by Vikramaditya

VIKRAMADITYA REDDY VARKALA

Z1973679

Data Visualization(CSCI 627)

2023-12-03

DataSet:Louisiana Road Home program

TASK 1

Whats the trend between Funds distributed and Damage caused by hurricane across each Parish?

Are there any regions that experienced severe damage but received comparatively less grants?

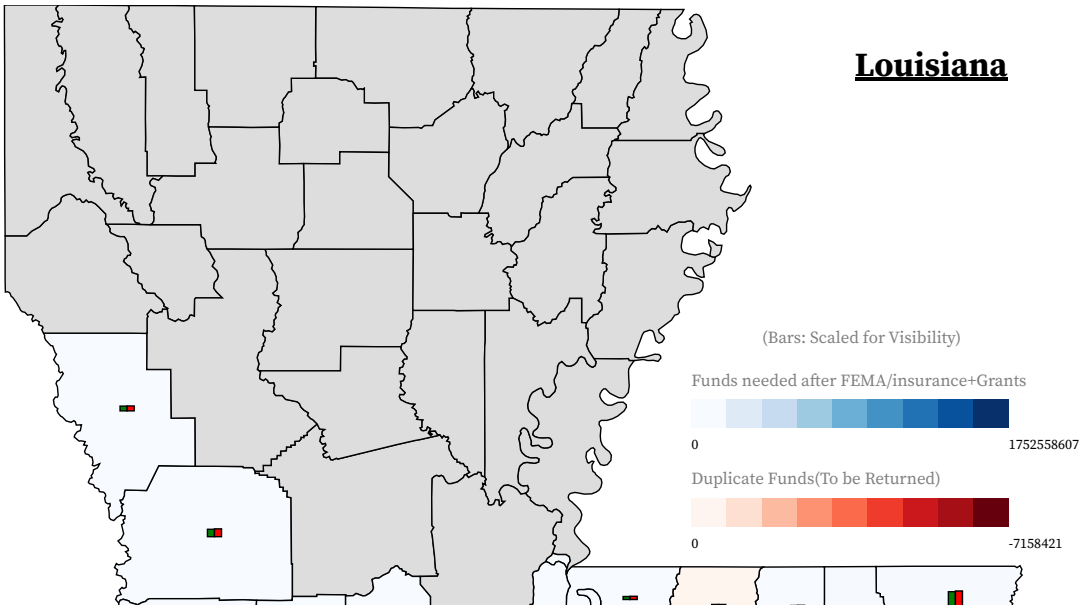
which structure type has the most damage among Parishes?

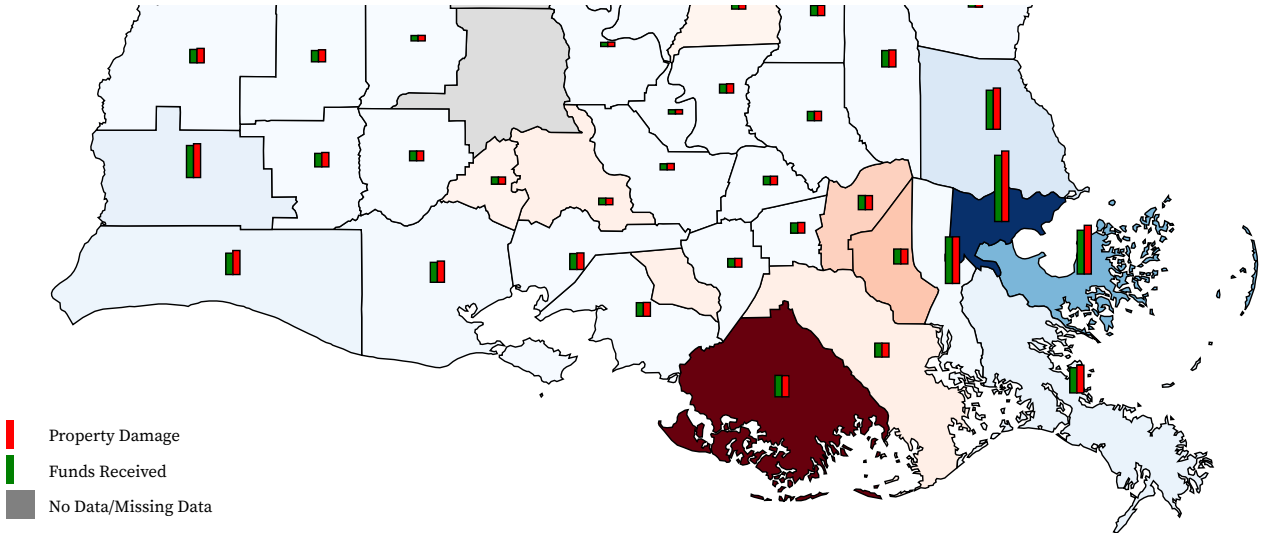
Viz Usage Description

Use the Slider to select the number of selections/clicks(parish comparisons), **click on the parish** to select parish and view different Structure damage comparisons on bar chart **click again** to deselect parish. **Hover over Grouped bars on map** to view funds received and Damage values. **Hover over parish** to view parish name and funds needed value .**Hover over bars on Bar chart** to view Parish Name and Structure Type Damage

No of Parishes to compare

2





TASK 2

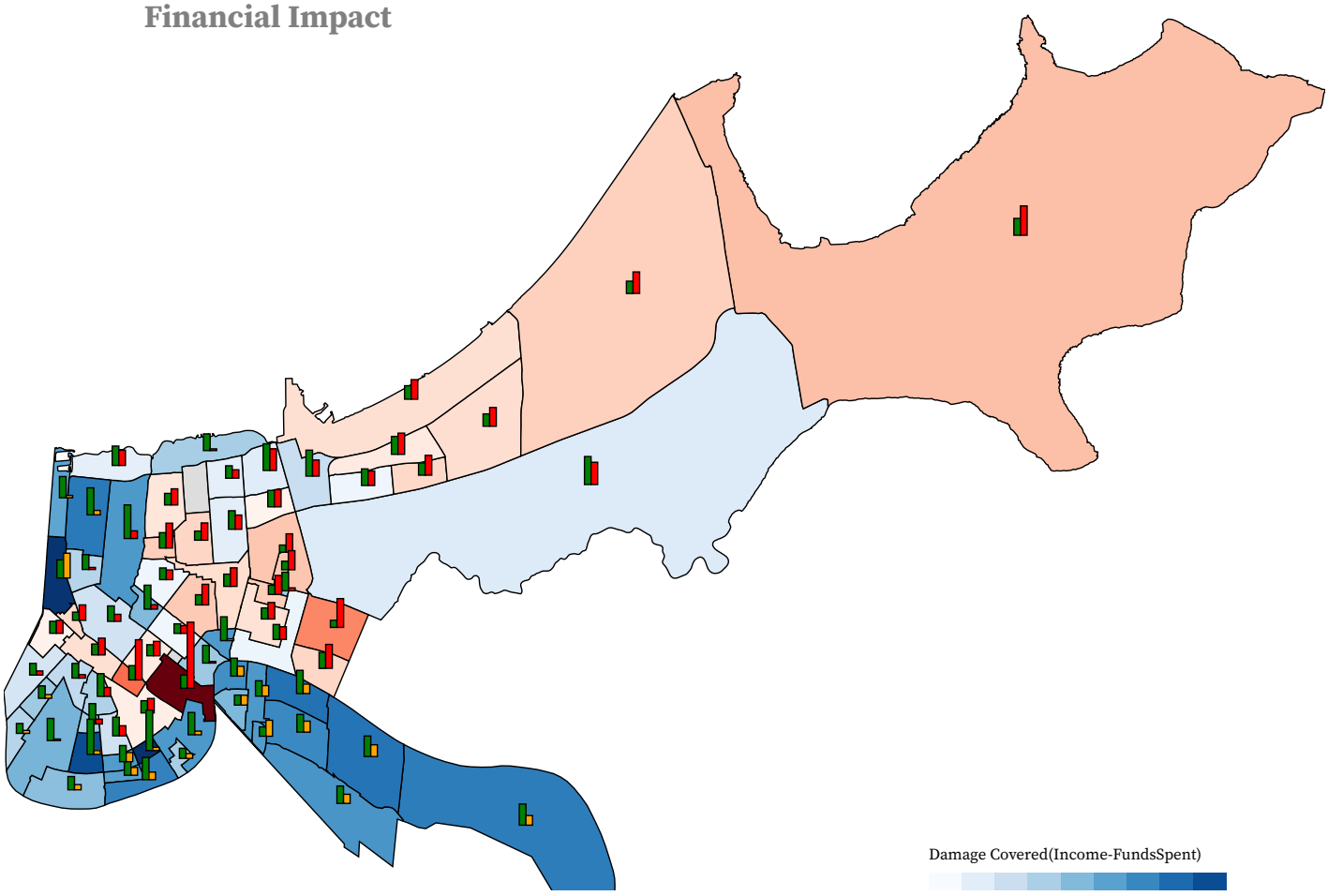
Which neighborhood of New Orleans have had the most financial Impact due to Hurricane.?

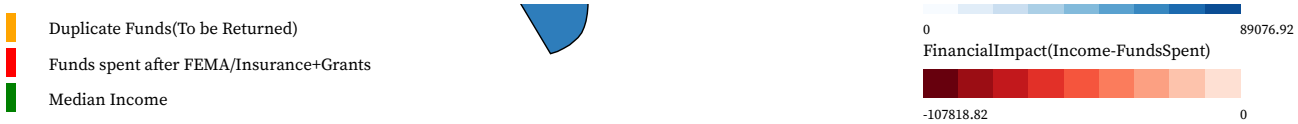
Viz Usage Description

Hover over Grouped bars on map to view Median Income and Median Funds Spent/Duplication Of funds. **Hover over parish** to view parish name and Financial Impact.

New Orleans Neighborhoods

Financial Impact





ROAD HOME GRANTS - DATASET

```
data = ▶ FileAttachment {name: "roadhome_for_datastore.csv", mimeType: "text/csv"}
```

```
pdata = ▶ Array(130053) [Object, Object, Object, Object, Object, Object, Object, Object, Obj
```

LOUISIANA

```
mapColor = ƒ(i)
```

```
uniqueNeighborhoods = ▶ Array(73) ["Read Blvd East", "Little Woods", "", "Broadmoor", "St. T
```

```
linedata = ▶ Array(130053) [Object, Object, Object, Object, Object, Object, Object, Object,
```

```
aggData = ▶ Array(9) [Array(2), Array(2), Array(2), Array(2), Array(2), Array(2), Array(2),
```

```
transformedData = ▶ Array(188) [Object, Object, Object, Object, Object, Object, Object, Obje
```

```
louisianaCounty = ▶ Object {type: "FeatureCollection", features: Array(64), style: Object}
```

```
countyNames = ▶ Array(64) ["Winn", "West Feliciana", "West Carroll", "West Baton Rouge", "We
```

```
uniqueParishes = ▶ Array(37) ["Orleans", "Jefferson", "St. Tammany", "Calcasieu", "Tangipaho
```

```
ndata = ▶ Array(130053) [Object, Object, Object, Object, Object, Object, Object, Object, Obj
```

	integer 0 140k	Structure Type string 9 categories	GIS City string 299 categories	GIS State string 4 categories	GIS string 54,919
0	0	Single (including mobile	NEW ORLEANS	LA	70128-
1	1	Single (including mobile	NEW ORLEANS	LA	70127-
2	2	Single (including mobile	HARVEY	LA	70058-
3	3	Single (including mobile	NEW ORLEANS	LA	70127-
4	4	Single (including mobile	NEW ORLEANS	LA	70125-
5	5	Single (including mobile	METAIRIE	LA	70006-
6	6	Duplex (with one owner-(	NEW ORLEANS	LA	70130-
7	7	Single (including mobile	HARVEY	LA	70058-
8	8	Duplex (with one owner-(	NEW ORLEANS	LA	70118-

roadhome_for_datastore Search 130,053 rows

```
height = 800
```

roadhome_for_datastore roadhom…

Filter Columns Sort Slice Save

```
width = 1000
```

```
centroid = ƒ(feature)
```

```
mapColor = d3.scaleOrdinal()
  .domain(selectedParishes)
  .range(d3.schemeSet2);
```

```
uniqueNeighborhoods = [...new Set(pdata.map(item => item['NOLA Neighborhood Name']))];
```

```
linedata = pdata.map(d => ({"Structure Type": d["Structure Type"],"TOTAL_CLOSING_AMOUNT":
d["TOTAL_CLOSING_AMOUNT"],"PARISH": d["PARISH"],"Current_Damage_Assessment": d["Current
Damage Assessment"]
                            }));



aggData = d3.rollups(linedata,
  v => d3.sum(v, d => +d.TOTAL_CLOSING_AMOUNT),
  d => d["Structure Type"],
  d => d.PARISH
)



transformedData = aggData.flatMap(([type, parishValues]) => {
  return parishValues.map(([parish, totalamount]) => ({
    "Structure Type": type,
    "TOTAL_CLOSING_AMOUNT": totalamount ,
    "PARISH": parish
  }));
});



louisianaCounty = FileAttachment("louisiana-with-county-boundaries_1101.geojson").json()



countyNames = louisianaCounty.features.map(d=> d.properties.name);



uniqueParishes = [...new Set(pdata.map(d => d.PARISH))];



ndata = {
  const updatedData = pdata.map(d => {
    return {
      ...d,
      PARISH: mapParishNames(d.PARISH)
    };
  });

  return updatedData;
}
```

```
// Parishes names on the map and on the dataset are not same..so changing them to have
common names
mapParishNames = (oldName) => {
  const nameMappings = {
    "St. Tammany": "Saint Tammany",
    "St. Bernard": "Saint Bernard",
    "St. James": "Saint James",
    "St. John the Baptist": "Saint John the Baptist",
    "St. Charles": "Saint Charles",
    "St. Martin": "Saint Martin",
    "St. Mary": "Saint Mary",
    "St. Helena": "Saint Helena"

  };

  return nameMappings[oldName] || oldName;
}
```

```
height = 800
```

```
rArray = ▶ Array(37) [Object, Object, Object, Object, Object, Object, Object, Object, Object
```

```
width = 1000
```

```
centroid = {
  const path = d3.geoPath();
  return feature => path.centroid(feature);
}
//to place bars on the map
```

```
aggregateByParish = (data) => {
  return data.reduce((s, d) => {
    const parish = d.PARISH;
    const closingAmount = parseFloat(d.TOTAL_CLOSING_AMOUNT) || 0;
    const damageAssessment = parseFloat(d['Current Damage Assessment']) || 0;
    const dobAmount = parseFloat(d['Current Total DOB Amount (no Legal Fees removed)']) || 0;

    if (!s[parish]) {
      s[parish] = {
        PARISH: parish,
        TotalClosingAmount: 0,
```

```
        TotalDamageAssessment: 0,
        TotalDOBAmount: 0
      };
    }

    s[parish].TotalClosingAmount += closingAmount;
    s[parish].TotalDamageAssessment += damageAssessment;
    s[parish].TotalDOBAmount += dobAmount;

    return s;
  }, {});
}
```

```
data0 = {

  const Data4 = aggregateByParish(ndata);
  const finalData = Object.keys(Data4).map(d => { const parishData = Data4[d];
    return {
      PARISH: parishData.PARISH,
      TotalClosingAmount: parishData.TotalClosingAmount,
      TotalDamageAssessment: parishData.TotalDamageAssessment, //total damage to structures.
      TotalDOBAmount: parishData.TotalDOBAmount,
      TotalFundsReceived: parishData.TotalClosingAmount + parishData.TotalDOBAmount,//
fema+insurance+ROad home grants
      FundsNeeded: parishData.TotalDamageAssessment - (parishData.TotalClosingAmount +
parishData.TotalDOBAmount)
    };
  });

  return finalData;
}
```

```
data2 = data0.map(d => ({...d,feature: louisianaCounty.features.find(f =>
f.properties.name === d.PARISH)})).filter(d => d.feature);
```

```
aggr = ndata.reduce((s, d) => {
```

```
    const parish = d.PARISH;
    const structureType = d['Structure Type'];
    const closingAmount = parseFloat(d.TOTAL_CLOSING_AMOUNT);

    if (!s[parish]) {
      s[parish] = {
        totalClosingAmount: 0,
        structureTypes: {}
      };
    }

    s[parish].totalClosingAmount += closingAmount;

    if (!s[parish].structureTypes[structureType]) {
      s[parish].structureTypes[structureType] = 0;
    }

    s[parish].structureTypes[structureType] += closingAmount;

    return s;
  }, {});




rArray = Object.entries(aggr).map(([parish, data]) => ({
  PARISH: parish,
  TOTAL_CLOSING_AMOUNT: data.totalClosingAmount,
  StructureTypes: data.structureTypes
}));



function transformData1(resultArray) {

  const sType = new Set();
  resultArray.forEach(d => {
    Object.keys(d.StructureTypes).forEach(type => sType.add(type));
  });

  const transformedData = [];
  resultArray.forEach(d => {
    sType.forEach(type => {
      const entry = {
        PARISH: d.PARISH,
        TOTAL_CLOSING_AMOUNT: d.TOTAL_CLOSING_AMOUNT,
        StructureType: type,
```

```
        StructureTypeAmount: d.StructureTypes[type] || 0
      };
      transformedData.push(entry);
    });
  });

  return transformedData;
}



transformedData1 = transformData1(rArray);



structureTypeMap = ({
  "Single (including mobile home)": "Single-mobile",
  "Duplex (with one owner-occupied unit)": "Duplex",
  "Townhouse": "Townhouse",
  "Single Family on Leased Land": "Single-Leased",
  "Mobile Home": "Mobile Home",
  "Condominium": "Condo",
  "Mobile Home on Leased Land": "Mobile-Leased",
  "Single": "Single",
  "Ineligible": "Ineligible"
})
//shortening the structure type name for labels  visibility on bar chart



transformedData2 = transformedData1.map(d => {
  if (structureTypeMap[d.StructureType]) {
    return {...d, StructureType: structureTypeMap[d.StructureType]};
  }
  return d;
})




<style>
.tooltip {
  position: absolute;
  text-align: center;
  width: auto;
  height: auto;
  padding: 2px;
  font: 12px sans-serif;
```

```
  background: lightsteelblue;
  border: 0px;
  border-radius: 8px;
  pointer-events: none;
}
</style>



function GroupedBarChart(selectedParishes, parishColorMap) {
  d3.select("#barchart").selectAll("*").remove();

  // Dimensions and margins
  const marginTop = 60;
  const marginRight = 30;
  const marginBottom = 50;
  const marginLeft = 60;
  const width = 928;
  const height = 600;

  const structureTypes = Array.from(new Set(transformedData2.map(d => d.StructureType)));


  const filteredData = structureTypes.map(type => {
    return {
      StructureType: type,
      parishes: selectedParishes.map(parish => {
        const parishData = transformedData2.find(d => d.PARISH === parish && d.StructureType
=== type);
        return {
          PARISH: parish,
          StructureTypeAmount: parishData ? parishData.StructureTypeAmount : 0
        };
      })
    };
  });

  const x0 = d3.scaleBand()
    .domain(structureTypes)
    .rangeRound([marginLeft, width - marginRight])
    .paddingInner(0.1);

  const x1 = d3.scaleBand()
    .domain(selectedParishes)
    .rangeRound([0, x0.bandwidth()])
    .padding(0.05);

  const y = d3.scaleLinear()
```

```
    .domain([0, d3.max(filteredData, d => d3.max(d.parishes, p =>
p.StructureTypeAmount))]).nice()
    .rangeRound([height - marginBottom, marginTop]);

  // SVG container
  const svg = d3.select("#barchart").append("svg")
    .attr("width", width)
    .attr("height", height)
    .attr("viewBox", [0, 0, width, height])
    .attr("style", "max-width: 100%; height: auto;");


  // Tooltip
  const tooltip = d3.select("body").append("div")
    .attr("class", "tooltip")
    .style("opacity", 0);

  const barGroups = svg.append("g")
    .selectAll("g")
    .data(filteredData)
    .join("g")
      .attr("transform", d => `translate(${x0(d.StructureType)},0)`);


  // Creating bars with transitions
  barGroups.selectAll("rect")
    .data(d => d.parishes)
    .join(
      enter => enter.append("rect")
        .attr("x", d => x1(d.PARISH))
        .attr("y", height - marginBottom)
        .attr("width", x1.bandwidth())
        .attr("height", 0)
        .attr("fill", d => parishColorMap.get(d.PARISH) || "#ddd")
        .call(enter => enter.transition()
          .duration(1000)
          .attr("y", d => y(d.StructureTypeAmount))
          .attr("height", d => y(0) - y(d.StructureTypeAmount))),

      update => update
        .attr("fill", d => parishColorMap.get(d.PARISH) || "#ddd")
        .call(update => update.transition()
          .duration(1000)
          .attr("x", d => x1(d.PARISH))
          .attr("width", x1.bandwidth())
          .attr("y", d => y(d.StructureTypeAmount))
          .attr("height", d => y(0) - y(d.StructureTypeAmount))),

      exit => exit
```

```
      .call(exit => exit.transition()
        .duration(1000)
        .attr("y", height - marginBottom)
        .attr("height", 0)
        .remove())
  );

// Tooltip
barGroups.selectAll("rect")
  .on("mouseover", function(event, d) {
    tooltip.transition()
      .duration(200)
      .style("opacity", .9);
    tooltip.html(`Parish: ${d.PARISH}<br/>Amount: ${d.StructureTypeAmount}`)
      .style("left", (event.pageX + 10) + "px")
      .style("top", (event.pageY - 28) + "px");
  })
  .on("mouseout", function(d) {
    tooltip.transition()
      .duration(500)
      .style("opacity", 0);
  });

// structure dmg value on bar
barGroups.selectAll("text.valueLabel")
  .data(d => d.parishes)
  .join("text")
    .attr("class", "valueLabel")
    .attr("x", d => x1(d.PARISH) + x1.bandwidth() / 2)
    .attr("y", d => y(d.StructureTypeAmount) - 5)
    .attr("text-anchor", "middle")
    .text(d => d3.format(".2s")(d.StructureTypeAmount))
    .attr("fill", "black")
    .attr("font-size", "10px");

// Append axes
const xAxis = svg.append("g")
    .attr("transform", `translate(0,${height - marginBottom})`)
    .call(d3.axisBottom(x0).tickSizeOuter(0));

xAxis.selectAll("text")
  .attr("transform", "rotate(0)")
  .attr("text-anchor", "middle");


svg.append("g")
  .attr("transform", `translate(${marginLeft},0)`)
  .call(d3.axisLeft(y).ticks(null, "s"));
```

```
    // X-axis label
    svg.append("text")
      .attr("transform", `translate(${width / 2}, ${height})`)
      .attr("dy", "-0.5em")
      .attr("text-anchor", "middle")
      .text("STRUCTURE TYPE");

    // Viz  Title
    svg.append("text")
      .attr("x", (width / 2))
      .attr("y", marginTop / 2 - 10)
      .attr("text-anchor", "middle")
      .style("font-size", "24px")
      .style("text-decoration", "underline")
      .style("font-weight", "bold")
      .text("Structure Type : Damage Comparison");

    const legend = svg.append("g")
      .attr("transform", `translate(${width - marginRight - 150}, ${marginTop})`);

    legend.selectAll("rect")
      .data(selectedParishes)
      .join("rect")
        .attr("x", 0)
        .attr("y", (d, i) => i * 20)
        .attr("width", 18)
        .attr("height", 18)
        .style("fill", d => parishColorMap.get(d) || "#ddd"); // Use color from parishColorMap

    legend.selectAll("text")
      .data(selectedParishes)
      .join("text")
        .attr("x", 24)
        .attr("y", (d, i) => i * 20 + 9)
        .attr("dy", ".35em")
        .style("text-anchor", "start")
        .text(d => d);

    return svg.node();
}
```

```
parishColorMap = new Map();
```

```
availableColors = d3.schemeSet2.slice(0, 4);
//only 4 colours because of 4 selecions
```

```
function assignColor(parish)
{
  if (!parishColorMap.has(parish))
  {
    let color = availableColors.shift()
    parishColorMap.set(parish, color);
  }
  return parishColorMap.get(parish);
}
```

```
newOrleans = ▶ Object {type: "FeatureCollection", features: Array(73)}
```

```
function removeColor(parish)
{
  if (parishColorMap.has(parish))
  {
    availableColors.push(parishColorMap.get(parish));
    parishColorMap.delete(parish);
  }
}
```

```
median = ƒ(d)
```

```
mapSelection = function(feature, element)
{
  const x = data2.find(p => p.PARISH === feature.properties.name);
  if (!x) return;

  const parishName = x.PARISH;
  const select = d3.select(element).classed("selected");

  if (!select && mutable selectedParishes.length < count)
  {

    d3.select(element)
```

```
        .classed("selected", true)
        .style("fill", assignColor(parishName));
      mutable selectedParishes = [...mutable selectedParishes, parishName];
    }
    else if (select)
    {
      d3.select(element)
        .classed("selected", false)
        .style("fill", "");
      removeColor(parishName);
      mutable selectedParishes = mutable selectedParishes.filter(p => p !== parishName);
    }

    GroupedBarChart(mutable selectedParishes,parishColorMap);
  };



  mutable selectedParishes = [];

  //mutates the array based on the selction from the map



  {
    const parishes = d3.select(LouiMap).selectAll("path");

    parishes.on("click", function(event, d) {
      mapSelection(d, this);
    });
  }
```

```
financialImpact = ▶ Array(72) [Object, Object, Object, Object, Object, Object, Object, Objec
```

Neighborhood	Median Incom…
string	string
73 unique values	73 unique values

	string (73 unique values)	string (73 unique values)
11	Little Woods	$27,514.00
12	Audubon	$45,174.00
13	French Quarter	$36,065.00
14	Bayou St. John	$48,816.00

neworleans_medianIncome Search 73 rows

```
data1 = FileAttachment("NewOrleans_Median income.csv")
```

```
medianIncome=data1.csv()
```

```
uniqueNeighborhood = [...new Set(medianIncome.map(item => item.Neighborhood))];
```

```
neighborhoodNames = ▶ Array(72) ["Read Blvd East", "Little Woods", "Broadmoor", "St. Thomas
```

```
newOrleans = FileAttachment("new-orleans@1.geojson").json()
```

```
combinedFinancialImpact = ▶ Array(72) [Object, Object, Object, Object, Object, Object, Objec
```

```
orleansData = pdata.filter(d => d.PARISH === 'Orleans');
```

```
function median(d)
{
  if (d.length === 0)
    return 0;
  d.sort((a, b) => a - b);

  const mid = Math.floor(d.length / 2);
  if (d.length % 2)
  {
    return d[mid];
  }
  return (d[mid-1]+d[mid])/2.0;
```

```
}
//funcgtion to calculate median


groupedByNeighborhoodMap= d3.group(orleansData, d => d['NOLA Neighborhood Name']);



groupedByNeighborhood = Object.fromEntries(groupedByNeighborhoodMap);


financialImp = Object.entries(groupedByNeighborhood).map(([d, v]) => {
  const damageAssessments = v.map(d => parseFloat(d['Current Damage Assessment'])); //
damage to structure types
  const currentDOBAmounts = v.map(d => parseFloat(d['Current Total DOB Amount (no Legal Fees
removed)'])); // fema+Insurance
  const totalClosingAmounts = v.map(d => parseFloat(d['TOTAL_CLOSING_AMOUNT']));// road home
grants

  //using median funcctionn to calculate median
  const medianDamageAssessment = median(damageAssessments);
  const mediancurrentDOBAmounts = median(currentDOBAmounts);
  const medianTotalClosingAmount = median(totalClosingAmounts);

  const Funds = medianDamageAssessment - (mediancurrentDOBAmounts  +
medianTotalClosingAmount); //can be negative or positive values meaning - to be  paid by
people or people have received duplicate funds and have to return them back.

  return {
    neighborhood: d,
    medianDamageAssessment,
    mediancurrentDOBAmounts ,
    medianTotalClosingAmount,
    financialImpact: Funds
  };
})



// Neighbohood names on the map and on the dataset are not same..so changing them to have
common names
financialImpact = (() => {
  const nameChange = {
    "St. Thomas Development": "St. Thomas Dev",
    "Lakeshore/Lake Vista": "Lakeshore - Lake Vista",
    "Village de l'est": "Village De Lest",
    "Lower Garden District Neighborh": "Lower Garden District",
```

```
    "Mid-City Neighborhood": "Mid-City",
    "Lakeview Neighborhood": "Lakeview",
    "New Aurora/English Turn": "New Aurora - English Turn",
    "Marlyville/Fountainebleau": "Marlyville - Fontainbleau",
    "Tall Timbers/Brechtel": "Tall Timbers - Brechtel",
    "McDonough": "Mcdonogh",
    "Treme'/Lafitte": "Treme - Lafitte",
    "Lower Ninth Ward Neighborhood": "Lower Ninth Ward",
    "Central City Neighborhood": "Central City",
    "Uptown Neighborhood": "Uptown",
    "Viavant/Venetian Isles": "Viavant",
    "U.S. Naval Base Neighborhood": "U.S. Naval Base",
    "Bywater Neighborhood": "Bywater",
    "Tulane/Gravier": "Tulane - Gravier",
    "Fischer Project": "Fischer Dev",
    "Desire Development": "Desire Dev",
    "Florida Development": "Florida Dev",
    "B.W. Cooper": "B. W. Cooper"
  };

  return financialImp.map(d => {
    const newName = nameChange[d.neighborhood];
    return {...d,neighborhood: newName ? newName : d.neighborhood};
  });
})();



neighborhoodNames= [...new Set(financialImpact.map(d=>d.neighborhood))];



combinedFinancialImpact = financialImpact.reduce((s,d) => {
  const x = medianIncome.find(inc => inc.Neighborhood === d.neighborhood);
  if (x)
  {
    s.push({ ...d, medianIncome2000: x['Median Income in 2000'] });
  }
  return s;
}, []);



OrleansPlotData = combinedFinancialImpact.map(d => {

  // Removing $ symbol before income
```

```
    const medianIncome = parseFloat(d.medianIncome2000.replace(/[$,]/g, ''));

    const PersonalFunds  = (medianIncome - d.financialImpact);

    return {
      neighborhood: d.neighborhood,
      medianIncome: medianIncome,
      impact: d.financialImpact,
      PersonalFunds: PersonalFunds
    };
  });



data3 = OrleansPlotData.map(d => ({...d,feature: newOrleans.features.find(f =>
f.properties.name === d.neighborhood)})).filter(d => d.feature);
```