STAT 479 HW2

Yifan Zhang

Problems

```r
#load some packages and libraries
library("dslabs")
library("dplyr")
library("tidyr")
library("ggplot2")
library("naniar")
library("UpSetR")
library("rpart")
library("simputation")
library("readr")
theme_set(theme_bw())
```

(1) Plant Growth Experiment

This problem will give you practice with tidying a dataset so that it can be easily visualized. The data describe the height of several plants measured every 7 days. The plants have been treated with different amounts of a growth stimulant. The first few rows are printed below – `height.x` denotes the height of t he plant on day `x`.

```r
library("readr")
plants <- read.csv("https://uwmadison.box.com/shared/static/qg9gwk2ldjdtcmmmiropcunf34ddonya.csv")
```

a. Propose an alternative arrangement of rows and columns that conforms to the tidy data principle.

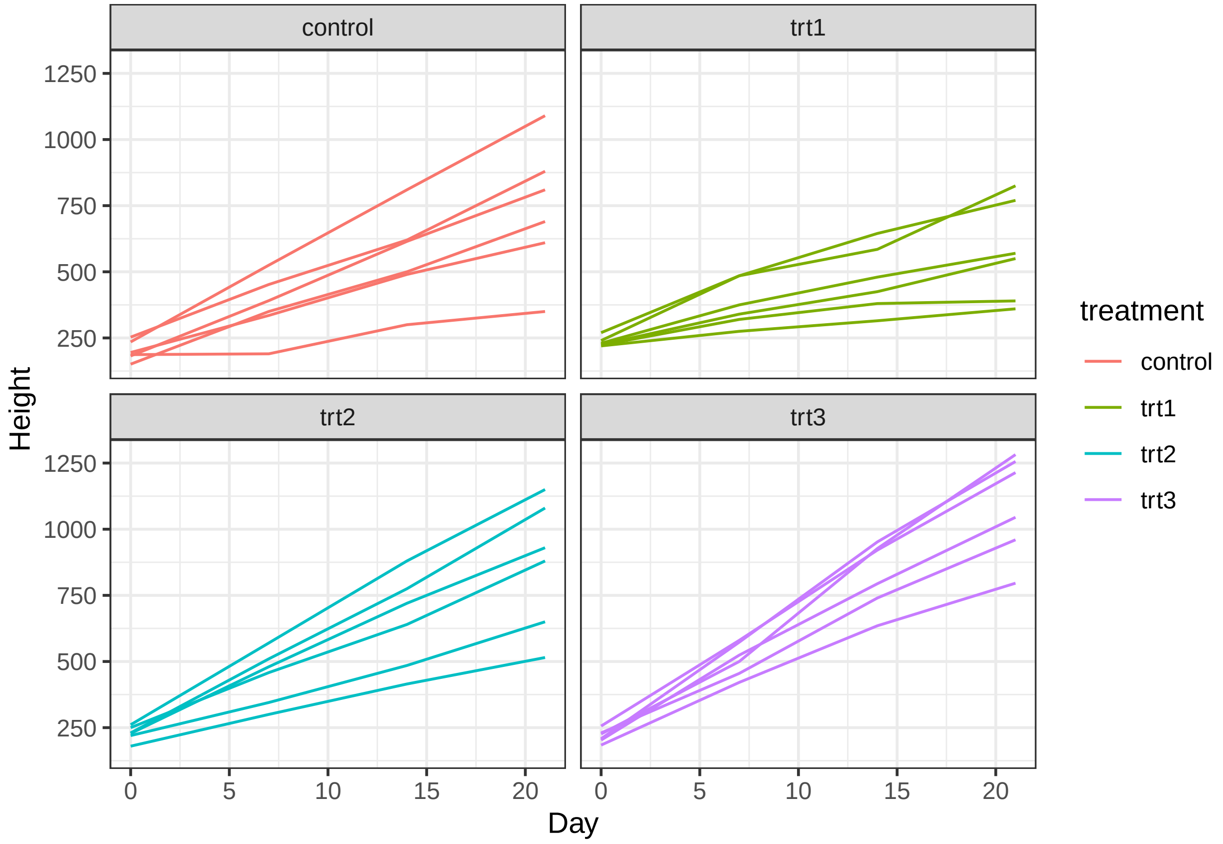
Create one column which is called "Days" represents the number of days the plants have grown. Create another column which is called "Height" represents the height of plants. Remove the columns "height.0", "height.7", "height.14", and "height21".

b. Implement your proposed arrangement from part (a).

```r
#change column names to int
colnames(plants)[c(3:6)] <- c(0, 7, 14, 21)
#lengthens data
plants.new <- pivot_longer(plants, c('0', '7', '14', '21'), names_to = "days", values_to = "height")
```

c. Using the dataset from (b), create a version of Figure 1, showing the growth of the plants over time according to different treatments.

```r
ggplot(plants.new) +
  geom_line(aes(as.numeric(days), height, col = treatment, group = plantid)) +
  facet_wrap(. ~ treatment, nrow = 2) +
  labs(
    x = "Day",
    y = "Height"
  ) +
  theme_bw()
```



(2) California Wildfires

In this problem, we will interactively visualize a dataset giving statistics of recent California wildfires. The steps below guide you through the process of building this visualization. Make sure to include your code, a screenshot, and a brief explanation of what you did for each step.

a. [Static version] Plot the day of the year that each fire started against the county within which it was located. Use the size of the mark to encode the number of acres burned in each fire. Sort the counties according to the average latitude of the fires within it. At this point, your figure should look something like Figure 2.

```
import { vl } from "@vega/vega-lite-api"
import { aq,op } from "@uwdata/arquero"
data = aq.fromCSV(await FileAttachment("fires.csv").text())
//Make a plot with "day_of_year" as x-axis and "Counties" as y-axis
vl.markPoint({filled: true, color: 'orange'})
  .data(data)
  .encode(
    vl.x().fieldQ("day_of_year").title("Day of Year"),
    //Sort y-axis with a descending order
    vl.y().fieldN("Counties").sort({op: 'median', field: 'Latitude', order: 'descending'}).title("Counti
    //Reflect the acres burned with the size of points.
    vl.size().fieldQ("AcresBurned").scale({range: [25, 1000]}).legend({tickCount: 5}).title("Acres Burne
  )
  .height(600)
  .width(500)
  .render()
```

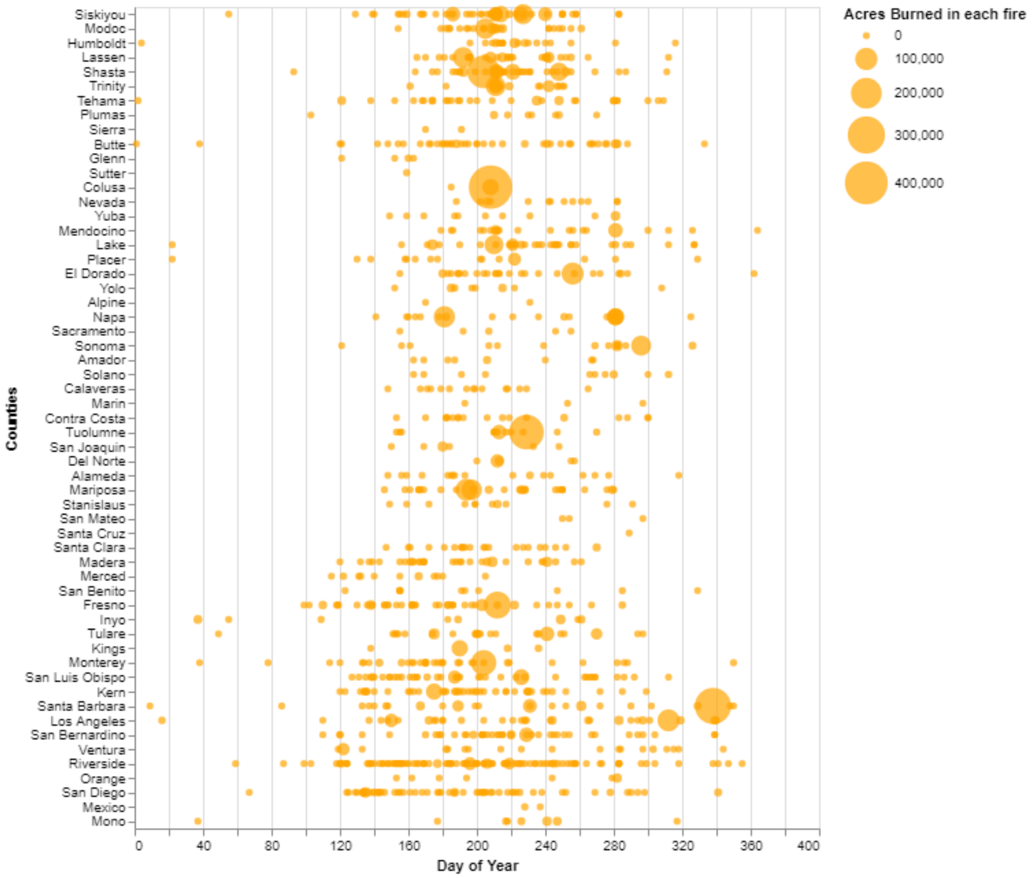
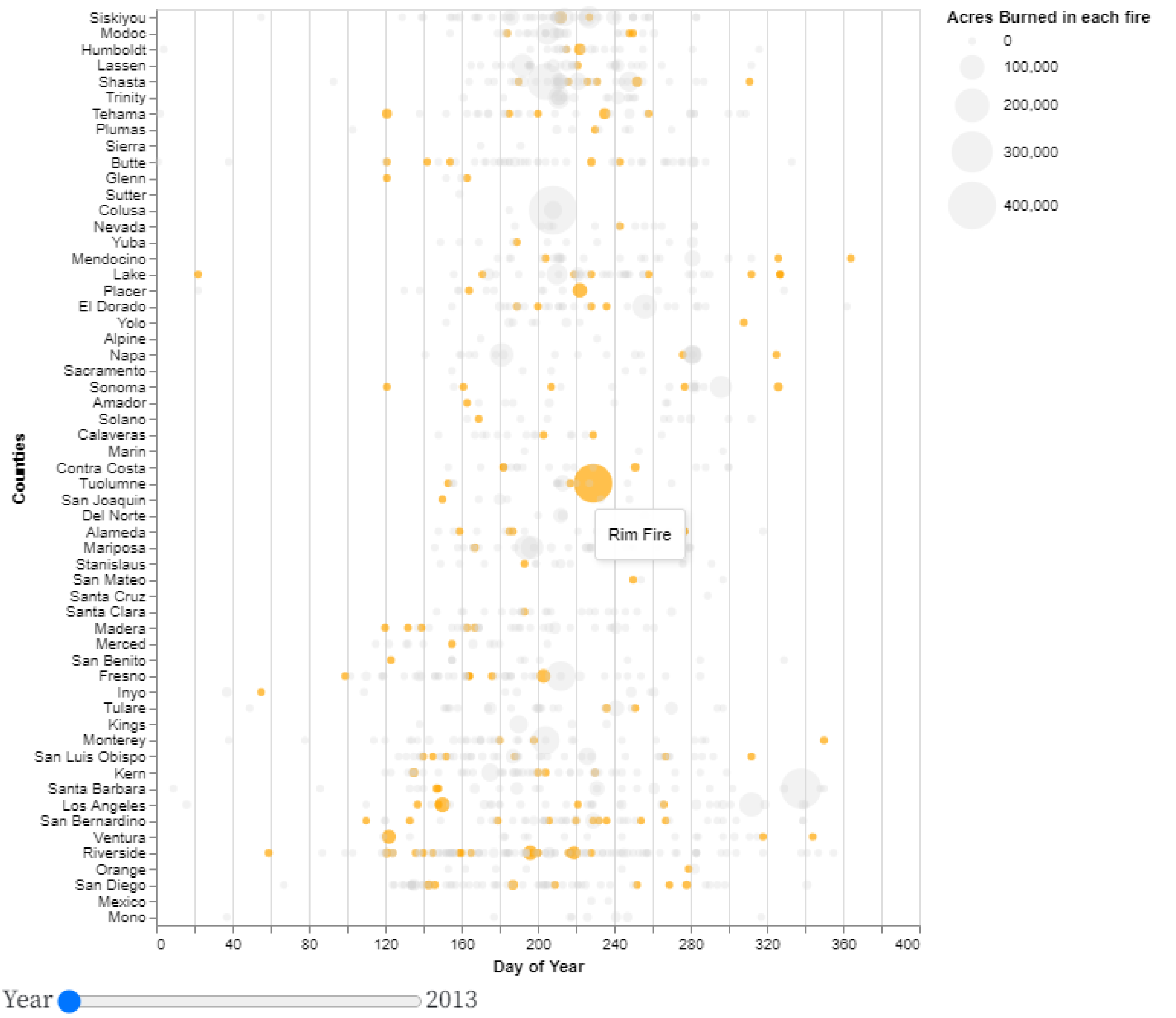
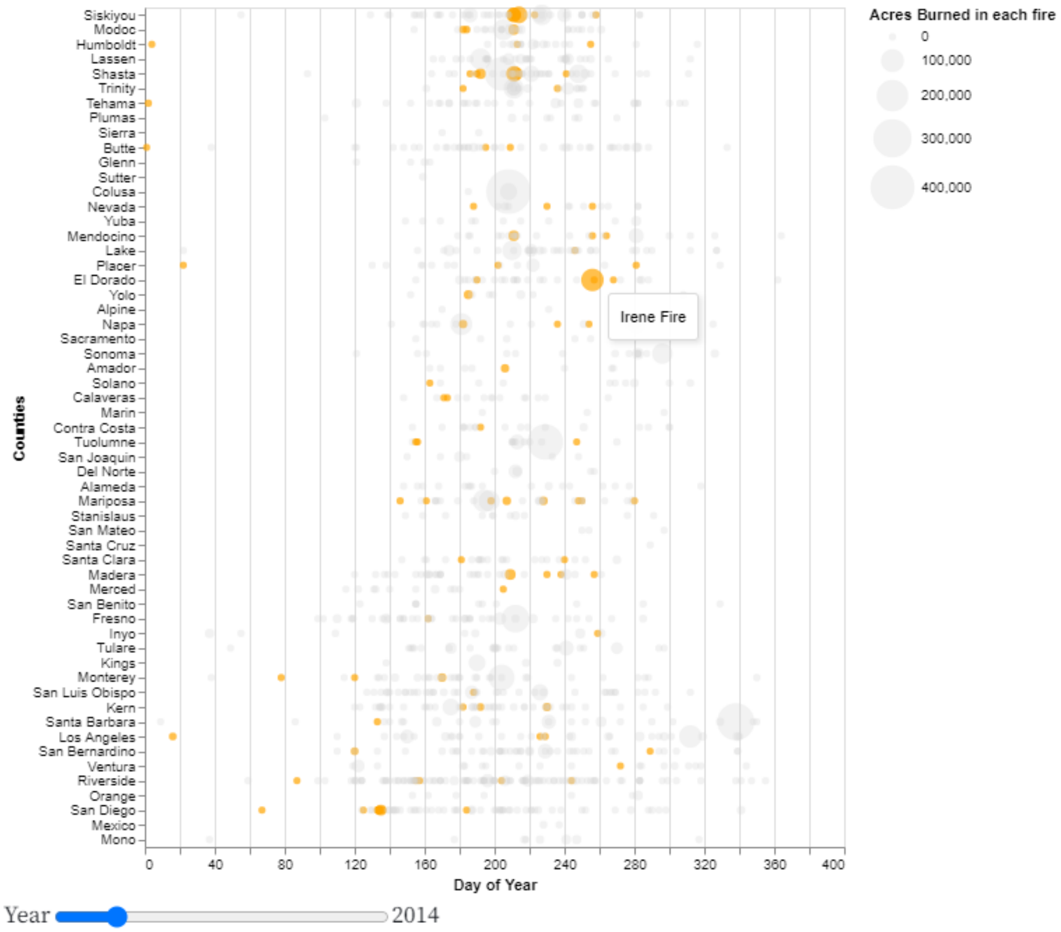


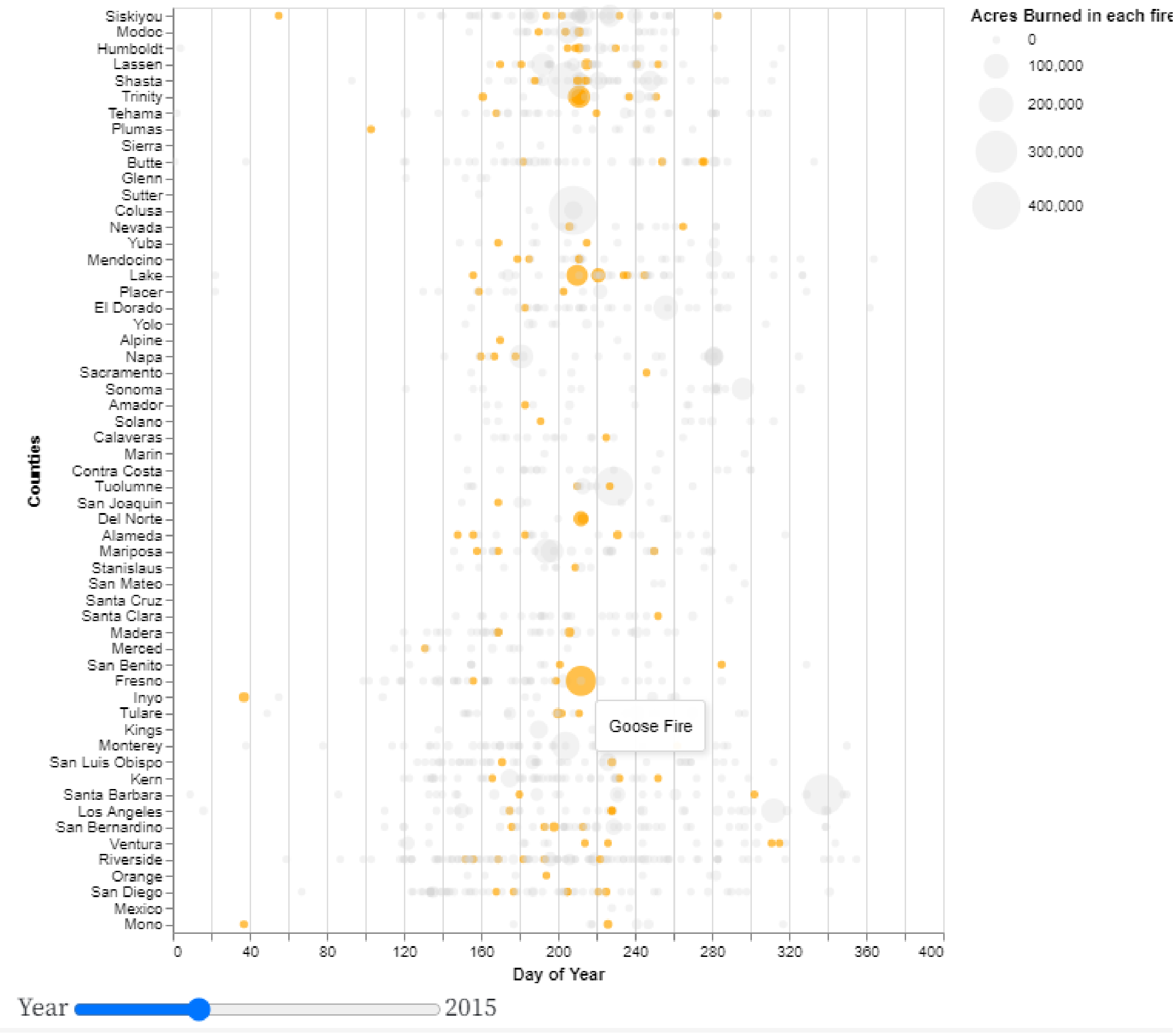
Figure 1: unchanged image

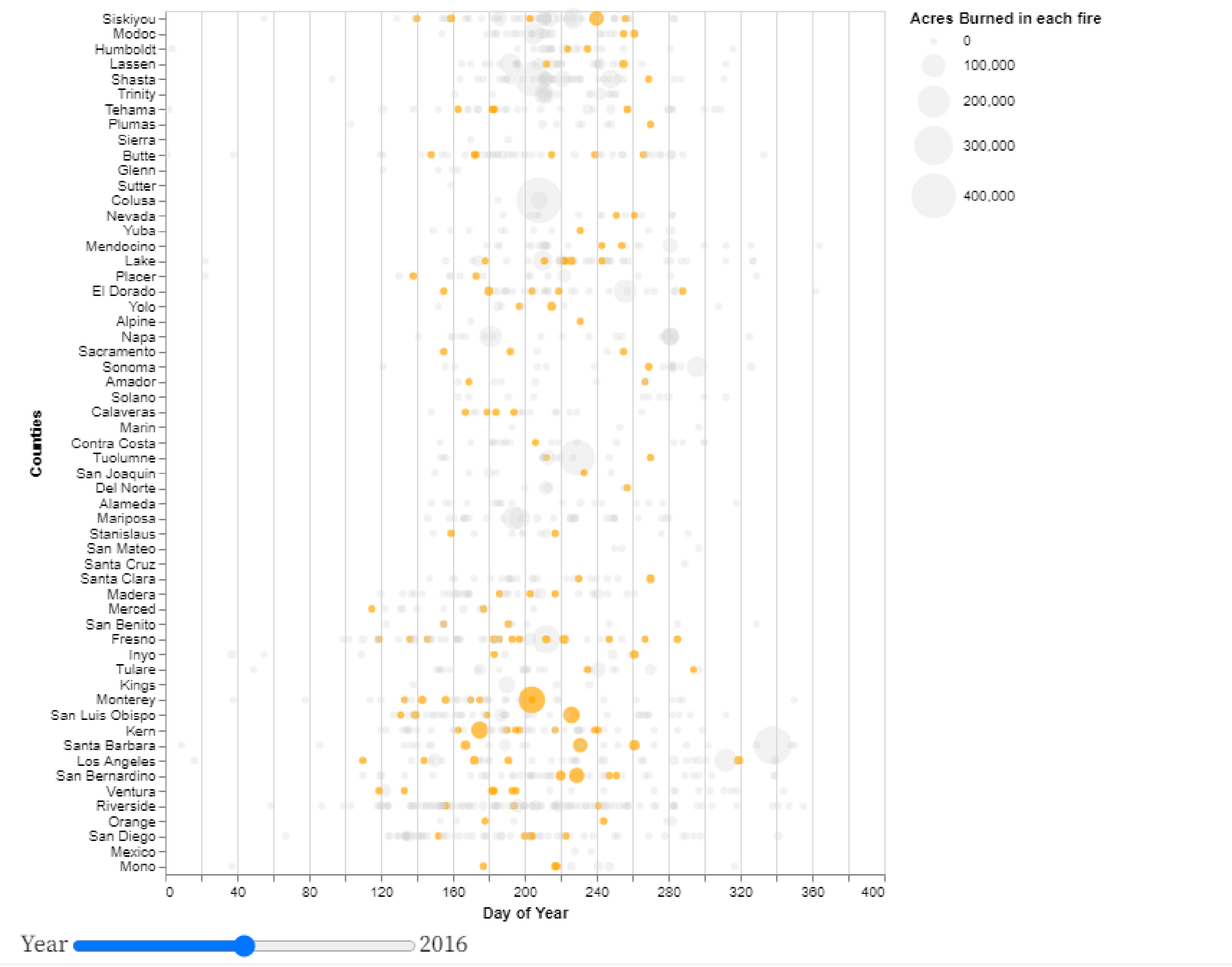
b. [Interactive] Provide a tooltip so that the name of the fire can be identified by hovering over the points. Introduce a slider to interactively highlight selected years. An interactive version is linked in the caption to Figure 2. *Hint*: The conditional encoding examples from Week 3 - 1 and the slider example from Week 1 - 3 may be useful references.

```
{
// define a slider selector object
let selectYear = vl.selectSingle("select").fields("year")
  .init({year: 2013})
  .bind(vl.slider().min(2013).max(2019).step(1).name("Year"))

// update the plot depending on the value of the slider
return vl.markPoint({filled: true})
  .data(data)
  .select(selectYear)// refer to the slider bar
  .encode(
    vl.x().fieldQ("day_of_year").title("Day of Year"),
    vl.y().fieldN("Counties").sort({op: 'median', field: 'Latitude', order: 'descending'}).title("Counti
    vl.size().fieldQ("AcresBurned").scale({range: [25, 1000]}).legend({tickCount: 5}).title("Acres Burne
  // add a tooltip
    vl.tooltip().fieldN("Name"),
    vl.color().if(selectYear, vl.value('orange')).value('lightgrey'),
    vl.opacity().if(selectYear).value('0.3')
  )
  .height(600)
  .width(500)
  .render()
}
```

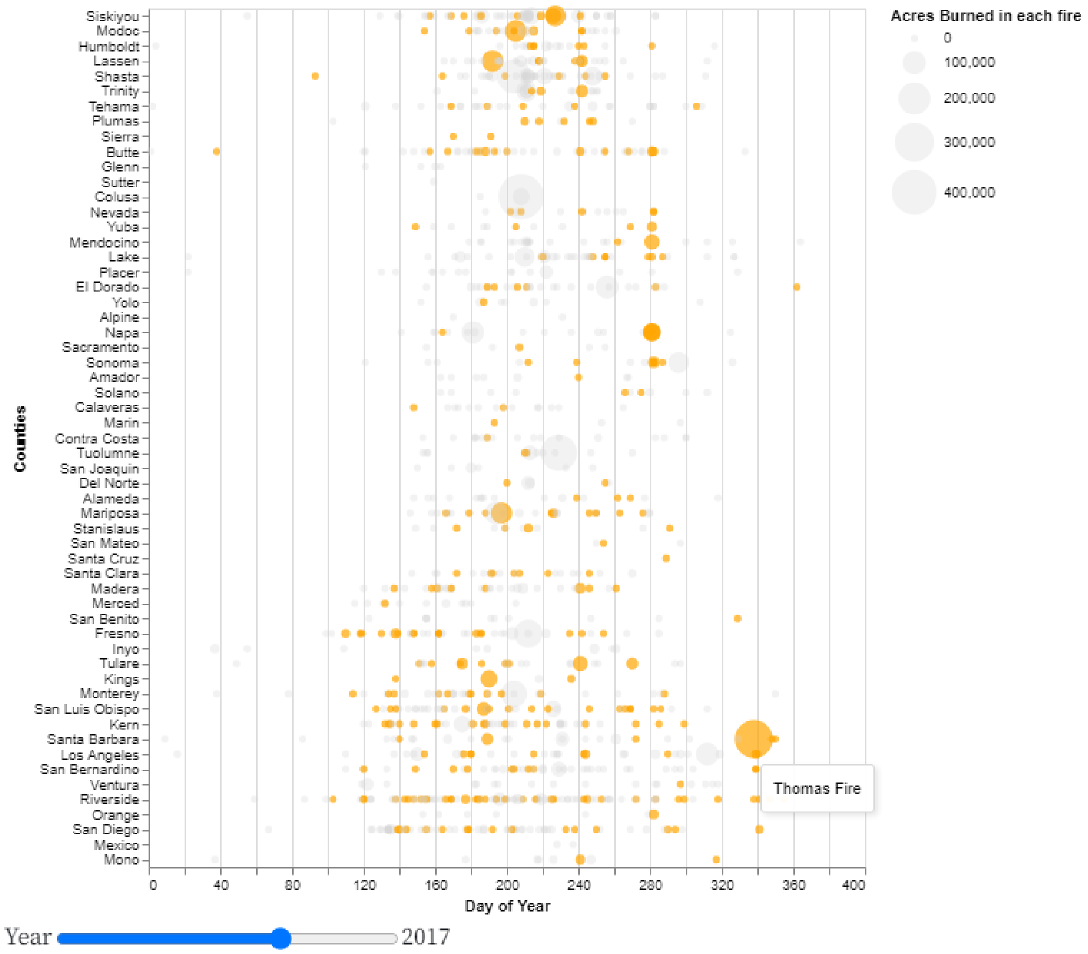


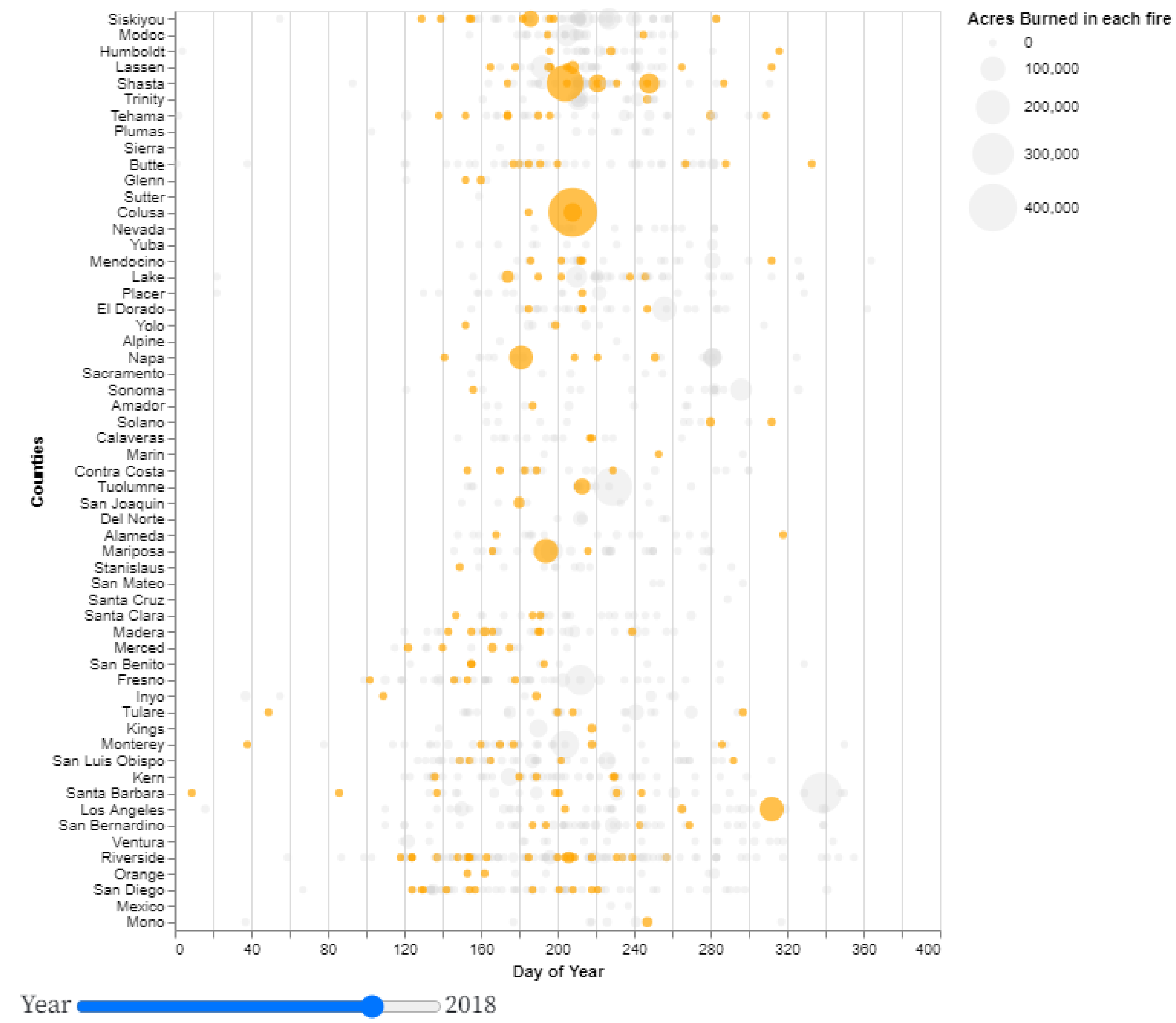

Siskiyou
Modoc
Humboldt
Lassen
Shasta
Trinity
Tehama
Plumas
Sierra
Butte
Glenn
Sutter
Colusa
Nevada
Yuba
Mendocino
Lake
Placer
El Dorado
Yolo
Alpine
Napa
Sacramento
Sonoma
Amador
Solano
Calaveras
Marin
Contra Costa
Tuolumne
San Joaquin
Del Norte
Alameda
Mariposa
Stanislaus
San Mateo
Santa Cruz
Santa Clara
Madera
Merced
San Benito
Fresno
Inyo
Tulare
Kings
Monterey
San Luis Obispo
Kern
Santa Barbara
Los Angeles
San Bernardino
Ventura
Riverside
Orange
San Diego
Mexico
Mono
Counties
0
40
80
120
160
200
240
280
320
360
400
Day of Year
Irene Fire
Acres Burned in each fire
0
100,000
200,000
300,000
400,000
Year
2014

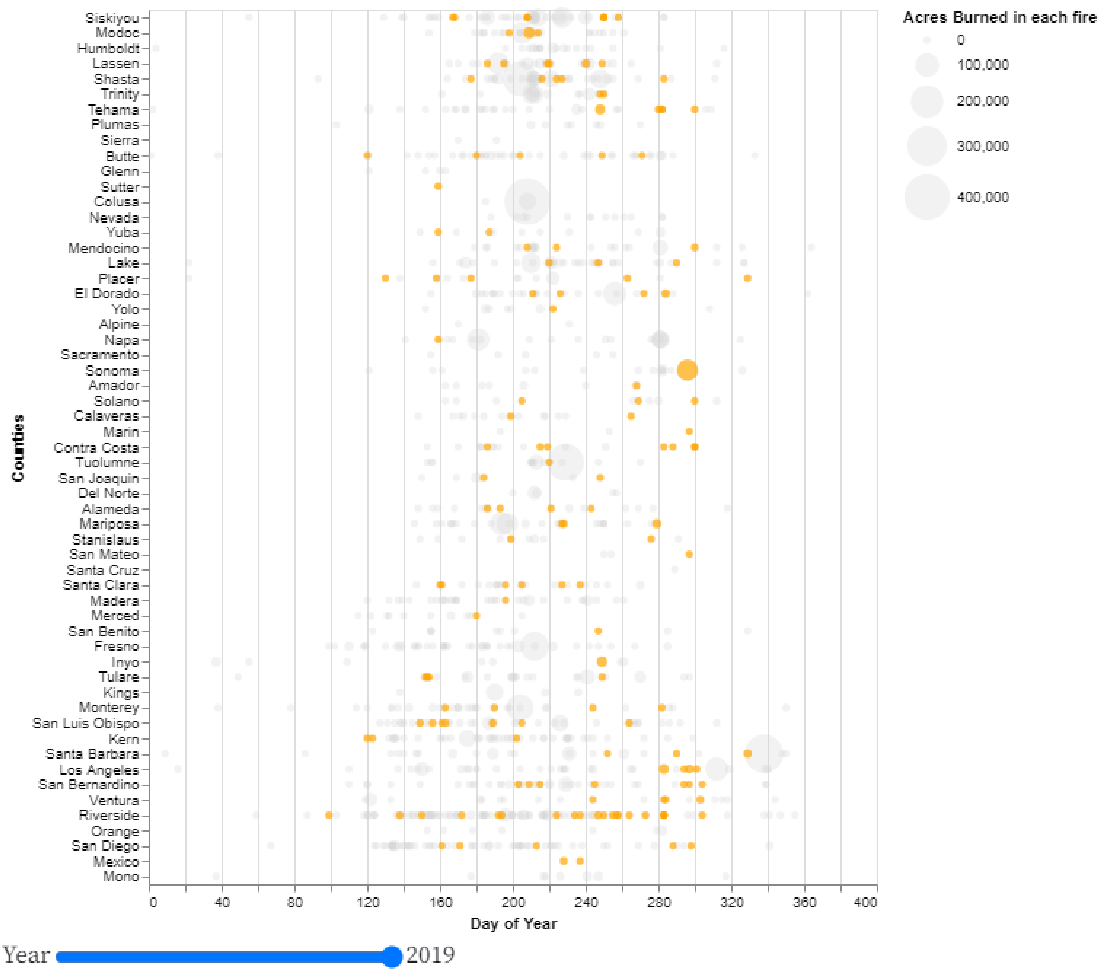

Siskiyou
Modoc
Humboldt
Lassen
Shasta
Trinity
Tehama
Plumas
Sierra
Butte
Glenn
Sutter
Colusa
Nevada
Yuba
Mendocino
Lake
Placer
El Dorado
Yolo
Alpine
Napa
Sacramento
Sonoma
Amador
Solano
Calaveras
Marin
Contra Costa
Tuolumne
San Joaquin
Del Norte
Alameda
Mariposa
Stanislaus
San Mateo
Santa Cruz
Santa Clara
Madera
Merced
San Benito
Fresno
Inyo
Tulare
Kings
Monterey
San Luis Obispo
Kern
Santa Barbara
Los Angeles
San Bernardino
Ventura
Riverside
Orange
San Diego
Mexico
Mono
Counties
Goose Fire
0
40
80
120
160
200
240
280
320
360
400
Day of Year
Acres Burned in each fire
0
100,000
200,000
300,000
400,000
Year
2015



Acres Burned in each fire
0
100,000
200,000
300,000
400,000
Counties
Siskiyou
Modoc
Humboldt
Lassen
Shasta
Trinity
Tehama
Plumas
Sierra
Butte
Glenn
Sutter
Colusa
Nevada
Yuba
Mendocino
Lake
Placer
El Dorado
Yolo
Alpine
Napa
Sacramento
Sonoma
Amador
Solano
Calaveras
Marin
Contra Costa
Tuolumne
San Joaquin
Del Norte
Alameda
Mariposa
Stanislaus
San Mateo
Santa Cruz
Santa Clara
Madera
Merced
San Benito
Fresno
Inyo
Tulare
Kings
Monterey
San Luis Obispo
Kern
Santa Barbara
Los Angeles
San Bernardino
Ventura
Riverside
Orange
San Diego
Mexico
Mono
0
40
80
120
160
200
240
280
320
360
400
Day of Year
Year
2016


Acres Burned in each fire
0
100,000
200,000
300,000
400,000
Counties
Siskiyou
Modoc
Humboldt
Lassen
Shasta
Trinity
Tehama
Plumas
Sierra
Butte
Glenn
Sutter
Colusa
Nevada
Yuba
Mendocino
Lake
Placer
El Dorado
Yolo
Alpine
Napa
Sacramento
Sonoma
Amador
Solano
Calaveras
Marin
Contra Costa
Tuolumne
San Joaquin
Del Norte
Alameda
Mariposa
Stanislaus
San Mateo
Santa Cruz
Santa Clara
Madera
Merced
San Benito
Fresno
Inyo
Tulare
Kings
Monterey
San Luis Obispo
Kern
Santa Barbara
Los Angeles
San Bernardino
Ventura
Riverside
Orange
San Diego
Mexico
Mono
0
40
80
120
160
200
240
280
320
360
400
Day of Year
Thomas Fire
Year
2017


Acres Burned in each fire
0
100,000
200,000
300,000
400,000
Counties
Siskiyou
Modoc
Humboldt
Lassen
Shasta
Trinity
Tehama
Plumas
Sierra
Butte
Glenn
Sutter
Colusa
Nevada
Yuba
Mendocino
Lake
Placer
El Dorado
Yolo
Alpine
Napa
Sacramento
Sonoma
Amador
Solano
Calaveras
Marin
Contra Costa
Tuolumne
San Joaquin
Del Norte
Alameda
Mariposa
Stanislaus
San Mateo
Santa Cruz
Santa Clara
Madera
Merced
San Benito
Fresno
Inyo
Tulare
Kings
Monterey
San Luis Obispo
Kern
Santa Barbara
Los Angeles
San Bernardino
Ventura
Riverside
Orange
San Diego
Mexico
Mono
0
40
80
120
160
200
240
280
320
360
400
Day of Year
Year
2018



c. What have you learned from this visualization? Is there additional information that is not described by this visualization or dataset that you think would enrich your interpretation?

I think this visualization can provide us with a dynamic view to see how the wildfires change over years in California. For example, the frequency of wildfires happen in different counties and the time intervals of most wildfires happen. This will be very helpful while managing the wildfires and preventing them. I think the information such as how these wildfires were stopped (naturally or manually), which is not described in the dataset, may help me enrich my interpretation.

(3) Pokemon

This problem gives practice in deriving new variables to improve a faceted plot. The data below give attack and defense statistics for Pokemon, along with their types. We will build a visualization to answer the question – how do the different types of Pokemon vary in their attack and defense potential?

```
pokemon <- read_csv("https://uwmadison.box.com/shared/static/hf5cmx3ew3ch0v6t0c2x56838er1lt2c.csv")
```

a. Derive a new column containing the attack-to-defense ratio, defined as $\frac{\text{Attack}}{\text{Defense}}$.

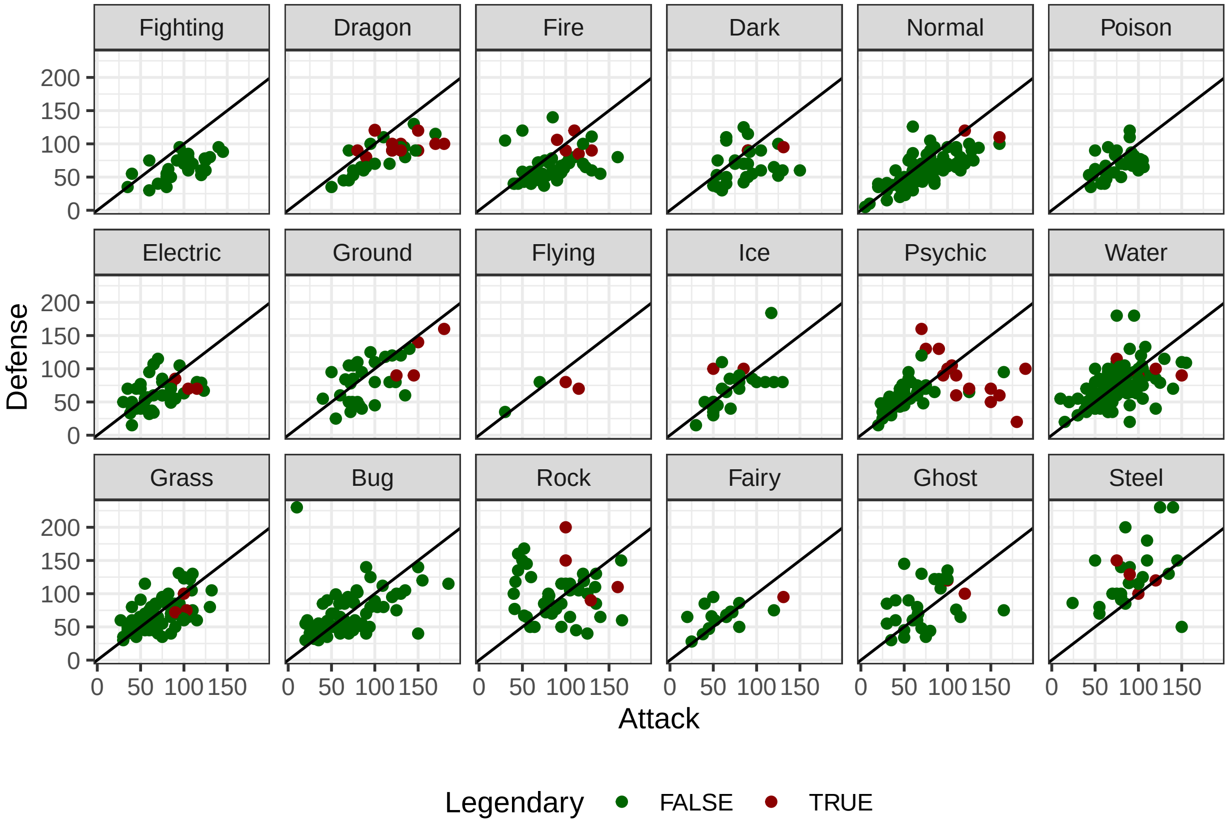
```
pokemon.new <- pokemon %>%
  #add the new column called ADR(attack-to-defense ratio)
  mutate(ADR = Attack / Defense)
```

b. For each `type_1` group of Pokemon, compute the median attack-to-defense ratio.

```r
pokemon.group <- pokemon.new %>%
  group_by(type_1) %>%
  summarise(medianADR = median(ADR)) #compute the median attack-to-defense ratio
```

c. Plot the attack vs. defense scores for each Pokemon, faceted by `type_1`. Use the result of (b) to ensure that the panels are sorted from types with highest to lowest attack-to-defense ratio. Your result should look similar to Figure 3.

```r
#provide an order of type_1 from the highest ADR to the lowest ADR
order <- pokemon.group %>%
  arrange(desc(medianADR)) %>%
  pull("type_1")

#make the plot
pokemon.new %>%
  mutate(type_1 = factor(type_1, levels = order)) %>%
  ggplot(aes(Attack, Defense, col = Legendary, group = type_1)) +
  geom_point() +
  facet_wrap(. ~ type_1, nrow = 3) +
  labs(
    x = "Attack",
    y = "Defense"
  ) +
  scale_color_manual(
    values = c("darkgreen", "darkred"),
    labels = c("FALSE", "TRUE")
  ) +
  geom_abline(slope = 1) +
  theme_bw() +
  theme(
    legend.position = "bottom"
  )
```



d. Propose, but do not implement, a visualization of this dataset that makes use of dynamic queries. What questions would the visualization answer? What would be the structure of interaction, and how would the display update when the user provides a cue?

I think we can make use of dynamic queries to visualize pokemons of different types. For example, if we want to see Fighting pokemons, we can select "Fighting", and pokemons of other types will be filtered out. We can also provide more interactions, including selection of type_1, type_2, and legendary or not. Questions like "how many fighting pokemons are legendary" or "who are them" will be answered. If the user choose one type 1, we will display the pokemons of this type. If he then choose one type 2, the display will update because only pokemons with this type 1 and this type 2 will be shown.

(4) NYC Airbnb Data

In this problem, we'll create a visualization to dynamically query a dataset of Airbnb rentals in Manhattan in 2019. The steps below guide you through the process of building this visualization. Make sure to include your code, a screenshot, and a brief explanation of what you did for each step.

a. Make a scatterplot of locations (Longitude vs. Latitude) for all the rentals, colored in by `room_type`.

```
{
  //make the scatterplot with markCircle() function
  return vl.markCircle({filled: true, size: 7, opacity: 0.8})
    .data(data)
    .encode(
      //define x-axis
      vl.x().fieldQ("longitude").scale({domain: [-74.02, -73.90]}),
      //define y-axis
      vl.y().fieldQ("latitude").scale({domain: [40.71, 40.88]}),
      vl.color().fieldN("room_type").scale({range: ["#74c8ab", "#Efa374", "#Ada9e9"]})
    )
    .width(500)
    .height(400)
    .render()
}
```

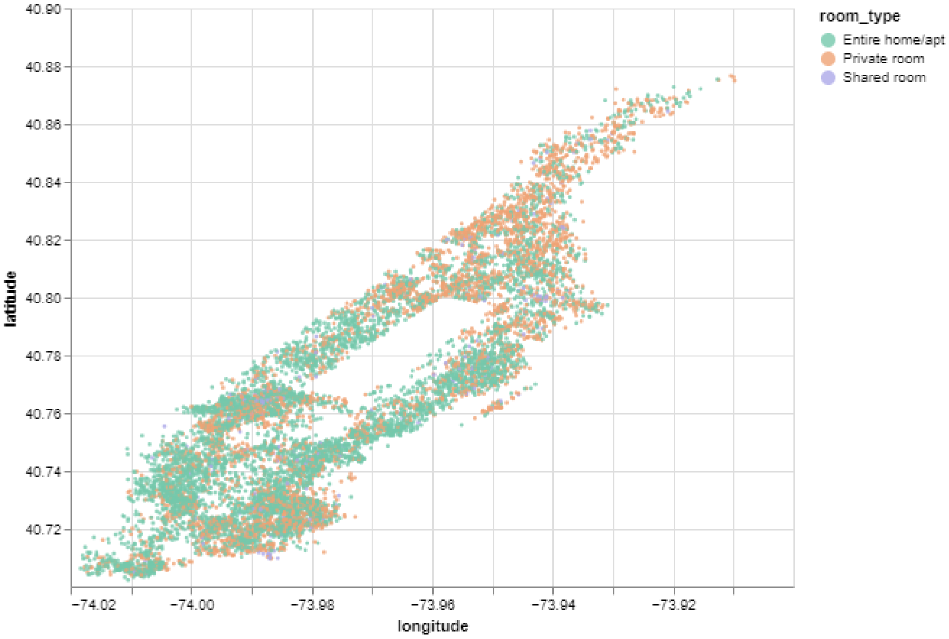


Figure 2: unchanged image

b. Make a histogram of the log-prices[1], with stacked colors to distinguish between room types. Vertically concatenate this histogram with the scatterplot from (a). An example of the resulting display is shown in Figure ??.

```
{
  //make the scatter point with markCircle() function
  const location = vl.markCircle({filled: true, size: 7, opacity: 0.8})
    .data(data)
    .encode(
      vl.x().fieldQ("longitude").scale({domain: [-74.02, -73.90]}),
      vl.y().fieldQ("latitude").scale({domain: [40.71, 40.88]}),
      vl.color().fieldN("room_type").scale({range: ["#74c8ab", "#Efa374", "#Ada9e9"]})
    )
    .width(500)
    .height(400);

  //make the histogram with markBar() function
  const logPrice = vl.markBar()
    .data(data)
    .encode(
      vl.x().fieldQ("log_price"),
      vl.y().count().title(null),
```

[1] We've applied a log-transform to the original rental prices because otherwise the prices are highly skewed.

```
      vl.color().fieldN("room_type").scale({range: ["#74c8ab", "#Efa374", "#Ada9e9"]})
    )
    .width(500)
    .height(80);

  //vertically concatenate these two plots
  return vl.vconcat(logPrice, location).render();
}
```

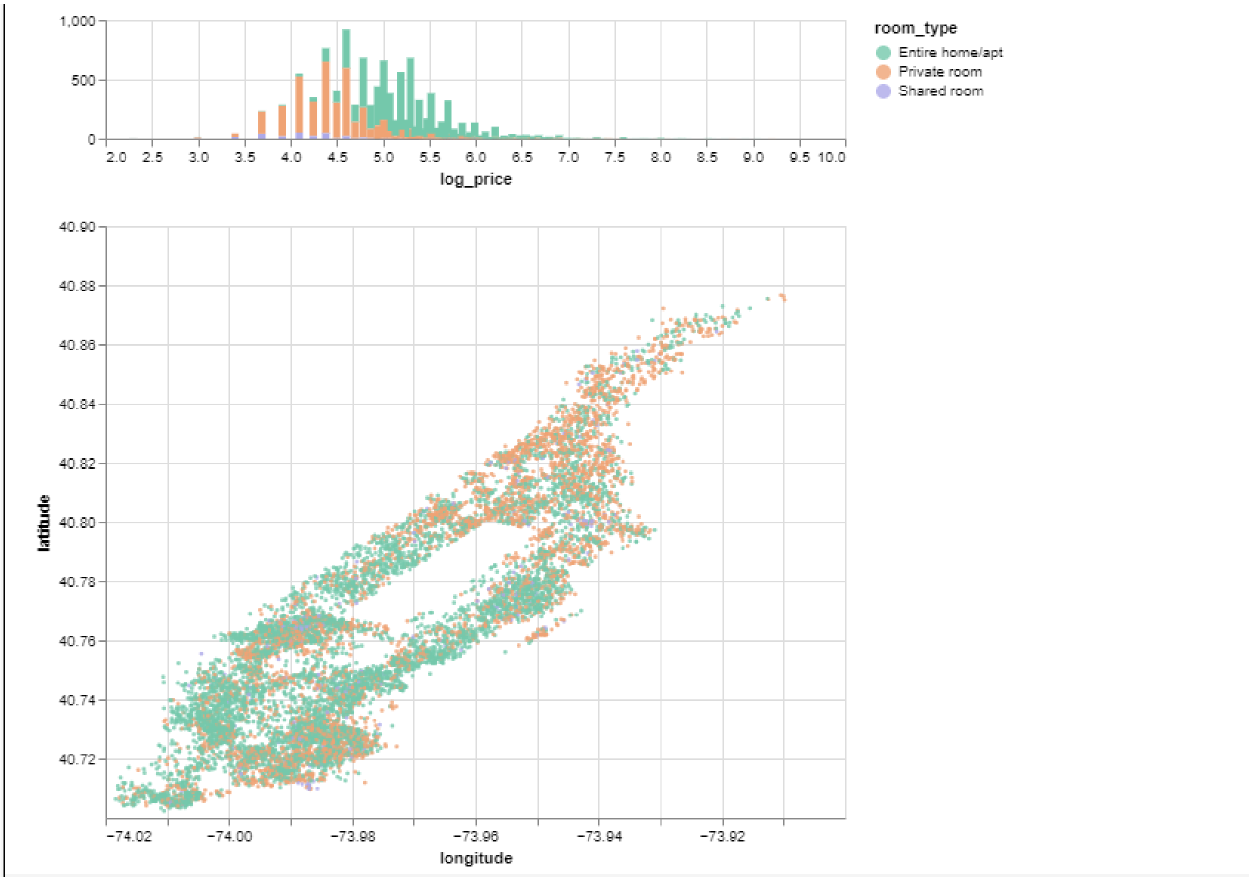
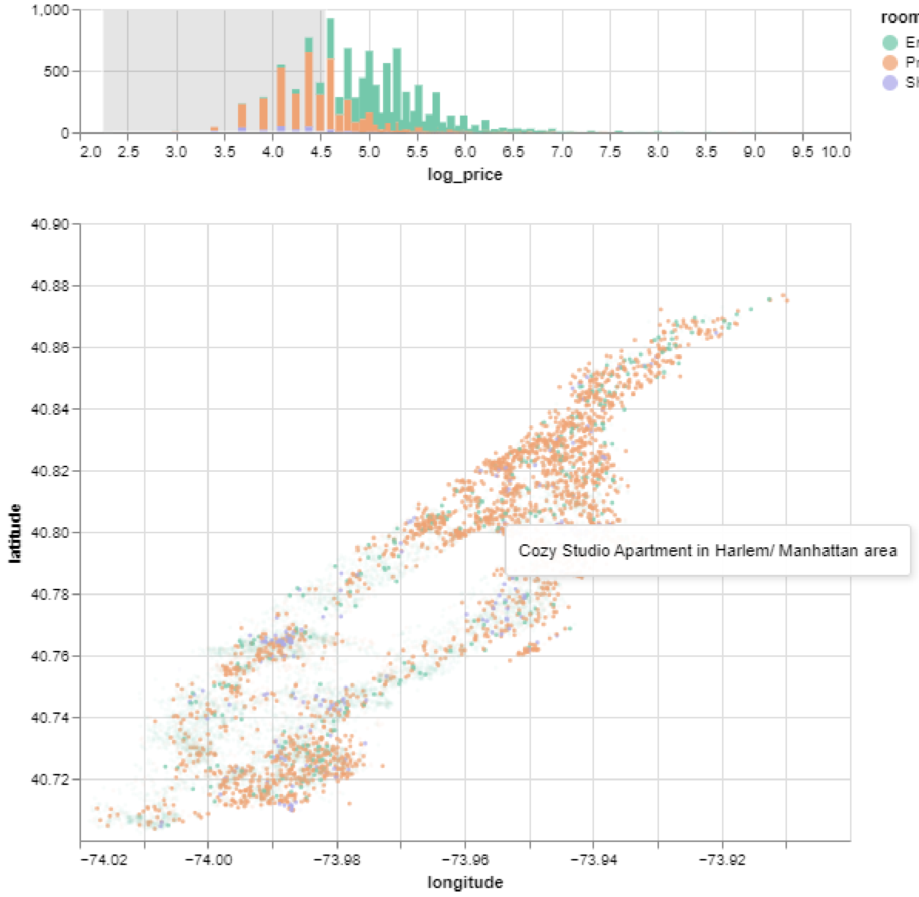
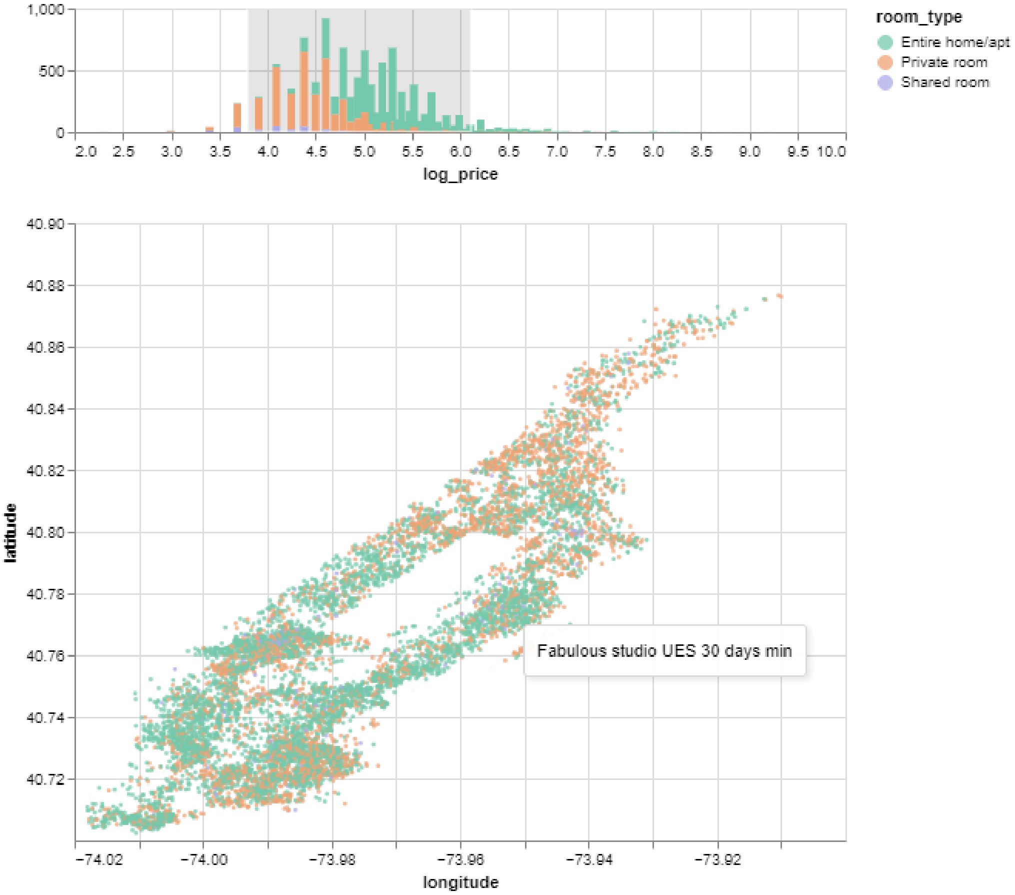


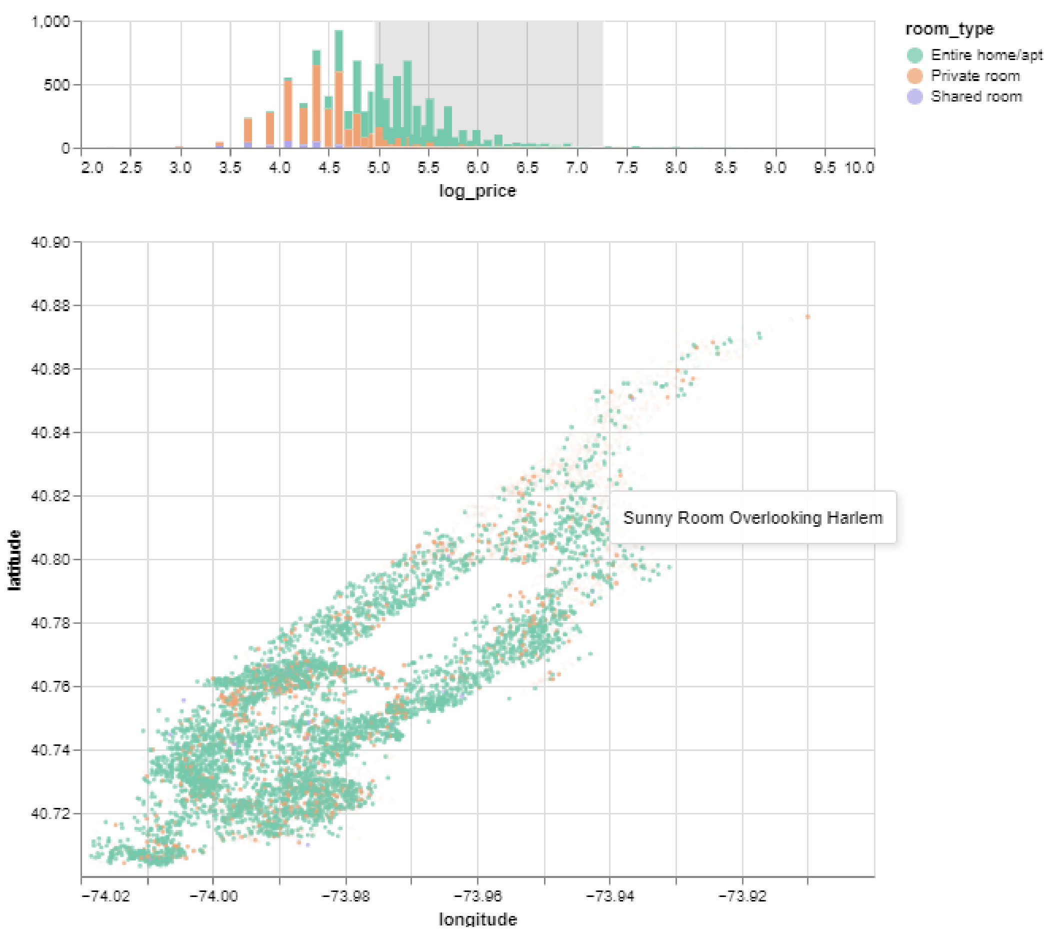
Figure 3: unchanged image

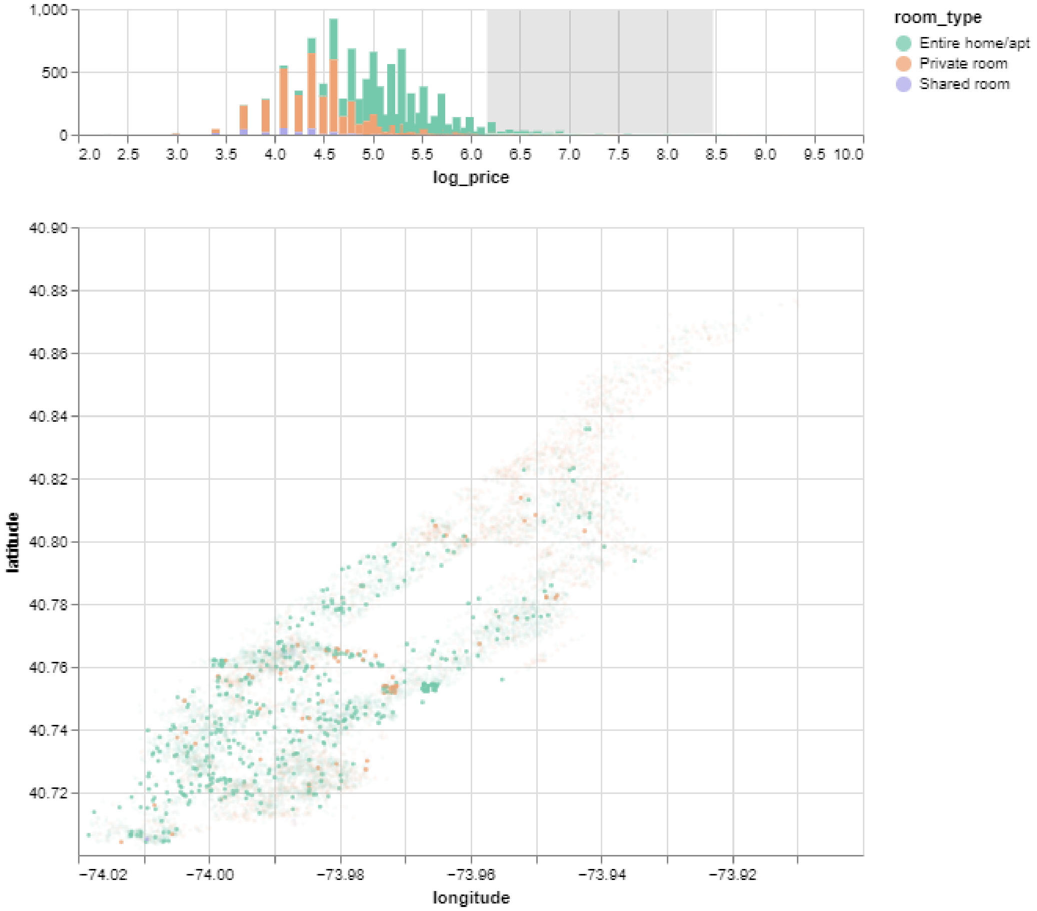
c. Introduce a selection so that brushing over price ranges highlights the associated rentals on the map. *Hint*: This is similar to the movie-ratings-over-time visualization from Week 3 [2].

```
{
  //add the selection
  const brush = vl.selectInterval()
    .encodings('x'); // limit selection to x-axis (year) values

  //update the scatter point with markCircle() function
  const location = vl.markCircle({filled: true, size: 7, opacity: 0.8})
    .data(data)
    .encode(
      vl.x().fieldQ("longitude").scale({domain: [-74.02, -73.90]}),
      vl.y().fieldQ("latitude").scale({domain: [40.71, 40.88]}),
      vl.tooltip().fieldN('name'),
      vl.color().fieldN("room_type").scale({range: ["#74c8ab", "#Efa374", "#Ada9e9"]}),
      vl.opacity().if(brush, vl.value(0.75)).value(0.05)
    )
    .width(500)
    .height(400);

  //update the histogram with markBar() function
  const logPrice = vl.markBar()
    .data(data)
    .select(brush)
    .encode(
      vl.x().fieldQ("log_price"),
      vl.y().count().title(null),
      vl.color().fieldN("room_type").scale({range: ["#74c8ab", "#Efa374", "#Ada9e9"]})
    )
    .width(500)
    .height(80);

  //vertically concatenate these two plots
  return vl.vconcat(logPrice, location).render();
}
```




1,000
500
0
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
9.5
10.0
log_price
room_type
● Entire home/apt
● Private room
● Shared room
40.90
40.88
40.86
40.84
40.82
40.80
40.78
40.76
40.74
40.72
latitude
−74.02
−74.00
−73.98
−73.96
−73.94
−73.92
longitude
Fabulous studio UES 30 days min


1,000
500
0
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
9.5
10.0
log_price
room_type
● Entire home/apt
● Private room
● Shared room
40.90
40.88
40.86
40.84
40.82
40.80
40.78
40.76
40.74
40.72
latitude
Sunny Room Overlooking Harlem
−74.02
−74.00
−73.98
−73.96
−73.94
−73.92
longitude



d. Comment on the resulting visualization. If you had a friend who was interested in renting an Airbnb in NYC, what would you tell them?

Based on this visualization, I can give him some advice about price and location. The shared rooms are always the cheapest. There are also some private rooms but the the entire apartments available below the average price. If he travels alone and does not want to spend much money, he can choose from these options. If he does not travels alone or he is rich, he can choose some more expensive rentals, including some private rooms and entire apartments. There is no shared room over the log price 5. However, to give him proper recommendations, I think more information, such as the room condition and the location information(like safety), should be integrated.

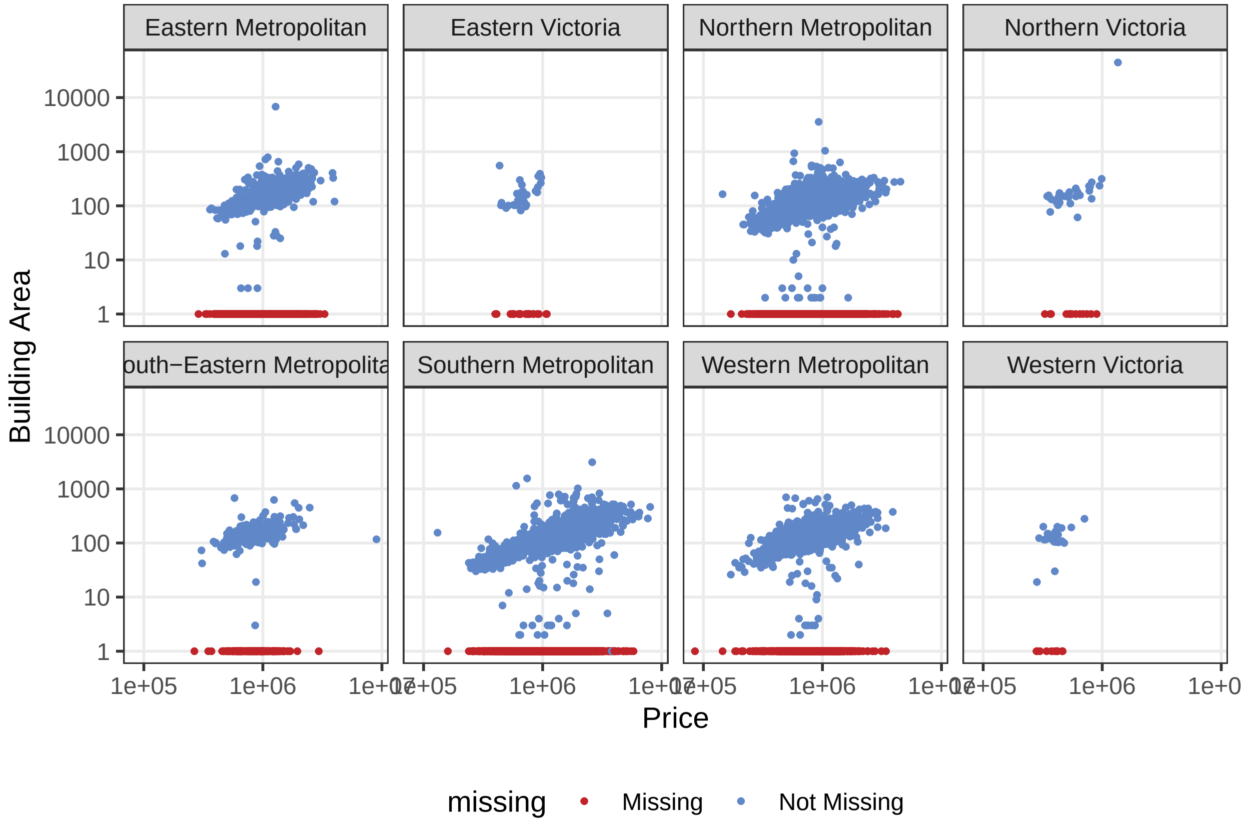
(5) Imputing Housing Data

This problem gives practice visualizing missing-data imputation. We will use another housing-themed dataset, this one describing homes for sale in Melbourne. Notice that the `BuildingArea` and `YearBuilt` variables have many missing values.

```r
housing <- read_csv("https://uwmadison.box.com/shared/static/h5u176syp4xkret4w89n70efsp1tubex.csv")
```

a. Using `geom_miss_point`, make a plot of `Price` against `BuildingArea`, faceted by the region from which the home is located. Make sure the observations with missing `BuildingArea` values are still displayed somewhere on the plot, as in Figure 5. What does the plot suggest about how you might impute the `BuildingArea` column?

```r
#make the plot
ggplot(housing) +
  geom_miss_point(aes(x = Price, y = BuildingArea), size = 0.7) +
  scale_x_log10() +
  scale_y_log10() +
  facet_wrap(. ~ Regionname, nrow = 2) +
  labs(
    x = "Price",
    y = "Building Area"
  ) +
  scale_color_manual(
    values = c("#C02428", "#6088c8"),
    labels = c("Missing", "Not Missing")
  ) +
  theme_bw() +
  theme(
    legend.position = "bottom",
    panel.grid.minor = element_blank()
  )
```

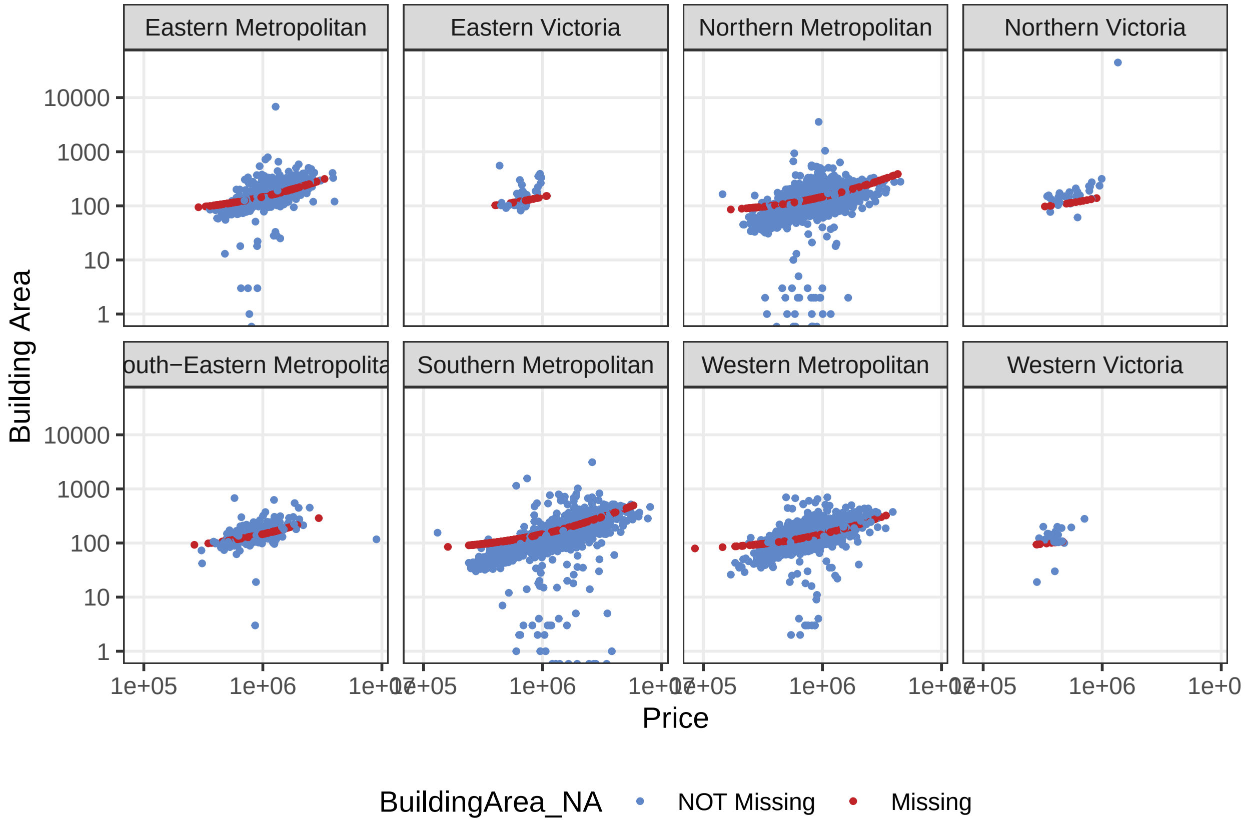


b. Using the `impute_lm` package, impute the `BuildingArea` variable. You may use whichever fully measured columns that you like – using `BuildingArea ~ Price` is a reasonable starting point.

```
housing_imputed <- housing %>%
  bind_shadow() %>%
  as.data.frame() %>%
  impute_lm(BuildingArea ~ Price)
```

c. Recreate the visualization from part (a), but with the imputed data included. Make sure to distinguish between values that were truly measured and those that were imputed in part (b). Comment on the quality of the results. Do you notice anything unusual about the imputations in Northern Victoria?

```
ggplot(housing_imputed) + #use inputed data
  #distinguish with colors
  geom_point(aes(x = Price, y = BuildingArea, col = BuildingArea_NA), size = 0.7) +
  scale_x_log10() +
  scale_y_log10() +
  facet_wrap(. ~ Regionname, nrow = 2) +
  labs(
    x = "Price",
    y = "Building Area"
  ) +
  scale_color_manual(
    values = c("#6088c8", "#C02428"),
    labels = c("NOT Missing", "Missing")
  ) +
  theme_bw() +
  theme(
    legend.position = "bottom",
    panel.grid.minor = element_blank()
  )
```



I think most of the imputed data can be merged with the original data. However, the imputations in Northern Victoria is a bit lower then most of measured data. Also, I think it is not as smooth as other imputations.

Feedback

a. How much time did you spend on this homework? At least eight hours, about one day.
b. Which problem did you find most valuable? I think the pokemon problem is the most valuable for me. I like playing pokemons and watching that anime. So I am really interested in this problems and enjoy the process of figuring it out.