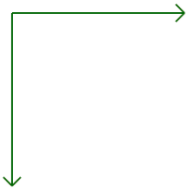


[`sudo sudo lsof -t -i tcp:8000 | xargs kill -9`](#)

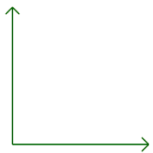
1.1.1 What's the difference between SVG Coordinate Space and Mathematical/Graph Coordinate Space?

The SVG coordinate system the point $x=0, y=0$ is the upper left corner. The y-axis is thus reversed compared to a normal graph coordinate system. As y increases in SVG, the points, shapes etc. move down, not up.



In a normal cartesian coordinate system the point $x=0, y=0$ is at the lower left corner of the graph. As x increases the points move to the right in the coordinate system. As x decreases the points move to the left in the coordinate system. As y increases the points move up in the coordinate system. As y decreases the points move down in the coordinate system.

Here is an illustration of a normal graph coordinate system with 0,0 at the lower left corner:



1.1.2 What is enter() and exit() in d3.js?

The enter function's parameter enter is the enter selection which represents the elements that need to be created. Also, the exit function's parameter exit is the exit selection and contains elements that need to be removed.

1.1.3 What is transform and translate in SVG?

The translate method means it moves the object by x and y. It is assumed to be 0 if y is not provided.

Also, the transform attribute states the list of transform definitions that are applied to an element and its children.

1.1.4 Try to understand the idea of anonymous function and its use in d3.js. If there is a list $a = [a, c, b, d, e]$, what is the return value of this anonymous function: `map(function(d,i) {return i+5})`

[5,6,7,8,9]

1.1.5 Compare the 2 code snippets below, what will be the output of the 2 code snippets will it be different or same. If it is different, give reasoning on why it is different. (Hint: what does the enter() function does)

Code snippet I:

```
<body>
<p>D3 Tutorials </p>
<script>
  var myData = ['big', 'data', 'assignment', 2, 'submission'];
  var p = d3.select("body")
    .selectAll("p")
    .data(myData)
    .text(function(d, i) {
      return d;
    });
</script>
</body>
```

Code snippet II:

```
<body>
<script>
  var data = ['big', 'data', 'assignment', 2, 'submission'];
  var body = d3.select("body")
    .selectAll("span")
    .data(data)
    .enter()
    .append("span")
    .text(function(d){return d + " "});
</script>
</body>
```

The enter function's parameter enter is the enter selection which represents the elements that need to be created.

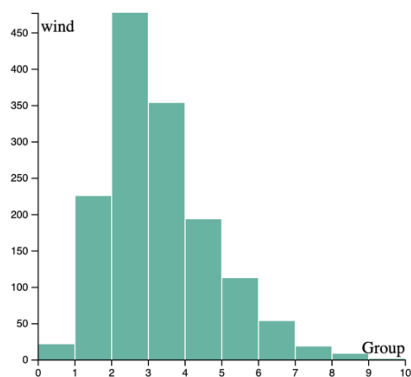
snippet 1 is big

snippet 2 have big data assignment 2 submission in the span element

Yes, the output of the Code snippets is different. When we selected our paragraph elements using `.selectAll("p")`, it returned only one element in the selection because we had only one paragraph element. Hence, when we bind the data to our paragraph selection, only the first data element is bound to the one available paragraph element. Rest of the data elements from the array were ignored because there were no other `<p>` elements.

Part II, there are 5 data values in our array. So enter() will create five reference placeholders and append will create five span elements. We provide our data array to the data() function. Since our array has six elements, the code after this will run six times for every element in the array. The enter() function checks for elements corresponding to our five array elements.

1.2.1



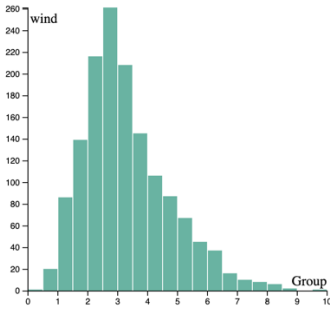
<p> The distribution is mainly around 1 -3, and the highest is around 450.</p>

```

8 <!-- Create a div where the graph will take place -->
9 <div id="dataviz"></div>
10 <p style="background-color:powderblue;"> The distribution is mainly around 1 -3, and the highest is around 450.</p>
11 <script>
12
13 // set the dimensions and margins of the graph
14 // You can change these values these are just sample values given
15 var margin = {top: 20, right: 50, bottom: 50, left: 60},
16   width = 460 - margin.left - margin.right,
17   height = 400 - margin.top - margin.bottom;
18
19 // append the svg object to the body of the page
20 var svg = d3.select("#dataviz")
21   .append("svg")
22   .attr("width", width + margin.left + margin.right)
23   .attr("height", height + margin.top + margin.bottom)
24   .append("g")
25   .attr("transform",
26     "translate(" + margin.left + "," + margin.top + ")");
27
28 // uncomment the function and complete this function to plot required graphs
29 // d3.csv("https://github.com/vega/vega/blob/main/docs/data/seattle-weather.csv", function(data) {
30 d3.csv("seattle-weather.csv", function(data) {
31   console.log(data[0])
32   // X axis: scale and draw:
33   var x = d3.scaleLinear()
34     .domain([0, 1000]) // return + d.wind }
35     .range([0, width]); // can use this instead of 1000 to have the max of data: d3.max(data, function(d) { return +d
36   // var xaxis=
37   .domain([0, 10])
38   .range([0, width]);
39   svg.append("g")
40     .attr("transform", "translate(0," + height + ")")
41     .call(d3.axisBottom(x));
42
43 // set the parameters for the histogram
44 var histogram = d3.histogram()

```

1.2.2



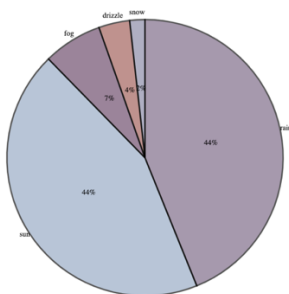
<p> The distribution is mainly around 1 -3, and the highest is around 260.<p>

```

19 // append the svg object to the body of the page
20 var svg = d3.select("#dataviz")
21   .append("svg")
22   .attr("width", width + margin.left + margin.right)
23   .attr("height", height + margin.top + margin.bottom)
24   .append("g")
25   .attr("transform",
26     "translate(" + margin.left + "," + margin.top + ")");
27
28 // uncomment the function and complete this function to plot required graphs
29 // d3.csv("https://github.com/yvesv/vega/blob/main/docs/data/seattle-weather.csv", function(data) {
30 d3.csv("seattle-weather.csv", function(data) {
31   console.log(data[0]);
32   // X axis: scale and draw:
33   var x = d3.scaleLinear()
34     .domain([0, 1000]) // can use this instead of 1000 to have the max of data: d3.max(data, function(d) { return +d.wind })
35     .range([0, width]);
36   // var axis = d3.max(data, function(d) { return +d.wind });
37   .domain([0, 10]);
38   .range([0, width]);
39   svg.append("g")
40     .attr("transform", "translate(0," + height + ")")
41     .call(d3.axisBottom(x));
42
43   // set the parameters for the histogram
44   var histogram = d3.histogram()
45     .value(function(d) { return d.wind; }) // I need to give the vector of value
46     .domain(x.domain()) // then the domain of the graphic
47     .thresholds(x.ticks(25)); // then the numbers of bins
48
49   // And apply this function to data to get the bins
50   var bins = histogram(data);
51
52   // Y axis: scale and draw:
53   var y = d3.scaleLinear()
54     .range([0, height]);

```

1.2.3



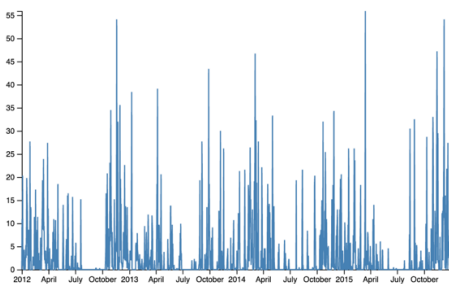
<p> The distribution is mainly occupied by rain and sun. The highest percent is 44%<p>

```

14 .domain(data)
15 .range(["#800000", "#f08080", "#f08080", "#800000", "#a5d5a5"])
16 var pie = d3.pie()
17 .value(function(d) { return d.value; })
18 var data_ready = pie(d3.entries(data))
19
20 // Build the pie chart. Basically, each part of the pie is a path that we build using the arc function.
21 svg1
22 .selectAll("whatever")
23 .data(data_ready)
24 .enter()
25 .append("path")
26 .attr("d", d3.arc()
27   .innerRadius(0)
28   .outerRadius(radius)
29   )
30 .attr("fill", function(d) { return color(d.data.key); })
31 .attr("stroke", "black")
32 .style("stroke-width", "2px")
33 .style("opacity", 0.7)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

1.2.4



<p> 2015 April and October have the high value around 55. </p>

```

<script>
- var margin = {top: 20, right: 50, bottom: 50, left: 60},
  width = 660 - margin.left - margin.right,
  height = 400 - margin.top - margin.bottom;

  var svg2 = d3.select("#my_data")
  .append("svg")
  .attr("width", width + margin.left + margin.right)
  .attr("height", height + margin.top + margin.bottom)
  .append("g")
  .attr("transform",
    "translate(" + margin.left + "," + margin.top + ")");

  d3.csv("seattle-weather.csv", function(d) {
    return {date: d3.timeParse("%Y-%m-%d")(d.date), value: d.precipitation}
  });

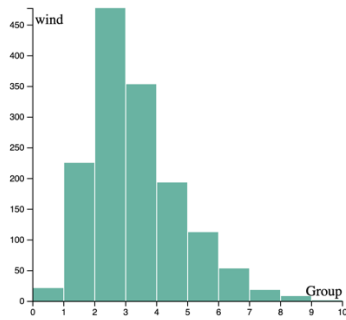
  function(data){
  - var x = d3.scaleTime()
    .domain(d3.extent(data, function(d) { return d.date; }))
    .range([0, width]);
    svg2.append("g")
    .attr("transform", "translate(0," + height + ")")
    .call(d3.axisBottom(x));

  - var y = d3.scaleLinear()
    .domain([0, d3.max(data, function(d) { return +d.value; })])
    .range([height, 0]);
    svg2.append("g")
    .call(d3.axisLeft(y));

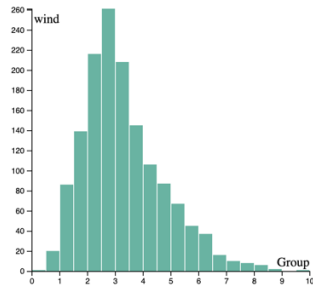
  - svg2.append("path")
    .datum(data)
    .attr("fill", "none")
    .attr("stroke", "steelblue")
    .attr("stroke-width", 1.5)
    .attr("d", d3.line()
      .x(function(d) { return x(d.date); })

```

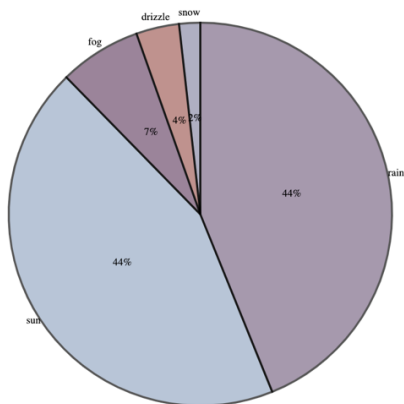
1.2.5 Add a paragraph to the webpage using the <p> tag to note down the main observations from the above obtained plots in 1.2.1, 1.2.2, 1.2.3, 1.2.4 (for eg. distribution, count, important understanding etc).



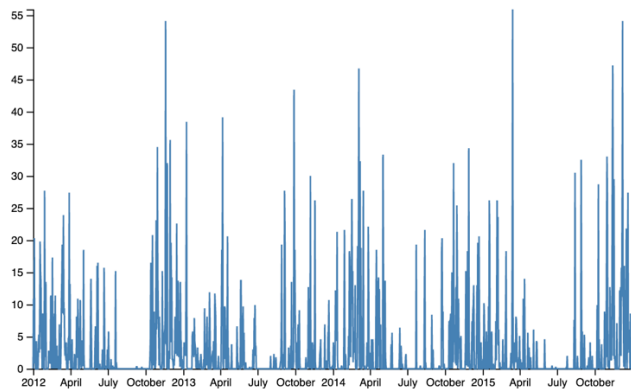
The distribution is mainly around 1 -3, and the highest is around 450.



The distribution is mainly around 1 -3, and the highest is around 260.



The distribution is mainly occupied by rain and sun. The highest percent is 44%



2015 April and October have the high value around 55.

```
<div id="dataviz1"></div>
<p> The distribution is mainly around 1 -3, and the highest is around 450.</p>
<div id="dataviz2"></div>
<p> The distribution is mainly around 1 -3, and the highest is around 260.</p>
<div id="dataviz3"></div>
<p> The distribution is mainly occupied by rain and sun. The highest percent is 44%</p>
<div id="dataviz4"></div>
<p> 2015 April and October have the high value around 55. </p>

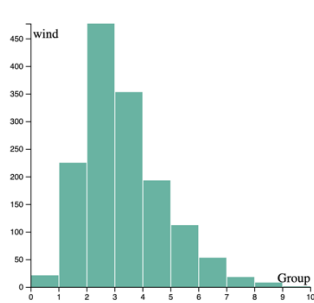
<script>

// set the dimensions and margins of the graph
// You can change these values these are just sample values given
var margin = {top: 20, right: 50, bottom: 50, left: 60},
    width = 400 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

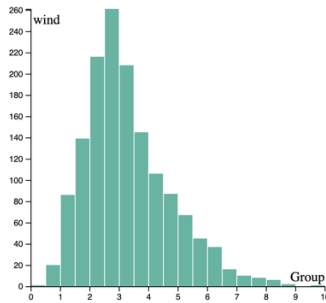
// append the svg object to the body of the page
var svg = d3.select("#dataviz")
    .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform",
        "translate(" + margin.left + "," + margin.top + ")");

// uncomment the function and complete this function to plot required graphs
// d3.csv("https://github.com/vega/vega/blob/main/docs/data/seattle-weather.csv", function(data) {
d3.csv("seattle-weather.csv", function(data) {
    console.log(data[0])
    // 2 axes scale and draw
    var x = d3.scaleLinear()
    // d3.max(data, function(d) { return + d.wind })
    // .domain([0, 1000]) // can use this instead of 1000 to have the max of data: d3.max(data, function(d) { return +d.pr
    // var xAxis= d3.max(data, function(d) { return + d.wind })
```

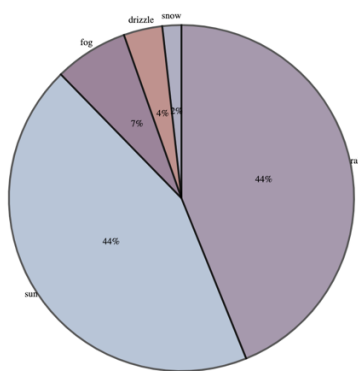
1.2.6 Use the internal style sheet to change the style (font size, font color, indentation) of the paragraph of the web page you created in 1.2.5.



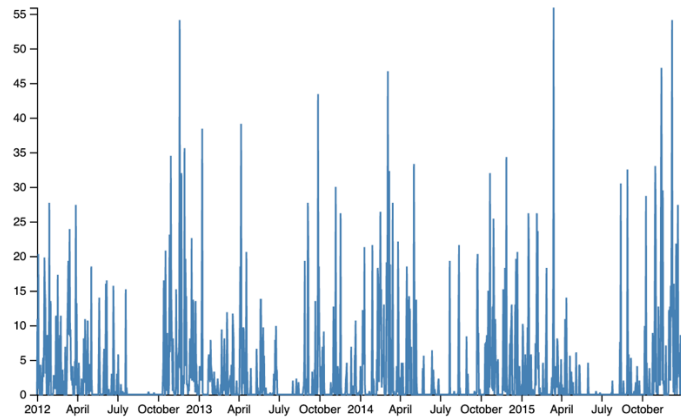
The distribution is mainly around 1 -3, and the highest is around 450.



The distribution is mainly around 1 -3, and the highest is around 260.



The distribution is mainly occupied by rain and sun. The highest percent is 44% 2015 April and October have the high value around 55.



```
<!DOCTYPE html>
<meta charset="UTF-8">

<!-- Load d3.js you can use other version if you want -->
<script src="https://d3js.org/d3.v4.js"></script>
<div id="dataviz1"></div>
<p style="background-color:powderblue;"> The distribution is mainly around 1 -3, and the highest is around 450.</p>
<div id="dataviz2"></div>
<p style="background-color:powderblue;"> The distribution is mainly around 1 -3, and the highest is around 260.</p>
<div id="dataviz3"></div>
<p style="color:blue;"> The distribution is mainly occupied by rain and sun. The highest percent is 44%</p>
<div id="dataviz4"></div>
<p style="color:blue;"> 2015 April and October have the high value around 55. </p>

<script>

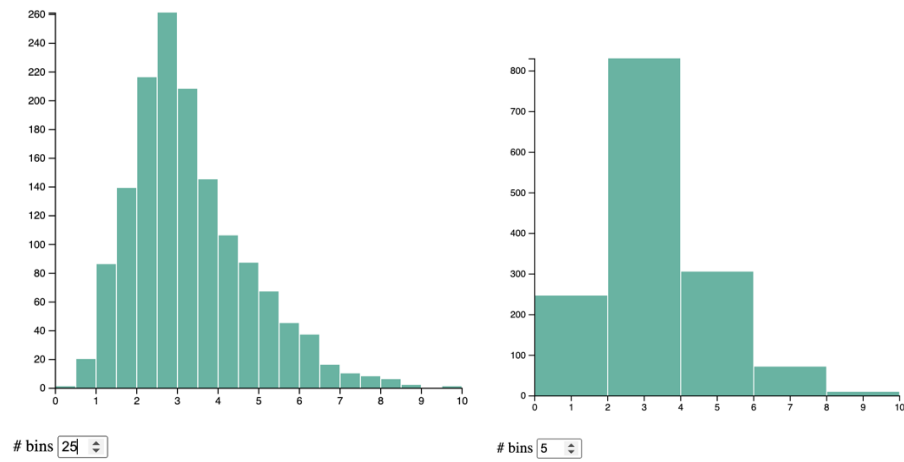
// set the dimensions and margins of the graph
// You can change these values these are just sample values given
var margin = {top: 20, right: 50, bottom: 50, left: 60},
    width = 400 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

// append the svg object to the body of the page
var svg = d3.select("#dataviz1")
    .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform",
        "translate(" + margin.left + "," + margin.top + ")");

// uncomment the function and complete this function to plot required graphs
d3.csv("https://github.com/vega/vega/blob/main/docs/data/seattle-weather.csv", function(data) {
    console.log(data[0])
    // X axis: scale and draw:

```

1.3.1



```
d3.csv("seattle-weather.csv", function(data) {
  // X axis: scale and draw
  var x = d3.scaleLinear()
    .domain([0, 10]) // can use this instead of 1000 to have the max of data: d3.max(data, function(d) { return +d.price })
    .range([0, width]);
  svg.append("g")
    .attr("transform", "translate(0, height + 10)")
    .call(d3.axisBottom(x));

  // Y axis: initialization
  var y = d3.scaleLinear()
    .range([height, 0]);
  var yAxis = svg.append("g");

  // A function that builds the graph for a specific value of bin
  function update(nBin) {
    // set the parameters for the histogram
    var histogram = d3.histogram()
      .value(function(d) { return d.wind; }) // I need to give the vector of value
      .domain(x.domain()) // then the domain of the graphic
      .thresholds(x.ticks(nBin)); // then the numbers of bins

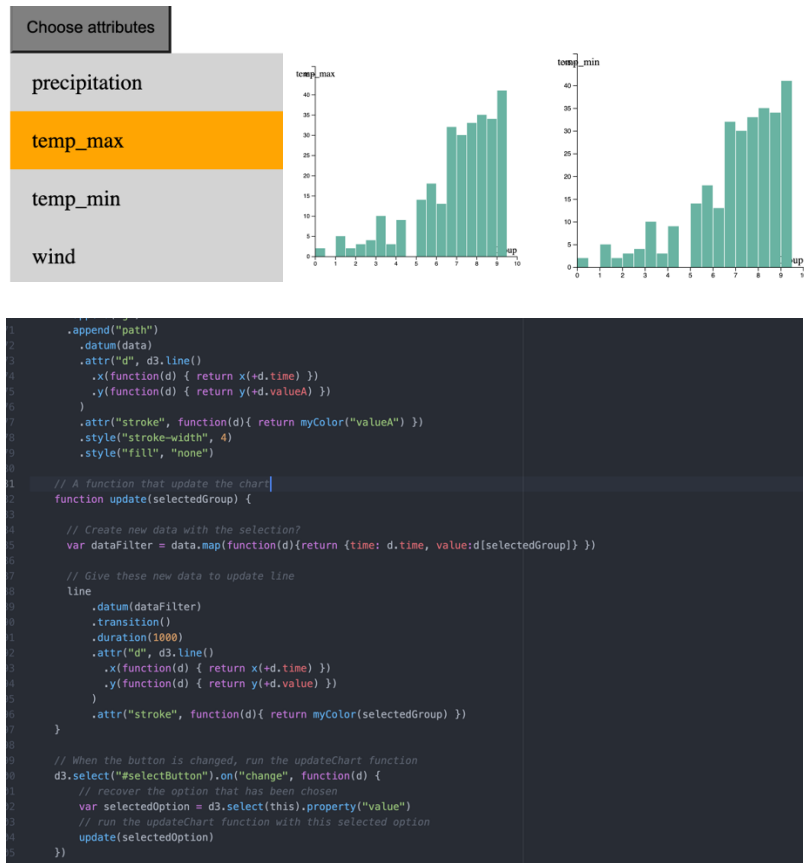
    // And apply this function to data to get the bins
    var bins = histogram(data);
    console.log(bins);

    // X axis: update new that we know the domain
    y.domain([0, d3.max(bins, function(d) { return d.length; })]); // d3.hist has to be called before the Y axis obviously
    yAxis
      .transition()
      .duration(100)
      .call(d3.axisLeft(y));

    // Join the rect with the bins data
    var u = svg.selectAll("rect");
  }
});
```

```
51 var bins = histogram(data);
52 console.log(bins);
53 // X axis: update new that we know the domain
54 y.domain([0, d3.max(bins, function(d) { return d.length; })]); // d3.hist has to be called before the Y axis obviously
55 yAxis
56   .transition()
57   .duration(100)
58   .call(d3.axisLeft(y));
59
60 // Join the rect with the bins data
61 var u = svg.selectAll("rect")
62   .data(bins);
63
64 // Manage the existing bars and eventually the new ones:
65 u
66   .enter()
67   .append("rect") // Add a new rect for each new elements
68   .merge(u) // get the already existing elements as well
69   .transition() // and apply changes to all of them
70   .duration(1000)
71   .attr("x", 1)
72   .attr("transform", function(d) { return "translate(" + x(d.x0) + "," + y(d.length) + ")"; })
73   .attr("width", function(d) { return x(d.x1) - x(d.x0) - 2; })
74   .attr("height", function(d) { return height - y(d.length); })
75   .style("fill", "#90b3a2");
76
77 // If less bar in the new histogram, I delete the ones not in use anymore
78 u
79   .exit()
80   .remove();
81
82 }
83
84 ]
85
86 }
```


1.3.2



Part II Tasks: (60 marks)

2.1

mpg	cyl	displ	hp	weight	accel	yr	origin	name
18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
15	8	350	165	3693	11.5	70	1	buick skylark 320
18	8	318	150	3436	11	70	1	plymouth satellite
16	8	304	150	3433	12	70	1	amc rebel sst

```

$(document).ready(function(){
  $('#load_data').click(function(){
    $.ajax({
      url:"auto-mpg.csv",
      dataType:"text",
      success:function(data)
      {
        var data = data.split(/\r?\n\r/);
        var table_data = '<table class="table table-bordered table-striped">';
        for(var count = 0; count< 6; count++)
        {
          var cell_data = data[count].split(",");
          table_data += '<tr>';
          for(var cell_count=0; cell_count<cell_data.length; cell_count++)
          {
            if(count == 0)
            {
              table_data += '<th>'+cell_data[cell_count]+'</th>';
            }
            else
            {
              table_data += '<td>'+cell_data[cell_count]+'</td>';
            }
          }
          table_data += '</tr>';
        }
        table_data += '</table>';
        // console.log(table_data)
        $('#table').html(table_data);
      }
    });
  });
}

```

2.1.2

each model year	number of cars	number of cars
70		29
71		28
72		28
73		40
74		27
75		30
76		34
77		28
78		29
79		29
80		29
81		31

```

<script>
  data = {70:29, 71:28, 72:28,73:40, 74:27, 75:30, 76:34 , 77:28 , 78:29 , 79:29, 80: 29, 81 : 31 }
  function showTableData() {
    document.getElementById('info').innerHTML = "";
    var myTab = document.getElementById('empTable');

    // LOOP THROUGH EACH ROW OF THE TABLE AFTER HEADER.
    for (i = 1; i < myTab.rows.length; i++) {

      // GET THE CELLS COLLECTION OF THE CURRENT ROW.
      var objCells = myTab.rows.item(i).cells;

      // LOOP THROUGH EACH CELL OF THE CURENT ROW TO READ CELL VALUES.
      for (var j = 0; j < objCells.length; j++) {
        info.innerHTML = info.innerHTML + ' ' + objCells.item(j).innerHTML;
      }
      info.innerHTML = info.innerHTML + '<br />';    // ADD A BREAK (TAG).
    }
  }
</script>
</html>

```

2.1.3

modelyear	cylinder
70	196
71	156
72	163
73	255
74	142
75	168
76	153
77	153
78	193
79	169
80	120
81	134
82	130

```
<script>
$(document).ready(function(){
  $('#load_data').click(function(){
    $.ajax({
      url:"2.1.4.csv",
      dataType:"text",
      success:function(data)
      {
        var data = data.split(/\r?\n|\r/);
        var table_data = '<table class="table table-bordered table-striped">';
        for(var count = 0; count<data.length; count++)
        {
          var cell_data = data[count].split(" ");
          table_data += '<tr>';
          for(var cell_count=0; cell_count<cell_data.length; cell_count++)
          {
            if(count === 0)
            {
              table_data += '<th>'+cell_data[cell_count]+'</th>';
            }
            else
            {
              table_data += '<td>'+cell_data[cell_count]+'</td>';
            }
          }
          table_data += '</tr>';
        }
        table_data += '</table>';
        // console.log(table_data)
        $('#employee_table').html(table_data);
      }
    });
  });
});
});
```

2.1.4

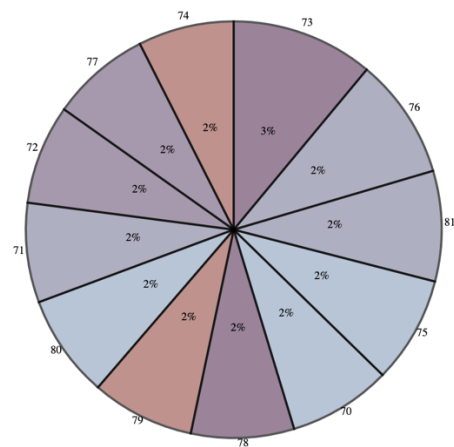
acceleration	numberofcar
8	1
8.5	2
9	1
9.5	2
10	4
10.5	1
11	7
11.1	1
11.2	1
11.3	1
11.4	2
11.5	7
11.6	1
12	10
12.1	1
12.2	2

```

<script>
$(document).ready(function(){
  $('#load_data').click(function(){
    $.ajax({
      url: '2.1.3_cur.csv',
      dataType: 'text',
      success: function(data) {
        var data = data.split(/\n|\r/);
        var table_data = '<table class="table table-bordered table-striped">';
        for(var count = 0; count<data.length; count++)
        {
          var cell_data = data[count].split(",");
          table_data += '<tr>';
          for(var cell_count=0; cell_count<cell_data.length; cell_count++)
          {
            if(count == 0)
            {
              table_data += '<th>'+cell_data[cell_count]+'</th>';
            }
            else
            {
              table_data += '<td>'+cell_data[cell_count]+'</td>';
            }
          }
          table_data += '</tr>';
        }
        table_data += '</table>';
        // console.log(table_data);
        $('#employee_table').html(table_data);
      }
    });
  });
});
</script>

```

2.2.1



```

- var color = d3.scaleOrdinal()
  .domain(data)
  .range(["#98abc5", "#a899a6", "#7b6888", "#6b486b", "#85d5e1"]);
- var pie = d3.pie()
  .value(function(d) {return d.value; });
var data_ready = pie(d3.entries(data))

// Build the pie chart: Basically, each part of the pie is a path that we build using the arc function.
- svg1
  .selectAll('whatever')
  .data(data_ready)
  .enter()
  .append('path')
  .attr('d', d3.arc()
    .innerRadius(0)
    .outerRadius(radius)
  )
  .attr('fill', function(d){ return color(d.data.key); })
  .attr('stroke', "black")
  .style('stroke-width', "2px")
  .style('opacity', 0.7)

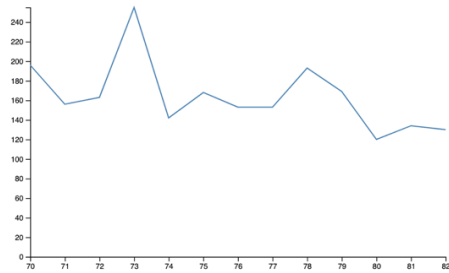
- svg1
  .selectAll('mySlices')
  .data(data_ready)
  .enter()
  .append('text')
  .text(function(d){ return d.data.key ; })
  .attr('transform', function(d) { return "translate(" + arcGenerator.centroid(d) + ")"; })
  .style('text-anchor', "middle")
  .style('font-size', 10)

- var arc = d3.arc()
  .innerRadius(0)
  .outerRadius(radius-5)

```

2.2.2

Using the array created in 2.1.3, plot a line graph using D3 along with the table from 2.1.3 to see the total number of cylinders for each model year, label the x and y axis and sort the x axis in decreasing order

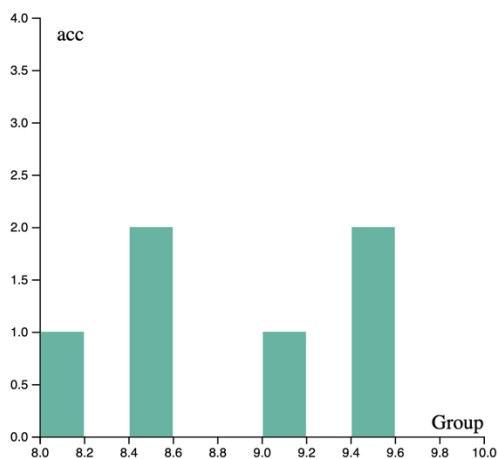


```

1 var svg2 = d3.select("#my_data")
2 .append("svg")
3   .attr("width", width + margin.left + margin.right)
4   .attr("height", height + margin.top + margin.bottom)
5   .append("g")
6   .attr("transform",
7     "translate(" + margin.left + "," + margin.top + ")");
8 d3.csv("2.1.3.csv", function(d) {
9   console.log(d)
10  return (date: d.modelyear, value: d.cylinder)
11 },
12 function(data) {
13   var x = d3.scaleLinear()
14     .domain(d3.extent(data, function(d) { return d.date; })))
15     .range([0, width]);
16   svg2.append("g")
17     .attr("transform", "translate(0," + height + ")")
18     .call(d3.axisBottom(x));
19
20   var y = d3.scaleLinear()
21     .domain([0, d3.max(data, function(d) { return d.value; })))
22     .range([height, 0]);
23   svg2.append("g")
24     .call(d3.axisLeft(y));
25
26   svg2.append("path")
27     .datum(data)
28     .attr("fill", "none")
29     .attr("stroke", "steelblue")
30     .attr("stroke-width", 1.5)
31     .attr("d", d3.line()
32       .x(function(d) { return x(d.date); })
33       .y(function(d) { return y(d.value); })
34     );
35 })

```

2.2.3



```

var histogram = d3.histogram()
  .value(function(d) { return d.accel; }) // I need to give the vector of value
  .domain(x.domain()) // then the domain of the graphic
  .thresholds(x.ticks(10)); // then the numbers of bins

// And apply this function to data to get the bins
var bins = histogram(data);

// Y axis: scale and draw:
var y = d3.scaleLinear()
  .range([height, 0]);
y.domain([0, d3.max(bins, function(d) { return d.length; })]); // d3.hist has to be called before

svg.append("g")
  .call(d3.axisLeft(y));
// append the bar rectangles to the svg element
svg.append("text")
  .attr("class", "x label")
  .attr("text-anchor", "end")
  .attr("x", width)
  .attr("y", height - 6)
  .text("Group");
svg.append("text")
  .attr("class", "y label")
  .attr("text-anchor", "end")
  .attr("x", 35)
  .attr("y", 6)
  .attr("dy", ".75em")
  // .attr("transform", "rotate(-90)")
  .text("acc");
svg.selectAll("rect")
  .data(bins)
  .enter()
  .append("rect")
  .attr("x", 1)
  .attr("transform", function(d) { return "translate(" + x(d.x0) + "," + y(d.length) + ")"; })

```

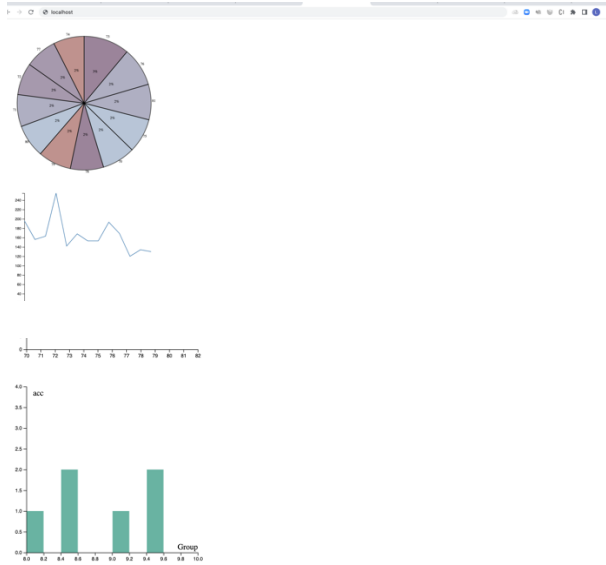
2.3.1 Follow the tutorial for installing Apache HTTP server &2.3.2 Create a Virtual Host

```

larry_1@liangdeMBP html % sudo apachectl start
Password:
larry_1@liangdeMBP html %

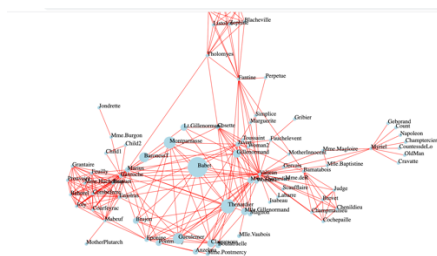
```

2.3.3 Host your webpage (webpage from Part II problem 2) on the virtual host.

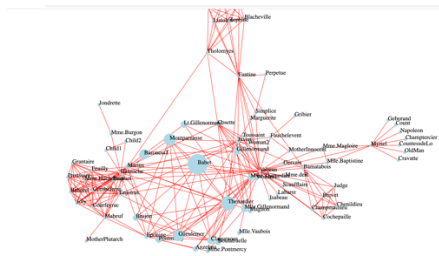


Part III Tasks: (30 marks)

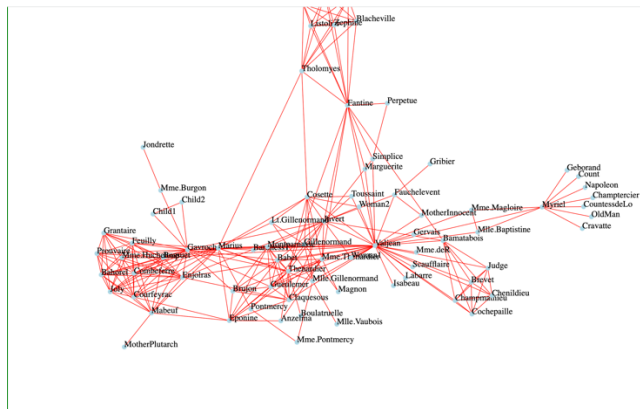
3.1.1 Plot a Network graph that shows character co-occurrence in Les
&3.1.2 Add labels to each node. (i.e. each node should show the name of the character)



3.1.3 It can be difficult to observe micro and macro features simultaneously with complex graphs. If you zoom in for detail, the graph is too big to view in its entirety. Use fisheye distortion (magnifies the local region around the mouse, while leaving the larger graph unaffected for context) for the above network graph. (you can use D3's fisheye plugin).

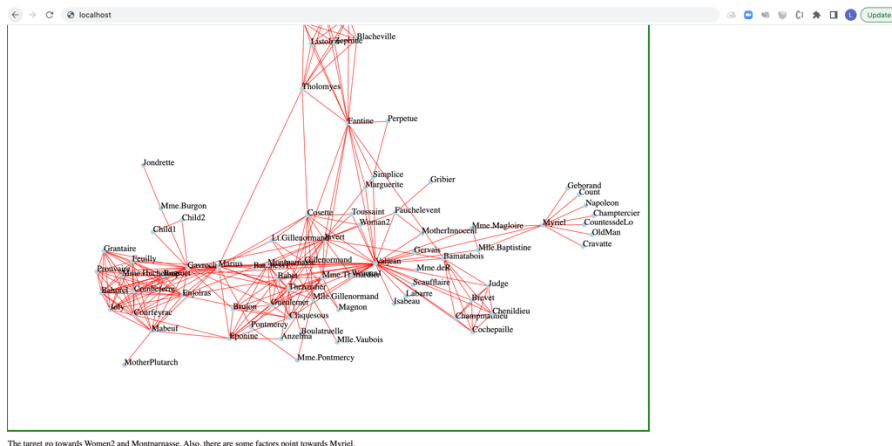


3.2.1 Add a <p> tag to your HTML file and infer what you see from the graph.



The target go towards Women2 and Montparnasse. Also, there are some factors point towards Myriel.

3.2.2 Host your webpage (from 3.1.3) on the virtual host.



The target go towards Women2 and Montparnasse. Also, there are some factors point towards Myriel.


```

0      const nodes = exportNodes(data);
1      console.log(data);
2      const links = data;
3
4      const fisheye = d3.fisheye.circular()
5        .radius(200)
6        .distortion(10);
7
8      const svg = sel.append("svg")
9        .attr("width", visConfig.width)
10       .attr("height", visConfig.height);
11
12      const lines = svg.selectAll("NONE")
13        .data(links)
14        .enter()
15        .append("line")
16        .style("stroke", "rgba(255,0,0,0.3)");
17
18      const circleGroup = svg.selectAll("NONE")
19        .data(nodes)
20        .enter()
21        .append("g");
22
23      const circles = circleGroup.append("circle")
24        .attr("r", 10)
25        .style("fill", "lightgreen");
26
27      circleGroup.append("text")
28        .style("user-select", "none")
29        .text(node => node.name);
30
31      const forceSimul = d3.forceSimulation(nodes)
32        .append("line")
33        .style("stroke", "rgba(255,0,0,0.3)");
34
35      const circleG = svg.selectAll("NONE")
36        .data(nodes)
37        .enter()
38        .append("g");
39
40      const circles = circleG.append("circle")
41        .attr("r", 10)
42        .style("fill", "lightgreen");
43
44      circleGroup.append("text")
45        .style("user-select", "none")
46        .text(node => node.name);
47
48      const forceSimul = d3.forceSimulation(nodes)
49        .force("center", d3.forceCenter(visConfig.width/2, visConfig.height/2+100))
50        .force("collide", d3.forceCollide().radius(12))
51        .force("charge", d3.forceManyBody().strength(-130))
52        .force("links", d3.forceLink(links).id(node=>node.name).distance(60))
53        .on("tick", tick);
54
55      svg.on("mousemove", function(e) {
56        fisheye.focus([e.offsetX, e.offsetY]);
57
58        circleGroup.each(function(d) { d.fisheye = fisheye(d); })
59
60        .attr("transform", function(d) {
61          const t = [d.fisheye.x, d.fisheye.y];
62          return `translate(${t})`;
63        })
64      });

```