In [1]:
```
using JuMP, Gurobi, LinearAlgebra, CSV, DataFrames, Pkg, Distances
```

In [2]:
```
his = CSV.read("/Users/bennetthellman/Desktop/OneDrive - Massachusetts Institute
fut = CSV.read("/Users/bennetthellman/Desktop/OneDrive - Massachusetts Institute
dc = CSV.read("/Users/bennetthellman/Desktop/OneDrive - Massachusetts Institute
pred_d = CSV.read("/Users/bennetthellman/Desktop/OneDrive - Massachusetts Instit
```

In [3]:
```
pred_d
```

Out[3]: 99,450 rows × 16 columns (omitted printing of 8 columns)

	Column1	FIPS_Code	State.Name	County.Name	Latitude	Longitude	Year	Week
	Int64	Int64	String	String	Float64	Float64	Int64	Int64
1	1	9001	Connecticut	Fairfield County	41.244	-73.363	2015	27
2	2	9001	Connecticut	Fairfield County	41.244	-73.363	2015	28
3	3	9001	Connecticut	Fairfield County	41.244	-73.363	2015	29
4	4	9001	Connecticut	Fairfield County	41.244	-73.363	2015	30
5	5	9001	Connecticut	Fairfield County	41.244	-73.363	2015	31
6	6	9001	Connecticut	Fairfield County	41.244	-73.363	2015	32
7	7	9001	Connecticut	Fairfield County	41.244	-73.363	2015	33
8	8	9001	Connecticut	Fairfield County	41.244	-73.363	2015	34
9	9	9001	Connecticut	Fairfield County	41.244	-73.363	2015	35
10	10	9001	Connecticut	Fairfield County	41.244	-73.363	2015	36
11	11	9001	Connecticut	Fairfield County	41.244	-73.363	2015	37
12	12	9001	Connecticut	Fairfield County	41.244	-73.363	2015	38
13	13	9001	Connecticut	Fairfield County	41.244	-73.363	2015	39
14	14	9001	Connecticut	Fairfield County	41.244	-73.363	2015	40
15	15	9001	Connecticut	Fairfield County	41.244	-73.363	2015	41
16	16	9001	Connecticut	Fairfield County	41.244	-73.363	2015	42
17	17	9001	Connecticut	Fairfield County	41.244	-73.363	2015	43
18	18	9001	Connecticut	Fairfield County	41.244	-73.363	2015	44
19	19	9001	Connecticut	Fairfield County	41.244	-73.363	2015	45
20	20	9001	Connecticut	Fairfield County	41.244	-73.363	2015	46
21	21	9001	Connecticut	Fairfield County	41.244	-73.363	2015	47
22	22	9001	Connecticut	Fairfield County	41.244	-73.363	2015	48
23	23	9001	Connecticut	Fairfield County	41.244	-73.363	2015	49
24	24	9001	Connecticut	Fairfield County	41.244	-73.363	2015	50

	Column1	FIPS_Code	State.Name	County.Name	Latitude	Longitude	Year	Week
	Int64	Int64	String	String	Float64	Float64	Int64	Int64
25	25	9001	Connecticut	Fairfield County	41.244	-73.363	2015	51
26	26	9001	Connecticut	Fairfield County	41.244	-73.363	2015	52
27	27	9003	Connecticut	Hartford County	41.82	-72.718	2015	27
28	28	9003	Connecticut	Hartford County	41.82	-72.718	2015	28
29	29	9003	Connecticut	Hartford County	41.82	-72.718	2015	29
30	30	9003	Connecticut	Hartford County	41.82	-72.718	2015	30
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

```
In [4]:
```

```julia
agg_county = groupby(pred_d, :FIPS_Code)
```

```
Out[4]:
```

GroupedDataFrame with 765 groups based on key: FIPS_Code

First Group (130 rows): FIPS_Code = 9001

	Column1	FIPS_Code	State.Name	County.Name	Latitude	Longitude	Year	Week
	Int64	Int64	String	String	Float64	Float64	Int64	Int64
1	1	9001	Connecticut	Fairfield County	41.244	-73.363	2015	27
2	2	9001	Connecticut	Fairfield County	41.244	-73.363	2015	28
3	3	9001	Connecticut	Fairfield County	41.244	-73.363	2015	29
4	4	9001	Connecticut	Fairfield County	41.244	-73.363	2015	30
5	5	9001	Connecticut	Fairfield County	41.244	-73.363	2015	31
6	6	9001	Connecticut	Fairfield County	41.244	-73.363	2015	32
7	7	9001	Connecticut	Fairfield County	41.244	-73.363	2015	33
8	8	9001	Connecticut	Fairfield County	41.244	-73.363	2015	34
9	9	9001	Connecticut	Fairfield County	41.244	-73.363	2015	35
10	10	9001	Connecticut	Fairfield County	41.244	-73.363	2015	36
11	11	9001	Connecticut	Fairfield County	41.244	-73.363	2015	37
12	12	9001	Connecticut	Fairfield County	41.244	-73.363	2015	38
13	13	9001	Connecticut	Fairfield County	41.244	-73.363	2015	39
14	14	9001	Connecticut	Fairfield County	41.244	-73.363	2015	40
15	15	9001	Connecticut	Fairfield County	41.244	-73.363	2015	41
16	16	9001	Connecticut	Fairfield County	41.244	-73.363	2015	42
17	17	9001	Connecticut	Fairfield County	41.244	-73.363	2015	43
18	18	9001	Connecticut	Fairfield County	41.244	-73.363	2015	44
19	19	9001	Connecticut	Fairfield County	41.244	-73.363	2015	45
20	20	9001	Connecticut	Fairfield County	41.244	-73.363	2015	46

	Column1	FIPS_Code	State.Name	County.Name	Latitude	Longitude	Year	Week
	Int64	Int64	String	String	Float64	Float64	Int64	Int64
21	21	9001	Connecticut	Fairfield County	41.244	-73.363	2015	47
22	22	9001	Connecticut	Fairfield County	41.244	-73.363	2015	48
23	23	9001	Connecticut	Fairfield County	41.244	-73.363	2015	49
24	24	9001	Connecticut	Fairfield County	41.244	-73.363	2015	50
25	25	9001	Connecticut	Fairfield County	41.244	-73.363	2015	51
26	26	9001	Connecticut	Fairfield County	41.244	-73.363	2015	52
27	19891	9001	Connecticut	Fairfield County	41.244	-73.363	2016	1
28	19892	9001	Connecticut	Fairfield County	41.244	-73.363	2016	2
29	19893	9001	Connecticut	Fairfield County	41.244	-73.363	2016	3
30	19894	9001	Connecticut	Fairfield County	41.244	-73.363	2016	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

⋮

Last Group (130 rows): FIPS_Code = 54109

	Column1	FIPS_Code	State.Name	County.Name	Latitude	Longitude	Year	Week
	Int64	Int64	String	String	Float64	Float64	Int64	Int64
1	19865	54109	West Virginia	Wyoming County	37.634	-81.539	2015	27
2	19866	54109	West Virginia	Wyoming County	37.634	-81.539	2015	28
3	19867	54109	West Virginia	Wyoming County	37.634	-81.539	2015	29
4	19868	54109	West Virginia	Wyoming County	37.634	-81.539	2015	30
5	19869	54109	West Virginia	Wyoming County	37.634	-81.539	2015	31
6	19870	54109	West Virginia	Wyoming County	37.634	-81.539	2015	32
7	19871	54109	West Virginia	Wyoming County	37.634	-81.539	2015	33
8	19872	54109	West Virginia	Wyoming County	37.634	-81.539	2015	34
9	19873	54109	West Virginia	Wyoming County	37.634	-81.539	2015	35
10	19874	54109	West Virginia	Wyoming County	37.634	-81.539	2015	36
11	19875	54109	West Virginia	Wyoming County	37.634	-81.539	2015	37
12	19876	54109	West Virginia	Wyoming County	37.634	-81.539	2015	38
13	19877	54109	West Virginia	Wyoming County	37.634	-81.539	2015	39
14	19878	54109	West Virginia	Wyoming County	37.634	-81.539	2015	40
15	19879	54109	West Virginia	Wyoming County	37.634	-81.539	2015	41
16	19880	54109	West Virginia	Wyoming County	37.634	-81.539	2015	42
17	19881	54109	West Virginia	Wyoming County	37.634	-81.539	2015	43
18	19882	54109	West Virginia	Wyoming County	37.634	-81.539	2015	44

	Column1	FIPS_Code	State.Name	County.Name	Latitude	Longitude	Year	Week
	Int64	Int64	String	String	Float64	Float64	Int64	Int64
19	19883	54109	West Virginia	Wyoming County	37.634	-81.539	2015	45
20	19884	54109	West Virginia	Wyoming County	37.634	-81.539	2015	46
21	19885	54109	West Virginia	Wyoming County	37.634	-81.539	2015	47
22	19886	54109	West Virginia	Wyoming County	37.634	-81.539	2015	48
23	19887	54109	West Virginia	Wyoming County	37.634	-81.539	2015	49
24	19888	54109	West Virginia	Wyoming County	37.634	-81.539	2015	50
25	19889	54109	West Virginia	Wyoming County	37.634	-81.539	2015	51
26	19890	54109	West Virginia	Wyoming County	37.634	-81.539	2015	52
27	59619	54109	West Virginia	Wyoming County	37.634	-81.539	2016	1
28	59620	54109	West Virginia	Wyoming County	37.634	-81.539	2016	2
29	59621	54109	West Virginia	Wyoming County	37.634	-81.539	2016	3
30	59622	54109	West Virginia	Wyoming County	37.634	-81.539	2016	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Part D

In [5]:

```julia
county_tot_d = combine(agg_county, :d_pallets => sum);
```

In [6]:

```julia
df = []
for i in 1:size(agg_county,1)
    row_num = size(agg_county[i])[1]
    x = sum(agg_county[i][row_num-7:row_num,:d_pallets])
    append!(df, x)
end
```

In [7]:

```julia
variable_cost = Vector(dc[:,:Variable_Cost]);
fixed_cost = Vector(dc[:,:Fixed_Cost]);
```

In [8]:

```julia
d_mat = zeros((size(fixed_cost,1), size(agg_county,1)))
for i in 1:size(fixed_cost,1)
    dc_lat = dc[i,:Latitude]
    dc_long = dc[i,:Longitude]
    for j in 1:size(agg_county,1)
        county_lat = agg_county[j][1,:Latitude]
        county_long = agg_county[j][1,:Longitude]
        d_mat[i,j]= haversine((dc_lat,dc_long),(county_lat,county_long), 3958.8)
    end
end
```

In [44]:

```julia
model = Model(with_optimizer(Gurobi.Optimizer, Gurobi.Env()))
```

```
set_optimizer_attribute(model, "OutputFlag", 0)

n = size(county_tot_d,1)

@variable(model, b[i=1:20], Bin)
@variable(model, u[i=1:20,j=1:n], Bin)
@variable(model, c[i=1:20]>=0)


@constraint(model, [i=1:3], b[i]==1)
@constraint(model, [i=1:2], c[i]==1200000)
@constraint(model, [i=3], c[i]==900000)
@constraint(model, [j=1:n], sum(u[:,j])== 1)
@constraint(model, [i=1:20, j=1:n], u[i,j] <= b[i])
@constraint(model, [i=1:20], c[i]*(5/13.5) >= sum(df[j]*u[i,j] for j=1:n))
@constraint(model, [i=1:20], c[i]<=b[i]*1200000)

@objective(model, Min, sum(variable_cost[i]*c[i] for i=1:20) + sum(fixed_cost[i]
    sum(sum((1.55/20)*d_mat[i,j]*county_tot_d[j,:d_pallets_sum]*u[i,j] for j=1:n

optimize!(model)
```

```
Academic license - for non-commercial use only - expires 2022-08-19
```

In [45]:

```
objective_value(model)
```

Out[45]:

```
7.533748614634609e8
```

In [46]:

```
b=value.(b)
```

Out[46]:

```
20-element Vector{Float64}:
 1.0
 1.0
 1.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 1.0
 0.0
 0.0
 1.0
 0.0
 0.0
 1.0
 1.0
 1.0
 0.0
```

In [47]:

```
u=value.(u)
```

Out[47]:

```
20×765 Matrix{Float64}:
 0.0  1.0  0.0  1.0  1.0  1.0  1.0  1.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     1.0  1.0  1.0  1.0  1.0  1.0  1.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 1.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

In [48]:

```
c=value.(c)
```

Out[48]:

```
20-element Vector{Float64}:
      1.2e6
      1.2e6
 900000.0
      0.0
      0.0
      0.0
      0.0
      0.0
      0.0
      0.0
      1.1999247319021097e6
      0.0
      0.0
 701461.9812655906
      0.0
      0.0
      1.199839817498397e6
      1.1992867442985747e6
      1.1999464575535741e6
      0.0
```

In [49]:

```
sum(u, dims=2)
```

Out[49]:

```
20×1 Matrix{Float64}:
  57.0
 122.0
 120.0
   0.0
   0.0
   0.0
   0.0
   0.0
   0.0
   0.0
 222.0
   0.0
   0.0
  41.0
   0.0
   0.0
  27.0
  60.0
 116.0
   0.0
```

In [50]:

```
construction_cost = sum(variable_cost[i]*c[i] for i=1:20) + sum(fixed_cost[i]*b[
```

Out[50]:

```
5.560500726415393e8
```

In [51]:

```
transpo_cost = sum(sum((1.55/20)*d_mat[i,j]*county_tot_d[j,:d_pallets_sum]*u[i,j
```

Out[51]:

```
1.9732478882192114e8
```

Part E

In [64]:

```
mod1 = Model(with_optimizer(Gurobi.Optimizer, Gurobi.Env()))
set_optimizer_attribute(mod1, "OutputFlag", 0)

n = size(county_tot_d,1)

@variable(mod1, c[i=1:20]>=0)
@variable(mod1, b[i=1:20], Bin)

@constraint(mod1, [i=1:2], c[i]==1200000)
@constraint(mod1, [i=3], c[i]==900000)
@constraint(mod1, [i=1:3], b[i]==1)
@constraint(mod1, [i=1:20], c[i]<=b[i]*1200000)
@constraint(mod1, sum(c[i]*(5/13.5) for i=1:20) >= sum(df[j] for j=1:n))

@objective(mod1, Min, sum(fixed_cost[i]*b[i] for i=1:20) + sum(variable_cost[i]*

optimize!(mod1)
```

```
Academic license - for non-commercial use only - expires 2022-08-19
```

In [65]:

```
objective_value(mod1)
```

Out[65]:

```
5.1707796649073195e8
```

In [66]:

```
b=value.(b)
```

Out[66]:

```
20-element Vector{Float64}:
  1.0
  1.0
  1.0
 -0.0
 -0.0
  1.0
 -0.0
 -0.0
  1.0
 -0.0
  1.0
  1.0
 -0.0
```

```
-0.0
-0.0
-0.0
-0.0
-0.0
 1.0
-0.0
```

In [67]:

```
c=value.(c)
```

Out[67]:

```
20-element Vector{Float64}:
      1.2e6
      1.2e6
 900000.0
      0.0
      0.0
      1.2e6
      0.0
      0.0
      1.2e6
      0.0
      1.2e6
 700237.024943693
      0.0
      0.0
      0.0
      0.0
      0.0
      0.0
      1.2e6
      0.0
```

In [68]:

```
mod2 = Model(with_optimizer(Gurobi.Optimizer, Gurobi.Env()))
set_optimizer_attribute(mod2, "OutputFlag", 0)

n = size(county_tot_d,1)

@variable(mod2,u[i=1:20,j=1:n], Bin)

b = value.(b)
c = value.(c)

@constraint(mod2, [j=1:n], sum(u[:,j])== 1)
@constraint(mod2, [i=1:20, j=1:n], sum(df[j]*u[i,j] for j=1:n) <= 1.001*c[i]*5/1

@objective(mod2, Min, sum(sum(1.55/20*d_mat[i,j]*county_tot_d[j,:d_pallets_sum]*

optimize!(mod2)
```

```
Academic license - for non-commercial use only - expires 2022-08-19
```

In [69]:

```
objective_value(mod2)
```

Out[69]:

```
5.059191728940411e8
```