Bennett Hellman

15.072: Advanced Analytics Edge Fall 2021
Homework 4: From Predictions to Prescriptions

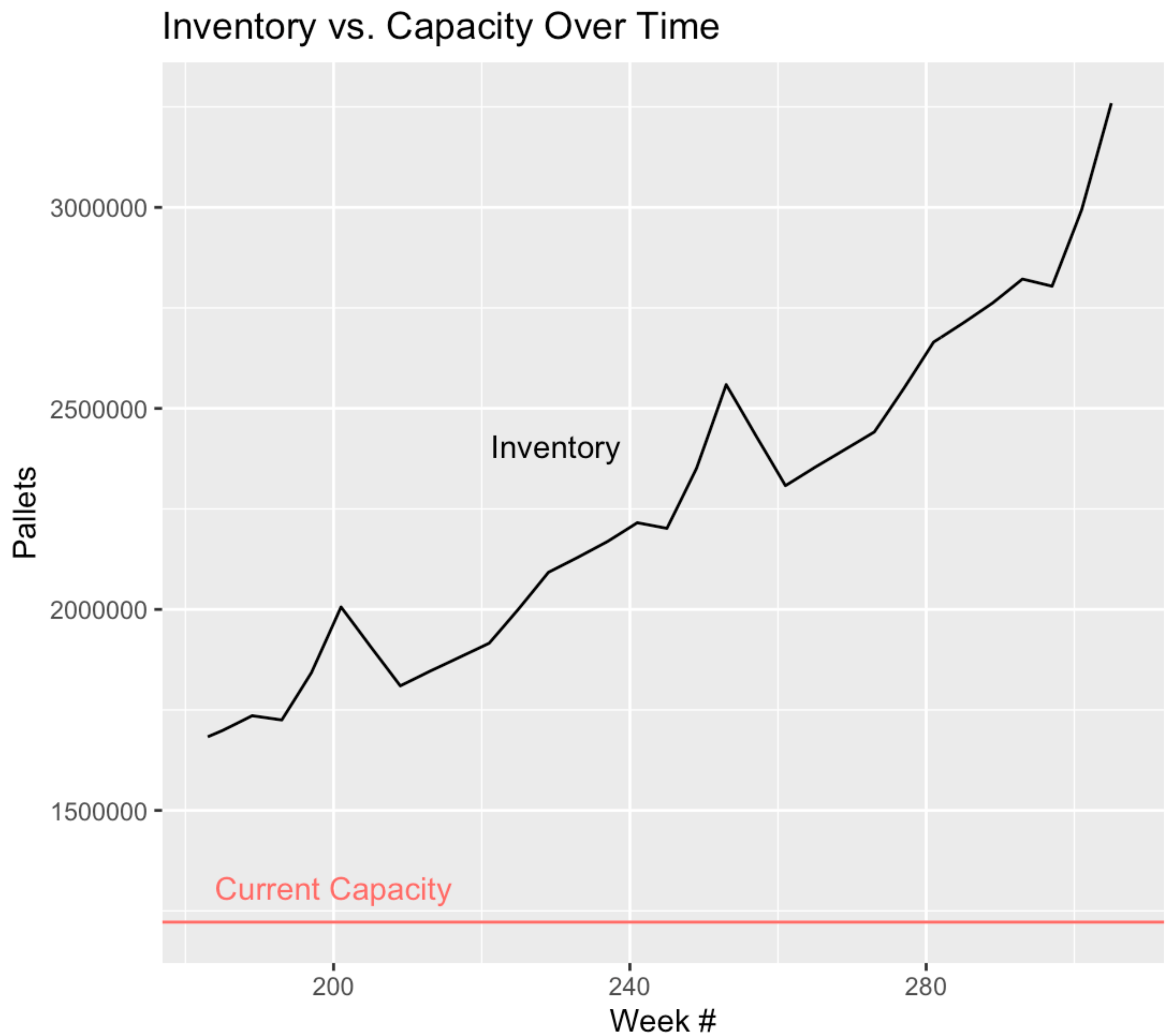
a.i)

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.312e+00  3.317e-02  69.712   <2e-16 ***
linc        -2.351e-01  3.147e-03 -74.711   <2e-16 ***
Week_Num     4.771e-03  2.832e-05 168.495   <2e-16 ***
bseason2     2.167e-03  3.757e-03   0.577    0.564    
bseason3     2.182e-03  3.762e-03   0.580    0.562    
bseason4     1.750e-03  3.770e-03   0.464    0.643    
bseason5     1.315e-03  3.782e-03   0.348    0.728    
bseason6     5.064e-02  3.797e-03  13.336   <2e-16 ***
bseason7     5.318e-02  3.816e-03  13.936   <2e-16 ***
bseason8     4.891e-02  3.838e-03  12.744   <2e-16 ***
bseason9     5.253e-02  3.863e-03  13.600   <2e-16 ***
bseason10    5.409e-02  3.891e-03  13.903   <2e-16 ***
bseason11    2.517e-03  3.922e-03   0.642    0.521    
bseason12    1.423e-01  3.956e-03  35.974   <2e-16 ***
bseason13    1.410e-01  3.993e-03  35.317   <2e-16 ***
```

```
his<-his%>%mutate(lsale = log(Sales/Population),saleper = Sales/Population,
            linc = log(Income), bseason = as.factor(Season))
train <- his%>%filter(Year <=2013)
test<-his%>%filter(Year ==2014)
mod <- lm(lsale~linc+Week_Num+bseason, data = train)
summary(mod)
```

a.ii)
It is important to note that none of these estimates are causal and can only be interpreted as correlation. A one percent increase in income per capita is associated with a 0.235% decrease in sales per capita. Another week away from the beginning of 2012 is associated with a .477% increase in sales per capita. Using season 2 as an example, being in season 2 is associated with a .217% increase in sales per capita. All other seasons can be interpreted the same was a 100*coefficient increase in sales per capita.

a.iii) **Out-of-sample performance: R^2 = 0.777, MAE = 0.163, RMSE = 0.209** b.i)



b.ii) **Shortage = 166.674%**

```
percent_cap_short <- (inv_df[123,3]-current_cap)/(current_cap)*100
```

c.i)

Decision Variables:

$b_i = 1\ if\ DC_i\ is\ chosen, 0\ otherwise$

$u_i = 1\ if\ DC_i\ \ is\ serves\ County_j, 0\ otherwise$

$c_i = the\ capacity\ utilized\ of\ DC_i$

Inputs:

$FixedCost_i = fixed\ cost\ of\ constructing\ DC_i$

$VariableCost_i = variable\ cost\ of\ constucting\ DC_i\ per\ sqft$

$d_{ij} = distance\ from\ DC_i\ to\ county_j$

$CapacityDC_i = maximum\ capacity\ of\ DC_i$

$x_{jt} = predicted\ demand\ of\ county\ j\ in\ week\ t$

$$\mathrm{Min}_{\mathrm{b,u,c}} \sum_{\mathrm{i}} FixedCost_i b_i + \sum_{\mathrm{i}} VariableCost_i c_i + \sum_{\mathrm{i}} \sum_{\mathrm{j}} \sum_{\mathrm{t}} \left(\frac{1.55}{20}\right) d_{ij} x_{jt} u_{ij}$$

$$b_i = 1\ \forall\ i = 1,2,3$$

$$c_i = 1{,}200{,}000\ \forall\ i = 1,2$$

$$c_i = 900{,}000\ \forall\ i = 3$$

$$\sum_{j}^{20} u_{ij} = 1\ \forall\ j$$

$$u_{ij} \le b_i\ \forall\ i, j$$

$$c_i = 1{,}200{,}000\ \forall\ i = 1,2$$

$$0 \le c_i \le CapacityDC_i b_i$$

$$\sum_{j} u_{ij} \le CapacityDC_i \left(\frac{5}{13.5}\right) \forall\ i$$

$$c_i \le 1{,}200{,}000 b_i \forall\ i$$

$$u_{ij}, b_i \in \{0,1\}$$

```
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]*b[i] for i=1:20) +
    sum(sum((1.55/20)*d_mat[i,j]*county_tot_d[j,:d_pallets_sum]*u[i,j] for j=1:n) for i=1:20))

optimize!(model)
```

d.i) **In addition to the 3 original DCs, there will also be DCs at Kalamazo, Lancaster, Scranton, Syracuse, Toledo.**

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

d.ii)
Providence = 57
Richmond = 122
Youngstown = 120
Kalamazo = 222
Lancaster = 41
Scranton = 27
Syracuse = 60
Toledo = 116

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

d.iii)
Construction Cost = $556,050,072.64
Transportation Cost = $197,324,788.82

```
construction_cost = sum(variable_cost[i]*c[i] for i=1:20) + sum(fixed_cost[i]*b[i] for i=1:20)
transpo_cost = sum(sum((1.55/20)*d_mat[i,j]*county_tot_d[j,:d_pallets_sum]*u[i,j] for j=1:n) for i=1:20)
```

e.i)
Cheapest DC's w/ capacity (in sqft)
- **Providence = 1,200,000**
- **Richmond = 1,200,000**
- **Youngstown = 900,000**
- **Bangor = 1,200,000**
- **Burlington = 1,200,000**
- **Dover = 1,200,000**

Kalamazoo = 700237.0249
Worcester = 1,200,000

Construction Costs = $517,077,966.49
Transportation Costs = $505,919,172.89

```
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]*c[i]
for i=1:20))

optimize!(mod1)

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]*5/13.5)

@objective(mod2, Min,
sum(sum(1.55/20*d_mat[i,j]*county_tot_d[j,:d_pallets_sum]*u[i,j] for j=1:n) for
i=1:20))

optimize!(mod2)
```

e.ii) **In the baseline solution, which first optimizes for construction cost and then generates transportation costs, the solution reflects the priorities. Construction cost is lower in the**

second model because it is explicitly minimizes it first. However, this comes at the cost of higher transportation costs. In part d, we were only concerned with the overall cost and thus that was decreased, and transportation costs were significantly less.

R-Code Appendix

```
library(tidyverse)
library(Metrics)
library(zoo)

his = read.csv("/Users/bennetthellman/Desktop/OneDrive - Massachusetts Institute of
Technology/AE/HWs/HW4/Dartboard_historical-1.csv");
fut = read.csv("/Users/bennetthellman/Desktop/OneDrive - Massachusetts Institute of
Technology/AE/HWs/HW4/Dartboard_future.csv.crdownload");
dc = read.csv("/Users/bennetthellman/Desktop/OneDrive - Massachusetts Institute of
Technology/AE/HWs/HW4/Dartboard_dcs.csv");

#dc<-dc%>%mutate(County.Name = Location)

#mdf = merge(his, dc, by.x ="State.Name", by.y = "Location", all.x = TRUE)

his<-his%>%mutate(lsale = log(Sales/Population),saleper = Sales/Population,
                  linc = log(Income), bseason = as.factor(Season))
train <- his%>%filter(Year <=2013)
test<-his%>%filter(Year ==2014)

mod <- lm(lsale~linc+Week_Num+bseason, data = train)
summary(mod)
summary(mod)$r.squared

pred <- predict(mod, newdata=test)
#OSR^2
1 - sum((pred - test$lsale)^2) / sum((mean(train$lsale) - test$lsale)^2)
#OMAE
mae(test$lsale, pred)
#ORMSE
rmse(test$lsale, pred)

fut<-fut%>%mutate(linc = log(Income), bseason = as.factor(Season))
fut$forecast <- exp(predict(mod, newdata=fut))*fut$Population

#b
inv_df <- fut %>% group_by(Week_Num) %>% summarize(tot_d= sum(forecast)) %>%
  mutate(inv = (rollapply(data = tot_d, FUN=sum, width=8, align="left", fill=NA)/1000))
#Claire Sailard helped me with this function
current_cap<- dc%>%summarise(tot_pallets_cap = sum(5*Current_Size/(13.5)))

ggplot(main = "Inventory vs. Capacity Over Time")+geom_line(aes(x = Week_Num, y = inv),
data = inv_df) + geom_hline(aes(yintercept=1222222, color = "red")) +
  geom_text(aes(200,1222222,label = "Current Capacity", vjust = -1, color = "red")) +
geom_text(aes(230,2322222,label = "Inventory", vjust = -1))+
```

```
 labs(x ="Week #", y="Pallets", title="Inventory vs. Capacity Over Time")+
theme(legend.position = "none")+xlim(183,306)

#bi
percent_cap_short <- (inv_df[123,3]-current_cap)/(current_cap)*100
percent_cap_short

#exportation
fut$d_pallets = fut$forecast/1000
write.csv(fut,'/Users/bennetthellman/Desktop/OneDrive - Massachusetts Institute of
Technology/AE/HWs/HW4/pred_d.csv')
```

Bennett Hellman

Julia Code Appendix

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]:

```
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]:

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

In [6]:

```
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]:

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

In [8]:

```
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]:

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

```julia
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]:

```julia
objective_value(model)
```

Out[45]:

```
7.533748614634609e8
```

In [46]:

```julia
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]:

```julia
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]:

```julia
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]:

```julia
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
```